Comparison of link layer of BLE and 802.15.4

Running on Contiki OS

Master Thesis

PrithviRaj Narendra

Supervisor
Simon Duquennoy
Senior Researcher, SICS

Academic Examiner
Mats Brorsson
Professor, KTH

EIT ICT Labs Master School Embedded Systems Program
School of Information and Communication Technology
KTH Royal Institute of Technology
Stockholm, Sweden

12th September, 2014

Abstract

Hello Abstract.

Sammanfattning

Hello Sammanfattning.

Acknowledgment

Foremost I would like to earnestly thank my supervisor at Swedish Institute of Computer Science (SICS), Simon Duquennoy, for agreeing to supervise me in a topic that I proposed, guiding me patiently, consistently reviewing and providing constructive feedback for my work. Next I would like to thank Thiemo Voigt and the Networked Embedded Systems group at SICS for their warm inclusiveness and assisting me whenever I needed help. As usual this open source work stands on the shoulders of giants, I'm thankful for their amazing work. For all the resources and guidance that I have received in various forms from my fellow netizens, I'm grateful to them. I thank EIT ICT Labs for believing in me and providing me this opportunity. As always, my family and friends have supported and encouraged me, and for that they have my heartfelt thanks.

Contents

Abstract i

Sammanfattning iii

Acknowledgment v

List of Abbreviations ix

List of Figures xi

List of Tables xii

1 Introduction 1
1.1 General introduction to the domain of the thesis . . . 1
1.2 Problem definition . . . 1
1.3 Goal . . . 1
1.4 Outline of the report . . . 1

2 Background 2
2.1 Porting the Contiki Operating System (OS) to a New Platform . . . 2
2.2 Overview of Bluetooth Low Energy . . . 3
2.2.1 Design objectives of Bluetooth Low Energy (BLE) . . . 3
2.2.2 BLE network architecture . . . 3
2.3 BLE stack overview . . . 3
2.4 Overview of Contiki . . . 4
2.5 Overview of 802.15.4 . . . 4
2.5.1 Physical layer . . . 5
2.5.2 Medium Access Control (MAC) layer . . . 5

3 Methodology 6
3.1 Research Process . . . 6
3.2 Research Method . . . 6

4 Literature Study 7
4.1 Evaluating the Performance of BLE . . . 7
4.2 Evaluating the Performance of 802.15.4 . . . 8
4.3 Comparison of BLE and 802.15.4 . . . 8

5 Porting Contiki OS to a BLE platform 10
5.1 BLE Hardware Platform . . . 10
5.1.1 Requirements of the hardware platform . . . 10
5.1.2 Comparison and selection of the hardware platform . . . 11
5.1.3 Overview of nrf51822 System-on-Chip (SoC) and its platform . . . 11

5.2 Porting PCA10000 platform of nrf51822 to Contiki 12
5.2.1 Development Setup 12
5.2.2 Peripherals required for Contiki 12

6 Test Cases 15
6.1 Performance Metrics Definition 15
6.2 Test setup 16
6.2.1 BLE platform 16
6.2.2 802.15.4 platform 16
6.2.3 WiFi Interference 17
6.3 High-Throughput (HT) Test Design 18
6.4 Request-Response (RR) Test Design 19

7 Results and Analysis 22
7.1 High-Throughput Test 22
7.1.1 Graphical Representation of the Data Acquired 22
7.1.2 Analysis 23
7.2 Request-Response Test 25
7.2.1 Graphical Representation of Data Acquired 25
7.2.2 Analysis 26
7.3 Evaluation with existing research 28

8 Other Contributions 31
8.1 Advertisement Logger 31
8.2 Firmware for an Alarm Device 32

9 Conclusion and Future Work 34

Appendix A Contiki Folder and Makefile Structure 37
A.1 Folder structure of Contiki 37

Appendix B Advertisement Logger's Implementation 38
B.1 Scanning BLE Advertisements 38
B.2 Software Architecture 39
B.2.1 Main Function 39
B.2.2 UART String Receive Handler 40
B.2.3 Scan Interval Timer Handler 41
B.2.4 Scan Window Timer Handler 41
B.2.5 Radio Interrupt Routine 42

Appendix C Appendix | BLE SoC Comparison 43

Appendix D Data Acquired 45

List of Abbreviations

6LoWPAN	IPv6 over Low power Wireless Personal Area Networks.
ADC	Analog to Digital Conversion.
AFH	Adaptive Frequency Hopping.
API	Application Programming Interface.
ATT	Attribute Protocol.
BER	Bit Error Rate.
BLE	Bluetooth Low Energy.
CCA	Clear Channel Assessment.
CCCD	Client Characteristic Configuration Descriptor.
CRC	Cyclic Redundancy Check.
CSMA	Carrier Sense Multiple Access.
GAP	Generic Access Profile.
GATT	Generic Attribute Profile.
GFSK	Gaussian Frequency Shift Keying.
GPIO	General Purpose Input-Output.
HF-CLK	High Frequency Clock.
HID	Human Interface Device.
HT	High-Throughput.
IC	Integerated Circuit.
IFS	Inter Frame Space.
IoT	Internet of Things.
ISM	Industrial, Scientific and Medical.
L2CAP	Logical Link Control and Adaptation Protocol.
LED	Light Emitting Diode.
LF-CLK	Low Frequency Clock.
MAC	Medium Access Control.
MCU	microcontroller.
NVIC	Nested Vectored Interrupt Controller.
OS	Operating System.
PAN	Personal Area Network.
PDR	Packet Delivery Ratio.
PHY	Physical Layer.
PRR	Packet Reception Ratio.
RAM	Random-Access Memory.
RDC	Radio Duty Cycle.
RR	Request-Response.
RSSI	Received Signal Strength Indicatior.
RTC	Real Time Clock.
SDK	Software Development Kit.
SIG	Special Interest Group.
SoC	System-on-Chip.

UART	Universal Asynchronous Receiver/Transmitter.
UDP	User Datagram Protocol.

List of Figures

2.1 Typical BLE network 4

3.1 Thesis' tasks and steps in the research process 6

5.1 PCA10000 development board 11

6.1 Illustration of latency measurement 15
6.2 TmoteSky platform 16
6.3 2.4 GHz environment captured by rssi-scanner 17
6.4 Test setup for High-Throughput (HT) test cases 18
6.5 Setup to measure data rate with BLE 18
6.6 Illustration of the RR Test operation 19
6.7 Test modes for measuring latency using BLE 20

7.1 Data Rate 22
7.2 Reliability 22
7.3 Energy Consumption 23
7.4 Latency 25
7.5 Energy Consumption 25
7.6 Energy Consumption vs Latency 26

8.1 User Interface of Advertisement Logger 31
8.2 State Diagram of an alarm device from an user's perspective 32
8.3 State Diagram of an alarm appcessory controlled by BLE 33

B.1 Scanning BLE advertisements 38
B.2 Main function flowchart 39
B.3 UART string receive handler flowchart 40
B.4 Scan interval timer handler flowchart 41
B.5 Scan window timer handler flowchart 41
B.6 Radio interrupt routine flowchart 42

List of Tables

6.1 List of Request-Response (RR) test cases using BLE 20

7.1 Latency in terms of master and slave Radio Duty Cycle (RDC) 26
7.2 Comparison of BLE data rate achieved 29

1 | Introduction

1.1 General introduction to the domain of the thesis

What is BLE? Why was it developed and who/where is it being used? [1]

What is Contiki OS? Why was it and who/where is it being used? Explain the amount of R&D done in terms of communication protocols based on 802.15.4 [2].

1.2 Problem definition

Need for BLE in Contiki as a platform for Internet of Things (IoT). To understand the characteristics of BLE and 802.15.4 so that we can identify the applications suitable for these protocols.

1.3 Goal

To choose and integrate a BLE hardware platform with Contiki. To compare BLE and 802 in various performance criteria of data-rate, latency, reliability and energy consumption in environments with and without external interference.

1.4 Outline of the report

1.method??

2 | Background*

Research articles involving the process of porting of Contiki OS to a hardware platform are presented in section 2.1.

2.1 Porting the Contiki OS to a New Platform

Contiki OS project originating in 2002 has been ported to an increasing number of hardware platforms [3]. Since Contiki is an open-source BSD Clause-3 licensed project, there are many projects in various hardware platforms based on a fork from the Contiki repository.

These hardware platforms' processors range across a spectrum of 8-bit (8051 and AVR), 16-bit (MSP430) to 32-bit (ARM Cortex-M, PIC32). Contiki has support for 802.15.4 based wireless communication with various external transceivers and SoCs with built in radio transceiver. Various common features of many platforms such as Light Emitting Diodes (LEDs), buttons and serial port have modules in Contiki for common Application Programming Interface (API) across platforms.

In [4] the authors describe Contiki's port to a CC2430 based platform manufactured by Sensinode Ltd. CC2430 is an enhanced Intel 8051 processor based SoC having 802.15.4 physical layer compatible radio transceiver. The authors fully debugged the port which had of a code footprint of about 100 kB, varying based on the compiler mode and the features enabled. Many new features were added to the port including support for Analog to Digital Conversion (ADC) unit, all the sensor available on the platform (accelerometer, light sensor, voltage and temperature sensors), watchdog timer and the general purpose buttons. Because of the limited stack availability of 233 bytes, many optimizations such as moving variables to external Random-Access Memory (RAM) memory space and re-writing the radio driver for CC2430 to prevent stack from overflowing.

Contiki was ported to two new platforms, namely MicaZ and TelosB in a Bachelor thesis [5]. TelosB, similar to the TMote-Sky platform, consists of a 8MHz 16-bit MSP430 microcontroller (MCU), a CC2400 transceiver with 802.15.4 Physical Layer (PHY) and a host of sensors to measure light, temperature and humidity. The port to TelosB was done by changing the port to the fully supported TMote-Sky platform. MicaZ platform consisted of a 8-bit Atmel ATMega128L MCU and a CC2400 transceiver. MicaZ platform needed rewriting of the code for processor abstraction so that the high level Contiki APIs could work.

Contiki has been ported into platform based on a ARM Cortex M3 processor to create a device wired with Ethernet to connected to the Internet [6]. Ethernet based networking with the libraries present in Contiki was developed in this project to demonstrate as a proof of concept.

There are many unofficial ports of Contiki to various hardware platforms, including ones based on ARM Cortex M3 based processors. This can be seen in the port to STM32F10x based platform [7] and LPC1768 based platform [8], both having Cortex M3 processor.

*Includes text and images from minor thesis 'Business ideation for BLE'

2.2 Overview of Bluetooth Low Energy

Rest of overview, from what is missed in 1st chapter.

2.2.1 Design objectives of BLE

BLE was designed by Bluetooth Special Interest Group (SIG) from the ground up, which helped it achieve certain design goals. These design goals for this wireless personal area network were *low cost, worldwide operation, short range, robustness* and *low power*[9].

Low Power BLE aims to use a tiny batteries such as button cell to keep a device operating for months to years. To achieve this goal, BLE was optimized to communicate small amounts of data, such as the states of devices. Also BLE is optimized to have lower peak power requirements, which allows use of button cells to be used with BLE devices.

Worldwide Operation For a technology to be adopted, it is important that there is uniform conformity to the regulations around the world. The 2.4 GHz Industrial, Scientific and Medical (ISM) radio band is the only one available license free worldwide. The technology to develop wireless devices in this band is mature making it the suitable radio band for BLE.

Short Range BLE was designed to be for personal area network like Classic Bluetooth, which means that it is not a network to work with a cellular base station network. This design criterion goes hand in hand with *low power*.

Low Cost Lower power requirements mean that the batteries in BLE devices need to smaller and have to be replaced less frequently, both resulting in a reduction of cost for both the manufacturer and the customer. The use of the ISM band for communication levels removes the licensing entry barrier for start-ups to develop BLE devices. BLE embraces simplicity in its pursuit to lower the cost. BLE supports only single-hop communication in a star network, which reduces the memory and processor requirement for supporting the protocol. Simplicity was the key factor for the choosing of Gaussian Frequency Shift Keying (GFSK) as the modulation scheme for BLE to result in low cost, small radio implementation of the radio in the Integerated Circuits (ICs) for BLE.

Robustness The 2.4 GHz space is crowded with devices communicating with various standards as well as spurious noise making the robustness a key criteria in developing BLE standard. BLE uses a multi-channel hopping mechanism called Adaptive Frequency Hopping (AFH) to detect, avoid and recover from interference. In addition to AFH, BLE uses Cyclic Redundancy Check (CRC) to detect and recover from bit-errors due to background noise.

2.2.2 BLE network architecture

Single and Dual devices, Advertisers and scanners, Master and Slave, Star Network, figure 2.1

2.3 BLE stack overview

Explain host and controller division. More details about link layer.

Image of the stack

The link layer of 802.15.4 consists of MAC layer on top of a RDC layer. For the RDC layer the Null-RDC and ContikiMAC driver will be tested in each of the test with Carrier Sense

Figure 2.1: Typical BLE network

Multiple Access (CSMA) as the MAC layer. In case of Contiki-MAC the receiving node switches on periodically to sense if there are any packets that need to be received. The default time of this period is 125 ms. In case of null-RDC the radio receiver is never switched off, as the name suggests.

In BLE, the devices can assume different roles in the different layers of the protocol. In the link layer a device can be a 'Master' or a 'Slave'. In the Attribute Protocol (ATT) layer, a device can be a 'Client' and/or 'Server'. A server contains data and the client can request data from the server.

Physical layer

Link layer

Logical Link Control and Adaptation Protocol (L2CAP)

Generic Access Profile (GAP)

ATT

Generic Attribute Profile (GATT)

Security Manager

2.4 Overview of Contiki

??????????????? Which aspects of Contiki here?

2.5 Overview of 802.15.4

Mention that in this thesis the Contiki specific implementations of the 802.15.4 layers will be tested.

2.5.1 Physical layer

2.5.2 MAC layer

2.5.2.1 RDC layer

2.5.2.2 CSMA

3 | Methodology

3.1 Research Process

Figure 3.1 shows tasks performed in this thesis as boxes of the flow-chart and the steps of the research process on the left. The thesis report is also organized in this manner.

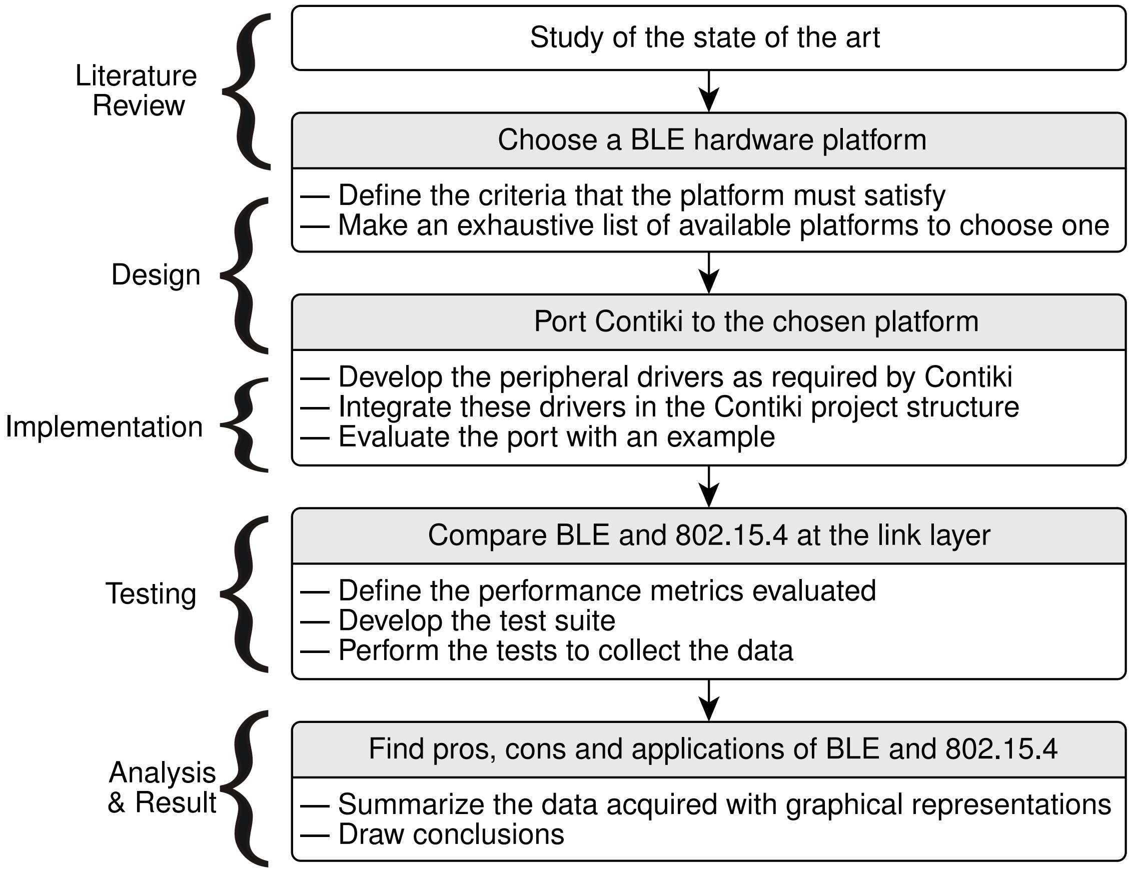

Literature Review
Study of the state of the art
Choose a BLE hardware platform
— Define the criteria that the platform must satisfy
— Make an exhaustive list of available platforms to choose one
Design
Port Contiki to the chosen platform
— Develop the peripheral drivers as required by Contiki
— Integrate these drivers in the Contiki project structure
— Evaluate the port with an example
Implementation
Testing
Compare BLE and 802.15.4 at the link layer
— Define the performance metrics evaluated
— Develop the test suite
— Perform the tests to collect the data
Analysis & Result
Find pros, cons and applications of BLE and 802.15.4
— Summarize the data acquired with graphical representations
— Draw conclusions

Figure 3.1: Thesis' tasks and steps in the research process

3.2 Research Method

In this experimental research large amount of data was generated to compare the two protocols mentioned, hence utilizing quantitative research method. The approach followed was deductive as the experiment started by defining the metric to measure, then designing the test to perform and finally deducting the conclusions based on the collected data from the tests.

