A GENERIC WIRELESS MODULE FOR INDUSTRIAL APPLICATIONS

A Dissertation submitted in fulfillment of the requirements for the degree of

MASTER OF ENGINEERING

in

Electronic Instrumentation and Control Engineering

Submitted by

Manik Grover

(801351009)

Under the guidance of

Mr. Nirbhow Jap Singh
Assistant Professor,
EIED, Thapar University,
Patiala

Dr. Suraj Kumar Pardeshi
Technology Manager,
EDC, Global R&D Centre,
Crompton Greaves Ltd.,
Mumbai

Electrical And Instrumentation Engineering Department
Thapar University, Patiala
(Declared as Deemed-to-be-University u/s 3 of the UGC Act., 1956)
Post Bag No. 32, Patiala – 147004
Punjab (India)

July, 2015

DECLARATION

I hereby certify that the work which is presented in dissertation entitled, "A Generic Wireless Module For Industrial Applications", in partial fulfillment of the requirements for the award of the degree of Master of Engineering in Electronics Instrumentation and Control, submitted to Electrical & Instrumentation Engineering Department of Thapar University, Patiala is as authentic record of my own work carried under the supervision of Mr. Nirbhow Jap Singh and Dr. Suraj Kumar Pardeshi. It refers others researcher's work which are duly listed in the reference section. The matter contained in this dissertation has not been submitted, neither in part nor in full to any other degree to any other university or institute except as reported in text and references.

Place: PATIALA
Date: 15-07-2015

MANIK GROVER
Roll No.:801351009

It is certified that the above statement made by the student is correct to the best of my knowledge and belief.

Nirbhow Jap Singh
Assistant Professor,
EIED, Thapar University,
Patiala

Date: 15-07-2015

Dr. Suraj Kumar Pardeshi
Technology Manager,
EDC, Global R&D Centre,
Crompton Greaves Ltd.,
Mumbai

Countersigned by:

Dr. Ravinder Agarwal
Head, EIED
Thapar University, Patiala

Dr. S.S Bhatia
Dean (Academic Affairs)
Thapar University, Patiala

i

ACKNOWLEDGEMENT

Achievement of a goal requires austerity along with hard work. This work is the outcome of that only. Work like this would have never been completed without the cooperation of different individuals. I am speechless to express my profound thanks to all those who directly or indirectly helped me throughout the duration of this work. I take the opportunity to express my sincere adulation and gratitude to my esteemed and worthy supervisors Dr. Suraj Kumar Pardeshi, Technology Manager, Electronic Design Centre, Global R&D, Crompton Greaves Ltd., and Mr. Nirbhow Jap Singh, Assistant Professor, Thapar University, Patiala for their invaluable guidance, constructive suggestions and cooperation which led this thesis report in the present form. Their consistent motivation and inspiration resulted in the completion of this work on time. It has been a benediction for me to spend many opportune moments at Global R&D Crompton Greaves Ltd. under the guidance of perfectionist at acme of professionalism. This work is testimony to their activity, inspiration and ardent personal interest taken by them during the course of this work. I am also very thankful to Dr. Samsul Ekram, Deputy General Manager, Global R&D, Crompton Greaves Ltd., Mumbai and also the HR team of Crompton Greaves especially Mr. Sumit for his support. I want to extend my thanks to Dr. Ravinder Aggarwal, Head, Electrical and Instrumentation Department and Dr. R.S. Kaler, Dean Resource Planning and Generation, Thapar University, Patiala, for his support. My heartiest thanks to all those who blessed me success especially my parents and above all the Almighty whose help is unmentionable in words.

Manik Grover
801351009

TABLE OF CONTENTS

LIST OF TABLES

Page

LIST OF FIGURES

NOMENCLATURE

BT: Bluetooth.

BLE: Bluetooth Low Energy.

SIG: Special Interest Group.

PLC: Power Line Communication.

CAN: Controller Area Network.

RSSI: Received Signal Strength Indicator

AFH: Adaptive Frequency Hopping.

eSCO: Extended Synchronous Connections.

HCI: Host Controller Interface.

SSP: Secure Simple Pairing.

EIR: Extended Enquiry Response.

AMP: Alternative MAC/PHY.

BR: Bluetooth Radio.

EDR: Extended Data Rate.

LE: Low Energy.

Ms/s: Megasymbol Per Second.

Mb/s: Megabit Per Second.

FDMA: Frequency Division Multiple Access.

TDMA: Time Division Multiple Access.

IC: Integrated Circuit.

LSOSC: Low Speed Frequency Oscillator.

HSOSC: High Speed Frequency Oscillator.

RF: Radio Frequency.

RAM: Random Access Memory.

SPI: Serial Peripheral Interface.

ESD: Electrostatic Discharge.

MIFA: Meandered Inverted F Antenna.

IFA: Inverted F Antenna.

HWD: Half Wave Dipole.

ACL: Asynchronous Connection Less.

A2DP: Advanced Audio Distribution Profile.

AVRCP: Audio/Video Remote Control Profile.

OPP: Object Push Profile.

PBAP: Phone Book Access Profile.

C2DM: Cloud to Device Messaging.

GCM: Google Cloud Messaging.

NFC: Near Field Communication.

P2P: Peer to Peer.

HID: Human Interface Device.

HOGP: HID Over Gatt Profile.

MAP: Message Access Profile.

TDLS: Tunneled Direct Link Setup.

OSAL: Operating System Abstraction Layer.

HAL: Hardware Abstraction Layer.

API: Application Programmable Interface.

ATT: Attribute Protocol.

GATT: Generic Attribute Protocol.

GAP: Generic Access Profile.

UART: Universal Asynchronous Receiver Transmitter.

ADT: Android Development Tools.

IDE: Integrated Development Environment.

ADC: Analog to Digital Converter.

LCD: Liquid Crystal Display.

ABSTRACT

The aim of this work is to provide a generic solution for wireless industrial applications. Wireless communication is desired in a lot of industrial applications such as biomedical, smart grid, and a range of sports products. This dissertation report provides hardware and software solution which can be integrated with existing products and facilitates the development of new products with wireless compatibility. The power consumption is a key issue for devices running on batteries. The present work provides a low power, low cost solution for short range wireless communications in industrial scenarios. The range of wireless communication is a factor of significance for the selection of a particular technology. The important factors that decide transmission range are transmitting power, gain of antennas and receiver sensitivity. This report also provides overview of different antennas that are suitable for 2.4 GHz applications. A complete system for BLE has been implemented. The experimental results exhibit successful operation of developed system.

Chapter 1

INTRODUCTION

Automation is an important aspect of industrial processes. In conventional systems hardware technologies are used in industrial automation for sensors, actuators and drives. In complex automation systems wireless technology plays a significant role both for acquiring data and transferring control signals. In this dissertation a wireless approach using Bluetooth Low energy is presented for communication of data or signals. Bluetooth Smart has a distinct advantage of operating on coin cell batteries and promising longer battery lives. Bluetooth Smart can be very useful for products operating on coin cell batteries. A generic wireless module is developed which can be integrated with industrial products such as biomedical equipments, smart grid devices, sports equipments, industrial grade sensors, motors and other equipments. An android application is developed to communicate with the wireless module.

1.1 Wired Communication

Wired technologies include fiber optic cable, telephone line, power line communication (PLC), Controller Area Network (CAN) and pair cables [1]:

1. Power Lines are used for supplying electric power to devices. The same lines can also be used for communication. Signals are injected on the power lines for communication but as the power lines are noisy they have interference problems and not very reliable.

2. Fiber optic cables are employed in critical environments with hard real time constraints requiring large amounts of data to be communicated. Fiber optics is costly to employ.

3. The advantage of telephone line is its wide service all over the country. But its cost of installation and monthly maintenance is high.

4. CAN was originally designed for automotive applications but now it is also used in industrial automation. It is used as a fieldbus in general automation environments due to low cost of some CAN controllers and processors [2].

1.2 Wireless Communication

The various wireless technologies are listed as follows [3]:

1. ZigBee: The ZigBee protocol is based on IEEE802.15.4-MAC- and PHY-layers. It includes the networking and application layers. It is suitable for low data rate, low power and low cost requirements. It has low Quality-of-Service (QoS) guarantee, it is not industrial grade, does not support determinism, and does not employ frequency hopping making it susceptible to interference.

2. Wireless HART:Wireless HART is a global IEC-approved standard (IEC 62591) that specifies an interoperable self-organizing mesh technology in which field devices form wireless networks that dynamically mitigate obstacles in the process environment. This technology is not designed to operate on coin cell batteries.

3. IEEE802.11a/b/g: IEEE 802.11: IEEE 802.11 describes the specification of wireless LAN. Factors important to automation industry such as energy efficiency, real time operation, reliability are not given much importance by WLAN.

4. ISA100.11a : ISA100.11a is an open-standard wireless networking technology developed by the ISA100 Committee of the ISA. ISA100.11a defines the OSI stack, system management, gateway, and security specifications for low data rate wireless connectivity with fixed, portable, and moving field devices. ISA100.11a cannot be easily interfaced with a mobile phone.

5. Bluetooth: Bluetooth 4.0 holds the promises for devices operating on battery life as it is designed to be energy efficient. It operates in 2.4GHz ISM band. Typical range with Bluetooth can be 40-50 meters. The Figure 1.1 exhibit that BLE has the longest battery life and data rate higher than zigbee but lower than classic Bluetooth.

The goal of this dissertation is development of a generic wireless module for industrial applications. This work is further divided into following objectives:

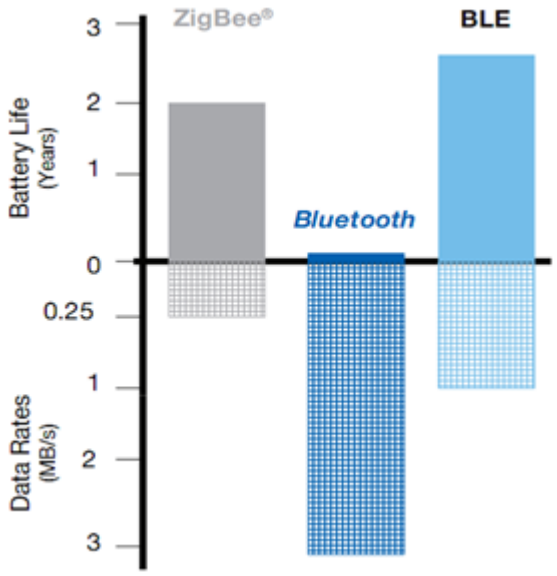

Figure 1.1: Comparison between Zigbee, Bluetooth and BLE [4].

1. Selection of BLE device.
2. Selection of 2.4 GHz antenna.
3. Schematic design for the selected device.
4. PCB design for the selected device.
5. Software development for BLE device.
6. Android application development.

Chapter 2

LITERATURE SURVEY

Bluetooth invented by Ericsson in 1994, is managed by Bluetooth special interest group (SIG) having more than 25000 member companies. The Bluetooth SIG oversees development of the specification, manages the qualification program, and trademark protection [5]. A manufactured product must meet Bluetooth SIG standards to market it as a Bluetooth device. Bluetooth has evolved a lot since its first implementation. Different versions of Bluetooth with their important features are listed in Table 2.1. The important terminology related to Bluetooth is explained as follows:

1. RSSI (dBm): RSSI refers to Received Signal Strength Indicator. It is a measurement of power present in a received radio signal and represents the power ratio of measured power in decibels referenced to one milliwatt. The relationship between RSSI and received power is given by Eq 2.1.

$$x = 10 \log_{10} \frac{P}{1mW} \tag{2.1}$$

   where x= measured RSSI, P= power received

2. AFH: It refers to Adaptive Frequency Hopping. This feature improves the resistance to radio frequency interference by avoiding use of crowded frequencies in hopping sequence [5]. Frequency Hopping Spread Spectrum (FHSS) is a method of transmitting radio signals by rapidly switching a carrier among many frequency channels, using a pseudorandom sequence known to both transmitter and receiver [6].

3. eSCO: eSCO refers to Extended Synchronous Connections. It improves voice quality of audio links by re-transmission of corrupted packets.

4. HCI: HCI refers to Host Controller Interface. UART as HCI was introduced in Bluetooth version 1.2. HCI provides a method of accessing the Bluetooth baseband capabilities. It provides a command interface to baseband controller, link manager and access to configuration settings [7].

5. SSP: SSP refers to Secure Simple Pairing. SSP strengthens the security and improves the pairing experience of Bluetooth devices [8].

6. EIR: EIR refers to Extended Enquiry Response. It provides more information during the enquiry procedure to allow better filtering of devices before connection [5].

7. Sniff Subrating: Sniff and Sniff-Subrating modes provide an effective means to reduce the power consumed by a pair of connected Bluetooth devices. SSR is particularly useful for devices that have periods of activity separated by long periods of inactivity. A computer mouse when used while working on a word processor is a good example of sniff subrating [9].

8. AMP: AMP refers to Alternative MAC/PHY. The Bluetooth radio is used for device discovery, initial connection and profile configuration. MAC PHY 802.11 (typically associated with Wi-Fi) is used when large quantities of data need to be transferred [5].

Table 2.1: Bluetooth versions

Version	Year	Important Features
1.0	1994	First Version [10].
1.1	2001	RSSI [11].
1.2	2003	AFH, 721Kbps data rate, eSCO, HCI UART [12].
2.0+EDR	2004	3 Mbps data rate [13].
2.1+EDR	2007	SSP,EIR, Sniff Subrating [14].
3.0+HS	2009	AMP, 24Mbps data rate [15].
4.0	2010	Classic BT + BT HS + BLE [16]
4.1	2013	Support for multiple roles, Co-existence for LTE and bulk data rates [17].
4.2	2014	Features for IOT, Data Length Extension, IP connectivity [18].