The data collected was condensed with descriptive statistics and graphical representation was used to showcase this information. Univariate analysis of all the metrics of the two protocols and bivariate analysis of some metrics was done to compare the two protocols from these graphical representation.

4 | Literature Study

This chapter provides a brief overview of the related work being done in the research community. Performance evaluation of BLE, 802.15.4 and their comparison are presented in section 4.1, 4.2 and 4.3 respectively.

4.1 Evaluating the Performance of BLE

In a paper [1] providing an overview and evaluation of BLE, its various parameters such as energy consumption, latency, maximum piconet size and throughput have been evaluated. Theoretical calculations were done to predict the lifetime of a BLE slave device running of a 230 mAh coin cell battery for different values of connection interval and slave latency. The average current consumption of a TI CC2540 BLE platform was plotted for the entire range of connection interval keeping the slave latency as zero. The estimate of the lifetime was also done with respect to various Bit Error Rate (BER). A latency measurement also has been done in this paper, where it measures the time to send a notification packet and receive the 'no more data' acknowledgment packet from the receiver in the link-layer. This interaction happening in the same connection event was measured as 676.7 µs.

The same experimental setup [1] shows the maximum throughput between two CC2540 devices as 58.48 kbps considering a payload of 20 bytes. This is attributed to the fact that in each connection event with an interval of 7.5 ms four packets are not transmitted as in an ideal case. An analytical model of the throughput [10] has been developed for different BER and connection interval. Simulation results validate this model developed. This model show that the maximum throughput in case the BER is zero is 236.7 kbps, independent of the connection interval. An assumption made in this paper is that the master and slave device do not have any limit on the number of packets communicated in a connection interval.

The energy consumption for an advertiser and scanner is modeled for the activity of the scanner discovering the advertiser [11]. The current consumption for each phase of advertisement and scanning is measured. Using this information a model of the energy consumption is developed taking into account the advertising and scanning interval. The model developed is compared with experiments and validated.

The current consumption in the different phases of a connection event, namely waking up, pre-processing, the transmission-reception cycles and the post-processing is measured [12] to estimate the lifetime. The same article found the throughput achievable for a payload of 20 bytes as around 40 kbps. A recent journal article [13] develops a precise model of the energy consumption of BLE devices. A payload (20 bytes) throughput of 102 kbps is achieved in this paper, consistent with the manufacturer's claims [14]. The model developed is unique in the sense that it is the first one which encompasses all the modes of operation, all the relevant parameters and their possible values. The model is based upon the actual measured duration of various parameters and measured current in various phases of operation. This leads to a model which at most has 6% variation from actual measurement. The code base for the model is available so that it can be ported to the system being evaluated.

Based on the model developed and evaluated, a set of guidelines are provided for developers of BLE system to reduce energy consumption [13]. In the unconnected mode, the scanner is recommended to be continuously scanning in case the advertisers is expected to be found soon. In case of where a lot of time is spent scanning idly, the parameters of advertising interval and duty cycle of the scanner can be tweaked to minimize both the energy consumption and latency of discovery. In the connected mode, the recommendation is to completely fill the payload as possible and communicate as much data as possible within a connection event. In both modes, it was noted that although reducing the transmission power appropriate to the the distance transmitted helped in reducing energy consumption, it was not as significant as the other factors.

The interference caused by WiFi over BLE advertisement packets and vice versa has been evaluated[15]. This paper tests two cases, one where a BLE advertising channel overlaps with a WiFi channel and one where it does not. In the case where the there was no overlap of the two wireless protocols, there was negligible influence of each other. Increasing the advertising devices from 1 to 21 increased the CRC error and decreased the packets sniffed by the scanner per device from 40 to 22 because of the collision of the advertising packets. In case where there was overlap of the WiFi and BLE channels, there the WiFi throughput decreased to 50% as the number of advertisers increased from 1 to 21. The scanner experience a greater number of CRC errors, thereby decreasing the number of packets sniffed per device, decreasing to almost 50% of the previous case.

4.2 Evaluating the Performance of 802.15.4

As mentioned in 2.background, this thesis will not use the standard 802.15.4 standard MAC layer, only the 802.15.4 physical layer. The MAC layers used with 802.15.4 physical layer are ContikiMAC and Null-RDC. This is because of the amount of activity in the research community with these layers as compared to the standard 802.15.4 MAC layer. This section will briefly explain the research done to evaluate the performance of ContikiMAC and Null-RDC.

ContikiMAC is detailed in a technical report [16] which also evaluates its performance with few benchmarks.

[16] Adam Dunkels. “The ContikiMAC Radio Duty Cycling Protocol”. In: (Dec. 2011).

[17] Mathieu Michel and Bruno Quoitin. “Technical Report : ContikiMAC vs X-MAC performance analysis”. In: arXiv preprint arXiv:1404.3589 (Apr. 2014). arXiv: 1404.3589.

[18] M.-P. Uwase, M. Bezunartea, T.L. Nguyen, et al. “Experimental evaluation of message latency and power usage in WSNs”. In: 2014 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom). IEEE, May 2014, pp. 69–72.

4.3 Comparison of BLE and 802.15.4

BLE is relatively new protocol compared to 802.15.4, due to which there are few studies conducted comparing the two low power wireless protocols. The energy consumption of BLE and 802.15.4 was compared in terms of the amount of payload can be communicated per Joule of energy i.e. the energy utility measured with unit kByte/J [19]. It was found that for BLE the energy utility was independent of the throughput and depended on the number of packets communicated per connection event. The energy utility varied from around 325 to 525 kByte/J when the packets per connection event rose from one to four respectively. In case of 802.15.4, the energy utility did depend on the throughput. Until 1 kBps the energy utility increased with respect to the throughput, then plateaued at 300 kByte/J.

In the same paper [19] the energy utility of both BLE and 802.15.4 is measured when transmitting a payload over the link layer and when transmitting payload with IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) with the application payload (up to 150 Bytes) as a factor. Both increase in a step-wise manner because of the maximum payload capacity in both protocols with BLE having larger number of step because of the lesser payload capacity of 27 byte at the link layer. The overheads of the different 6LoWPAN frames can also be seen in both the protocols.

The effect of interference caused by User Datagram Protocol (UDP) packets sent over WiFi on the reception of packets is also considered for the two protocols [19]. It should be noted that the test build did not support AFH to avoid interference, so the authors tested only the non-connected mode for BLE. In that case, up to 1 m the interference resulted in only 5 to 20% of the packets being received successfully, depending on the overlap of the interfering WiFi channel over the advertising BLE channel. 1.5 m and above the WiFi interference affected the communication little. In case of 802.15.4, the WiFi interference closer than 0.5 m resulted in only 35% of the packets being communicated. At a distance of 1 m and above, the interference had negligible effect on the packets being communicated.

Another journal article [20] compares BLE, 802.15.4 and another wireless protocol SimpliciTI over various criteria of throughput (theoretical and experimental), minimum turnaround time, energy consumption of the transceivers and the memory resources required for the stack of these protocols. Minimum turnaround time here is the time required for requesting data and receiving it. SimpliciTI is an flexible open-source low-power proprietary radio protocol developed by TI for their wireless products, compatible with 802.15.4 transceivers. Similar to [10], a the calculation of the maximum throughput for BLE provided a value greater than 300 kbps, now at the link layer. Since SimpliciTI does not have rigid specification, its parameters can be tweaked for the maximum throughput to be calculated as 350 kbps. 802.15.4's maximum throughput was calculated to be between 150 and 200 kbps. The experimental evaluation of the throughput for SimpliciTI and 802.15.4 peaked at about 160 kbps and 145 kbps respectively accounting to the pre-processing operations and CCA inc case of 802.15.4. In case of BLE, the stack allowed different number of packets per connection event based on the link layer payload size. This resulted in a the throughput increasing irregularly with the payload, peaking at 122.6 kbps.

The minimum turnaround time measured for BLE was estimated to be below 1 ms since the reply was expected after the Inter Frame Space (IFS) in the same connection event. Experiments show that this is actually 7.6 ms. This is consistent with the minimum connection interval of 7.5 ms after which the reply is received in the next connection event. The minimum turnaround time was estimated as 1.92 to 10.08 ms and 0.7 to 5 ms for 802.15.4 and SimpliciTI respectively based on the mode of operation. The measured value was between 1.5 and 3 ms higher than the estimated value for both 802.15.4 and SimpliciTI. The energy consumption for the three protocols were measure per transmission and per byte transmitted. It was found that BLE transceiver consumed 2 to 7 times lower energy than the other two transceivers depending on the mode of operation.

One more article compare the energy consumption of BLE, ZigBee and ANT wireless protocols for a low duty cycle application sending few bytes of data periodically [21]. Standardized in 2003, ZigBee is a wireless, low cost, low power, mesh networking standard [22] typically used in home automation, industrial control, wireless sensor networks and the like. ZigBee uses 802.15.4 for its physical and MAC layer. ANT is another 2.4 GHz ISM based wireless sensor network protocol supporting many network architectures as point to point, broadcasting and mesh. Its typical applications are in fitness and sports Personal Area Network (PAN). In the article [21] the test case involves devices based on these three protocols initiating connection and transferring 8 bytes of data and then disconnecting periodically at an interval ranging from 5 to 120 seconds. From this test case, the authors found that the average current consumed by BLE was the lowest, followed by ZigBee and then ANT. The paper finds that on an average BLE takes the longest to connect at 1150 ms, followed by ANT at 930 ms and Zigbee being the fastest at 250 ms. The BLE test setup's parameters such as advertising interval, scanning interval and scanning duty cycle which effect the time for BLE connection to be established are not mentioned in this paper.

5 | Porting Contiki OS to a BLE platform

This chapter describes the initial task of choosing the hardware platform supporting BLE and porting Contiki OS to this platform. This task was a pre-requisite to the next task of testing detailed in chapter 6. Section 5.1 explains the process of choosing a hardware platform and describes the chosen one. Section 5.2 describes the different aspects of porting Contiki to a new platform, with details of the specific port done in this project.

5.1 BLE Hardware Platform

As mentioned in section **3.ref4Porting** Contiki has been ported to many hardware platforms. Since one of the main feature of Contiki is its communication stack capable of running in resource constrained systems, the primary aspects of a hardware platform are the MCU and the communication interface. When choosing a hardware platform there are many other aspects that need to be considered as well. These include the availability of source code for peripheral drivers and examples, development environment (compiler, linker, programmer and debugger), development boards, documentation of the entire system and online forum for discussion.

There are two types of platforms, namely platforms which consist of circuit board with a discrete radio transceiver with a MCU controlling it and platforms which consist of a SoC containing both the MCU and radio transceiver in the same IC. For including BLE support in Contiki, a hardware platform needed to be chosen and this process is explained in this section.

5.1.1 Requirements of the hardware platform

The *mandatory* requirements for the platform would be:

- Must have a well supported and documented processor with good specifications.
- Availability of well documented datasheet and user manual.
- Availability of an evaluation/development kit.
- Enough memory to accommodate Contiki and BLE stack's requirements.
- Presence of basic peripherals such as timers and serial port required for Contiki.

The non-mandatory, although *nice to have* requirements for the platform would be:

- A SoC based platform. A SoC solution is preferred because of characteristics such as lower power consumption, lower circuit board area and lower total cost.
- For an open source project such as Contiki, a free and preferably open source development toolchain must support the platform.
- Presence of flexible power modes with low active and sleep power consumption.
- Availability of a good set of peripherals.
- Availability of BLE stack from the vendor, preferably with source code.

5.1.2 Comparison and selection of the hardware platform

With these requirements, based on the exhaustive comparison in Appendix C of the available SoC (BLE+MCU) solutions available today, a platform based on nRF51822 from Nordic-Semiconductors would be a suitable option. As seen from the table in Appendix C, this platform would satisfy all the requirements mentioned above except for that the BLE stack would be available as a binary file, without the source code.

As shown in the table in Appendix C, recently many new promising BLE based SoCs have be released such as Quintic 9020, Dialog Semiconductor DA14580, Lapis MLA7105 and Broadcom BCM20732. From the limited technical information available about them, their technical specifications would be suitable for a project like this. But because of the limited documentation about them, scarce availability and nascent support they are not suitable.

5.1.3 Overview of nrf51822 SoC and its platform

nrf51822 is a SoC made by Nordic Semiconductor for developing BLE and 2.4 GHz based wireless systems [23]. Most of the specification of this SoC can be found in the table in Appendix C. The ARM Cortex M0 present is a 32 bit, 3 stage pipeline processor with Von Neumann architecture. It is designed for low silicon die size, low cost and power. It has an integrated Nested Vectored Interrupt Controller (NVIC) responsible for handling processor exceptions and peripheral interrupts.

The development boards in the form of a USB dongle used for this thesis are called PCA10000. As seen in the figure 5.1, the top side of PCA10000 contains nrf51822 at the centre, powered from the USB port through a voltage regulator. This SoC is connected to a tri-colour RGB led, a 16 MHz crystal, a 32.768 kHz crystal and a PCB antenna with its matching network. On the other side of the PCB is the SEGGER JLink Lite Cortex M unit. This can program and debug using the Serial Wire Debug (SWD) port of nrf51822. Another useful feature of this board is that the SEGGER JLink unit provides a serial port over USB with hardware flow control (HWFC) to the computer that this dongle is connected to. This serial port is connected to the Universal Asynchronous Receiver/Transmitter (UART) port of nrf51822 [24].

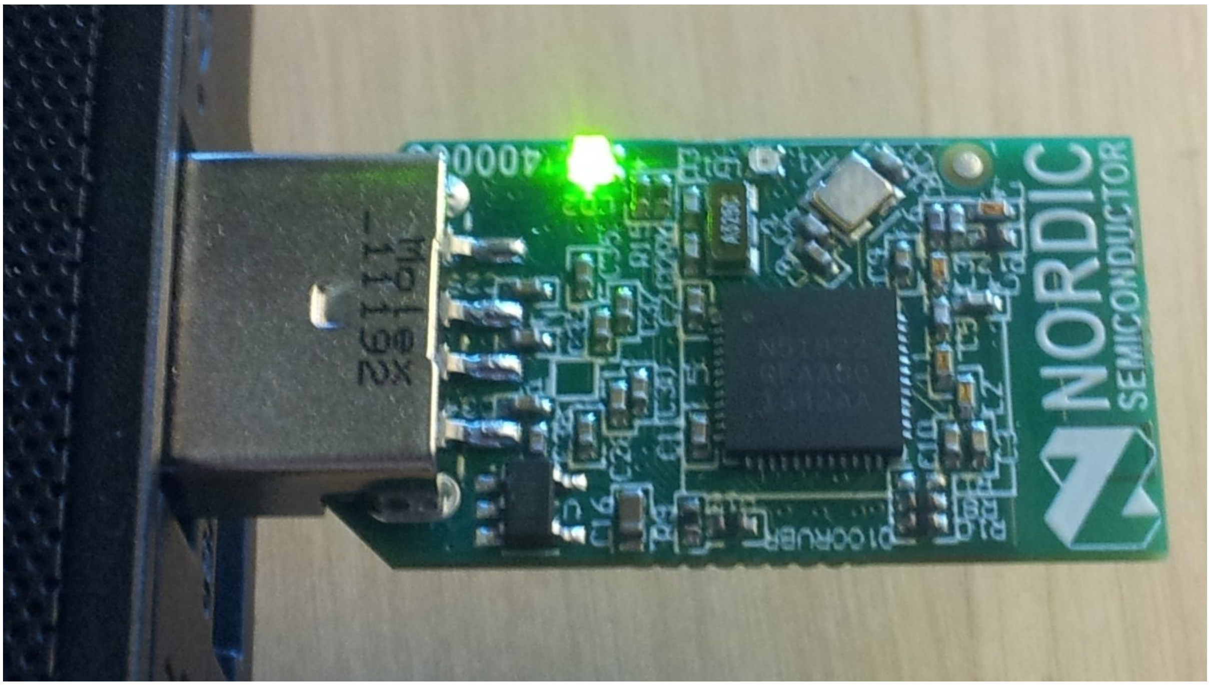


Figure 5.1: PCA10000 development board

Nordic Semiconductor provides BLE stack as a precompiled and linked binary file called *SoftDevice*. There are various of these SoftDevice binaries with different aspects of the BLE stack. The SoftDevice is stored in a protected area in the flash memory and has access to a protected section of RAM memory, preventing unauthorized access by the application code. The SoftDevice can be accessed through a specified set of API calls. These APIs are accessed by making a supervisor call to the processor causing a exception handler to run the SoftDevice. All calls are non-blocking, which means that the call will not stall the application making the call. And there are synchronous and non-synchronous calls, where the synchronous calls immediately return the result while the asynchronous calls start an operation that will send the result as an event to the application. The SoftDevice is used for accessing the BLE stack as it was out of the scope of this thesis project to implement a BLE stack from scratch.