2.1 Bluetooth Smart Architecture

Bluetooth Smart is general-public face of Bluetooth Low Energy. Like the BR/EDR radio, the LE radio operates in the unlicensed 2.4 GHz ISM band. The LE system employs a frequency hopping transceiver to combat interference and fading and provides

many FHSS carriers. LE radio operation uses a shaped, binary frequency modulation to minimize transceiver complexity. The symbol rate is 1 Megasymbol per second (Ms/s) supporting the bit rate of 1 Megabit per second (Mb/s) [16] . LE employs two multiple access schemes: Frequency division multiple access (FDMA) and time division multiple access (TDMA). Forty (40) physical channels, separated by 2 MHz, are used in the FDMA scheme. Three (3) are used as advertising channels and 37 are used as data channels. A TDMA based polling scheme is in used in which one device transmits a packet at a predetermined time and a corresponding device responds with a packet after a predetermined interval [16]. Bluetooth Core System architecture is shown in Fig. 2.1.

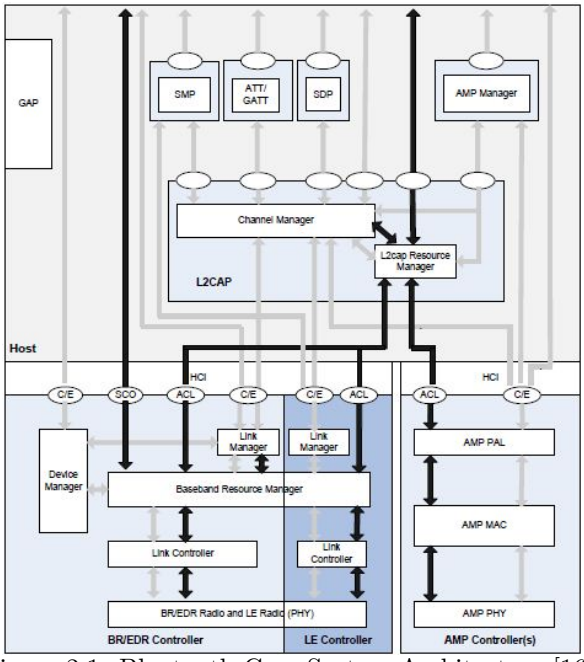

Figure 2.1: Bluetooth Core System Architecture [16].

2.2 Study of various bluetooth devices

Bluetooth Low Energy devices are available from a number of vendors that vary in their hardware and software features. Some of the BLE ICs are studied in this work. **CC2540** is a single chip solution which can run both BLE stack and application. It is integrated with 8051 core. Software development for CC2540 requires IAR embedded Workbench for 8051. BLE protocol stack is provides as a library file [19]. **STBLC01** is a very low power Bluetooth low energy (BLE) controller compliant with Bluetooth specification 4.0. The STBLC01 integrates a low power physical layer, a link layer with an embedded security engine, a host controller interface (HCI), and power manage-

ment [20]. **BLUENRGQTR** is Bluetooth low energy network processor. It requires an external microcontroller for application while BLE stack runs on BLUENRGQTR [21]. **CC2541** is another IC provided by Texas instrument for BLE. The features of different BLE devices are compared in Table 2.2.

2.2.1 Clock and Modes of BLE devices

BLE devices use number of oscillators for optimum performance. For example BlueNrg uses four oscillators, two low speed frequency oscillators (LSOSC) and two high speed frequency oscillators (HSOSC). The values of these oscillators are given in Table 2.3. In different modes of BLE devices different oscillators are used. For example BlueNRG can be in one of the five modes. These modes are explained below [21]:

1. Reset Mode: Reset mode is ultra low power consumption mode which is entered by external reset signal. After reset device goes to active mode. In reset mode all voltage regulators, clocks and RF interface are off.

2. Sleep Mode: In sleep mode Low speed crystal or ring oscillator is on while high speed oscillators, and RF interface are off. The state of BlueNrg and RAM is retained.

3. Standby Mode: Stand-by mode is similar to SLEEP mode except that in Stand By mode low speed oscillators are also powered down. BlueNrg can be activated through SPI interface in Stand-by mode.

4. Active Mode: In Active mode high speed oscillator, SPI, RF interface and MCU core are On.

5. Radio Mode: Radio Mode is like active mode except that in Radio mode RF transmitter is also active and it is either transmitting or receiving.

Table 2.2: Features of different BLE devices.

Device	**Company**	**Price (for 100 units)**	**USART**	**USB**	**Uc**	**Adc**	**timer**	**dma**	**Ram (Kb)**	**Rom (Kb)**	**i/o pins**	**I2c**
cc2540	TI	4.325	2	1	8051	12 bit 8-channels	16 bit-1, 8bit-2	5	8	128 Or 256	21	NA
cc2541	TI	4.34	2	NA	8051	12 bit 8-channels	16 bit-1, 8bit-2	5	8	128 Or 256	23	1
STBLC01	ST	4.29	1	NA	NA	NA	NA	NA	NA	NA	NA	NA
BlueNRG	ST	2.82	SPI		Cortex-M0 32 bit	NA	NA	NA	12	64	6	NA
ble113	Bluegiga	14.98	2	NA	8051	12 bit	4	5	8	128or 256	21	1

2.2.2 Bluetooth Low Energy devices: A comparison

The different devices are compared (Table 2.3) on the basis of various operational parameters. Active-Mode TX refers to the current required by the device when it is transmitting at transmit power of 0 dbm. Similarly Active-Mode RX parameter refers to the current when device is receiving in standard gain mode. ESD refers to electrostatic discharge. ESD is the current that flows between two differently charged objects. One object has excess of electrons and other lack them. Electrostatic discharge may occur through direct contact, electric short or dielectric breakdown. ESD can occur when a charged body touches and IC, a charged IC touches a grounded surface, a charged machine touches an IC or due to an Electrostatic field [22]. Spurious emissions are unintended out of the bandwidth signals that result from harmonics, intermodulation, frequency conversion and Electromagnetic interference [23][24]. The comparison of Bluetooth devices (Table 2.4) shows that maximum transmit power of CC2540 is better than all except BlueNrg but CC2540 contains 8051 microcontroller which can be used along with peripherals and BLE controller as a single device. But with BlueNRG we need an external microcontroller. It is for this reason that CC2540 has been chosen for implementing the BLE system.

Table 2.3: Comparison of different BLE devices

Parameter	CC2540	BLUENRG	CC2541
Active-Mode TX (0 dBm)	27 mA	8.2 mA	18.2 mA
Active-Mode RX	19.6 mA	7.3 mA	17.9 mA
RF frequency range	2402 - 2480 Mhz	2400 - 2483.5 Mhz	2379 - 2496 Mhz
Channel spacing	2 Mhz	2 Mhz	1 Mhz
Data rate	1 Mbps	1 Mbps	250 Kbps, 500 Kbps, 1 Mbps, 2 Mbps
Operating Supply Voltage	2-3.6 V	2-3.6 V	2-3.6 V
Voltage on any digital pin	-0.3 V - Vdd+0.3 <= 3.9V	-0.3 - 3.9 V	-0.3 V - Vdd+0.3 <= 3.9V
ESD (all pins)	2 KV	2 KV	1 KV
Oscillators	Crystal- 32 Mhz, 32.768 Khz RC- 32 Khz, 16 Mhz	Crystal- 16/32 Mhz, 32.768 Khz Ring- 16 Mhz, 37.4 Khz	Crystal- 32 Mhz, 32.768 Khz RC- 32 Khz, 16 Mhz
Spurious emission	-41 dbm	-41 dbm	-48 dbm
Optimum load impedance	70 + j30	25.9 + j44.4	70 +j30
Saturation	6 dbm(1 Mbps, GFSK,250 kHz deviation)	8 dbm(1 Mbps, GFSK, 150 kHz deviation)	5 dbm(1 Mbps, GFSK, 250-kHz deviation)

Table 2.4: Power comparison of different bluetooth devices.

Device	**Min Transmit Power**	**Max Transmit Power**	**Receiver Senstivity**	**Units**
Cc2540	-23	4	-93	dbm
Cc2541	-23	0	-90	dbm
STBLC01	-18	3	-80	dbm
BlueNRG	-18	8	-88	dbm
ble113	-23	0	-93	dbm

2.2.3 Study of Bluetooth antennas

Antennas used for the Bluetooth can be categorized into PCB antennas, Chip antennas or Whip antennas. PCB antennas are printed on the PCB and are least costly. Chip antennas come like an IC package while whip antennas are external antennas such as Lambda/2 antennas. A comparison of different antennas for Bluetooth is shown in Table 2.5.

Table 2.5: Gain comparison of different antennas.

Antenna	***Type***	***Average gain(dbi)***	***Dimensions(mm)***
Meandered Inverted F Antenna	Single-ended	5.033	15.2x5.7
Inverted F Antenna	Single ended	2	25.7x7.5
Half Wave Dipole Antenna	Differential	7.16	44.6
Folded Dipole Antenna	Differential	6.5	46x9
Yagi PCB antenna	Directional	7.22	100x150
W3008	Chip	1.7	7x3x2

Description of Bluetooth antennas

The 'Meandered Inverted F Antenna' abbreviated as **MIFA** requires 15.2 x 5.7 mm of pcb space and has a gain of about 5 dbi. It is a single ended antenna which ensures VSWR of less than 2 across the 2.4 GHz ISM band and impedance of 50 ohm so that no additional components are required when connected to a 50 ohm source [25]. A MIFA is shown in Fig.2.2 with its dimensions in Table 2.6 [25].

The 'Inverted F Antenna' **(IFA)** has a gain of 2 dbi occupying 25.7 x 7.5 mm of pcb space. The IFA is shown in Fig.2.3 with its dimensions in Table 2.7.

The 'Half Wave Dipole Antenna' **(HWD)** is a differential antenna with gain of about 7 dbi. This antenna can be used without a balun. A HWD antenna is shown in Fig.2.4 [27]. This antenna is composed of two arc segments. Each segment has a radius of 20mm with 300 um trace thickness. The angular length is 64 degrees corresponding

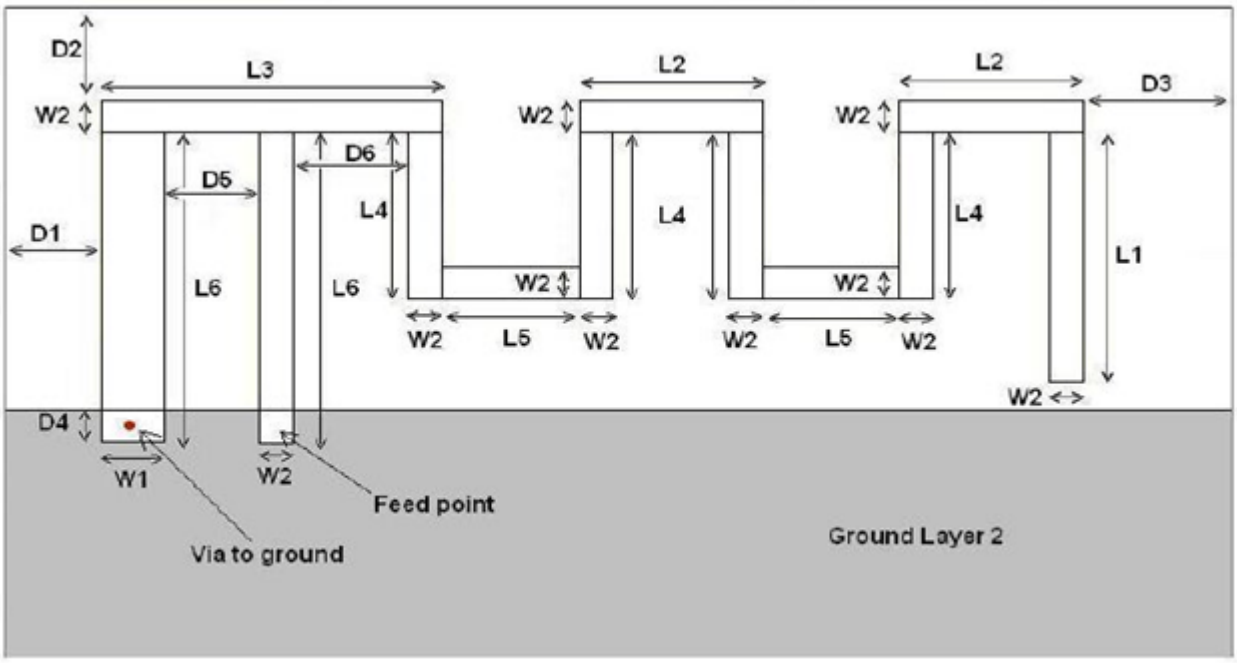

Figure 2.2: Meandered Inverted F antenna [25]

Table 2.6: Dimensions of Meandered Inverted F antenna

Label	Length(mm)
L1	3.94
L2	2.70
L3	5.00
L4	2.64
L5	2.00
L6	4.90
W1	0.90
W2	0.50
D1	0.50
D2	0.30
D3	0.30
D4	0.50
D5	1.40
D6	1.70

to 22.3 mm of arc length and 44.6 mm of total antenna length [27].

The 'Folded Dipole Antenna' **(FDA)** is another differential antenna with average gain of 6.5 dbi. It requires 46 x 9 mm of pcb space. The FDA is shown in Fig.2.5 with its dimensions in Table 2.8.

Yagi PCB antenna is a directional antenna with average directional gain of about 7.22 dbi. It can be used with any 2.4 Ghz radio design. Fig.2.6 shows Yagi PCB antenna with its dimensions in Table 2.7. A Yagi PCB antenna board occupies 100 x 150 mm of space.

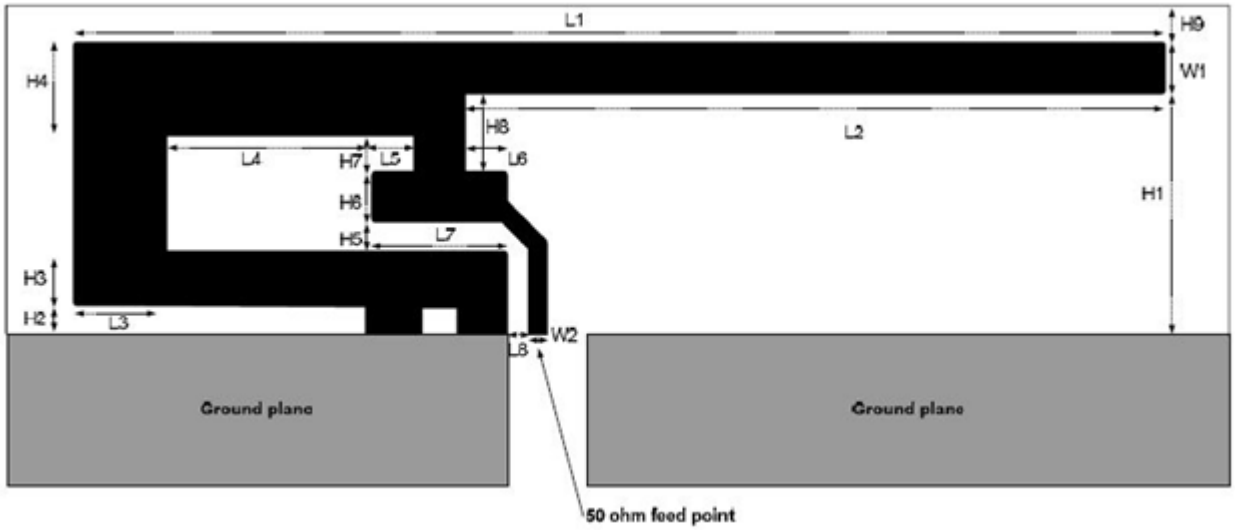

Figure 2.3: Inverted F antenna [26].

Table 2.7: Dimensions of Inverted F antenna

Label	Length(mm)	Label	Length(mm)
H1	5.70	W2	0.46
H2	0.74	L1	25.58
H3	1.29	L2	16.40
H4	2.21	L3	2.18
H5	0.66	L4	4.80
H6	1.21	L5	1.00
H7	0.80	L6	1.00
H8	1.80	L7	3.20
H9	0.61	L8	0.45
W1	1.21		

Table 2.8: Dimensions of Folded Dipole antenna.

Label	Length(mm)	Label	Length(mm)
L1	44.7	H1	2.4
L2	21.0	H2	3.1
D1	1.5	H3	6.0
D2	0.9	H4	2.8

Chip antenna is a ready to use packaged antenna. Chip antennas are available from a number of vendors these days. Specifications of Fractus Compact Reach Xtend Antenna are shown in Table 2.9 . The features of Fractus Compact Reach Xtend Antenna are:

1. High efficiency
2. Small form factor
3. Cost-effective

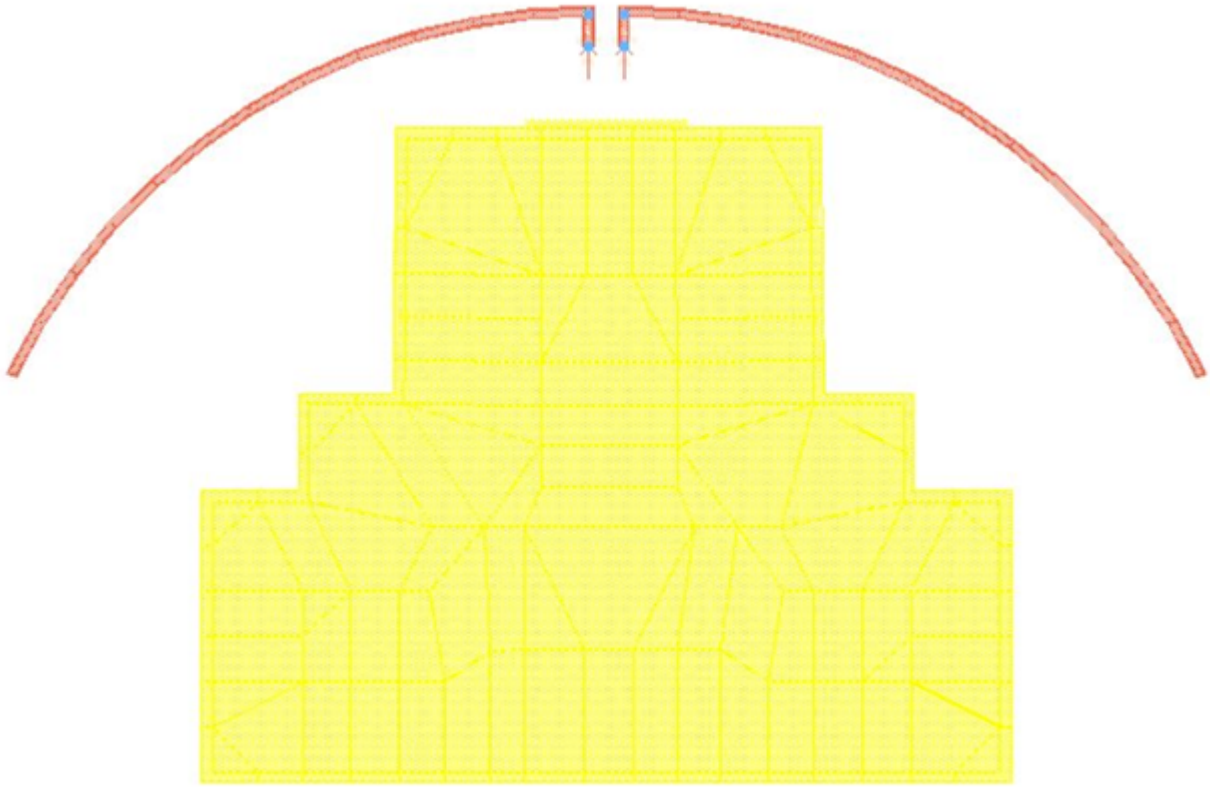

Figure 2.4: Half Wave Dipole antenna [27].

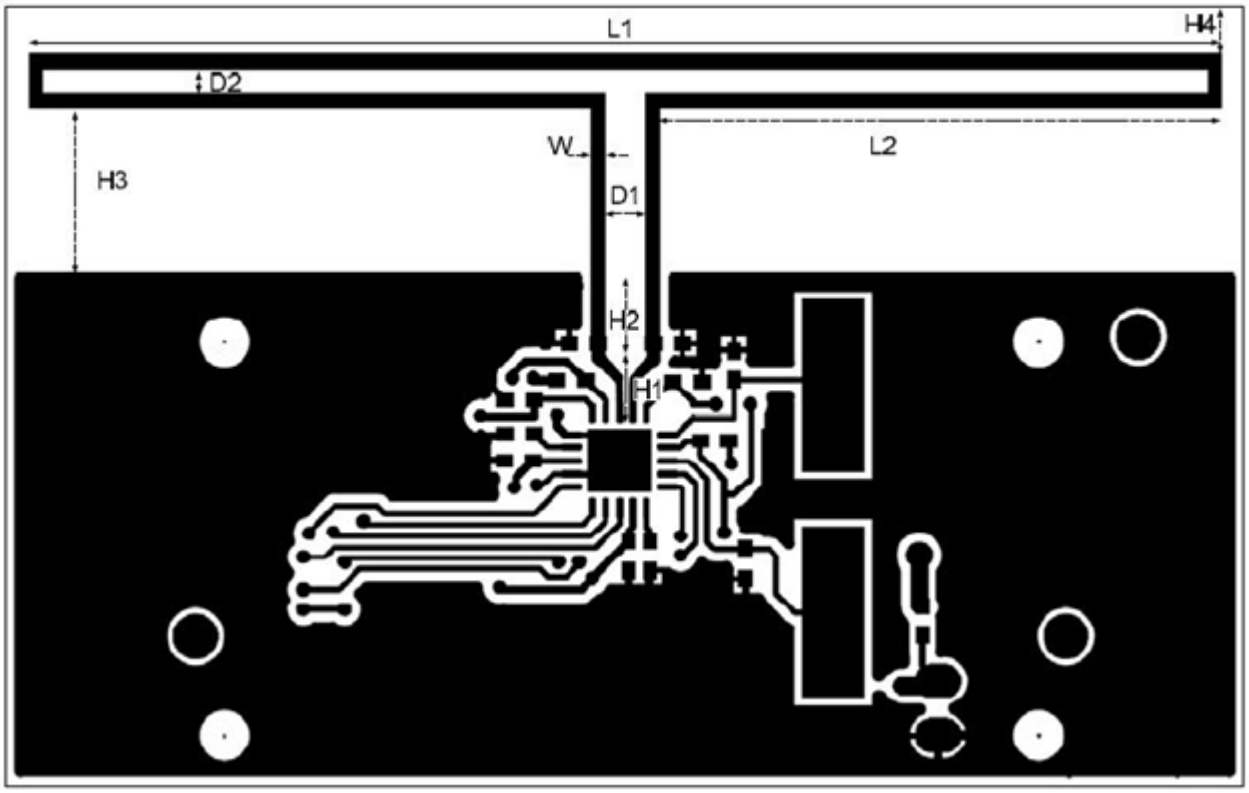

Figure 2.5: Folded Dipole antenna [28].

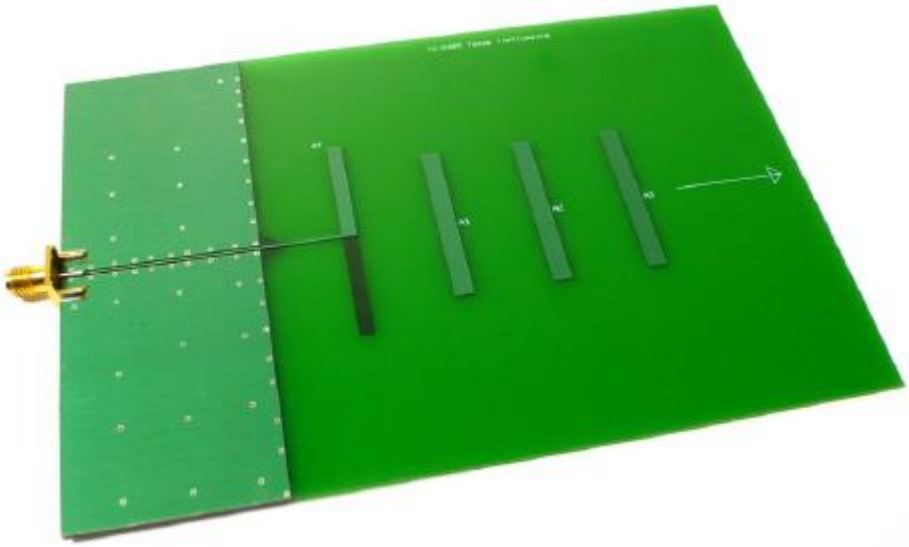

Figure 2.6: Yagi PCB antenna [29].

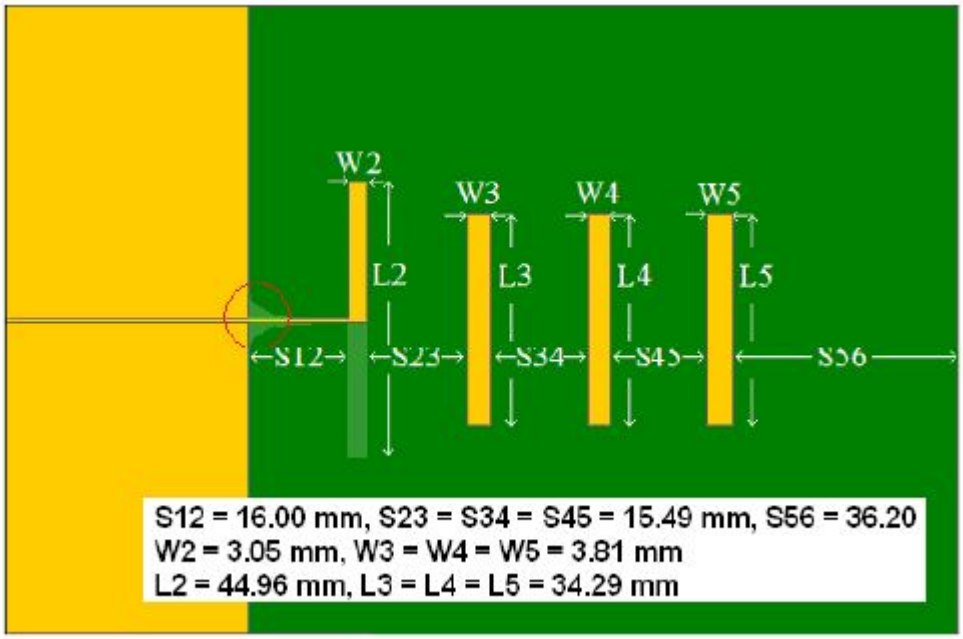

Figure 2.7: Dimensions of Yagi PCB antenna [29].

4. Small clearance needed

5. Easy-to-use

2.3 Android Versions

Android is a linux kernel based mobile operating system. Android 1.0 was released in September, 2008 and the recent version Android 5.0 was released in November, 2014 [30]. Different Android versions with their year of release and important features are listed in Table 2.10. Following terms are explained in reference to Table 2.10:

Table 2.9: Specifications of Fractus Compact Reach Xtend antenna [31].

Parameter	**Specification**
Frequency Range	2400-2500 MHz
Radiation Efficiency	$> 50\%$
Peak Gain	> 0 dBi
VSWR	$< 2{:}1$
Polarization	Linear
Weight	0.1 g
Temperature	-40 to + 85°C
Impedance	50O
Dimensions	7x3x2 mm

1. ACL: It refers to Asynchronous Connection less. It is the normal type of radio link used for general data packets using a polling TDMA scheme to arbitrate access. ACL packets are retransmitted automatically if unacknowledged, allowing for correction of a radio link that is subject to interference [32].

2. A2DP: The Advanced Audio Distribution Profile (A2DP) defines the protocols and procedures that realize distribution of audio content of high-quality in mono or stereo mode on ACL channels [33].