A Software Development Kit (SDK) is provided for nrf51822 which contains the peripheral drivers, examples, the interface header files for the SoftDevices and their documentation. Two tools provided by Nordic Semiconductor has been extensively used in this thesis. *nRF Sniffer* is a tool used with the PCA10000 board and *Wireshark* application to capture, view and save the information of BLE packets being sent between two devices. This greatly helps in learning about BLE packets and debugging problems. This tool is capable of providing detailed information about almost all the segments of the captured packets. It was found in during the thesis project that the nRF-Sniffer is not entirely capable of capture each and every packet being communicated, which does limit its use as tool to get feedback as one develops low level BLE drivers. Another invaluable tool provided by Nordic Semiconductor is *Master Control Panel*. It is an Android application which acts a generic BLE application capable of discovering BLE devices, connect and communicate with them while providing an overview of their Attribute database.

5.2 Porting PCA10000 platform of nrf51822 to Contiki[*]

5.2.1 Development Setup

Contiki was initially developed in a Linux based operating system, although Windows is also supported now. The porting of PCA10000 platform to Contiki was done in Ubuntu 13.10 and 14.04. The compiler suite used is GNU Tools for ARM Embedded Processors Version 4.8.3. The program used for programming the binary files compiled was SEGGER J-Link Commander V4.90 using the Segger JLink programmer on PCA10000. The front end used to write the code is the Eclipse IDE for C/C++ Developers Version Kepler Service Release 1 and Sublime Text. The porting was done to Contiki version 3.x, which is under active development.

5.2.2 Peripherals required for Contiki

This section details the implementation of the peripheral drivers done in this thesis so that the peripherals can be controlled with APIs specified by Contiki.

Contiki clock Contiki clock, which is the source for all the timers, except the rtimer, is provided by the RTC1 peripheral of nrf51822. RTC1 is chosen because when a SoftDevice is used, RTC0 will not be accessible for the user application. The Real Time Clock (RTC) peripheral uses the Low Frequency Clock (LF-CLK) of nrf51822, which can be generated by a crystal or RC oscillator to produce 32.768 kHz. For Contiki clock, in this thesis there are two driver implementations made, namely Tickless and Ticks as described below.

[*] Available at `https://github.com/EarthLord/contiki` with complete Doxygen documentation

Ticks For this implementation the RTC1 peripheral is configured such that an interrupt is called for its every increment or *tick*, hence the name. This happens at 64 Hz, which is the default value of `CLOCK_SECOND`. In the interrupt routine, the current clock tick is incremented, an etimer poll is requested if an etimer has expired and every `CLOCK_SECOND`$^{\text{th}}$ interrupt the second count is incremented. The variable storing the clock 'ticks' and 'seconds' is returned upon their request from any other process.

Tickless In this implementation, the processor will not be woken up at every tick of the RTC peripheral, hence the name. To enable this, a small addition is required to the etimer and clock module present at `CONTIKI/core/sys`. Every time the next etimer expiration is computed, the clock module is informed of it. With this additional information the RTC can configure a compare interrupt to poll the etimer upon its expiry. The RTC1 peripheral is also configured to provide an interrupt only on its overflow so that it can be accounted for when calculating the seconds elapsed. For the 24-bit timer of RTC1, with `CLOCK_SECOND` as 64, the overflow interrupt would happen only once every $(2^{24}/64)$ second, that is almost every three days. Compared to interrupt happening few times a second in the 'Ticks' case this is a long time without the need for processor's involvement. With a Tickless implementation, both keeping track of the value of 'clock ticks' and polling the etimer upon its expiry is handled by the RTC peripheral rather than the processor, which would significantly reduce the power consumption without sacrificing any performance. Since changes need to be made to core Contiki library as mentioned above, this implementation is not supported natively by Contiki.

Rtimer An rtimer is implemented in Contiki's port to nrf51822 by using the *TIMER1* peripheral, which runs on the High Frequency Clock (HF-CLK) of 16 MHz. TIMER1 was chosen as TIMER0 is required by SoftDevice, in case it is used. Since rtimer is required with higher granularity than Contiki Clock, rtimer increments at rate of 62.5 kHz in its current implementation, which is every 16 µs. This is achieved by TIMER1 being configured as a 8-bit timer and providing an interrupt on overflow where the rtimer count is incremented. In the interrupt routine there is also a check to see if the scheduled rtimer task needs to be run by comparing the current rtimer count. It should be noted that Rtimer being run by TIMER1 peripheral requires the HF-CLK to be active, thereby consuming additional power.

LEDs The PCA10000 board has a RGB LED unit on it. The port allows it to be controlled by the Contiki API. This is done through reading and writing to the General Purpose Input-Output (GPIO) port when the LED's status is read and LED's state is changed respectively. This implementation is present in 'platform' folder since they are a property of PCA10000.

Button(s) The PCA10000 board does not have any buttons. In case of porting a platform with physical buttons, its implementation would be also in the platform folder.

Serial Port The port of Contiki to PCA10000 platform offers abstraction of the UART peripheral of nrf51822, which will communicate with the serial port of the computer to which PCA10000 is connected to. Writing to the serial port is achieved by using the `printf()` function present in the `stdio.h` library by redirecting `printf` call's character stream to the UART peripheral. Instead of the default NewLib library, the Newlib-nano library is used so that the amount of memory required is reduced.

For receiving the data from the the *serial-line* module of Contiki is used. This module broadcasts an event when a series of characters are received ending with a 'newline' (\n) character. To use this module, it is initialized on boot and all the characters received by the UART port is sent to this module in the UART receive interrupt routine.

Radio Since the radio peripheral is completely controlled by the SoftDevice for implementing the BLE stack, it is untouched in this port. To use the radio peripheral, the APIs provided by the SoftDevice is used. To see the work done in this thesis regarding the radio peripheral, refer section 8.1.

6 | Test Cases

This chapter details the reasoning and design of the test cases used in this thesis. To provide a common platform for comparing BLE and 802.15.4, Contiki was ported to a platform supporting BLE as described in chapter 5. This chapter starts off with section 6.1 explaining all the metrics measured in the experiments performed. Section 6.2 follows by explaining the test platforms and tools used. Section 6.3 and 6.4 explains the two test suite designed to measure the metrics defined.

6.1 Performance Metrics Definition

The aim of this thesis is to compare the key characteristics of the link layer of BLE and 802.15.4. Note that as explained in 4.background, 802.15.4 here means the physical layer of 802.15.4 standard. The 802.15.4 standard MAC layer is not used, rather Null-RDC and ContikiMAC was used, both with CSMA layer. The scenario looked into in this thesis is of the case where there is communication between two device, one constrained in terms of power and the other not. This is typical of *appcessories*, where one device is a powerful mobile device and the other is a battery operated single purpose device. In this section the four key performance metrics measured for both the wireless protocols are defined in the context of this thesis.

Data rate Data rate at the link layer is defined as the maximum amount of data transmitted per second for a protocol with packets containing the maximum allowed payload for the link layer. The data rate is expressed as both kilobits per second and packets per second.

Latency In this test the aim is to measure the effect of the link layer architecture and configuration on the latency observed by an layer on top when requesting data from another node. Latency is defined as the time from which the layer(s) above the link layer provides it the packet which asks for data from another node to the time the data from the other node is received. This is better understood by a generic illustration in figure 6.1.

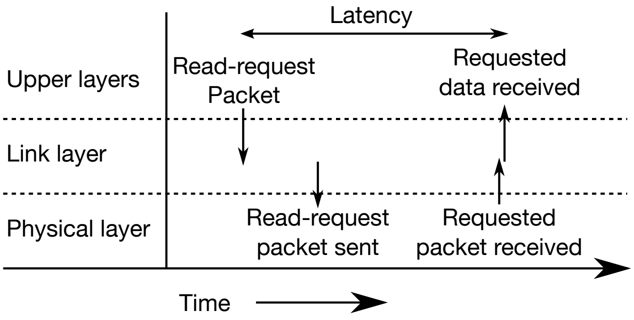


Figure 6.1: Illustration of latency measurement

Reliability Two criteria was used for measuring reliability when two nodes are communicating. Packet Reception Ratio (PRR) is defined as the ratio of the number of packets successfully received to the number of packets sent, both by the link layer. Packet Delivery Ratio (PDR) is defined as the ratio of the number of packets delivered to the number of packets sent, by the layer above the link layer in both the sender and receiver. While PRR measures the reliability of transfer of a single transfer of a packet, PDR measures the reliability of the link layer to communicate a packet as seen by the layer above it.

Energy Consumption The energy consumption in this project is indirectly measured as the Radio Duty Cycle (RDC), which is the ratio of the time for which the radio is switched on to the total time of the experiment. Measuring the energy consumption as RDC for different scenarios will be more useful to developers as this information will change little across platforms for the same protocol, whereas the actual energy consumed is reducing with every new platform introduced in the market, as seen in this presentation[25]. The processor time utilized by the protocol stack is not considered as this information could not be extracted from the SoftDevice APIs and also because of the different processor types in the two platforms (32-bit vs 16 bit).

6.2 Test setup

6.2.1 BLE platform

The platform used for BLE is PCA10000, which is described in section 5.1.3. Since BLE is has an asymmetric architecture, a master device is required to communicate with a slave device. In all the tests the slave device is a PCA10000 board running on Contiki and peripheral BLE stack provided by the SoftDevice *S110* version 6.0.0. Depending on the test, the master device varied between either a PCA10000 board or an Android device. When a PCA10000 was used as master device, it ran on bare-metal code with central BLE stack provided by the SoftDevice *S120* version 1.0.0. This choice of use of Contiki for slave and bare-metal code for master is consistent with the assumption that the master is unconstrained and slave is not. The transmission power was 0 dBm for all the tests using PCA10000. Nexus 7 and Nexus 4, both at Android version 4.4.4 were the Android devices used for performing certain tests.

Communication was done at the Generic Attribute (GATT) layer because the binary from Nordic Semiconductor used in this project does not provide APIs to access the lower layers in both peripheral and central devices, hence evaluation at the link layer was done indirectly based on the inferences from the collected data. The default transmission power of 0 dBm was used. The radio state change notification event in the SoftDevice was enabled. This allowed logging of a timer to measure the energy consumption in terms of RDC.

6.2.2 802.15.4 platform

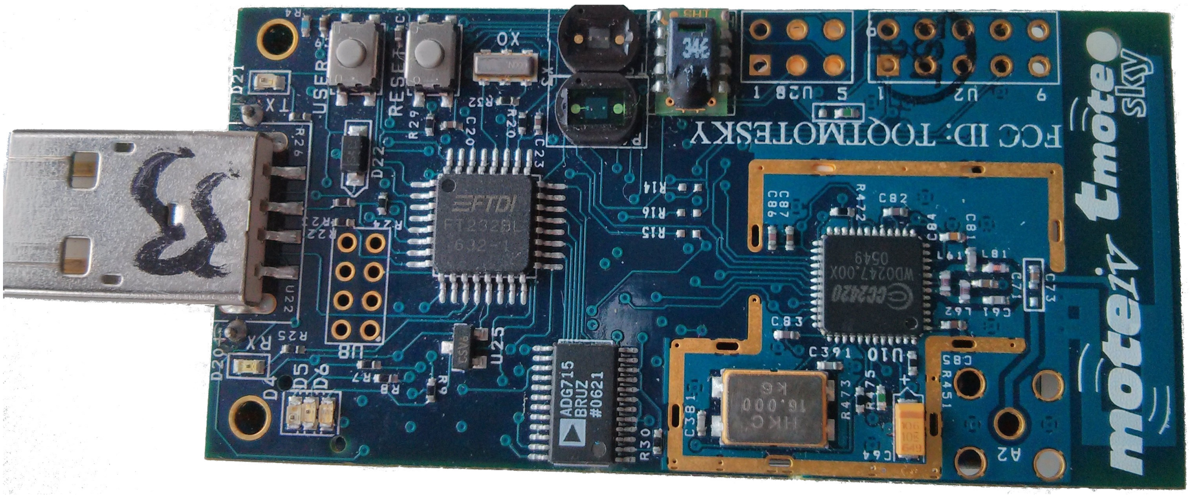


Figure 6.2: TmoteSky platform

Tmote-Sky platform, which is widely used and supports all the features of Contiki was used as the platform to test 802.15.4 performance. Shown in figure 6.2, Tmote-Sky is based on

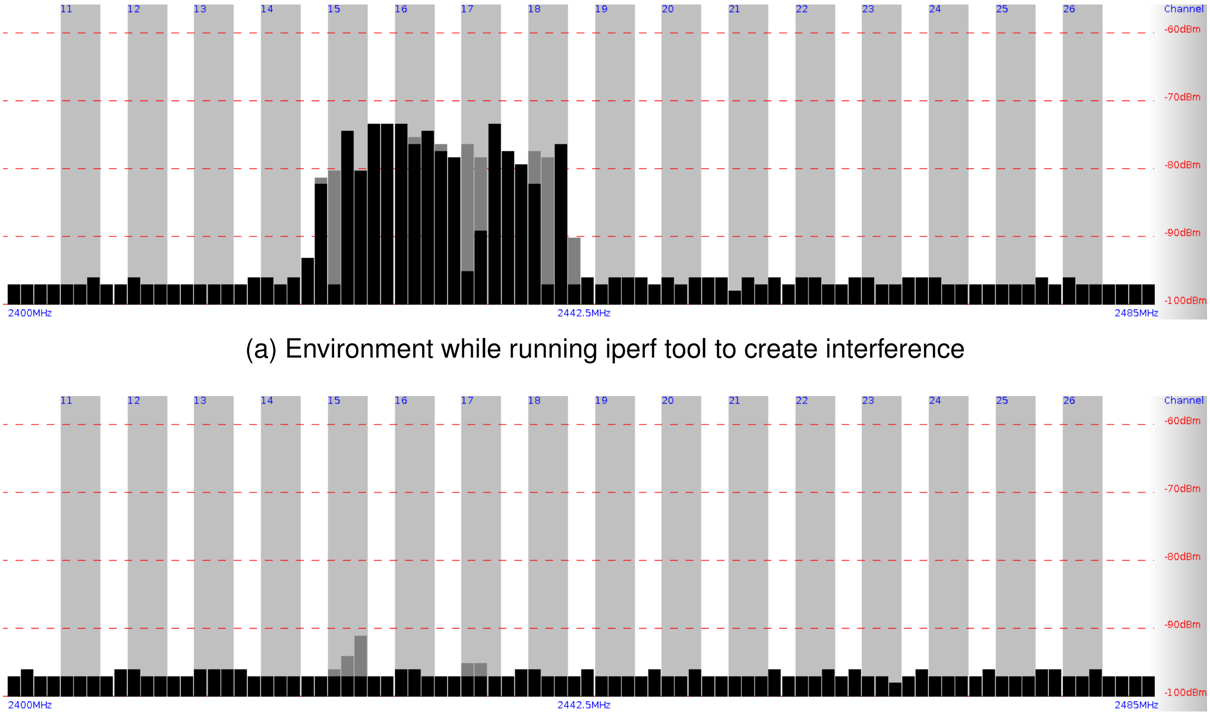
MSP430-F1611 MCU and CC2420 2.4 GHz transceiver. It has a USB to serial converter on board to allow the MSP430 MCU to communicate with the serial port of a computer. It has additional sensors, buttons and memory which are not used in this project. The transmission power was unchanged from the default value of 0 dBm. The ‘energest’ module of Contiki was used to log the radio on and total time to calculate the energy consumption in terms of RDC.

6.2.3 WiFi Interference

In some of the tests described in the following sections, WiFi interference needed to be created to test the reliability of the wireless protocols with external interference. A open-source tool called `iperf` was used for creating this interference by sending UDP packets to the IP address of a router. The computer and router used are ThinkPad T420 and Netgear N300 router respectively, both supporting WiFi 802.11b/g/n standard. The aim of the interference pattern is to simulate streaming of large amount of data over WiFi. This was achieved by this command:

```
sudo iperf -c 192.168.1.1 -u -P 1 -i 1 -p 5001 -f M -b 54.0M -t 64 -T 1
```

Where ‘-c 192.168.1.1’ specifies that the packets are sent to the IP address specified (router’s IP), ‘-u’ to use UDP packets, ‘-P 1’ specifies one client thread to run, ‘-i 1’ specifies the bandwidth reporting interval as one second, ‘-p 5001’ specifies the server port to connect to as 5001, ‘-f M’ specifies the reporting format as MegaByte per second, ‘-b 54.0M’ specifies the data rate to be used as 54 Megabit per second, ‘-t 64’ specifies the transmission to happen for 64 second and ‘-T 1’ specifies that only one hop is allowed to reach destination. Although the data rate was specified as 54 Mbps i.e. 6.75 MBps, the actual data rate of transmission reported was around 3 MBps.



(a) Environment while running iperf tool to create interference

(b) Environment without creating interference

Figure 6.3: 2.4 GHz environment captured by rssi-scanner

To verify that the interference was created and to know which frequency range was occupied, the ‘rssi-scanner’ project was used*. This project is based on Tmote-Sky platform running on Contiki. The screen-shot of rssi-scanner shows a snapshot of the environment where the

*Available at: http://sourceforge.net/p/contikiprojects/code/HEAD/tree/sics.se/rssi-scanner/

test were conducted for the cases of with and without interference in figure 6.3a and 6.3b respectively, where y-axis shows the signal strength measured and x-axis is the frequency of the measurement.