3. AVRCP: AVRCP stands for Audio/Video Remote Control Profile. AVRCP is designed to provide a standard interface to control TVs, Hi-Fi equipment, or others to allow a single remote control (or other device) to control all the A/V equipment to which a user has access [34].

4. Object Push Profile (OPP): OPP defines the roles of push server and push client. It is used for exchange of a contact or appointment between two mobile phones or a mobile phone and a PC. Push server is a device that provides an object exchange server. Push client pushes and pulls objects to and from the Push Server [45].

5. PBAP: It stands for Phone book Access Profile. The Phone Book Access Profile (PBAP) defines the protocols and procedures that shall be used by devices for the retrieval of phone book objects. This profile is used to access phone book objects in a car environment with hands free mode [46].

6. C2DM: C2DM stands for Cloud to Device Messaging. Android C2DM is now known as Google Cloud Messaging(GCM). It is a service that enables developers

Table 2.10: Android versions

Version	Year	Important Features
1.0 (Alpha)	September 2008	Gmail, Google contacts, Google maps, Google Search, Google talk and Google calendar Synchronization [30].
1.1 (Beta)	February 2009	Updates to MAPS, Dial pad and MMS services [35].
1.5 (Cupcake)	April 2009	Stereo BT support: A2DP and AVCRP profiles [36].
1.6 (Donut)	September 2009	Gesture framework [37].
2.0 (Eclair)	October 2009	BT 2.1, New BT profiles: Object Push Profile (OPP) and Phone Book Access Profile (PBAP) [38].
2.2-2.2.3 (Froyo)	May 2010	Android Cloud to Device Messaging (C2DM) service, USB tethering, Wifi hotspot, BT enabled car and desk docks [30].
2.3-2.3.2 (Gingerbread)	December 2010	API support: gyroscope, rotation vector, linear acceleration, gravity and barometer sensors, Khronos OpenSL ES [39].
3.0 (Honeycomb)	February 2011	Multitasking, Built-in support for Media/Picture transfer protocol [40].
4.0-4.0.2 (Ice Cream Sandwich)	October 2011	Android Beam for NFC data sharing [41].
4.1 (Jelly Bean)	July 2012	Support for Wifi P2P [42].
4.4 (Kit Kat)	September 2013	Secure NFC transactions, BLE, HOGP, BT MAP, IR blasters, Wifi tunneled direct link setup (TDLS) [43].
5.0-5.0.2 (Lollipop)	November 2014	Concurrent operations with BLE, Tilt detector and heart rate sensor, wake up, pick up and glance gestures recognition [44].

to send data from servers to both Android applications or Chrome apps and extensions. This service provides a mechanism that tells android app about the updated application or user data on the server [47].

7. OpenSL ES: OpenSL ES is a native language application-level audio API for embedded mobile multimedia devices. It provides a device independent cross platform interface for applications to access a device's audio capabilities. Features provided by OpenSL ES include playback of audio and MIDI, Effects and controls, Advanced MIDI, 3D audio [48].

8. NFC: NFC stands for Near Field Communication. It is a set of short range wireless technologies that require a distance of 4cm or less to initiate a connection. NFC allows data sharing between two Android powered devices or between an NFC tag and android device [49].

9. Wi-Fi P2P: Wi-Fi P2P is available in devices with minimum Android 4.0. It allows the devices to share data using Wi-Fi link. This feature can be useful for multiplayer games, photo sharing application and general data sharing [50].

10. HID: The Human Interface Device (HID) profile specification defines the protocols, procedures, and features to be used by Bluetooth HID devices and Blue-

tooth HID Hosts. A Bluetooth HID device can be a mice, keyboard, gamepads and sensors. A Bluetooth HID host can be a computer, gaming console and industrial machine [51].

11. HOGP: The HID over GATT profile defines the procedures and features to be used by Bluetooth low energy HID Devices using GATT and Bluetooth HID hosts using GATT [52].

12. Bluetooth MAP: Bluetooth Message Access Profile is used for communication between different devices in scenarios such as in Car, used for porting call and message information to inbuilt speakers . It is also used to send/receive text messages between a Palm/HP smartphone to an HP TouchPad tablet [32].

13. Wi-Fi TDLS: Wifi Tunneled Direct Link Setup is used to transfer data between devices that are on the same wifi network. It can be used for video streaming, data syncing and data transfer [53].

Chapter 3

PROBLEM FORMULATION AND SOLUTION APPROACH

The present work aims to develop a generic wireless module which can be easily integrated with existing and new embedded products. The various steps involved in implementation of system are (i) Selection of BLE device. (ii) Selection of 2.4 GHz antenna. (iii) Schematic design for the selected device. (iv) PCB design for the selected device. (v) Software development for BLE device. (vi) Android application development. The whole work has been divided into various parts as shown in Fig. 3.1.

Among wireless communication technologies (section 1.3), it is found that BLE is well suited for devices operating on coin cell batteries. BLE devices available from major vendors are compared on the basis of various technical aspects (section 2.2.2). It is found that transmit power and receiver sensitivity decides the maximum range available with the device. CC2540 is chosen for the hardware implementation and further study.

Bluetooth has broad spectrum of applications out of which household electronics, biomedical devices and process control equipment are significant. Among the features of Bluetooth, one of the areas of concern is power consumption of Bluetooth modules. The objective of this work is to develop a generic wireless module which can be easily interfaced (Fig. 3.2) with other electronic modules and has low energy consumption. To successfully implement this objective Bluetooth low energy technology has been chosen as the preferred solution. On the basis of literature survey (Section 2.2.2), the CC2540 IC is one of the possible option for implementation. In order to add Bluetooth functionality, BLE hardware is interfaced with the embedded product which is capable of communicating with other electronic devices that support BLE (Fig. 3.2). An application is developed for a smartphone to communicate with the BLE hardware.

The solution methodology is divided into three sections hardware implementation, software implementation and android application development. All these components are detailed in the following sections.

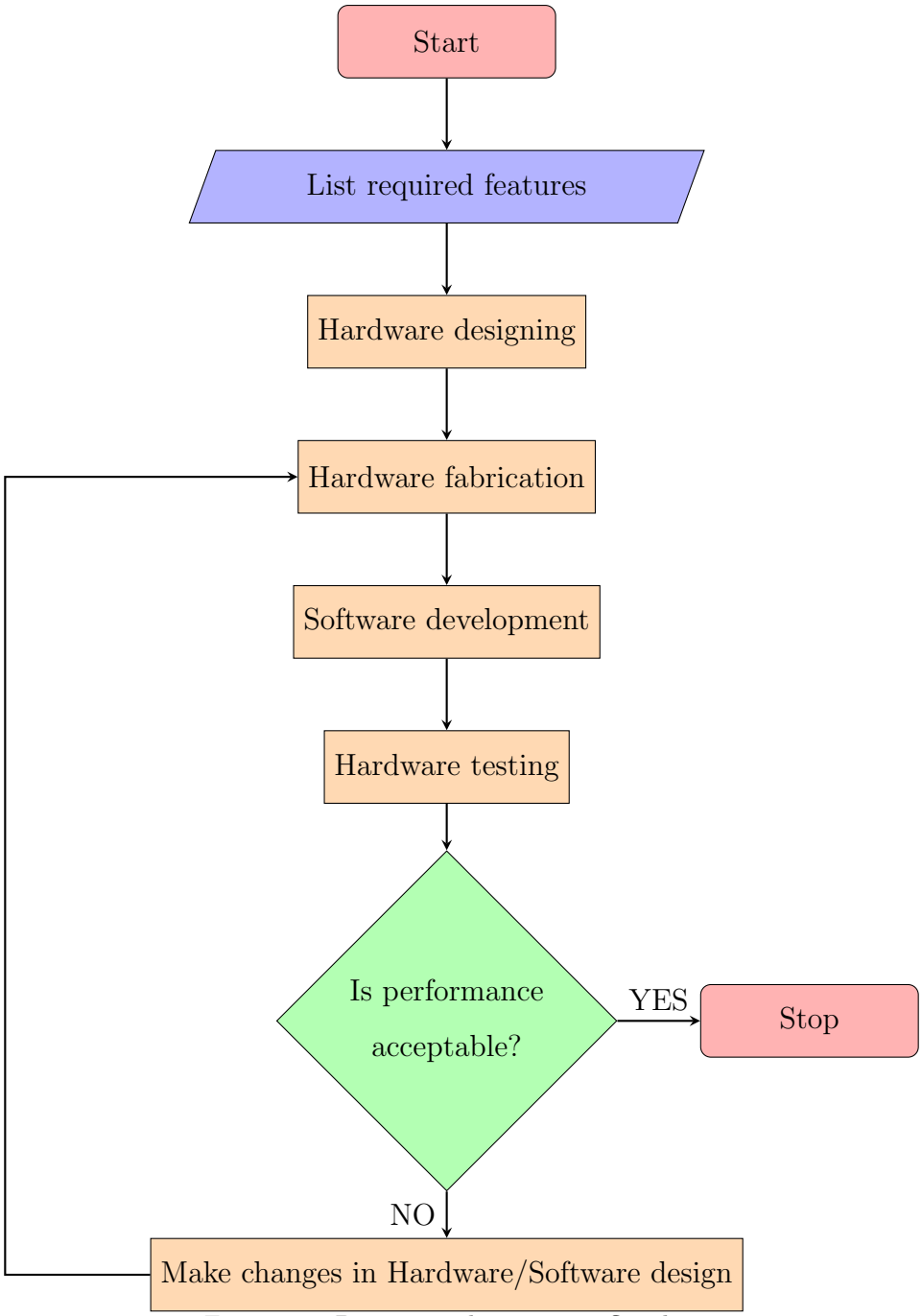

Figure 3.1: Project implementation flowchart

3.1 Hardware Implementation

The hardware implementation of the circuit for above stated objectives include fabrication of supporting components around the CC2540 Bluetooth device. It is observed that CC2540 has the maximum transmit power (Table 2.4) and comes integrated with

an 8051 microcontroller (Table 2.2). The factors that determine the device selection for BLE are transmission range, initial cost, total power consumption, software tools support, inbuild microcontroller feature and peripherals provided by it (Fig. 3.3). The important points considered in device selection are as follows:

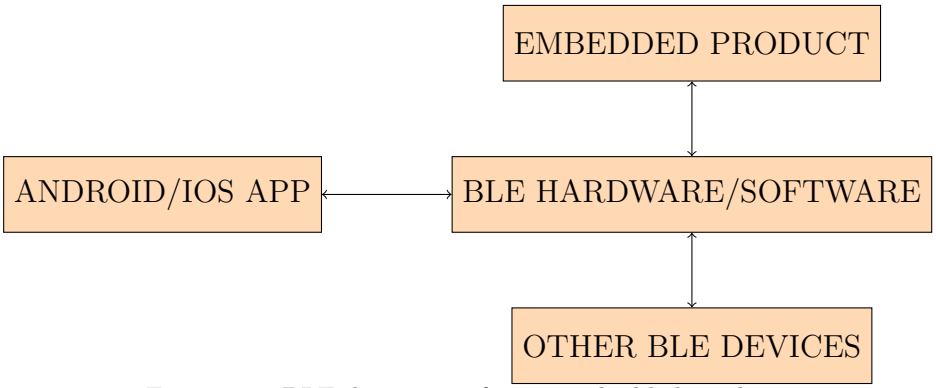

Figure 3.2: BLE device interface to embedded product.

1. The devices such as ST Bluenrg implement Bluetooth stack in the BLE IC requiring an external microcontroller to implement BLE profiles. On the other hand Texas CC2540 has integrated microcontroller to implement BLE profiles.

2. The Power Consumption is a key factor for devices running on coin cell batteries. Power consumption of BLE device depends on transmit Power, hardware and software design.

3. The range of a BLE device is decided by transmission power, antenna gain and environmental conditions. BlueNrg has maximum transmit power (Table 2.4) but it doesnot have inbuild microcontroller for BLE application.

4. The cost of a BLE device depends upon available peripherals and features. A comparison of various devices on the basis of available peripherals and cost has been discussed (section 2.2). It is observed that BlueNrg has the lowest cost but it doesnot have inbuild microcontroller for BLE applications, thus requiring large pcb space for hardware implementation.

5. The choice of BLE device depends upon the peripherals required if using standalone solution. The peripherals available with different BLE devices is given in Table 2.2.

6. The software support tools is another deciding factor for selection of BLE device.

Before designing one must ensure that the adequate and affordable software support is available for the chosen device.

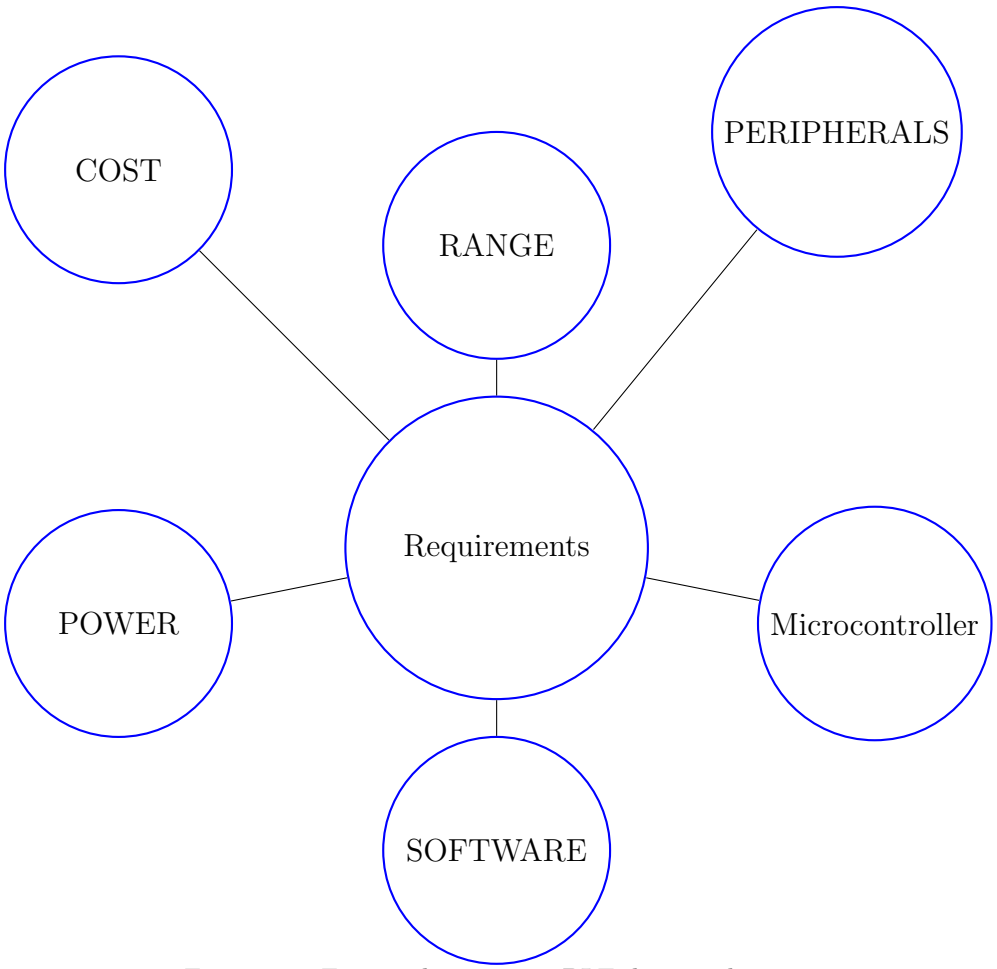

Figure 3.3: Factors determining BLE device selection.

As per above discussion the factors such as transmission range, initial cost and pcb size is a major concern in product development. The CC2540 has an inbuild microcontroller, thus saving pcb space and provides best range in class of BLE devices with inbuild microcontroller. The minimum connections required for CC2540 are reproduced in Fig. 3.4 [19]. CC2540 requires two external crystal oscillators (Section 2.2.1), a bias resistor for reference current when using analog I/O, a balun circuit for 50 ohm impedance matching, decoupling capacitors and power supply rails. A PCB is designed for CC2540 as shown in Fig. 3.5. The design includes two pcb antennas 'Meandered Inverted F Antenna' and 'Half Wave Dipole Antenna'. The relevance of adding two antennas is discussed in Chapter 5. A UART port is provided for data transfer between BLE hardware and embedded product. BLE hardware on receiving data through UART port, transmits it to other BLE devices or an android smart phone.

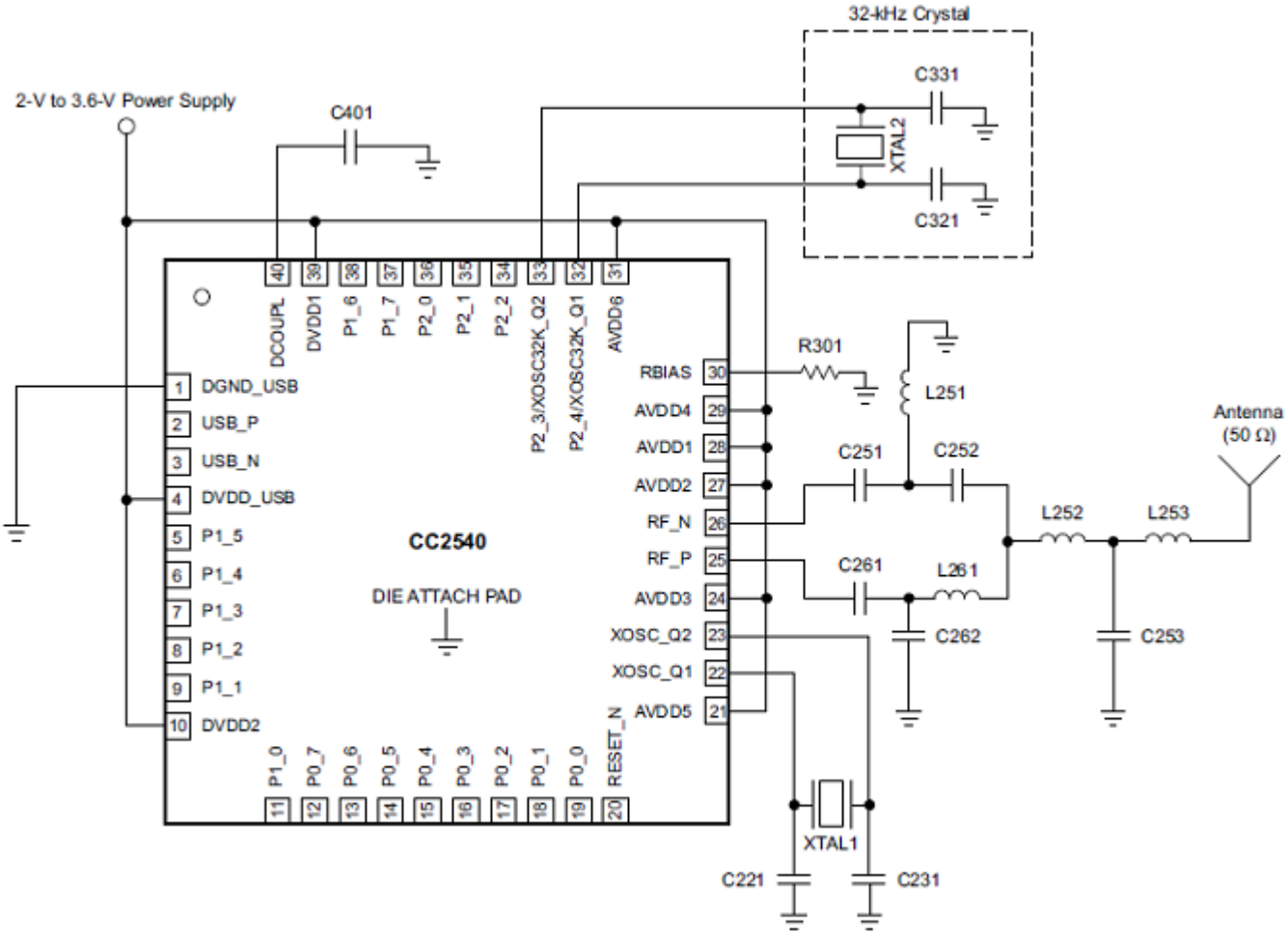

Figure 3.4: Minimum connections for CC2540 [19]

3.1.1 Antenna Selection

Antenna selection is a crucial design parameter in wireless hardware development. Table 2.5 shows the different antennas available for Bluetooth implementation. A single ended antenna requires impedance matching requiring balun for this purpose. A balun consists of discrete components or an integrated package. A differential antenna doesnot require balun thus saving pcb cost and size. The first five antennas shown in Table 2.5 are pcb antennas i.e. they can be implemented as a pcb layout. The last antenna is a chip antenna which can be used as an IC package.

A 'Meandered Inverted F Antenna' (MIFA) and 'Half Wave Dipole Antenna' (HWD) are chosen for implementation in this work. It is observed that among single ended antennas MIFA has maximum gain of 5dbi and HWD antenna has gain of 7 dbi, maximum among differential antennas (Table 2.5). Design of MIFA and HWD antenna are given in Fig. 2.2 and Fig. 2.4 respectively.

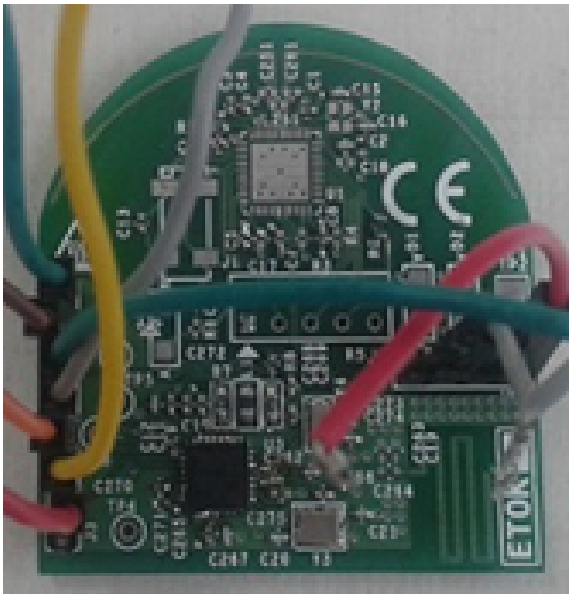

Figure 3.5: PCB design of CC2540

3.2 Software Implementation

The finished pcb when given 3.3V power supply can be used for Bluetooth connection, disconnection and data transfer according to the software programmed in CC2540. The software architecture of CC2540 is built around real time operating system. Fig. 3.6 shows the software architecture for CC2540. After selecting the suitable BLE

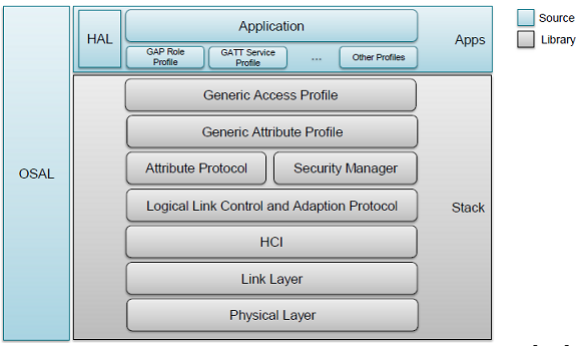

Figure 3.6: Software Architecture of CC2540 [54].

device, a schematic is designed according to the specified requirement. A PCB is designed, fabricated and is ready for testing. In the implementation, BLE hardware is interfaced with other electronics using UART port available with CC2540. The electronic device transfers data to CC2540 through UART port. CC2540 transfers the received data to other BLE devices. Texas instruments provide API's that are useful in programming of CC2540. HAL API's are used to access different hardware peripherals of CC2540. OSAL API's are used to utilize operating system functions. A software is developed which is used to communicate with an android application.

Following points are important wrt software development for CC2540:

1. Use the HAL APIs for hardware peripherals.
2. OSAL APIs are used to set the event flags.
3. OSAL acts as a control loop that allows software to setup execution of events. User has to define different services for his application.

The software that is flashed in the BLE IC includes the BLE protocol stack and BLE application including BLE profiles. The BLE protocol stack is provided by the device manufacturer. The important terminology related to BLE software development is as follows [55] :

1. Attribute: Attributes are the basic elements used by server to send/receive data. Figure 3.7 shows the attribute representation:

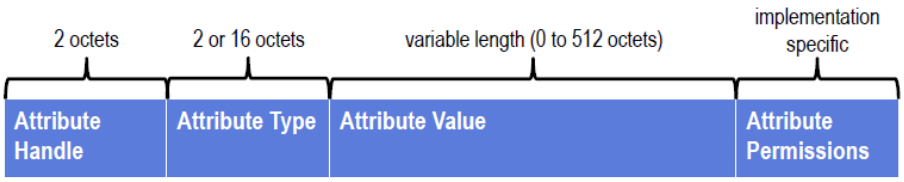

Figure 3.7: Attribute representation in BLE [56].

An attribute is composed of a handle, type, value and permissions. Attribute is an array of octets which can be of any length between 0 to 512 octets. Following points explain the different parts of an attribute [56].

(a) Attribute Handle: Each attribute has a handle which is used by a client to address it. Handle is a 16 bit value of which 0x0000 is reserved. The values between 0x0001 to 0xFFFF can be assigned to any attributes.Also the handles are to be used in a sequential manner.

(b) Attribute Type: Attribute type is a UUID which is a 16 bit or 128-bit assigned number. Attribute type defines how the value is used. It is defined in the GATT, GAP or GATT based profile, service or characteristic specifications.

(c) Attribute Permissions: Attribute Permissions are used to protect the Attribute values. The server may not reveal any values to the client, it doesnot trust enough. Attribute permissions are of following types:

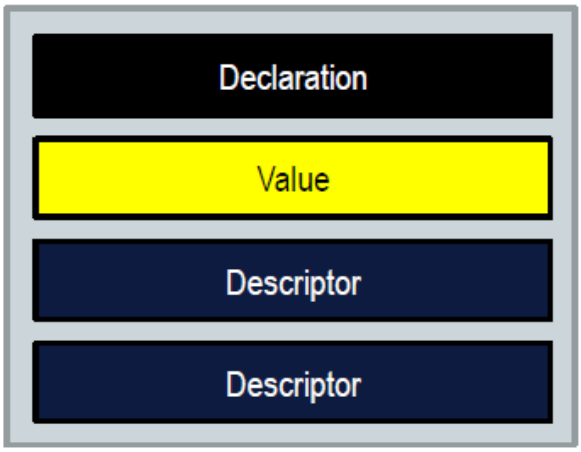

Figure 3.8: Characteristic specification [56].

i. Readable/not Readable

ii. Writable/not Writable

iii. Readable and Writable/not Readable and not Writable.

iv. Authentication to read/write.

v. Authorization to read/write.

vi. Encryption/pairing with sufficient strength to read/write.

2. Characterstic: Characteristic is value with a known type and format defined in characteristic specification. It includes declaration, value and descriptor fields. Fig.3.8 shows the characteristic specification. Descriptors define any additional information about the characteristics.

3. Service: Characteristics are grouped together to form a service. Each service serves a specific purpose. Some standard services include Link Loss Service, Immediate Alert Service.

4. Attribute Protocol (ATT): It is a protocol for discovering, reading and writing attributes between an attribute server and an attribute client.

5. Generic Attribute Protocol (GATT): GATT is build on top of ATT describing the hierarchy of services, characteristics and attributes used in the attribute server. It provide interfaces for discovering, reading, writing and indicating of service characteristics and attributes [56].

6. Generic Access Profile (GAP): GAP helps in discovering identities, names and

basic capabilities. It provides the functionalities such as creating bonds, exchange of security information and establishing connections.

The following text explains different parts of software implementation in CC2540.