6.3 High-Throughput (HT) Test Design

HT test aims to measure the maximum possible data rate of BLE and 802.15.4 at the LL with and without WiFi interference. For both protocols, the link layer contains the maximum payload allowed. Each run of the test will last for one minute to collect enough data to calculate the mean data rate. In each of the tests, the amount of time the radio is switched on (RDC), either transmitting or receiving, is logged to measure the energy consumption. The metrics measured in this test are data rate, reliability and energy consumption. The two nodes were kept 1 m apart and in case of environment with interference, the router to which the UDP packets were sent over WiFi was about 2.5 m from these nodes. In the cases where an Android device is not involved, they were conducted for 10 times and the median value of the 10 was used. The test with an Android device could not be automated, so the test was conducted once.

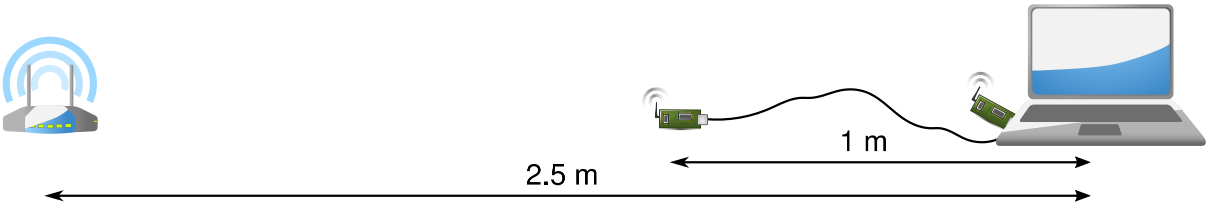


Figure 6.4: Test setup for HT test cases

BLE As shown in figure 6.5, the data rate is measured by sending data from the server to the client using *Notifications*, which are packets which are not acknowledged in the ATT layer. Since the link layer is not accessible with the SoftDevice used, a 20 byte payload was sent at the GATT layer as notification so that 27 byte data for the link layer's playload is achieved. The data rate is measured at the receiver i.e the Master device.

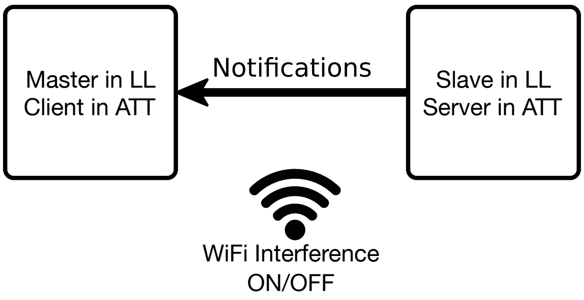


Figure 6.5: Setup to measure data rate with BLE

The number of packets sent and received at the ATT layer in the two nodes are logged to calculate the PDR. Also by logging the number of times the radio was switched on, the number of packets actually transmitted can be inferred, by which the PRR can be calculated.

BLE specification tells that a master device must change the channel map of a connection according the 2.4 GHz environment so as to not face any interference. This is the *adaptive* part of AFH, which requires the master device to have information about the activity in the frequency spectrum used by BLE . S120 SoftDevice used for the central BLE stack by default uses the complete channel map and has an option for configuring it manually, but the channel map does not change automatically based on external interference. So in the tests without WiFi interference, the channel map consisted of all the channels. When WiFi interference was introduced, one test had the complete channel map and another had a *WiFi free* map of channels (0 to 6) and (20 to 36), which is 2404 to 2416 MHz and 2446 to 2478 MHz. This map was based on the interference noticed in figure 6.3a.

802.15.4 In case of 802.15.4, Null-RDC layer was modified to send packets with maximum payload size as soon as possible. One node is configured as transmitter and the other as receiver, with the data rate measured at the receiver. A payload of 110 byte was necessary for the link layer to achieve the maximum 127 byte packet size.

Three tests were designed to measure the data rate achievable with 802.15.4. In the first test without WiFi interference, channel 26 (2480 MHz) of 802.15.4 was used. In the second test with WiFi interference, channel 15 (2425 MHz) of 802.15.4 was used. In the third test, the Clear Channel Assessment (CCA) was disabled, so that the transmitting node ignores the external environment.

The various test cases conducted in the HT test suite are

- sky_sky_N_!CCA
- sky_sky_Y_26
- sky_sky_Y_15
- nrf_nrf_N_all
- nrf_nrf_Y_all
- nrf_nrf_Y_!Wifi
- nrf_N4_N
- nrf_N7_N

The legend of the naming of the test cases is

`{source}_{destination}_{presence of WiFi interference(Y/N)}_{other conditions}`

Where, sky = Tmote-Sky; nrf = nrf51822; N4 = Nexus4; N7 = Nexus7; !CCA = CCA disabled; 26,15 = channel used; all = All 40 BLE channels; !WiFi = WiFi free channels in BLE

6.4 Request-Response (RR) Test Design

The RR test aims to determine the latency for reading data from another node as explained in the latency paragraph in section 6.1. In all the tests the nodes were kept at a distance less than 10 cm from one another, since the tests we aimed to reflect an error-free environment. In this tests also the time the radio is switched on is logged in both the nodes to calculate RDC to interpret the energy consumption. The idea is to find the relation between energy consumption and latency based on different link layer configurations. Thus, the metrics evaluated in this test are latency and energy consumption.

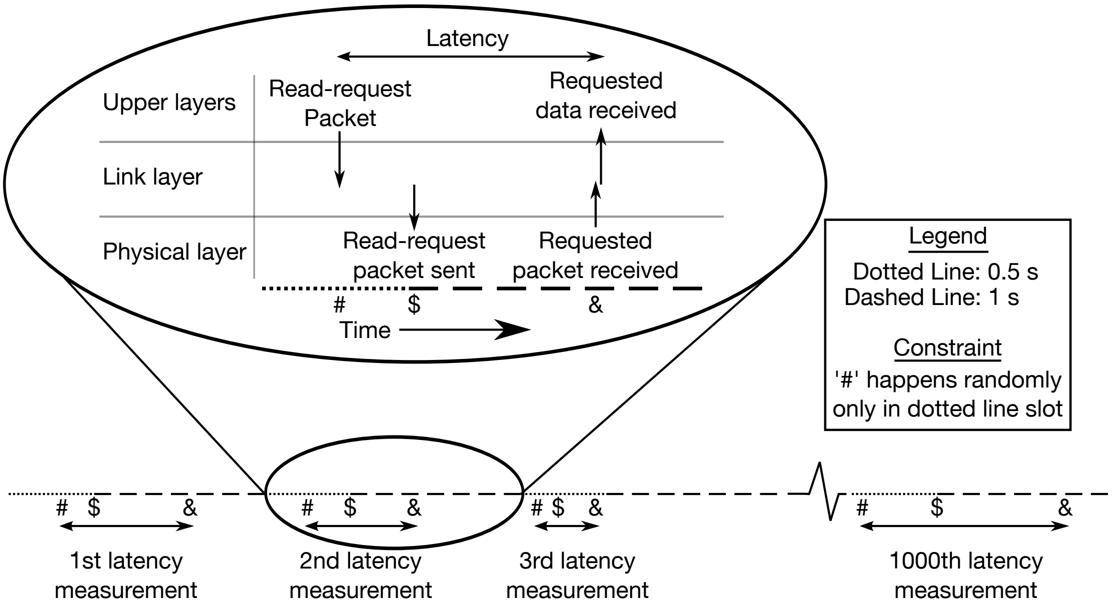
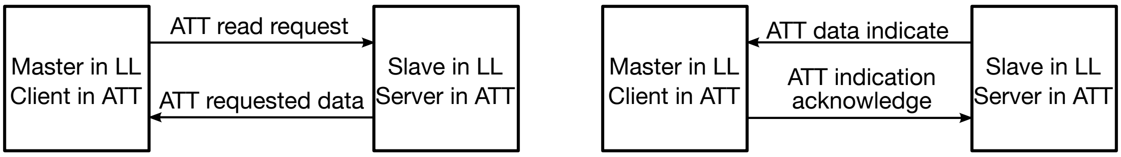


Figure 6.6: Illustration of the RR Test operation

The design of the test for both the protocols is shown in a generic diagram in figure 6.6. A latency measurement is done in every 1.5 second period, repeated thousand times. The link layer is sent a ‘read-request’ packet from its upper layer after a random time in the first 0.5 second slot of this 1.5 second period, represented by ‘#’ in figure 6.6. This randomization of the issuing of the read request (‘#’ event) ensures that is there is any periodic communication between the nodes, the thousand measurements of latency will provide the range of values that can expected with a particular scenario along with the power consumption. As explained in the latency paragraph in section 6.1, the latency measurement is the difference of the time between the ‘&’ and ‘#’ events shown in figure 6.6.

BLE In case of BLE , the use case used for the latency test is a central device which is master at link layer and client at ATT layer gets data from a peripheral device which is slave at link layer and server at ATT. This can be achieved in two modes as shown in figure 6.7a and 6.7b. In both cases it is the link ATT client which gets the data from the server. In the first case, the client asks for the data from the server, which responds with data. The latency measurement for this interaction happens at the client. In the second case, the server sends the data in an ‘indication’ packet, which is acknowledged by the client. In this case since it is the server that is getting the response from the master, the latency measurement happens in the server.



(a) Read-response test case

(b) Indicate-acknowledge test case

Figure 6.7: Test modes for measuring latency using BLE

In all the tests, the complete payload of 20 bytes for the ATT layer was used, which corresponds to 27 byte in the link layer. Two connection intervals of 7.5 ms and 125 ms were used for the tests. For each ’connection interval’, tests were conducted with two different ‘slave latency’ parameter. The table 6.1 shows these values and the configuration for all the test cases in RR test using BLE. The intent here is to find the effect of the link layer configuration on the latency. By using different ‘slave latency’ values, the energy consumption of the asymmetric connecting devices can be evaluated. The value of the non-zero slave latency value is decided such that the worst case theoretical latency in a error free environment is less than one second. This is the reason for choosing the ‘supervision timeout’ of 1.5 s.

Test Case	Connection Interval	Slave Latency	Test Mode
7.5_0_r	7.5 ms	0	Read
7.5_0_i	7.5 ms	0	Indicate
7.5_65_r	7.5 ms	65	Read
7.5_65_i	7.5 ms	65	Indicate
125_0_r	125 ms	0	Read
125_0_i	125 ms	0	Indicate
125_3_r	125 ms	3	Read
125_3_i	125 ms	3	Indicate

Table 6.1: List of RR test cases using BLE

802.15.4 In the case of 802.15.4, one node is configured to send unicast messages and another is configured to receive these and send a response. The latency was measured using both ContikiMAC and Null-RDC. ContikiMAC uses the default 125 ms wake up interval for the receiver. For payload size is same as in the test cases with BLE for comparing the latency for the same payload. The BLE test cases are designed such that the 7.5 ms connection interval tests can compare with Null-RDC test of 802.15.4 while the 125 ms connection interval tests can compare with the ContikiMAC test of 802.15.4.

The sending of the unicast packet and getting its response followed the same thousand iterations of 1.5 s intervals containing a 0.5 s period where the packet was randomly requested to be sent. The latency measurement happens as illustrated by figure 6.1 used to define the term latency in section 6.1.

7 | Results and Analysis

This chapter provides a graphical representation of the data acquired from conducting the tests designed in the last chapter. The raw data used to generate this graphs can be accessed in Appendix D. After the graphs their analysis is done to explain the various characteristics of the two wireless protocols according to the experimental results.

7.1 High-Throughput Test

The data acquired from the different test cases of the HT test suite is represented graphically below. The legend used for naming the test cases is same as described in section 6.3.

7.1.1 Graphical Representation of the Data Acquired

Note that in figure 7.1 there are two Y-axis representing the data rate in packets per second and kilobit per second. The link layer payload of 27 bytes and 110 byte was used for BLE and 802.15.4 respectively to calculate the data rate in kbps. The data used for the plot is the median of the 10 tests conducted, each of one minute. The standard deviation of the data rate from these 10 tests are also plotted.

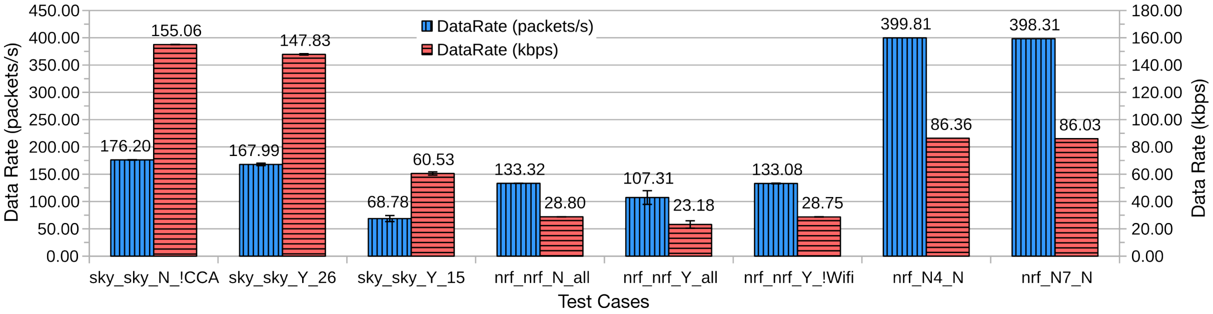


Figure 7.1: Data Rate

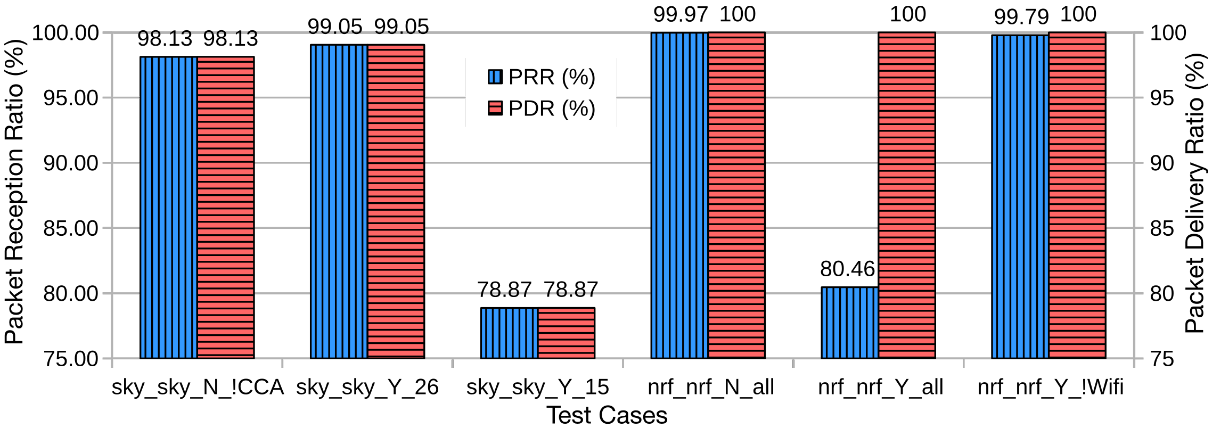


Figure 7.2: Reliability

The graph in figure 7.2 has Y-axes from 75 to 100 % for clearer representation of the PDR and PRR. As mentioned in section 6.3 the reliability was calculate in BLE by logging the number of times the radio was switched on as this information was accessible from the SoftDevice. This allowed the calculation of PRR with one packet per connection event. Since the android devices can communicate multiple packets per connection event, the PRR could not be calculated. Hence, those test cases are not plotted.

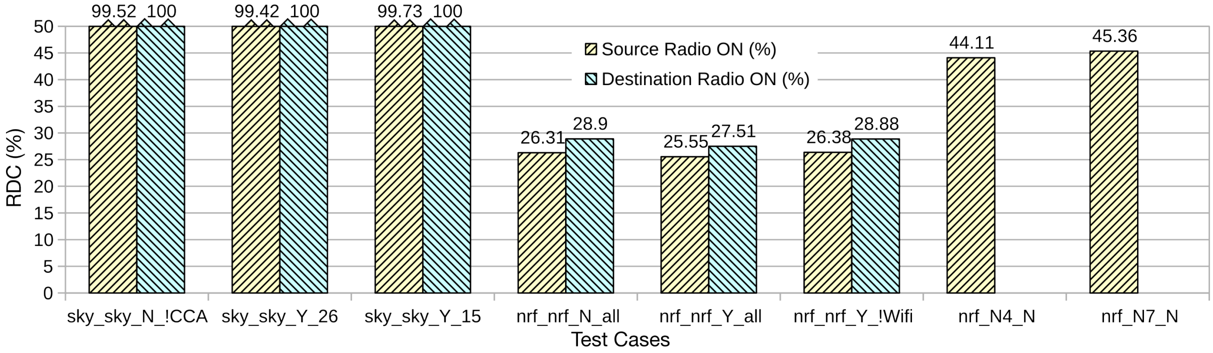


Figure 7.3: Energy Consumption

For a clearer representation of the graph, the y-axis in figure 7.3 is limited to 50%. The break in the graph shows when Tmote-Sky uses ~100% of radio duty cycle. In all the tests with BLE the source node was the slave device sending the notifications, while the destination node was the master device receiving them.

There was no easy way of accessing the low level information of BLE RDC on an Android device, so the energy consumption of these devices is not plotted. Moreover the energy consumption of such multi-purpose master device is less important than a single purpose slave device running on a meager battery.

7.1.2 Analysis

Data Rate The absolute data rate is higher in 802.15.4 than BLE although the number of packets transmitter per second is higher in BLE as seen in figure 7.1. This is because of the higher packet size of 802.15.4 and the bit rate of transmission of BLE is four time the bit rate of 802.15.4 at 1 Mbps.