3.2.1 HAL Drivers

HAL drivers are used to access different hardware peripherals of CC2540. For example sensor data can be read and converted to digital using ADC service, can be displayed on BLE device using LCD service and can be transferred to Android smartphone using BLE link. The services provided by HAL drivers are ADC service, LCD service, LED service, Key service, Sleep service, Timer service, UART service, PA/LNA service, I2C service and IR signal generation service. ADC service and LCD service are provided as reference in Appendix A.1. In software development we are using key service to toggle advertisement status of BLE hardware, UART service to read/write at UART port and timer service to provide time delays. The functions provided by different services of HAL drivers can be grouped into three categories [57]:

1. **Initialization function calls:**These function calls are used to initialize a service and /or to setup optional parameters for platform-specific data. Initialization functions are often called at the beginning stage when the device powers up.

2. **Service access function calls:**These function calls can directly access hardware registers to get/set certain value of the hardware (i.e. ADC) or control the hardware components (i.e. LED) .

3. **Callback function calls:**These functions must be implemented by the application and are used to pass events that are generated by the hardware (interrupts, counters, timers) or by polling mechanism (UART poll, Timer poll) to upper layers. Data accessed through callback function parameters (such as a pointer to data) are only valid for the execution of the function and should not be considered valid when the function returns. If these functions execute in the context of the interrupt, it must be efficient and not perform CPU-intensive operations or use critical sections.

3.2.2 OSAL APIs

The OS abstraction layer is used to shield the TI stack software components from the specifics of the processing environment. The operation of OSAL is shown in Fig. 3.9. Each subsystem of the software is assigned a task ID. A task array contains

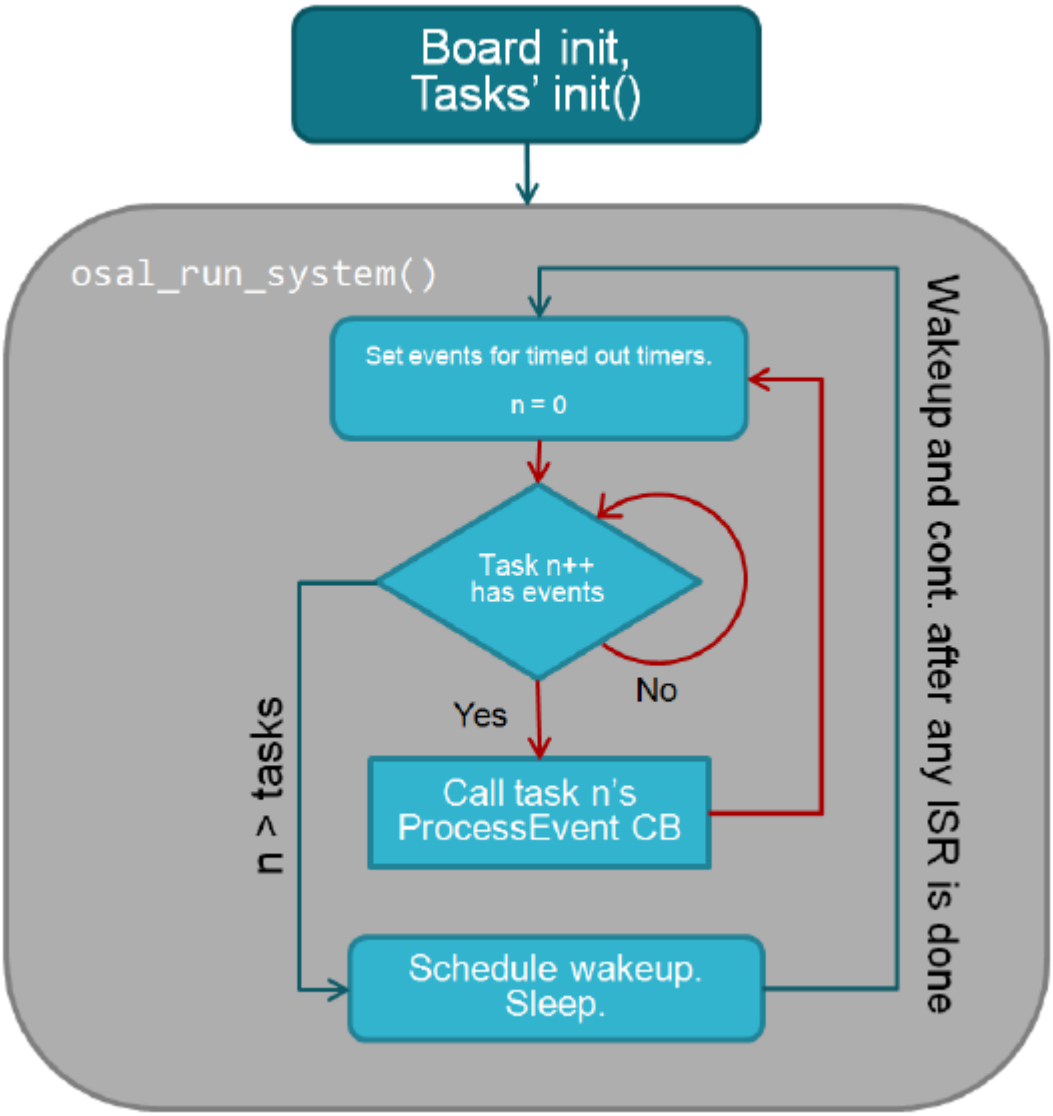

Figure 3.9: OSAL operation [58].

the pointers to event handler functions of every task. A task event array is used to track the various events within a task. Callback functions are provided for various events which ease the programming. It provides functionalities such as task registration, initialization, synchronization, interrupt handling, timers, message exchange and memory allocation. According to these functionalities OSAL API's are grouped into categories that are message management, task synchronization, timer management, interrupt management, task management, memory management, power management,

non volatile memory, osal clock system and osal misc APIs [59]. Two examples from each of the above category are presented in Appendix-B.

After the board, hardware, drivers and OSAL initialization, osal control loop starts running waiting for the interrupts and expired timers. One bit is reserved for each task event in a 16 bit variable. An array of pointers called tasks array holds the addresses of the event processing routines for each task. When an event of a task is detected by the OSAL, the function pointed by the tasks array is called and the event is served. The one event flag called SYS_EVENT_MSG is reserved for message passing between tasks and cannot be defined by the application.

In this work our BLE hardware is taking data through UART from other electronic devices and transferring it to android mobile. After a UART receive timeout interrupt is detected by the OSAL, it transfers the program execution to UART call back function. The UART call back function reads the data placed in the UART buffer and writes it to the characteristic value of a service. This characteristic value is read by an android application and displayed on the screen of an android device. The flowchart outlining this process is shown in Fig. 3.10. Android application development is explained in next section.

3.3 Android Application Development

Android application is needed to make BLE hardware communicate with a smart phone. Smart Phone can be used to send commands and data to remote BLE device. Android Studio is needed for android app development or Eclipse IDE with android ADT bundle serves the purpose. Layout is designed in XML while dynamic actions are programmed in JAVA. One should be familiar with XML and JAVA for android app development. Android is providing BLE APIs for android version 4.3 and above [60]. A flowchart outlining the process of android application development is shown in Fig. 3.11. The important stages in android application development are explained below [60]:

1. When user clicks on the icon of the application, it should start only if Bluetooth is supported by the device. If Bluetooth is not supported a message “Bluetooth not supported” should be displayed on the user screen. If Bluetooth is supported it should On the Bluetooth with user permission and display the startup screen.

2. The startup screen contains the scan option which scans for nearby BLE devices. On successful scanning a list of near by Bluetooth devices is displayed on user screen. The user is able to select a BLE device and is able to connect to it.

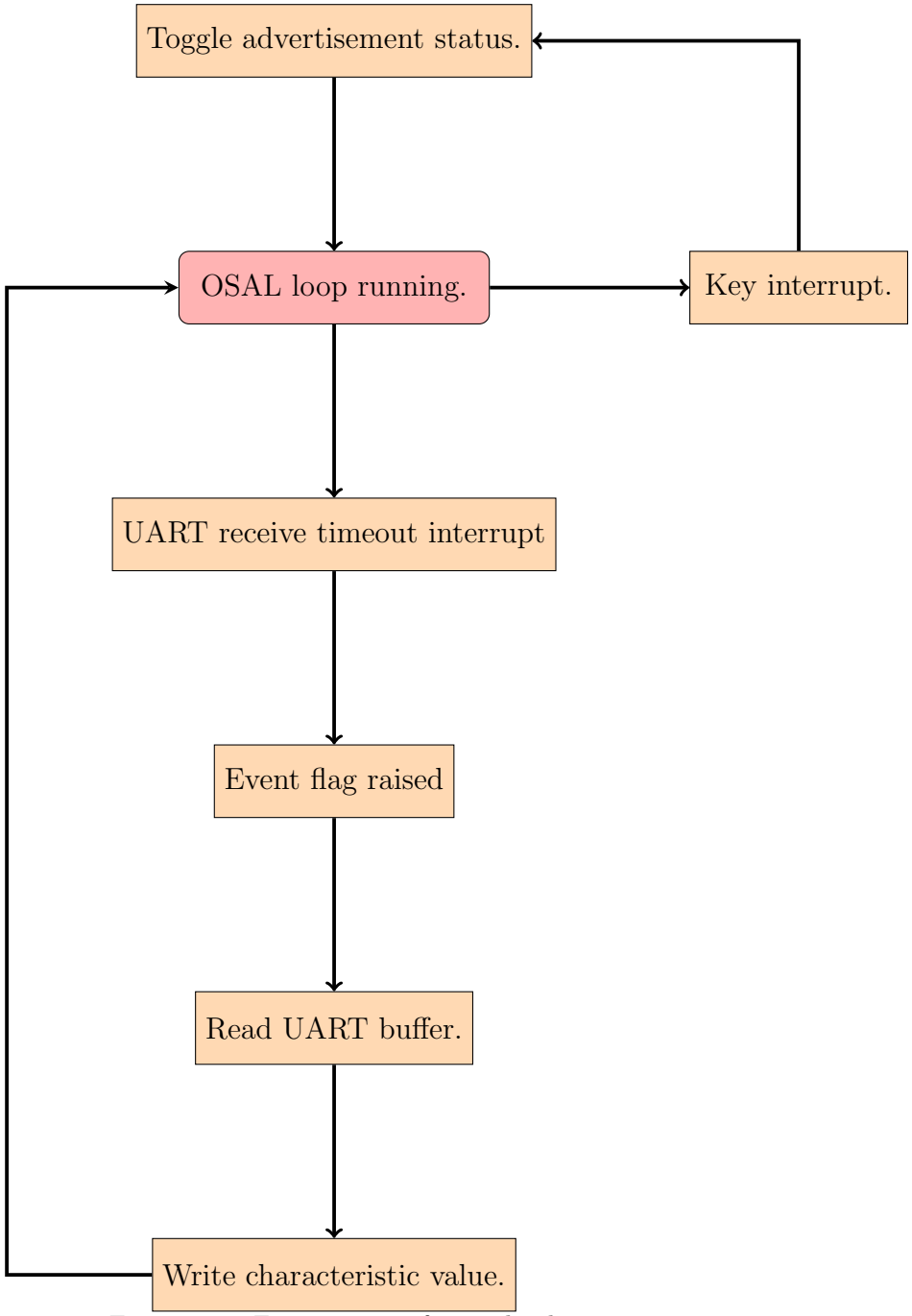

Figure 3.10: Footsteps in software development.

3. The android device on successful connection displays the services available with the BLE hardware. On selection of particular service, a screen shows the number of characteristics and descriptors for that service. Characteristic value can be written or read from the available screen.

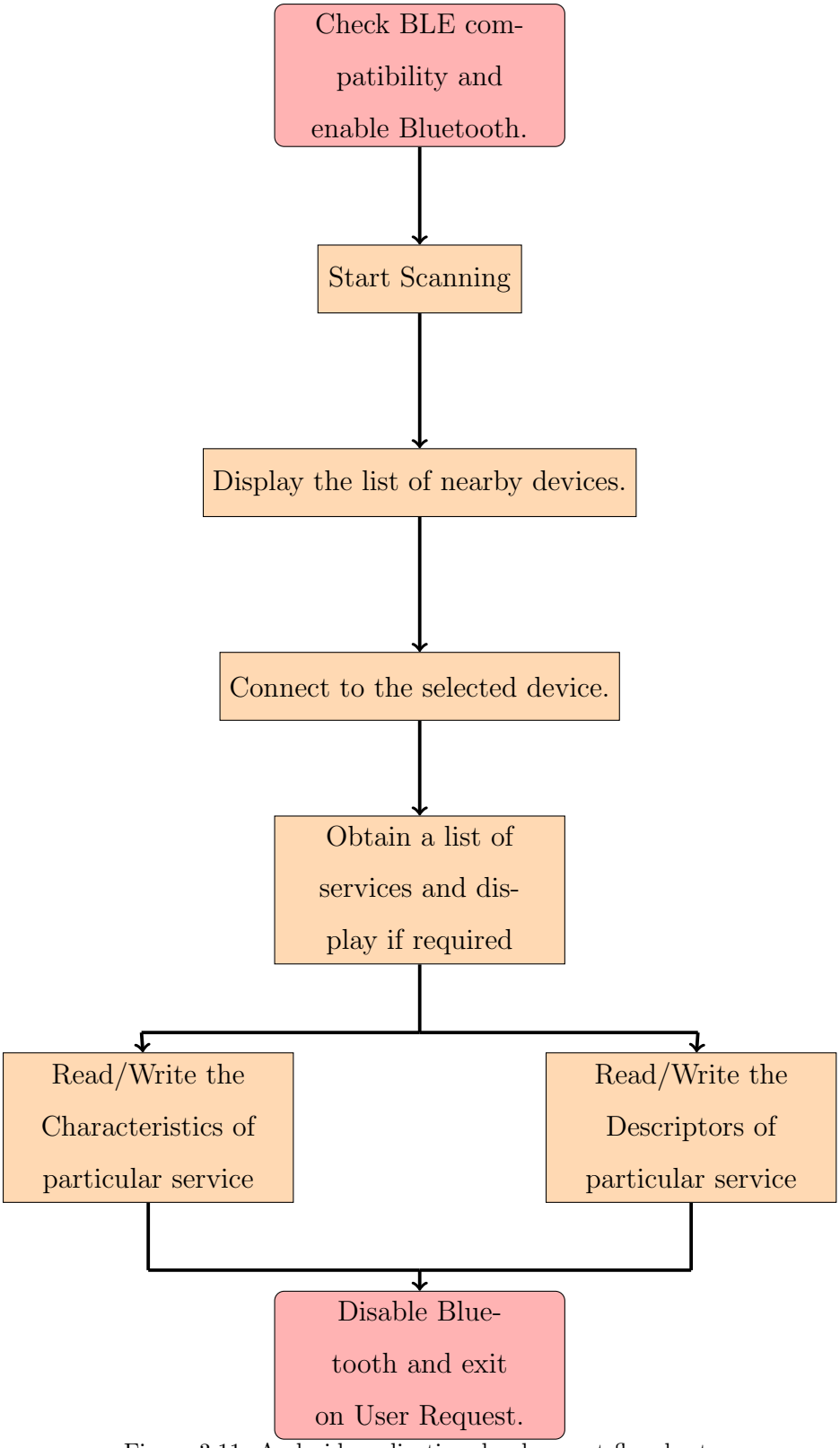

Figure 3.11: Android application development flowchart.

BLE hardware on getting data from UART buffer stores it in characteristic value of UART data transfer service. This change in characteristic value is detected by the android application and the data is displayed on the user device. The android

application that is developed as part of this project is shown in Fig. 3.12.

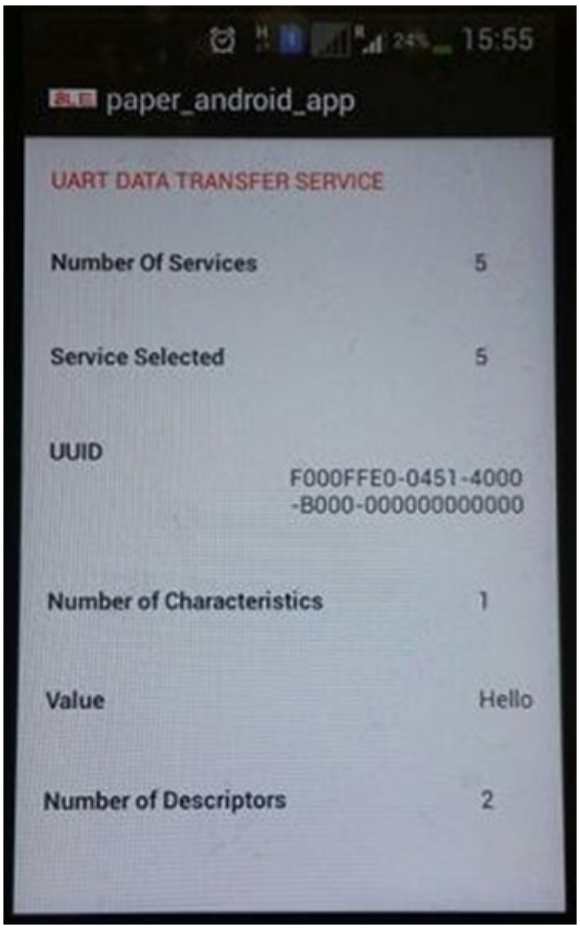

Figure 3.12: Android application.

Chapter 4

RESULTS AND DISCUSSION

The hardware design includes a schematic design followed by pcb design. The wireless module was designed with two pcb antennas 'Meandered Inverted F Antenna' (gain=4dbi) and 'Half Wave Dipole Antenna' (gain=7dbi). A MIFA is shown in Fig. 2.2 and 'Half Wave Dipole Antenna' in Fig. 2.4. These two antennas were designed on pcb according to the dimensions mentioned in (Section 2.2.3). After successful fabrication of PCB, it is send to testing phase. The testing of PCB is divided into two cases. Case I considers the testing of PCB with 'Meandered Inverted F Antenna' and case II uses 'Half Wave Dipole Antenna'. RSSI of wireless module was measured using Packet sniffer software provided by Texas Instruments. Fig. 4.1 shows a typical screen of BLE packet sniffer software.

Case I: Wireless module is tested with 'Meandered Inverted F Antenna' in this case with transmission power of 1 mw. The wireless module when interfaced with electronic device is placed in chassis of electronic device. Table 4.1 shows the performance of the wireless module when placed in a metallic chassis of the electronic product. It was observed that, wireless module placed in the chassis of the product did not function properly. Range of the wireless module has been drastically decreased to a mere 5 m compared to 45 m in open condition. Thus, the performance of wireless module also depends upon the properties of the material of the chassis.

Table 4.1: Range reduction of BLE module when placed in chassis.

Hardware	Tx. Power	Antenna Gain	Distance	RSSI(dbm)	Power Received	Estimated Range
Open	1mw	4 dbi	0.5 m	-46	25.1 nw	45 m
Chassis	1mw	4 dbi	0.5 m	-65	0.3162 nw	5 m

It was estimated that if we increase the transmit power to 2.5mw and increase the antenna gain to 7 dbi using half wave dipole antenna, performance can be considerably improved. Range was calculated theoretically before implementing the presented

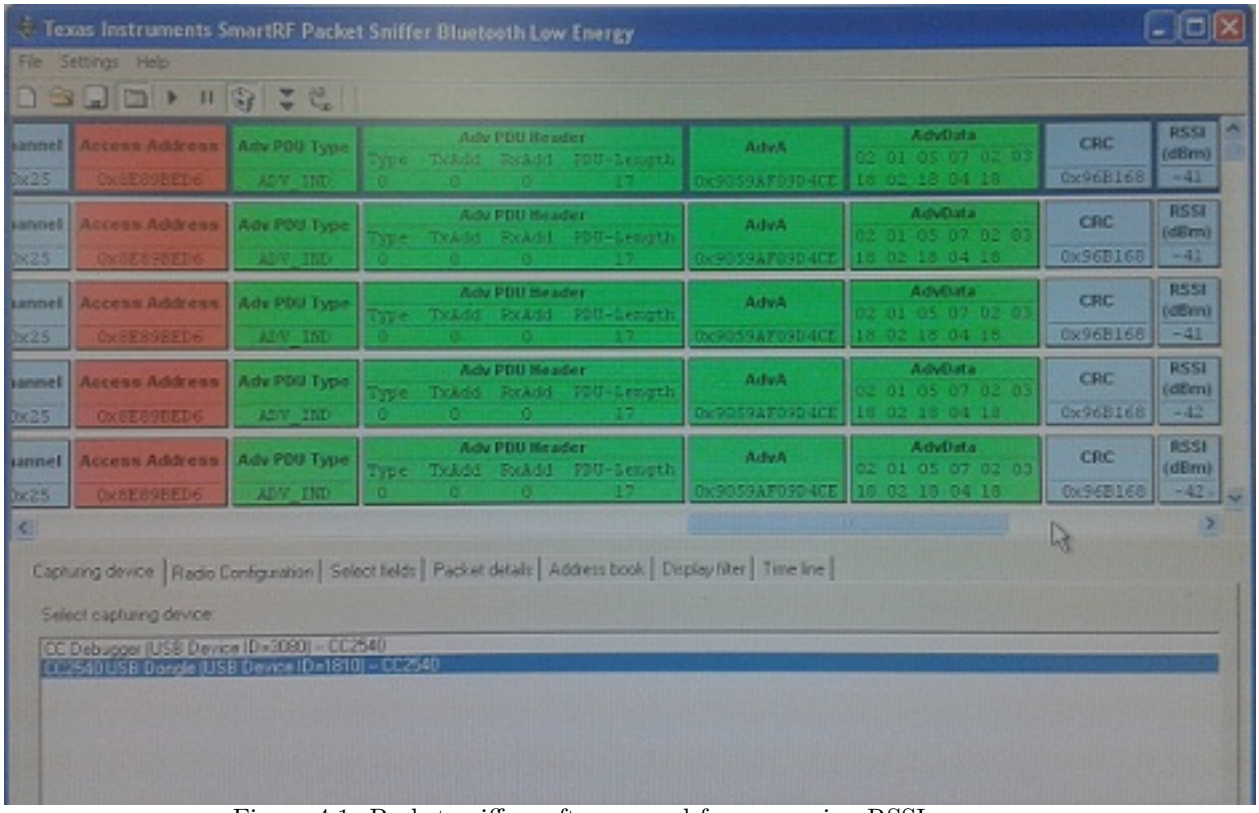

Figure 4.1: Packet sniffer software used for measuring RSSI.

solution.Table 4.2 shows the theoretically estimated results. Transmit Power is fixed at 2.5mw and RSSI is measured at 0.5m while taking these readings.

Table 4.2: Theoretically estimated range with increased transmit power and antenna gain.

Hardware	Antenna Gain	RSSI	Power Received	Estimated Range
Open	4 dbi	-42.003	63.0483 nw	71.305 m
Open	7 dbi	-39.01	125.5 nw	100.603 m
Chassis	4 dbi	-61.02	0.7905 nw	8 m
Chassis	7 dbi	-58.01	1.581 nw	11 m

Case II: The hardware was tested with HWD pcb antenna and transmission power of 2.5 mw. Table 4.3 shows the results obtained. The wireless link is maintained when the wireless module is inside chassis with about 10 meters of range. The results obtained are acceptable. Thus half wave dipole antenna is found to be a suitable option in this case.

Table 4.3: Calculated Range after implementing the solution.

Haedware	Antenna Gain	RSSI	Power Received	Estimated Range
Open	4 dbi	-43.22	47.643 nw	61.985 m
Open	7 dbi	-39.25	118.85 nw	97.901 m
Chassis	4 dbi	-60.51	0.8892 nw	8.4681 m
Chassis	7 dbi	-59.02	1.2531 nw	10.052 m

Chapter 5

CONCLUSION AND FUTURE SCOPE

In this work a generic approach to provide wireless interface for industrial products is implemented and tested. The developed system is centered around CC2540, a BLE device. CC2540 is concluded to be suitable according to literature survey in Section 2.2.2. The supporting circuit is designed and fabricated to investigate the performance of developed system. Experimental trials are performed under different conditions to corroborate the behaviour of wireless module. The experimental results exhibit that a HWD antenna provides a transmission range of 10 meteres with the wireless module placed in chassis of the product. Hence, from these observations it is concluded that the designed and fabricated system is suitable for desired objectives.

The future scope of this project may include extending Bluetooth Low energy for internet of things. Consumer electronic products in a household environment can be connected to each other using BLE. A Bluetooth Smart Ready phone can be used to provide one of the interface for internet of things.

Bibliography

[1] J. Singh, M. Tiwari, and M. Shrivastava, "Industrial automation- a review," *International journal of Engineering Trends and Technology*, vol. 4, pp. 3516–3520, Aug. 2013.

[2] *Can bus.* [Online]. Available: `https://en.wikipedia.org/wiki/CAN_bus#Applications`.

[3] W. Ikram and N. Thornhil, "Wireless communication in process automation: A survey of opportunities, requirements, concerns and challenges," in *UKACC International Conference on Control 2010*, 2010, pp. 1–6.

[4] *Panasonic white paper,"moving forward with bluetooth low energy".*

[5] *Bluetooth.* [Online]. Available: `http://en.wikipedia.org/wiki/Bluetooth#Specifications_and_features`.

[6] *Frequency-hopping spread spectrum.* [Online]. Available: `https://en.wikipedia.org/wiki/Frequency-hopping_spread_spectrum`.

[7] *Host controller interface (hci) architecture.* [Online]. Available: `https://developer.bluetooth.org/TechnologyOverview/Pages/HCI.aspx`.

[8] B. SIG, *Simple pairing whitepaper.*

[9] *Bluetooth master/slave communications and sniff/sniff sub-rating modes.*

[10] *Bluetooth specification version 1.0 and 1.0b.*

[11] *Bluetooth specification version 1.1.*

[12] *Bluetooth specification version 1.2.*

[13] *Bluetooth specification version 2.0+edr.*

[14] *Bluetooth specification version 2.1+edr.*

[15] *Bluetooth specification version 3.0+hs.*

[16] *Bluetooth specification version 4.0.*

[17] *Bluetooth specification version 4.1.*

[18] *Bluetooth specification version 4.2.*

[19] *Texas instruments,"2.4 g-hz bluetooth low energy system on chip". document number-swrs084f.*

[20] *St microelectronics," bluetooth low energy controller", stblc01.*

[21] *St microelectronics," bluetooth low energy processor" , bluenrg.*

[22] *Electrostatic discharge (esd) application report by texas instruments-ssya010a.*

[23] *Spurious emission.* [Online]. Available: `https://en.wikipedia.org/wiki/Spurious_emission`.

[24] *Spurious emissions.* [Online]. Available: `http://www.ntrc.vc/spurs.html`.

[25] *Audun andersen,"application note an-043 small size 2.4 ghz pcb antenna".*

[26] *Espen wium,"design note dn-0007 2.4 ghz inverted f antenna".*

[27] *Espen wium,"design note dn-041using cc253x or cc254x with dipole pcb antennas".*

[28] *Audun andersen,"design note dn-004 folded dipole antenna for cc25xx".*

[29] *Richard wallace, steve dunbar"application note dn-034 2.4 ghz yagi pcb antenna.".*

[30] *Android version history.* [Online]. Available: `http://en.wikipedia.org/wiki/Android_version_history`.

[31] *Fractus compact reach xtend, "bluetooth , zigbee, 802.11 b/g/n wlan chip antenna".*

[32] *List of bluetooth protocols.* [Online]. Available: `https://en.wikipedia.org/wiki/List_of_Bluetooth_protocols`.

[33] *Advanced audio distribution profile specification version 1.0.*

[34] *Audio/video remote control profile (avrcp).* [Online]. Available: `https://developer.bluetooth.org/TechnologyOverview/Pages/AVRCP.aspx`.