For 802.15.4 the peak data rate of 155.06 kbps is achieved when CCA is not used. The more practical approach of using CCA decreased in the data rate slightly to 147.83 kbps. When WiFi interference was introduced, most of the transmission slots were not utilized because of backing off by CCA. Appendix D shows that only 37% of the transmission slots were used, dropping the data rate considerably. With this 60.53 kbps data rate was achieved, which is higher than the data rate achieved by BLE communication between nodes of PCA10000 platform in any test case. Since 802.15.4's Null-RDC tries to send a packet as soon as possible, we can approximately calculate that the inference of WiFi traffic was not present 40.1% of the time by taking the ratio of the data rate with and without WiFi interference (60.53/147.85). **5.can do this?**

The SoftDevice used to implement the central stack allows only one packet to be communicated per connection interval. With this the theoretical data rate that can be achieved can be found out by

$$\text{Data Rate (packet/second)} = \frac{1}{\text{connection interval}} = \frac{1}{7.5ms} = 133.33\,packet/second$$

$$\text{Data Rate (kbps)} = \frac{\text{(bits per byte)}\times\text{(link layer payload in bytes)}}{\text{connection interval}} = \frac{8\times 27}{7.5ms} = 28.8\,kbps$$

The experimental results seen in figure 7.1 shows this is achieved accurately. When WiFi is introduced, the data rate drops from 28.8 kbps to 23.18 kbps, when all the channels are used. This is 81% of the WiFi free data rate. Figure 6.3a shows that 10 out of the 37 data channels are interfered by WiFi traffic. This means that the data rate should reduce by (37-10)/37, which is about 73% of the WiFi free data rate. The greater data rate achieved experimentally can be accounted to the fact that the WiFi traffic wasn't interfering 100% of the time as inferred earlier with 802.15.4 data rate.

When the WiFi free channels were used the data rate recorded was close to the theoretical data rate predicted. There was negligible influence of the WiFi traffic at all for BLE communication in this case. This shows that when frequency hopping channel map is chosen to avoid the interference, there is no effect on the data rate.

Communication of the PCA10000 with a Nexus4/7 master device achieved a greater data rate of about 86 kbps or ~400 packet/second. This is because the BLE stack in these Android devices support communication of multiple packets in a connection event, increasing the data rate achievable. With a connection interval of 7.5 ms, at an average there were around $(400 \times 7.5/1000)$ i.e. 3 packets of data received by the Android device per connection event.

Reliability In case of 802.15.4, the PDR and PRR is same since it does not have any communication failure detection and retransmission mechanism. This is left to the upper layer to handle. The link layer just tries initiate communication when there is no external interference present with its CCA. Figure 7.2 shows that the PRR achieved when CCA is switched off is 98.13%, the value being high since there was little external interference. While it can be seen that without CCA the data-rate is higher, the reliability decreases. By using CCA in a WiFi free channel, the PRR is 99.05%, which means that the CCA is effective in case of less interference in the operated channel. When there was heavy WiFi traffic in the same channel of communication, the CCA did work since only 36.82% of the transmission slots were used. Of these transmissions the PRR was 78.87%. This could happen when the interference starts after the carrier assessment, causing the packet to get corrupted. Here it can be seen that CCA of 802.15.4 has reduced effectiveness in case of heavy interference.

In case of BLE, figure 7.2 shows that the PDR is always 100% since BLE employs a simple acknowledgment scheme to detect if the communication has not been successful so that the packet can be retransmitted. This ensures that the upper layers can safely assume that the link layer is completely reliable. Without WiFi interference PRR was about 99.97%, which means that there were few retransmissions. With all the channels employed and with WiFi interference, the PRR reduced to 80.46%. This number is not low because of the frequency hopping used by BLE, which ensures that interference in one frequency range only effects the communication happening over the channels in that frequency range. The other data channels can still keep the communication flow intact. This shows that the frequency hopping when used over the complete channel map helps in sustaining the communication when there is external interference. When only the WiFi free channels were employed, there was no influence of the WiFi interference on PRR, which was 99.79%. This shows that when a master device has the capability of detecting external interference, it can adapt the frequency hopping channels so that there is negligible effect on the communication. Note that AFH can only help negate narrow band interference in the 2.4 GHz ISM band.

Energy Consumption As seen in figure 7.3, the radio of Tmote-Sky nodes is used nearly 100% the time in all the test cases of the HT test suite. This shows that in 802.15.4 the data rate can be maximized, by choosing an appropriate RDC layer like Null-RDC, although at the expense of energy consumption.

When the test cases where the communication happened between PCA10000 node, the source node had RDC between 25.5% and 26.5%. The destination nodes had RDC between

27.5% and 29%. From this we can see that when streaming data every connection interval, the energy consumption of the master and slave is similar, with the master node being a few percent higher. The energy consumption isn't affected by the presence of external interference in this case of the stack supporting one packet per connection event. This is because the radio must switch on anyway in each connection event to detect if the communication is successful. In case multiple packets per connection interval is supported, the energy consumption for streaming of data with external interference could be lower as the connection event ends prematurely. This could not be tested with the existing setup.

In case of communication with the Android devices, the source nodes had RDC of 44% to 45%. This is higher because multiple packets were communicated per connection interval, which also increased the data rate. This shows that when a master device can support communication of greater number packets per connection event, both the data rate and energy consumption increases.

7.2 Request-Response Test

The data acquired from the different test cases of the RR test suite is represented graphically below. The legend used for naming the test cases is same as described in section 6.4.

7.2.1 Graphical Representation of Data Acquired

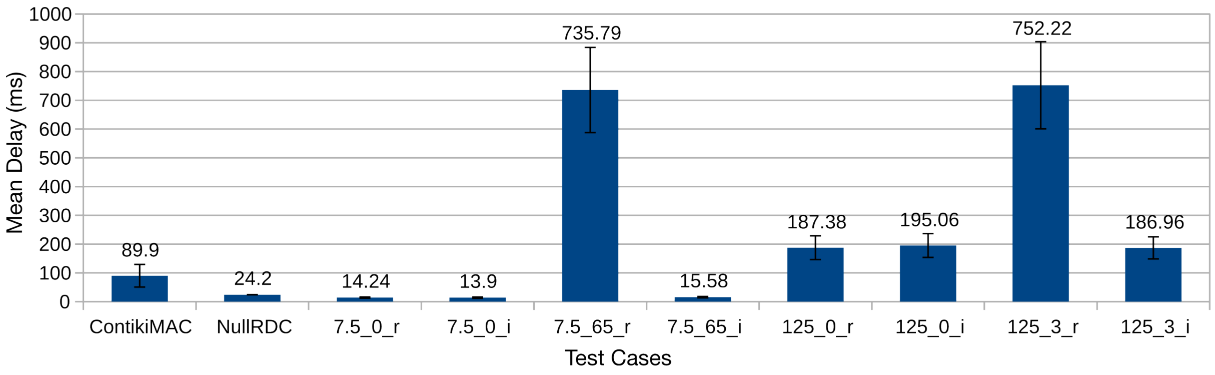


Figure 7.4: Latency

Figure 7.4 shows the mean of the thousand measurement of latency in ms for different cases of the RR test suite along with the standard deviation.

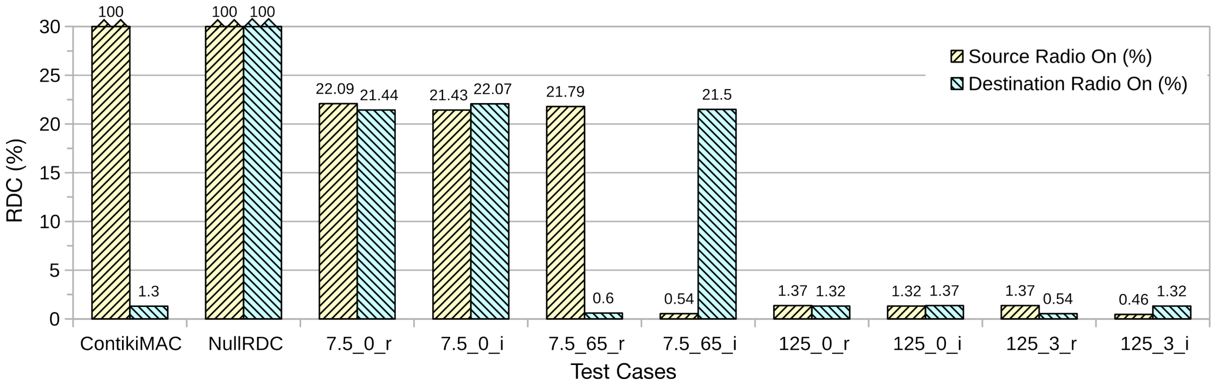


Figure 7.5: Energy Consumption

Figure 7.5 shows the energy consumption of the nodes in terms of the percentage of time the radio was switched on for the different cases of the RR test suite. Similar to the graph for the HT tests, this graph is limited to 30% for clearer representation, especially of in test cases with low duty cycle.

In 802.15.4 the source and destination nodes can be identified based on who's sending and who's receiving. In case of BLE it is a bit more complicated since source and destination nodes change based on where the latency is being measured. In the tests with the master reading a packet from the slave, the master is the source while the slave is the destination. In tests with the slave sending an indication packet to the master and receiving an acknowledgment from it, the source and destination are inverted.

Latency (ms)	Master RDC (%)	Slave RDC (%)	Test case
14.24	22.09	21.44	7.5_0_r
13.9	22.07	21.43	7.5_0_i
735.79	21.79	0.6	7.5_65_r
15.58	21.5	0.54	7.5_65_i
187.38	1.37	1.32	125_0_r
195.06	1.37	1.32	125_0_i
752.22	1.37	0.54	125_3_r
186.96	1.32	0.46	125_3_i

Table 7.1: Latency in terms of master and slave RDC

Table 7.1 shows the latency measurement from the data acquired in the RR test in terms of BLE master and slave RDC in different test cases. With this data the graph in figure 7.6 was plotted to show the various possible latency values for different master and slave's energy consumption. Note that this table and graph includes data from the BLE test cases as it is made to evaluate the impact of the various configurations on latency in terms of energy consumption.

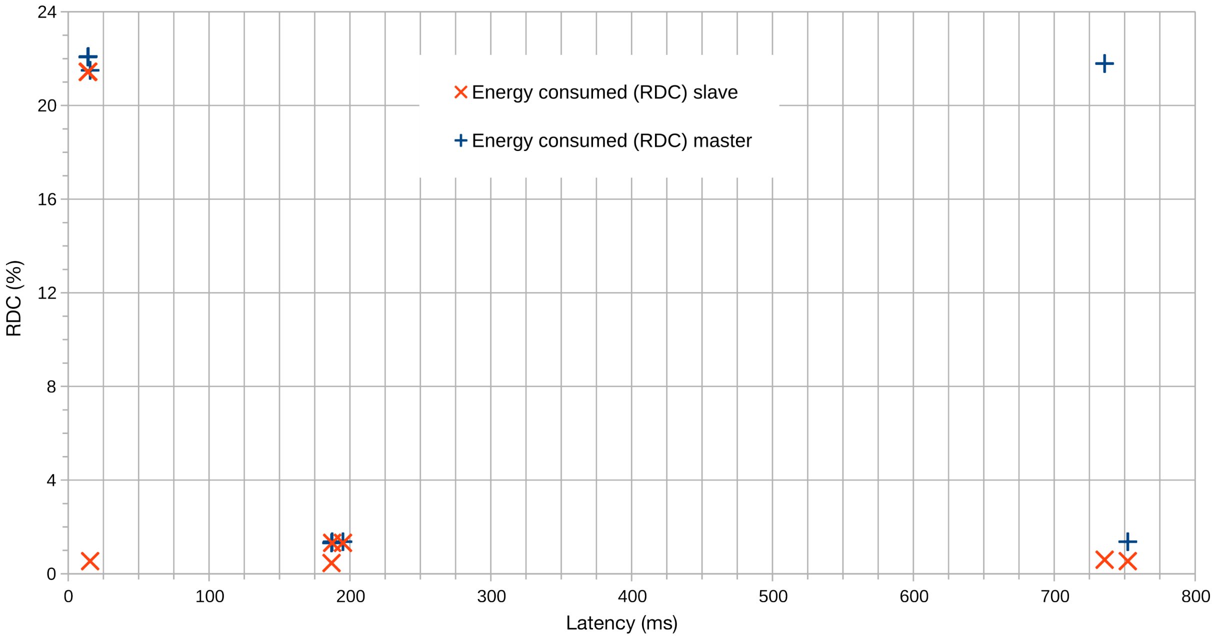


Figure 7.6: Energy Consumption vs Latency

7.2.2 Analysis

Latency The latency measurement as seen from figure 7.4 can vary in the order of magnitude among different tests. The least latency can be achieved using 802.15.4 when using Null-RDC, which is about 24 ms. When using ContikiMAC, the average latency was about 90 ms, which can be attributed to the 125 ms interval with which the receiver wakes up. In this test case where the source was assumed to be unconstrained in terms of power, it's radio was always listening, if not transmitting. This means that when the constrained destination node when ready to send the reply, the source node will be listening and get the reply immediately. This is the reason the average latency is less than the waking interval of the receiver. Note that in case both the nodes have a sleep and wake up routine, the latency would be greater than the waking interval. One disadvantage of various 802.15.4 based MAC standards is that there is no way of dynamically changing the link layer parameters. This is because there is no hierarchal master and slave concept to force particular link layer parameters.

Based on the link layer configurations used for BLE, the latency varied from a minimum of about 14 ms to a maximum of about 750 ms. The provision in BLE's link layer for changing these configurations while in a connection is an advantage since the latency can be adapted according to the requirement dynamically. The configuration of the slave latency value determines if there is any difference in the latency experienced based on whether the communication happens with a read request or an indication. With slave latency as zero, the latency of the read and indication case is almost same, with the average latency being around 14 ms with 7.5 ms connection interval and 190 ms with a connection interval of 125 ms. With a non-zero slave latency value, the read-request from the master node (client in ATT) has high latency since the slave node (server at ATT)will not react to the master at every connection interval as it is sleeping. With a slave latency value of 65, the master node can read a value from the slave node with a latency of about 736 ms, even with a connection interval of 7.5 ms. Similarly, with a slave latency of 3 and connection interval of 125 ms, the master will read a value from the slave with a latency of about 752 ms. When using indication to communicate data to a master, the latency can be low since the slave can decide to respond to the master when necessary communicate. With a 7.5 ms connection interval and slave latency of 65, the latency is about 15 ms, which is similar to the latency experience with zero slave latency. Similarly, an indication providing the master with data experiences a latency of 187 ms with a connection interval of 125 ms and slave latency of 3. This is almost equal to the latency experienced with zero slave latency. This shows that use of indication does not effect the latency experienced to send data to a master node in a connection with non-zero slave latency.

With slave latency as zero, it can be seen that for an application layer the latency experienced is between one and two times the connection interval, the best case being close to one connection interval. This shows that a read-request communication happening in the same connection event is not possible. The read request in sent in one connection event and the requested data is sent back in the next connection event. This is because the 150 µs inter frame time in a connection event is not enough for the resource constrained MCUs respond back with the requested data.

Energy Consumption The energy consumption of the 802.15.4 source and destination node is 100% in terms of RDC in case of Null-RDC. This is as expected because of the nature of Null-RDC implementation. In case of ContikiMAC, the source node has 100% RDC while the destination has 1.3%. The source node has this high energy consumption because it is configured so. This in accordance with the assumption of one node being unconstrained in terms of power. Usually the root node of a 802.15.4 network has access to a wall socket and gathers data from the other nodes. It is the destination node which is battery operated and has to conserve energy.

In case of BLE, the energy consumption of the nodes which have to switch on their radio every 7.5 ms have the highest RDC at around 21-22%. This includes both the source and destination when the slave latency is zero and connection interval is 7.5 ms. In case the slave latency of 65 is used, the master has RDC of around 21%, while the slave has a RDC of only 0.6%. This is since the slave can choose only to respond to the master every 66th connection interval and sleep the rest of the time, thereby creating an asymmetric architecture. Similarly when the connection interval is 125 ms and slave latency is zero, both master and slave node have approximately equal RDC of 1.32 to 1.37%. By using a slave latency of 3 and the same connection interval, the slave nodes were able to sleep for longer duration leading to the reduction of RDC to around 0.6%.

Latency vs Energy Consumption Figure 7.6 clearly showcases the flexibility available in BLE in configuring the link layer dynamically. The parameter that allows that is slave latency. A symmetric connection is created when the slave latency is zero, causing slave to respond to the master in every connection interval. This scenario will result in the latency for a read response from either the master or the slave being almost equal. Also the RDC of the master and slave device will be similar. In this case there is a direct relation between the latency and connection interval, which means that if lower latency is required then more energy is consumed. This scenario is useful when the latency requirement for any communication started by the master is greater than that of the slave. An example for this scenario is a Human Interface Device (HID) that also provides feedback to the user, like as a game controller as a slave communicating with a gaming console as master. In this case there needs to be low latency for both the commands sent from the controller to the console, as well for the vibration feedback sent to the controller from the console. The test case that represent this in figure 7.6 are the ones with 7.5 ms connection interval and zero slave latency to achieve near instantaneous response.

By using a non-zero slave latency, the slave node can be sleepy by not having the responsibility to respond in every connection event. As figure 7.6 shows, this asymmetric connection creates a situation where the communication started from the slave has a latency as if the slave latency parameter was zero, since the master is communicating in all connection events. On the other hand, communication started by the master can experience high latency, since it has to wait till the master responds. This can be seen in the case `7.5_65_i` where the slave can communicate with 14 ms latency with the master while having a RDC of only 0.54%, although the master's radio needs to be on for about 21.5% of the time. This scenario is useful with HID devices such as keyboards where there is usually no feedback to be the sleepy slave devices. This will allow them to have low latency to communicate the keys pressed while allowing them to sleep when the keyboard isn't used, hence extending battery life.

7.3 Evaluation with existing research

Data Rate Considering the research that looked into the data rate achievable, the data rate achieved in this project falls well within the maximum data rate achievable[10] of 236.7 kbps. The data rate in experiments have looked at the data rate from the link layer[20] or application layer[1][12][13], all of them by sending notification packets. To equalize the result with this experiment, the data rate is compared in terms of packets per second and presented in table 7.2.

This comparison shows that the data rate achieved is highly dependent on the BLE stack used. The BLE stack used, on both the master and slave device, determines parameters such as connection interval and packets consistently allowed to be transmitted per connection event with full application payload of 20 bytes. As seen from the table above, even though all the devices are following the BLE core specification, the limitations of the stacks for these two parameters determines the maximum data rate achievable. By improvements in the stack implementation, the data rate can be improved to reach closer to the limit of 236.7 kbps, thus extending BLE to application which require intermittent streaming of data.

This thesis project extends the research to data rate achievable in the presence of WiFi interference when using both complete and WiFi free BLE channel map. The energy con-

Article	Throughput (packet/second)	Devices communicating	Master device's stack
This expt.	133.3	PCA10000 to PCA10000	S120 v1.0.0
This expt.	399.8	PCA10000 to Nexus4/7	Android 4.4.4
[1]	365.5	CC2540 to CC2540 (dev board)	TI BLE Stack
[12]	250	CC2540 to CC2540 (dev board)	TI BLE Stack
[13]	637.5	Bluegiga BLE112 to BLE112	Bluegiga stack [14]
[20]	567.6	CC2540 to CC2540 (dev board)	TI BLE Stack v1.21

Table 7.2: Comparison of BLE data rate achieved

sumption in terms of RDC has been measured when sending data at maximum data rate, to two different master devices at different data rates.

External Interference As mentioned in section 4.3, in an experiment[19] the PRR was measured for receiving the advertisements of a slave by a master, while varying the distance of the source of interference. Similarly, the influence of WiFi on BLE advertisement and vise versa has been evaluated for up to 21 advertisers[15]. This thesis extends the research of the effect of WiFi interference on BLE in connection mode in terms of both PRR and PDR. Although automatic AFH was not available in the test setup in this project, the effect of AFH was analyzed by manually selecting the channel map without WiFi interference.

Reliability There has been research on reliability of receiving the advertisement packets by scanners in the presence of WiFi interference[15][19]. To the best of the author's knowledge, this is the first experiment which finds the reliability of the link layer of BLE when in a connection. This has been looked into with and without external WiFi interference.

Latency The latency for detection of a BLE advertiser by a scanner has been extensively modeled and verified[13][26]. Latency for periodic formation of BLE connection and transfer of data has been measured[21], although the advertising and link layer configuration has not been detailed. The time for a connection event with sending a packet with maximum payload is measured as latency[1]. One article has measured the latency for a read-request operation, where the time measurement starts and ends with sending and receiving the data at the physical layer to/from the link-layer[20]. This thesis project extends the research by exploring the influence of the link-layer parameters of connection interval and slave latency on the latency measurement for a read-request operation.

Energy Consumption Consistent with the name of the protocol, there has been extensive research on the energy consumption of a BLE devices. A recent paper has extensively and precisely modeled the energy consumption of BLE devices in during a connection and when being advertisers and scanners[13]. Various BLE ICs have their power consumption compared in a compared[25]. The battery life of a BLE device has been estimated for various connection and packet configuration[1]. There have been many research papers where the energy required for the different phases of a BLE transmission is accurately measured[20][19][12][11] for the CC254x SoC from Texas Instruments.

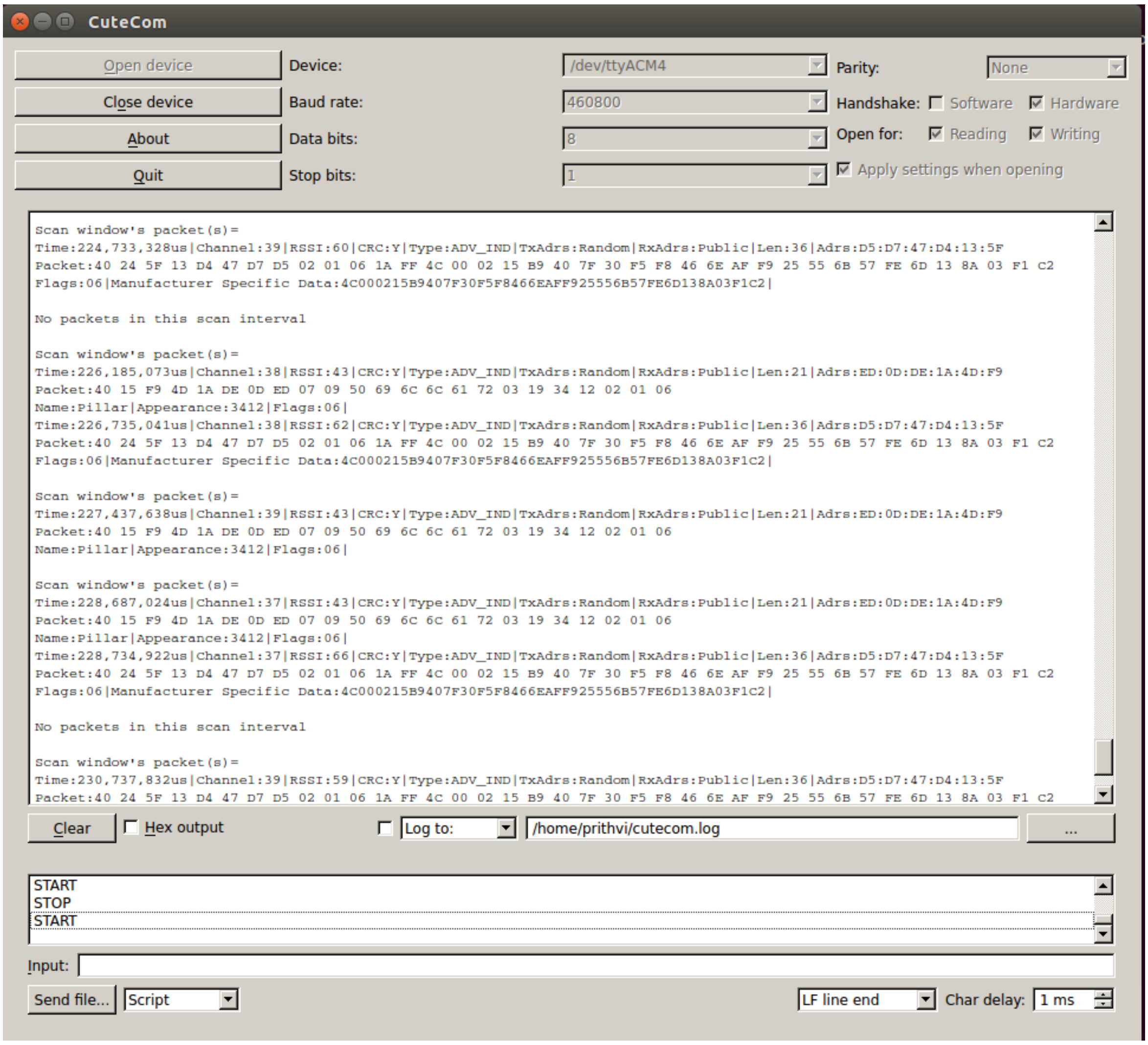
This thesis project complements this existing research by providing an idea of the energy consumed for various scenarios, especially useful for bivariate analysis with other metrics. With the existing models available, this information in terms of RDC can easily be converted to precise energy consumed for the platform used.

Comparison of BLE and 802.15.4 There has been research which compares BLE link layer with standard 802.15.4[20][19] and SimpliciTI[20]. To the best of our knowledge, this is the first experiment which compares BLE and 802.15.4 which uses ContikiMAC and Null-RDC MAC layer. This thesis project explores test case scenarios which are practically useful to a BLE device and application developer such as streaming data to different devices, effect of heavy WiFi interference and latency for a read-request operation, all with various link-layer configuration.

8 | Other Contributions

8.1 Advertisement Logger[*]

During the exploration phase of the thesis, the plan was to implement the required parts of the link layer of BLE from scratch. This included developing the entire radio driver. After looking into the radio peripheral of nrf51822 and working with it, this plan was abandoned due to lack of time to finish these tasks and the test cases. The work with the radio peripheral did result in developing a full-fledged BLE advertisement packet logger, whose interface (using CuteCom serial terminal) can be seen in figure 8.1. Detailed description can be found in Appendix B.



Device: /dev/ttyACM4
Baud rate: 460800
Data bits: 8
Stop bits: 1
Parity: None
Handshake: □ Software ☑ Hardware
Open for: ☑ Reading ☑ Writing
☑ Apply settings when opening

```
Scan window's packet(s)=
Time:224,733,328us|Channel:39|RSSI:60|CRC:Y|Type:ADV_IND|TxAdrs:Random|RxAdrs:Public|Len:36|Adrs:D5:D7:47:D4:13:5F
Packet:40 24 5F 13 D4 47 D7 D5 02 01 06 1A FF 4C 00 02 15 B9 40 7F 30 F5 F8 46 6E AF F9 25 55 6B 57 FE 6D 13 8A 03 F1 C2
Flags:06|Manufacturer Specific Data:4C000215B9407F30F5F8466EAFF925556B57FE6D138A03F1C2|

No packets in this scan interval

Scan window's packet(s)=
Time:226,185,073us|Channel:38|RSSI:43|CRC:Y|Type:ADV_IND|TxAdrs:Random|RxAdrs:Public|Len:21|Adrs:ED:0D:DE:1A:4D:F9
Packet:40 15 F9 4D 1A DE 0D ED 07 09 50 69 6C 6C 61 72 03 19 34 12 02 01 06
Name:Pillar|Appearance:3412|Flags:06|
Time:226,735,041us|Channel:38|RSSI:62|CRC:Y|Type:ADV_IND|TxAdrs:Random|RxAdrs:Public|Len:36|Adrs:D5:D7:47:D4:13:5F
Packet:40 24 5F 13 D4 47 D7 D5 02 01 06 1A FF 4C 00 02 15 B9 40 7F 30 F5 F8 46 6E AF F9 25 55 6B 57 FE 6D 13 8A 03 F1 C2
Flags:06|Manufacturer Specific Data:4C000215B9407F30F5F8466EAFF925556B57FE6D138A03F1C2|

Scan window's packet(s)=
Time:227,437,638us|Channel:39|RSSI:43|CRC:Y|Type:ADV_IND|TxAdrs:Random|RxAdrs:Public|Len:21|Adrs:ED:0D:DE:1A:4D:F9
Packet:40 15 F9 4D 1A DE 0D ED 07 09 50 69 6C 6C 61 72 03 19 34 12 02 01 06
Name:Pillar|Appearance:3412|Flags:06|

Scan window's packet(s)=
Time:228,687,024us|Channel:37|RSSI:43|CRC:Y|Type:ADV_IND|TxAdrs:Random|RxAdrs:Public|Len:21|Adrs:ED:0D:DE:1A:4D:F9
Packet:40 15 F9 4D 1A DE 0D ED 07 09 50 69 6C 6C 61 72 03 19 34 12 02 01 06
Name:Pillar|Appearance:3412|Flags:06|
Time:228,734,922us|Channel:37|RSSI:66|CRC:Y|Type:ADV_IND|TxAdrs:Random|RxAdrs:Public|Len:36|Adrs:D5:D7:47:D4:13:5F
Packet:40 24 5F 13 D4 47 D7 D5 02 01 06 1A FF 4C 00 02 15 B9 40 7F 30 F5 F8 46 6E AF F9 25 55 6B 57 FE 6D 13 8A 03 F1 C2
Flags:06|Manufacturer Specific Data:4C000215B9407F30F5F8466EAFF925556B57FE6D138A03F1C2|

No packets in this scan interval

Scan window's packet(s)=
Time:230,737,832us|Channel:39|RSSI:59|CRC:Y|Type:ADV_IND|TxAdrs:Random|RxAdrs:Public|Len:36|Adrs:D5:D7:47:D4:13:5F
Packet:40 24 5F 13 D4 47 D7 D5 02 01 06 1A FF 4C 00 02 15 B9 40 7F 30 F5 F8 46 6E AF F9 25 55 6B 57 FE 6D 13 8A 03 F1 C2
```

□ Hex output
□ Log to: /home/prithvi/cutecom.log

```
START
STOP
START
```

Input:

LF line end
Char delay: 1 ms

Figure 8.1: User Interface of Advertisement Logger

[*] Available at `https://github.com/EarthLord/nrf51AdvLogger` with Doxygen documentation

8.2 Firmware for an Alarm Device[†]

A demonstration application was developed to test the port of Contiki on the nrf51822 platform. This application consisted of an alarm device controlled by a mobile application. The PCA10000 platform was used for this development too. The operations of setting of the next alarm time, canceling the alarm and also silencing the alarm can be done from a mobile application. The *Master Control Panel* application by Nordic Semiconductor was used to test all the functionality.

The functioning of the alarm device with the mobile *App* controlling it from a user's perspective can be seen in the state diagram in figure 8.2

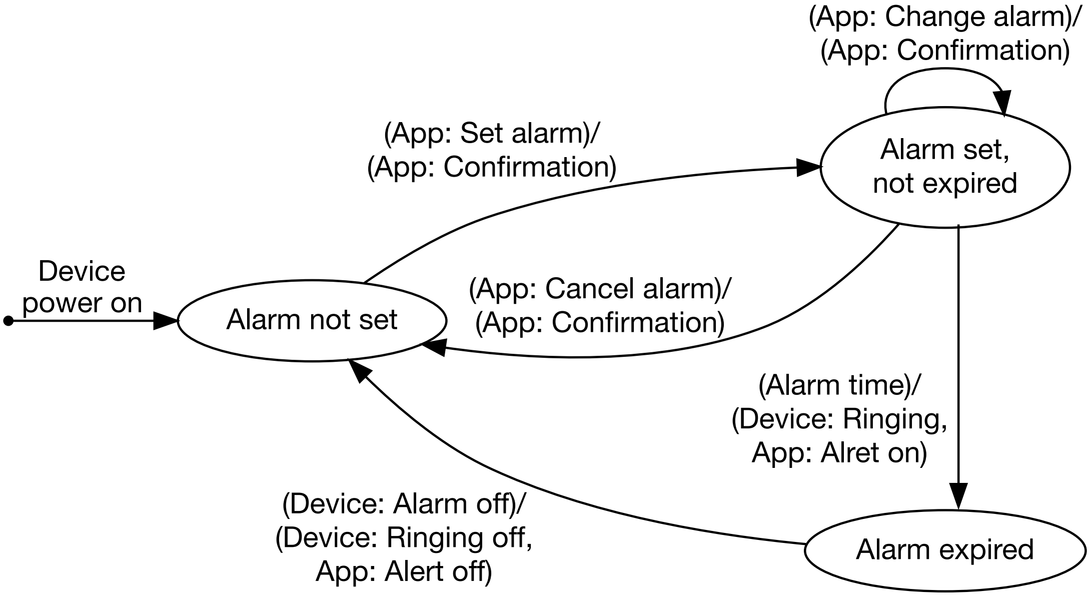


Figure 8.2: State Diagram of an alarm device from an user's perspective

To achieve this a custom *service* was created to contain two custom *characteristics*. One characteristic contained the time to the next alarm and the other characteristic contained the status of whether the alarm is set or not, both with Client Characteristic Configuration Descriptor (CCCD). The details of the two characteristics are:

Next alarm time (§)

- Read, write and notify allowed
- CCCD to disable/enable notifications
- Open link security mode
- No read/write authorization required
- 32-bit value
  - –0 implies no alarm set
  - –Seconds to next alarm, if not 0

Alarm status (¶)

- Read and notify allowed
- CCCD to disable/enable notifications
- Open link security mode
- No read/write authorization required
- 8-bit value
  - –1 if alarm is set or ringing
  - –0 if alarm is not set

Apart from these characteristics, Contiki etimers and Contiki serial-line module from the Contiki port to nrf51822 are used for this alarm. Since the PCA10000 platform is used for this alarm device's development, the 'ringing' of the alarm is done by flashing the LED on board and 'silencing' the alarm is done by sending a string through the serial port. The state diagram in figure 8.3 shows the implementation done for this alarm device. Note that (§) and (¶) represent the characteristics described above.