[35] *Android 1.1 version notes.* [Online]. Available: `http://en.wikipedia.org/wiki/Android_version_history#cite_note-29`.

[36] *Android 1.5 platform highlights.* [Online]. Available: `http://developer.android.com/about/versions/android-1.5-highlights.html`.

[37] *Android 1.6 platform highlights.* [Online]. Available: `http://developer.android.com/about/versions/android-1.6-highlights.html`.

[38] *Android 2.0 platform highlights.* [Online]. Available: `http://developer.android.com/about/versions/android-2.0-highlights.html`.

[39] *Gingerbread.* [Online]. Available: http://developer.android.com/about/versions/android-2.3-highlights.html.

[40] *Honeycomb.* [Online]. Available: http://developer.android.com/about/versions/android-3.0-highlights.html.

[41] *Ice cream sandwich.* [Online]. Available: http://developer.android.com/about/versions/android-4.0-highlights.html.

[42] *Jelly bean.* [Online]. Available: http://developer.android.com/about/versions/jelly-bean.html#media.

[43] *Android kitkat.* [Online]. Available: http://developer.android.com/about/versions/kitkat.html.

[44] *Android lollipop.* [Online]. Available: http://developer.android.com/about/versions/lollipop.html.

[45] *Object push profile.* [Online]. Available: https://developer.bluetooth.org/TechnologyOverview/Pages/OPP.aspx.

[46] *Phone book access profile specification.*

[47] *Google cloud messaging.* [Online]. Available: https://en.wikipedia.org/wiki/Google_Cloud_Messaging.

[48] *Opensl es specification version 1.0.1.*

[49] *Near field communication.* [Online]. Available: https://developer.android.com/guide/topics/connectivity/nfc/index.html.

[50] *Wi-fi peer-to-peer.* [Online]. Available: http://developer.android.com/guide/topics/connectivity/wifip2p.html.

[51] *Human interface device profile 1.1.*

[52] *Hid over gatt profile specification.*

[53] *Tdls.* [Online]. Available: https://en.wikipedia.org/wiki/TDLS.

[54] *Greg stewart,"ti bluetooth low energy training october 2013".*

[55] M. Grover, S. Pardeshi, and N. Singh, "A guide to bluetooth low energy technology," in *2nd IEEE International Conference on Innovations in Information, Embedded and Communications Systems, ICIIECS'15*, pp. 557–560.

[56] *Attribute profile(att) and generic attribute profile(gatt), bluetooth special interest group.*

[57] *Texas instruments,"hal drivers application programming interface". document number- swra193.*

[58] *Texas instruments cc2540/41 bluetooth low energy software developer's guide v1.4.*

[59] *Texas instruments, "os abstraction layer application programming interface". document number- swra194.*

[60] *Bluetooth low energy.* [Online]. Available: `https://developer.android.com/guide/topics/connectivity/bluetooth-le.html`.

Appendix- A

HAL DRIVERS

The required information to use the ADC and LCD services of HAL drivers is as explained below. This will give a general idea on using the HAL drivers.

A.1 ADC Service

This service supports 8, 10, and 14-bit analog to digital conversion on 8 channels (0-7).

1. **HalAdcInit() :**This ADC initialization function is called once at the startup. This function has to be called before any other ADC functions can be called. It enables ADC to be initialized with both required and optional parameters.
   Prototype: void HalAdcInit (void)
   Parameter Details: None
   Return: None

2. **HalAdcRead():** This function reads and returns the value of the ADC conversion at specified channel and resolution.
   Prototype: uint16 HalAdcRead (uint8 channel, uint8 resolution)
   Parameter Details:
   channel– input channels
   resolution– resolution of the conversion.
   Return: Return 16-bit value of the conversion at the given channel and resolution.
   Constants: The constants with their description are provided in Table A.1 and Table A.2.

Table A.1: ADC channel constants.

Parameter	Description
HAL_ADC_CHANNEL_0	Input Channel 0
HAL_ADC_CHANNEL_1	Input Channel 1
HAL_ADC_CHANNEL_2	Input Channel 2
HAL_ADC_CHANNEL_3	Input Channel 3
HAL_ADC_CHANNEL_4	Input Channel 4
HAL_ADC_CHANNEL_5	Input Channel 5
HAL_ADC_CHANNEL_6	Input Channel 6
HAL_ADC_CHANNEL_7	Input Channel 7

Table A.2: ADC resolution constants.

Parameter	Description
HAL_ADC_RESOLUTION_8	8-bit resolution
HAL_ADC_RESOLUTION_10	10-bit resolution
HAL_ADC_RESOLUTION_12	12-bit resolution
HAL_ADC_RESOLUTION_14	14-bit resolution

A.2 LCD Service

This service allows writing text on the LCD .

1. **HalLcdInit():** This initialization function is called once at the startup. This function has to be called before any other LCD function can be called. It enables LCD to be initialized with both required and optional parameters.
   Prototype: void HalLcdInit (void);
   Paraneter Details: None
   Return: None

2. **HalLcdWriteString():** This routine writes a string to the LCD.
   Prototype: void HalLcdWriteString (unit8* str, uint8 option);
   Parameter Details:
   str – pointer to the string that will be displayed on the LCD. Max length of the string is defined under HAL_LCD_MAX_CHARS. If the length is greater than HAL_LCD_MAX_CHARS, then only HAL_LCD_MAX_CHARS will be displayed.
   option– option for the string to be displayed on the LCD.
   Return: None.

3. **HalLcdWriteValue() :** This routine writes a 32-bit value to the LCD.
   Prototype: void HalLcdWriteValue (uint32 value, uint8 radix, uint8 option);
   Parameter Details:

value – value that will be displayed on the LCD.

radix– representation of the value. It can be 8, 10 or 16. (Octal, Decimal, or Hex)

option – option for the value to be displayed on the LCD.

Appendix- B

OSAL APIs

The OSAL APIs are discussed below:

B.1 Message Management APIs

1. **osal_msg_allocate( ) :** This function is called by a task to allocate a message buffer, the task/function will then fill in the message and call osal_msg_send() to send the message to another task. If the buffer cannot be allocated, msg_ptr will be set to NULL.

   Prototype: uint8 *osal_msg_allocate(uint16 len)

   Parameters:

   len: It is the length of the message.

   Return: The return value is a pointer to the buffer allocated for the message. A NULL return indicates the message allocation operation failed.

2. **osal_msg_deallocate( ) :** This function is used to de-allocate a message buffer. This function is called by a task (or processing element) after it has finished processing a received message.

   Prototype: uint8 osal_msg_deallocate(uint8 *msg_ptr)

   Parameter Details:

   msg_ptr: It is a pointer to the message buffer that needs to be de-allocated.

   Return: Return value indicates the result of the operation.

B.2 Task Synchronization APIs

1. **osal_set_event( ) :** This function is called to set the event flags for a task.

   Prototype : uint8 osal_set_event(uint8 task_id, uint16 event_flag)

   Parameter Details:

   task_id: It is the identifier of the task for which the event is to be set.

event_flag: It is a 2-byte bitmap with each bit specifying an event. There is only one system event (SYS_EVENT_MSG), the rest of the events/bits are defined by the receiving task.

Return: Return value indicates the result of the operation.

B.3 Timer Management APIs

1. **osal_start_timerEx( ) :** This function is called to start a timer. When the timer expires, the given event bit will be set. The event will be set for the task specified by taskID. This timer is a one shot timer, meaning that when the timer expires it isn't reloaded.

   Prototype : uint8 osal_start_timerEx(uint8 taskID, uint16 event_id, uint32 timeout_value);

   Parameter Details:

   taskID: It is the task ID of the task that is to get the event when the timer expires.

   event_id is a user defined event bit. When the timer expires, the calling task will be notified (event).

   timeout_value is the amount of time (in milliseconds) before the timer event is set.

   Return: Return value indicates the result of the operation.

2. **osal_start_reload_timer( ) :** Call this function to start a timer that, when it expires, will set an event bit and reload the timeout value automatically. The event will be set for the task specified by taskID.

   Prototype : uint8 osal_start_reload_timer(uint8 taskID, uint16 event_id, uint32 timeout_value);

   Parameter Details:

   taskID: is the task ID of the task that is to get the event when the timer expires.

   event_id: is a user defined event bit. When the timer expires, the calling task will be notified (event).

   timeout_value: is the amount of time (in milliseconds) before the timer event is set. This value is reloaded into the timer when the timer expires.

   Return: Return value indicates the result of the operation.

B.4 Interrupt Management APIs

1. **osal_int_enable( ) :** This function is called to enable an interrupt. Once enabled, occurrence of the interrupt causes the service routine associated with that interrupt to be called.
   Prototype: uint8 osal_int_enable(uint8 interrupt_id)
   Parameter Details: interrupt_id identifies the interrupt to be enabled.
   Return: Return value indicates the result of the operation.

2. **osal_int_disable( ) :** This function is called to disable an interrupt. When a disabled interrupt occurs, the service routine associated with that interrupt is not called.
   Prototype : uint8 osal_int_disable(uint8 interrupt_id)
   Parameter Details:
   interrupt_id identifies the interrupt to be disabled.
   Return: Return value indicates the result of the operation.

B.5 Task Management APIs

1. **osal_init_system( ) :** This function initializes the OSAL system. The function must be called at startup prior to using any other OSAL function.
   Prototype : uint8 osal_init_system(void)
   Parameter Details: None
   Return: Return value indicates the result of the operation.

2. **osal_start_system( ) :** This function is the main loop function of the task system, repeatedly calling osal_run_system(), from an infinite loop. This function never returns. When using a different scheduler, it should call osal_run_system() directly and not use this function.
   Prototype: void osal_start_system(void)
   Parameter Details: None
   Return : None

B.6 Memory Management APIs

1. **osal_mem_alloc( ) :** This function is a simple memory allocation function that returns a pointer to a buffer (if successful).

Prototype: void *osal_mem_alloc(uint16 size)

Parameter Details :

size – the number of bytes wanted in the buffer.

Return: A void pointer (which should be cast to the intended buffer type) to the newly allocated buffer. A NULL pointer is returned if there is not enough memory to allocate.

2. **osal_mem_free( ) :** This function frees the allocated memory to be used again. This only works if the memory had already been allocated with osal_mem_alloc().

   Prototype: void osal_mem_free(void *ptr)

   Parameter Details :

   ptr – pointer to the buffer to be freed. Buffer must have been previously allocated with osal_mem_alloc().

   Return: None

B.7 Power Management APIs

1. **osal_pwrmgr_init( ) :** This function will initialize variables used by the power management system. Do not call this function, it is already called by osal_init_system().

   Prototype: void osal_pwrmgr_init(void)

   Parameter Details:None

   Return: None

2. **osal_pwrmgr_powerconserve( ) :** This function is called to go into power down mode. Do not call this function, it is already called in the OSAL main loop [osal_start_system()].

   Prototype : void osal_pwrmgr_powerconserve(void);

   Parameter Details : None

   Return: None

B.8 Non-Volatile Memory APIs

1. **osal_nv_item_init( ) :** Initialize an item in NV. This function checks for the presence of an item in NV. If it does not exist, it is created and initialized with the data passed to the function, if any.

Prototype: uint8 osal_nv_item_init(uint16 id, uint16 len, void *buf);

Parameter Details:

id – User-defined item ID.

len – Item length in bytes.

buf – Pointer to item initialization data. If no initialization data, set to NULL.

Return: Return value indicates the result of the operation.

2. **osal_nv_read( ) :** Read data from NV. This function can be used to read an entire item from NV or an element of an item by indexing into the item with an offset. Read data is copied into *buf.

   Prototype: uint8 osal_nv_read(uint16 id, uint16 offset, uint16 len, void *buf)

   Parameter Details:

   id – User-defined item ID.

   offset – Memory offset into item in bytes.

   len – Item length in bytes.

   buf – Data is read into this buffer.

   Return: Return value indicates the result of the operation.

B.9 Simple Non-Volatile Memory APIs

1. **osal_snv_read( ) :** Read data from NV. This function can be used to read an entire item from NV. Read data is copied into *pBuf.

   Prototype : uint8 osal_snv_read(osalSnvId_t id, osalSnvLen_t len, void *pBuf);

   Parameter Details:

   id – User-defined item ID.

   len – Item length in bytes.

   pBuf – Data is read into this buffer.

   Return: Return value indicates the result of the operation. Note that an attempt to read an item, which was never written before, would result into the NV_OPER_FAILED return code.

2. **osal_snv_write( ) :** Write data to NV. This function can be used to write an entire item to NV.

   Prototype: uint8 osal_snv_write(osalSnvId_t id, osalSnvLen_t len, void *pBuf);

Parameter Details :

id – User-defined item ID.

len – Item length in bytes.

pBuf – Data to write.

Return: Return value indicates the result of the operation. Note that it is allowed to write an item that was never before initialized into the NV system by other means.

B.10 OSAL Clock System

1. **osalTimeUpdate( ) :** Called from osal_run_system() to update the time. This function reads the number of MAC 320usec ticks to maintain the OSAL Clock time. Do not call this function anywhere else.
   Prototype: void osalTimeUpdate(void);
   Parameter Details : None
   Return: None

2. **osal_setClock( ) :** Call this function to initialize the device's time.
   Prototype : void osal_setClock(UTCTime newTime);
   Parameter Details :
   newTime – new time in seconds since 0 hrs, 0 minutes, 0 seconds, on 1 January 2000 UTC
   Return : None

B.11 OSAL Misc

1. **osal_rand( ) :** This function returns a 16-bit random number.
   Prototype : uint16 osal_rand(void);
   Parameter Details : None
   Return : Returns a random number.

2. **osal_memcmp( ) :** Compare two memory sections.
   Prototype: uint8 osal_memcmp(const void GENERIC *src1, const void GENERIC *src2, unsigned int len);
   Parameter Details :

src1 – memory compare location 1.

src2 – memory compare location 2.

len – length of compare.

Return: TRUE - same, FALSE - different.