[†] Available at `https://github.com/EarthLord/contiki/tree/master/examples/PCA10000-nrf/Pillar`

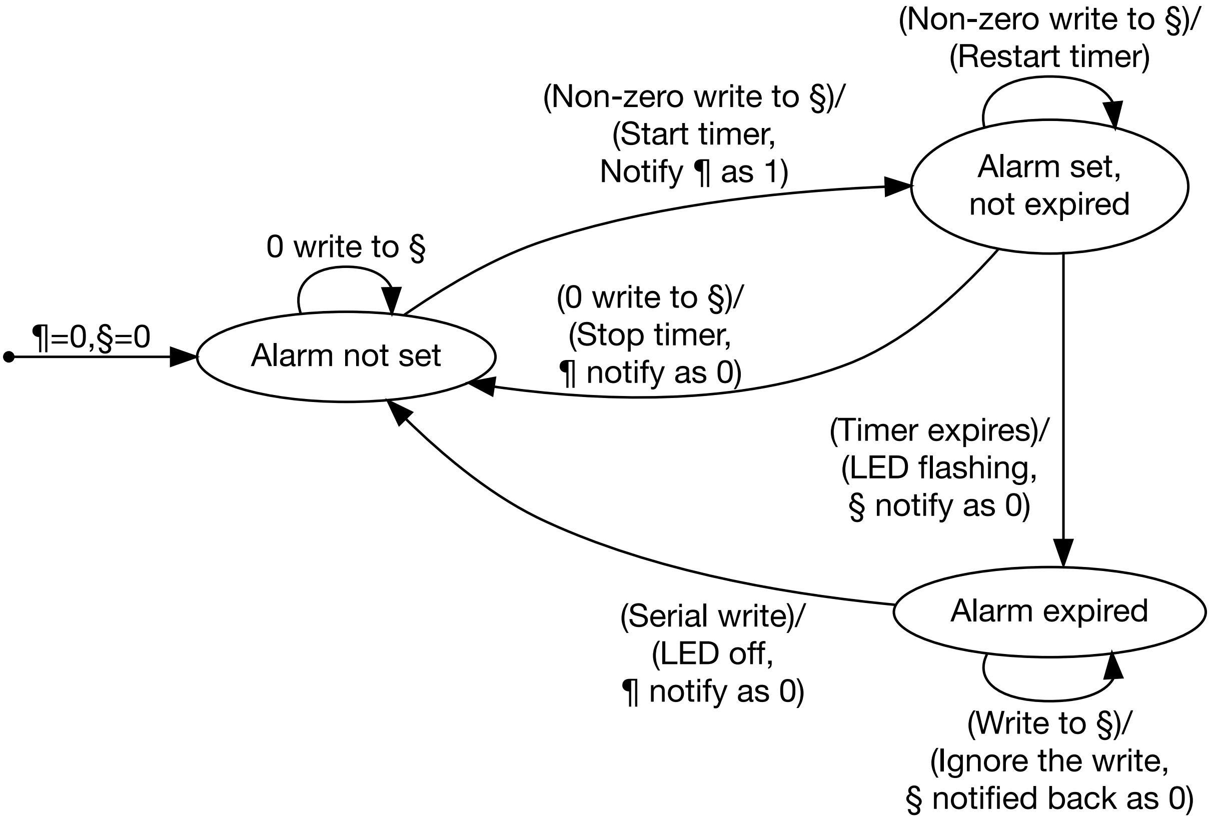


Figure 8.3: State Diagram of an alarm appcessory controlled by BLE

9 | Conclusion and Future Work

References

[1] Carles Gomez, Joaquim Oller, and Josep Paradells. “Overview and evaluation of bluetooth low energy: an emerging low-power wireless technology.” In: *Sensors (Basel, Switzerland)* 12.9 (Jan. 2012), pp. 11734–53.

[2] *Contiki: The Open Source Operating System for the Internet of Things*. URL: `http://contiki-os.org/` (Retrieved on Feb. 17, 2014).

[3] *Contiki Hardware*. URL: `http://contiki-os.org/hardware.html` (Retrieved on Feb. 17, 2014).

[4] George Oikonomou and Iain Phillips. “Experiences from porting the Contiki operating system to a popular hardware platform”. English. In: *2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*. IEEE, June 2011, pp. 1–6.

[5] Alexandru Stan. “Porting the Core of the Contiki operating system to the TelosB and MicaZ platforms”. Bachelor Thesis. International University Bremen, 2007.

[6] Adriana Wilde, Richard Oliver, and Ed Zaluska. “Developing a low-cost general-purpose device for the Internet of Things”. English. In: *2013 Seventh International Conference on Sensing Technology (ICST)*. IEEE, Dec. 2013, pp. 490–494.

[7] Zoltan Padrah, Ales Verbic, Marko Mihelin, et al. “Connecting Contiki enabled Versatile Sensor Nodes via CC1101 radio”. In: *NEWCOM++* (2011).

[8] Voravit Tanyingyong, Robert Olsson, Markus Hidell, et al. “Design and Implementation of an IoT-controlled DC-DC Converter”. eng. In: *2013 Sustainable Internet and ICT for Sustainability, SustainIT 2013* (2013), p. 6685199.

[9] Robin Heydon. *Bluetooth Low Energy: The Developer’s Handbook*. 1st ed. Prentice Hall, 2012, p. 368.

[10] Carles Gomez, Ilker Demirkol, and Josep Paradells. “Modeling the Maximum Throughput of Bluetooth Low Energy in an Error-Prone Link”. In: *IEEE Communications Letters* 15.11 (Nov. 2011), pp. 1187–1189.

[11] Jia Liu and Canfeng Chen. “Energy analysis of neighbor discovery in Bluetooth Low Energy networks”. In: *Nokia.(nd)* (2012).

[12] Elke Mackensen, Matthias Lai, and Thomas M. Wendt. “Performance analysis of an Bluetooth Low Energy sensor system”. In: *2012 IEEE 1st International Symposium on Wireless Systems (IDAACS-SWS)*. September. IEEE, Sept. 2012, pp. 62–66.

[13] Philipp Kindt, Daniel Yunge, Robert Diemer, et al. “Precise Energy Modeling for the Bluetooth Low Energy Protocol”. In: *CoRR* abs/1403.2 (Mar. 2014). arXiv: `1403.2919`.

[14] Mikko Savolainen. *Throughput with Bluetooth Smart technology : Bluegiga Technologies*. 2013. URL: `https://bluegiga.zendesk.com/entries/24646818-Throughput-with-Bluetooth-Smart-technology` (Retrieved on Aug. 5, 2014).

[15] Jeroen Wyffels, Jean Pierre Goemaere, Bart Nauwelaers, et al. “Influence of Bluetooth Low Energy on WIFI communications and vice versa”. en. In: ().

[16] Adam Dunkels. “The ContikiMAC Radio Duty Cycling Protocol”. In: (Dec. 2011).

[17] Mathieu Michel and Bruno Quoitin. “Technical Report : ContikiMAC vs X-MAC performance analysis”. In: *arXiv preprint arXiv:1404.3589* (Apr. 2014). arXiv: `1404.3589`.

[18] M.-P. Uwase, M. Bezunartea, T.L. Nguyen, et al. “Experimental evaluation of message latency and power usage in WSNs”. In: *2014 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*. IEEE, May 2014, pp. 69–72.

[19] Matti Siekkinen, Markus Hiienkari, Jukka K Nurminen, et al. “How low energy is bluetooth low energy? Comparative measurements with ZigBee/802.15.4”. In: *2012 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. IEEE, Apr. 2012, pp. 232–237.

[20] Konstantin Mikhaylov, Nikolaos Plevritakis, and Jouni Tervonen. “Performance Analysis and Comparison of Bluetooth Low Energy with IEEE 802.15.4 and SimpliciTI”. In: *Journal of Sensor and Actuator Networks* 2.3 (Aug. 2013), pp. 589–613.

[21] Artem Dementyev, Steve Hodges, Stuart Taylor, et al. “Power consumption analysis of Bluetooth Low Energy, ZigBee and ANT sensor nodes in a cyclic sleep scenario”. English. In: *2013 IEEE International Wireless Symposium (IWS)*. IEEE, Apr. 2013, pp. 1–4.

[22] ZigBee Alliance. *ZigBee Specification*. URL: `http://www.zigbee.org/Specifications.aspx` (Retrieved on Aug. 6, 2014).

[23] Nordic Semiconductor. *nRF51822*. URL: `https://www.nordicsemi.com/eng/Products/Bluetooth-R-low-energy/nRF51822` (Retrieved on Mar. 9, 2014).

[24] PrithviRaj Narendra. *Introduction to PCA10000 and using JLink Software with it in Ubuntu*. 2014. URL: `http://rumblingsofearthlord.wordpress.com/2014/01/22/introduction-to-pca10000-and-using-jlink-software-with-it-in-ubuntu/` (Retrieved on Aug. 10, 2014).

[25] Jachen Bernegger and Marcel Meli. “Comparing the energy requirements of current Bluetooth Smart solutions”. In: InES Institute of Embedded Systems. Nuremberg: Embedded World 2014, 2014.

[26] Jia Liu, Canfeng Chen, and Yan Ma. “Modeling Neighbor Discovery in Bluetooth Low Energy Networks”. In: *IEEE Communications Letters* 16.9 (Sept. 2012), pp. 1439–1441.

Appendix A | Contiki Folder and Makefile Structure

A.1 Folder structure of Contiki

When porting a new platform to Contiki, the three folders inside a Contiki distribution where additions need to be made are cpu, examples and platform. The content in these folders are as described below. The location of the Contiki distribution is represented by the path ‘CONIKI’. The port of nrf51822 SoC with its PCA10000 board also follows this convention.

CONTIKI/cpu In this folder, all the files contain implementation which is solely dependent on the SoC is present. This includes the code for the processor abstraction, the drivers of the peripherals of the SoC, makefile with commands for compiling and linking the code, linker file and the documentation of these implementation. The boot-loader or start-up code, if required is also present here. The peripheral drivers for peripherals such as timers and serial port must use the API format of Contiki so that the libraries of Contiki using these peripherals can operate correctly. The exact implementations of these drivers is explained in section 5.2.2.

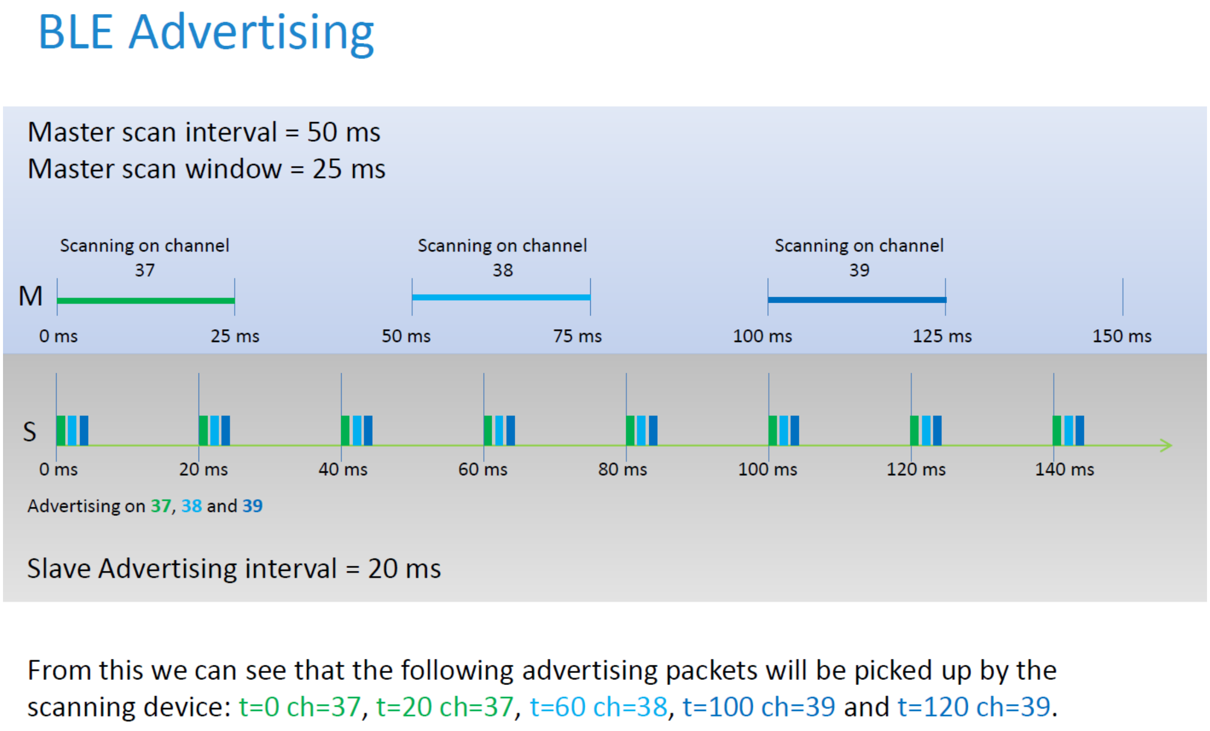
CONTIKI/platform An SoC over time has increasing number of boards or systems that are based on it and these are referred as platforms in this folder. Every board has its own folder in this platform folder which contains files which contain implementation which is dependant on the particular board. These include the specification of the connections to the LEDs, buttons, sensors and serial ports, power sources, the clock sources, memories present and any other board specific details. Any peripheral driver implementation specific to the board is present here. The default project specifications such as the source and frequency of clock and serial baud-rate are defined here. The main function where the execution of the program starts and initialization of all the peripherals and libraries used by Contiki happens is present here.

CONTIKI/examples The examples folder contains all the files related to projects that are implemented using Contiki with any of the supported hardware platforms. The makefile where the make command is executed is present here. The compiled object and binary files are also stored here. The default specifications of the platform can be overridden by creating a `project-conf.h` in the specific example.

Appendix B | Advertisement Logger's Implementation

B.1 Scanning BLE Advertisements

According to the Bluetooth 4.0 Core specifications, an advertisement event consists of an advertising device periodically send advertisement packets. The interval of these advertising events can range from 20 ms to 10 s. In each of these advertising events an advertiser sends an advertising packet in each channel in an interval of 10 ms starting with channel 37. The objective of a BLE advertisement scanner would be to be able to receive the packets from at least one channel in an advertisement event, while saving power. To achieve this trade off, the radio duty cycles between receiving and disabled state, while being in the next advertising channel every time it is switched on. The total time of this periodic event is known as the *Scan Interval*, while the time the radio is on is known as *Scan Window*. The figure* B.1 depicts this process.



BLE Advertising

Master scan interval = 50 ms
Master scan window = 25 ms

Slave Advertising interval = 20 ms

From this we can see that the following advertising packets will be picked up by the scanning device: t=0 ch=37, t=20 ch=37, t=60 ch=38, t=100 ch=39 and t=120 ch=39.

Figure B.1: Scanning BLE advertisements

*https://devzone.nordicsemi.com/question/2535/what-is-the-minimum-time-for-a-app-and-a-peripheral-to-create-a-connection/

B.2 Software Architecture

The following flow charts explain the software architecture of the *Advertisement Logger*.

B.2.1 Main Function

The execution of the program starts with the main function after reset as depicted by figure B.2. Note that the radio is initialized to continuously receive packets once it has started while also measuring the received signal strength RSSI.

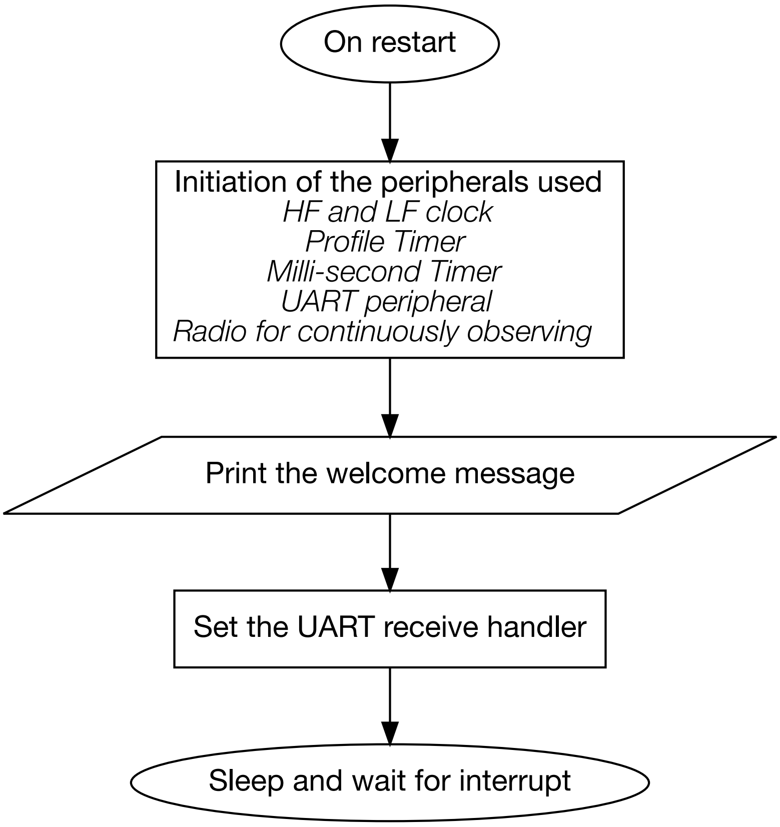

On restart
Initiation of the peripherals used
HF and LF clock
Profile Timer
Milli-second Timer
UART peripheral
Radio for continuously observing
Print the welcome message
Set the UART receive handler
Sleep and wait for interrupt

Figure B.2: Main function flowchart

B.2.2 UART String Receive Handler

The UART String Receive handler which gets a string sent to the SoC through an unsigned character pointer checks if the string is either `START` or `STOP` so that the scanning of the advertisements can be started or stopped respectively.

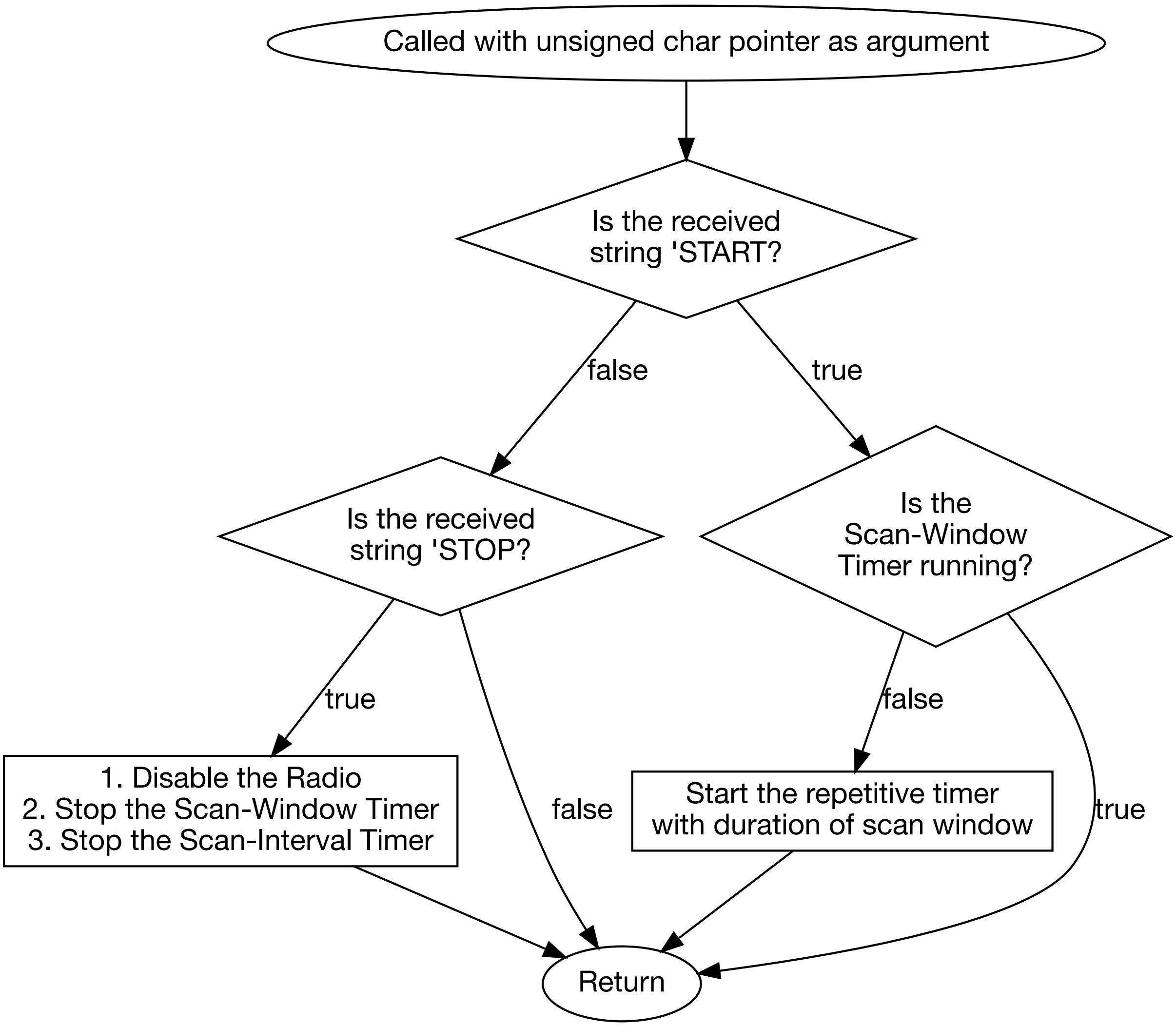

Called with unsigned char pointer as argument
Is the received string 'START?
false
true
Is the received string 'STOP?
Is the Scan-Window Timer running?
true
false
1. Disable the Radio
2. Stop the Scan-Window Timer
3. Stop the Scan-Interval Timer
false
Start the repetitive timer with duration of scan window
true
Return

Figure B.3: UART string receive handler flowchart

B.2.3 Scan Interval Timer Handler

The scan interval timer is a repetitive timer with an interval equal to the scan interval. At the beginning of the scan interval this function performs 5 different tasks as shown in figure B.4.

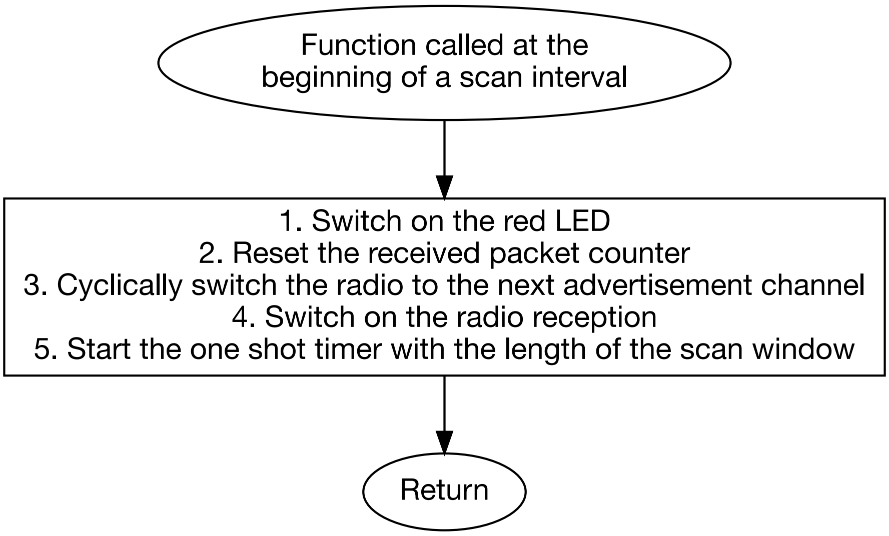

Function called at the beginning of a scan interval
1. Switch on the red LED
2. Reset the received packet counter
3. Cyclically switch the radio to the next advertisement channel
4. Switch on the radio reception
5. Start the one shot timer with the length of the scan window
Return

Figure B.4: Scan interval timer handler flowchart

B.2.4 Scan Window Timer Handler

The scan window timer is a single shot timer with an interval equal to the scan window. At the end of the scan window this function performs 3 different tasks as shown B.5.

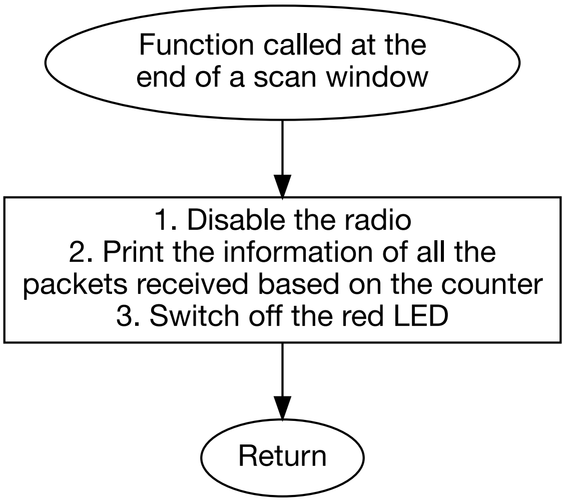

Function called at the end of a scan window
1. Disable the radio
2. Print the information of all the packets received based on the counter
3. Switch off the red LED
Return

Figure B.5: Scan window timer handler flowchart

B.2.5 Radio Interrupt Routine

The radio peripheral is configured such that the end of a packet and the end of RSSI measurement trigger the interrupt. At the end of a packet, the packet's data is collected if there is space in the buffer. At the end of the RSSI measurement the RSSI value is saved. Note that when receiving a packet the 'end of RSSI measurement' event happens before the 'end of packet' event.

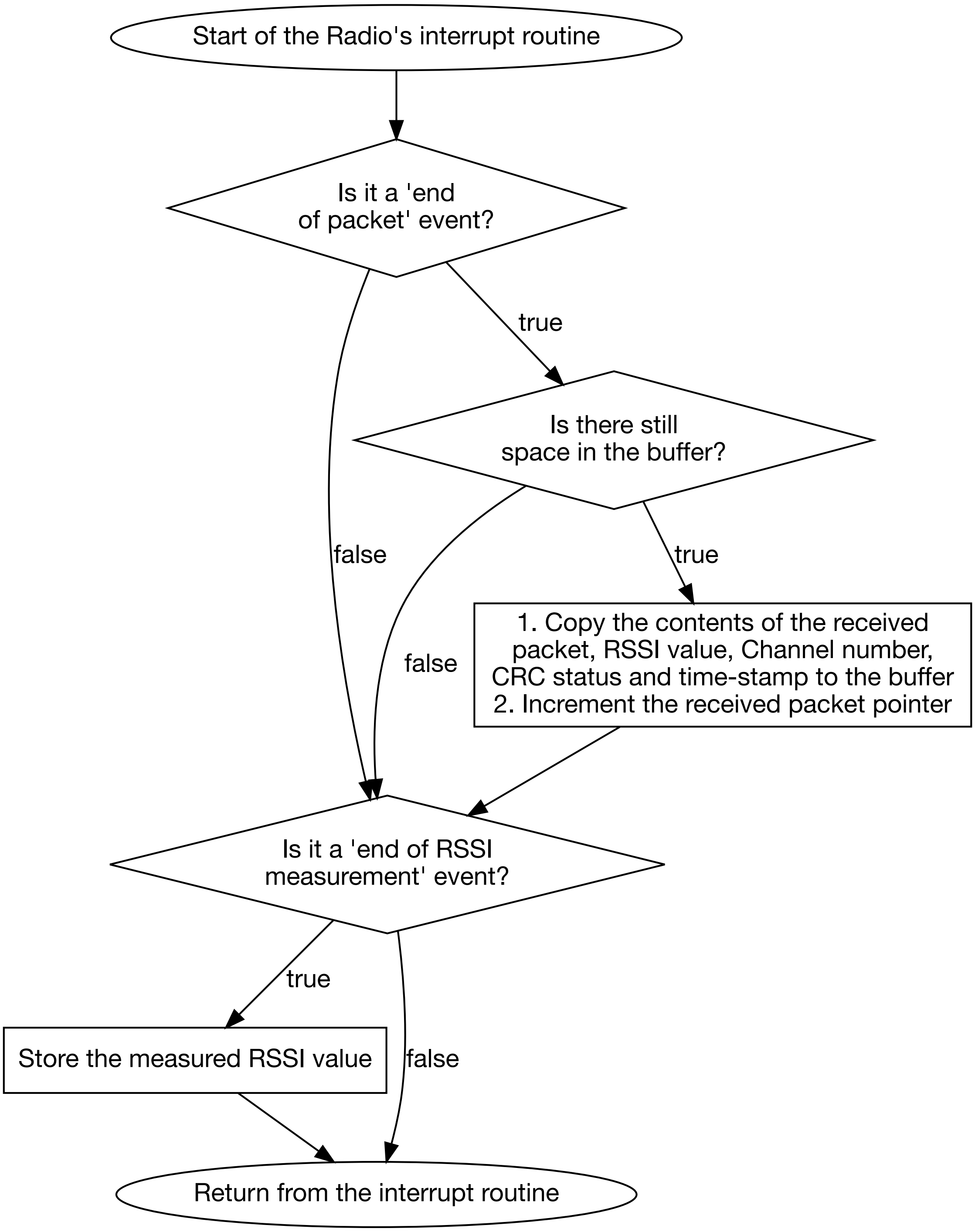

Start of the Radio's interrupt routine
Is it a 'end of packet' event?
true
Is there still space in the buffer?
false
true
false
1. Copy the contents of the received packet, RSSI value, Channel number, CRC status and time-stamp to the buffer
2. Increment the received packet pointer
Is it a 'end of RSSI measurement' event?
true
Store the measured RSSI value
false
Return from the interrupt routine

Figure B.6: Radio interrupt routine flowchart

Appendix C | Appendix | BLE SoC Comparison

The dense table in the next page compares many different parameters of the BLE SoCs available. This table was compiled in the first half of 2014. Because of the high pace at which the industry around BLE is progressing, many details here might not be accurate as time progresses.

The legend for the table is

NA	: The information is not available
-	: The feature is not available

Platform	Unit	TI CC2540/1	CSR1001/ CSR1000	NordicSemi nRF51822	DialogSemi DA14580	Broadcom BCM20732	Quintic 9020	Lapis ML7105
Processor		8051	16 bit proprietary	Arm Cortex M0	Arm Cortex M0	Arm Cortex M3	Arm Cortex M0	Arm Cortex M0
Flash size	kB	128/256	64 (ROM)	128/256	32 (OTP)	NA	NA	64
RAM size	kB	8	64	16	42K SRAM & 8K Retention RAM	NA	NA	12 User code, 16 Data
RX Sensitivity	dBm	-93	-92.5	-93	-93	NA	-95	-85
TX Max Power		4	7.5	4	NA	NA	NA	3
GPIO		21	32	31	32	NA	31	
ADC		12 bit, 8 chl, up to 30 kHz	10 bit, 3 ch, up to 700 Hz	10 bit, 8 ch, upto 14 kHz	10 bit	Yes	Yes	-
I2C		1, if not USB	1	2	1	1	Yes	1
SPI		-	1	2	1	1	Yes	1
UART		2	1	1	1	1	Yes	1
USB		Full-Speed	-	-	-	-	-	-
DMA		5 Channel	1 channel	16 PPI,No DMA	NA	NA		-
Timer 32 bit		-	-	1				
Timer 16 bit		1	-	2	2+wakeup timer			
Timer 8 bit		2	-	-				
Other peripherals		OPAMP, Comparator, WDT, AES Encryption	AES, LED PWM	AES, WDT, Internal DC/DC regulator, RNG QDEC, 4 PWM channels,	AES, WDT, Internal DC/DC regulator, QDEC	DAC	DC/DC Regulator	AES, WDT
Active RX	mA	19.6	16	13	3.8	NA	8	9
Active TX at 0 dBm	mA	27	16	10.5	3.8	NA	8	9
Idle	mA		1	275 uA/MHz				3
Processor Off	uA	235						
RTC Timer on	uA	0.9	1.5	2.3 uA	0.6			2.9
External Interrupt	uA	0.4	0.4	0.5	0.6			0.7
Package		40 pin QFN	56 pin QFN	48 pin QFN	48&40 pin QFN, 34 WLCSP	32 pin QFN		WQFN32
BLE stack		Free, from TI	From CSR	From NordicSemi	Integratedin ROM	Yes, already in ROM		
Support Forum		TI E2E Community		Nordic DevZone	support.dialog-semiconudctor.com/			
IDE to use		IAR Studio	Proprietary IDE	Keil, Eclipse+GCC	Keil	NA		
Development Board		CC2540DK, CC2540EMK	DK-CSR1000-10048	nRF51822-EK	DA14580 Motherboard + WCSP or QFN daughterboards	BCM920732_BLE_KIT		
Price of Dev Board		101	300	108	NA	166		
Price of IC (single IC)	$	5.62	3.1	3.84	NA	NA	NA	NA
Price of IC (bulk quantiy)	$	2.59	NA	1.91	NA	NA	NA	NA

Appendix D | Data Acquired

The entire data acquired and results computed for both High-Throughput test and Request-Response test can be found in the tables in the following pages.

High-Throughput Test

Test Cases	**sky_sky_N_!CCA**	**sky_sky_Y_26**	**sky_sky_Y_15**	**nrf_nrf_N_all**	**nrf_nrf_Y_all**	**nrf_nrf_Y_!Wifi**	**nrf_N4_N**	**nrf_N7_N**
Source	Tmote-Sky	Tmote-Sky	Tmote-Sky	PCA10000	PCA10000	PCA10000	PCA10000	PCA10000
Destination	Tmote-Sky	Tmote-Sky	Tmote-Sky	PCA10000	PCA10000	PCA10000	Nexus4	Nexus7
WiFi	No	Yes	Yes	No	Yes	Yes	No	No
Condition	No CCA	CCA on, ch 26	CCA on, ch 15		All channels	Wifi free channels	Wifi OFF in device	Wifi OFF in device
DataRate (packets/s)	176.20	167.99	68.78	133.32	107.31	133.08	399.81	398.31
DataRate (kbps)	155.06	147.83	60.53	28.80	23.18	28.75	86.36	86.03
Conn Interval (ms)	NA	NA	NA	7.5	7.5	7.5	7.5	7.5
TX slots	11008	10496	13982	7998	7998	7998	8006	8004
Packets sent	11008	10467	5148	7996	6435	7981	23997	23907
Packets received	10802	10368	4060	7996	6435	7981	23997	23907
PRR (%)	98.13	99.05	78.87	99.97	80.46	99.79		
PDR (%)	98.13	99.05	78.87	100	100	100	100	100
TX slot utilization (%)	100	99.72	36.82	99.64	81.94	99.95		
Source RDC (%)	99.52	99.42	99.73	26.31	25.55	26.38	44.11	45.36
Destination RDC (%)	100	100	100	28.9	27.51	28.88	NA	NA
Packet size (Bytes)	110	110	110	27	27	27	27	27

Request-Response Test

Cases	ContikiMAC	NullRDC	7.5_0_r	7.5_0_i	7.5_65_r	7.5_65_i	125_0_r	125_0_i	125_3_r	125_3_i
Mean delay (ms)	89.9	24.2	14.24	13.9	735.79	15.58	187.38	195.06	752.22	186.96
Stan Dev (σ) (ms)	39.3	0.2	2.02	2.23	147.93	2.44	41.52	41.44	151.45	38.42
Max delay (ms)	159.7	25.4	17.19	24.08	1077.96	19.11	399.92	376.54	1431.17	273.71
Min delay (ms)	39.8	23.7	10.54	10.15	498.04	12.11	128.41	127.7	128.33	137.04
Source Radio On (%)	100	100	22.09	21.43	21.79	0.54	1.37	1.32	1.37	0.46
Destination Radio On (%)	1.3	100	21.44	22.07	0.6	21.5	1.32	1.37	0.54	1.32
Slave Latency (count)	NA	NA	0	0	65	65	0	0	3	3
Connection Interval (ms)	NA	NA	7.5	7.5	7.5	7.5	125	125	125	125
Supervision Timeout (ms)	NA	NA	1250	1250	1250	1250	1250	1250	1250	1250