A SUB-SAMPLING PULSE-RESONANCE OOK MODULATED DIGITAL ULTRASOUND COMMUNICATION SYSTEM FOR MEDICAL IOT

A Thesis

by

Mohamed O.Abouzeid

Submitted to the
Graduate School of Sciences and Engineering
In Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in the
Department of Electrical and Electronics Engineering

Özyeğin University
April 2018

A SUB-SAMPLING PULSE-RESONANCE OOK MODULATED DIGITAL ULTRASOUND COMMUNICATION SYSTEM FOR MEDICAL IOT

Approved by:

Assistant Professor Ahmet Tekin, M.Sc.
Advisor
Department of Electrical and Electronics Engineering
Özyeğin University

Associate Professor H.Fatih Uğurdağ
Department of Electrical and Electronics Engineering
Özyeğin University

Assistant Professor Enver Cavus
Department of Electrical and Electronics Engineering
Ankara, Yildirim Beyazit University

Date Approved: 9 April 2018

To my parents ......

ABSTRACT

This dissertation presents a new ultrasound communication microsystem (Pulsed Resonance On-Off Keying) as a viable alternative to today's widely used RF technologies in order to avoid the associated health risks. Special circuit techniques were proposed to overcome the drawbacks of classical ultrasound communications; such as echoes and excess ringing, achieving a measured communication range of 28m with a 50 bits/s data rate and BER of 0.01. Targeting mainly medical sensor devices, the technology had to be insulated, small size and low power. Utilizing a 40 kHz ultrasound transducer and an 8-pin low power controller, wireless charged, high accuracy remote temperature sensor system with nominal average current consumption of 0.5uA was designed and tested. Multiple subsystems were all merged in a total volume of 12mm diameter and 15mm height, excluding the charging coil. Each of the ultrasound communication, temperature sensor and battery measurement functions do use the same circuit pins with special circuit configurations. Thanks to echoes avoidance, ringing suppression, dynamic detection threshold adjustment techniques along with 3-bit preamble synchronization; the proposed low-power sub-sampling IQ demodulation of OOK bits resulted in high sensitivity robust ultrasound communication system without any alignment requirement for the transducers. The lifetime of the prototyped sensors with an 8mAh LiR battery was about 27 months corresponding to sensory data update frequency of 1 sample/minute.

ÖZETÇE

Bu tez, günümüzde yaygın şekilde kullanılan RF teknolojilerine, bu sistemden dolayı oluşan sağlık sorunlarını engellemek amacıyla, RF sistemlerine uygun bir alternatif olabilecek bir ultrason iletişim mikrosistemini takdim etmektedir. Klasik ultrason iletişim sistemlerinin yol açtığı sıkıntıların üstesinden gelmek için özel tasarım devre sistemleri önerilmektedir, bu sıkıntılara örnek olarak eko ve istenmeyen salınımlar örnek gösterilebilir. Bu özel tasarım devre 50bit/s data oranı ve 0.01 BER ile 28m iletişim menziline ulaşabilmektedir. Medikal sensör cihazlarına küçük boyutta ve düşük güçte entegre edilmiştir. 40kHz ultrason dönüştürücü ve 8-pinli düşük güç kontrolü kullanılarak, suya dayanıklı, kablosuz şarj edilebilen, yüksek hassasiyetli uzaktan kontrol edilebilen, ortalama 0.5uA akım tüketen ısı sensör sistemi dizayn ve test edilmiştir. Birden fazla alt sistem, şarj bobini hariç 12mm yarıçap ve 15mm uzunluğu olan bir hacme toplanmıştır. Tüm ultrason iletişim sistemi, ısı sensörü ve batarya ölçer fonksiyonlar aynı özel konfigürasyonlu devre pin'ini kullanmaktadır. Ekoyu ve ringing'i engellemesi, 3-bit giriş senkronizasyonuyla dinamik sezim eşik ayarlama teknikleri sayesinde, sunulan OOK düşük-güçlü alt-örnekleme IQ demodülasyonu bitleri, transdüserleri hizalamaya gerek kalmadan yüksek hassasiyetli ve güçlü bir ultrason iletişim sistemi elde etmemize olanak sağlamıştır. Dakikada 1 örnek data frekans güncellemesi yapan 8mAh LiR bataryanın ömrü yaklaşık 27 aydır.

ACKNOWLEDGEMENTS

Firstly, I would like to thank Allah for his mercy and help being with me along the way till the end.

Secondly, I would like to express my sincere appreciation and deepest gratitude to my supervisor, Dr.Ahmet Tekin for his invaluable guidance and support which inspired me as a student, I would like to thank him for planning and executing the work in the best timely manner. The confidence of getting his advice at any time will always be a valuable memory. His technical and editorial points were crucial for the completion of this dissertation and taught me innumerable lessons on academic research work in general.

Furthermore, I would like to thank Dr.H.Fatih Uğurdağ and Dr.Enver Cavus for being my thesis committee and their valuable and editorial guidance.

In addition, I would like to express my gratitude to WAVEWORKS Inc. for funding this project with the necessary tools and materials.

Finally, I cannot find enough words to express my sincere gratitude that I owe to my parents, my brothers and my friends for the motivation and encouragement they always do for me.

Mohamed O.Abouzeid

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1 Problem Definition

The revolution of biomedical IoT electronics has been rising remarkably in terms of number of products and the remarkably lower costs associated. These biomedical IoT applications mainly utilize RF based high frequency data transmission and reception. However, harmful radio frequency exposure is a main setback in most of the prominent technologies of today. The human exposure to electromagnetic radiation had been in the rise last few decades. However, it is widely accepted that, long-term exposure to high intensity EM field may cause health problems. As proposed in [1], many organizations carried out different projects and experiments to find out ways to reduce the specific absorb rate (SAR) and EM exposure and to define the guidelines for minimizing this excess exposure. For example, as it was discussed in [2], experimental studies carried out by the International Commission on Non-Ionizing Radiation Protection (ICNIRP) on volunteering humans, who are exposed to low level of EM radiation over a period of 30 minutes, show that this sheer exposure results up to 0.5 °C rise in body temperature. Nevertheless, harmful tissue heating levels have been remarked for SAR values over 4 W/Kg. Furthermore, some researchers in [3] had concluded that excess EM exposure to a standard GSM mobile phone has adverse effects on the human cognitive functions. The National Radiological Protection Board (NRPB) of the UK has given its recommendations on the limits of exposure to EM from mobile phones, base stations and other sources of EM radiation in different reports [4][5][6].

1.2 Proposed Solution

In the quest for a remedy for the mentioned daily exposure problems; ultrasound has been investigated as a viable candidate for short-range wireless communication over the air to be an alternative to radio frequency(RF). It has several advantages over radio frequency; ultrasound waves are unregulated and EMI-interference free to most sensitive electronic devices. Furthermore, as a result of being hard to intercept through solid barriers like walls, ultrasonic signals provide an extremely good privacy in air, thus enhancing the security of the communication.

The goal of this research is to develop a means of communication using radio frequency and in the same time harmless. Especially if the target application is related to young generations. Ultrasound transducers can be used as an intermediate step between the radio frequency device and the biomedical sensor close by the patient.

This dissertation proposes a new ultrasound communication micro system (PR−OOK) as a viable alternative to today's widely used RF technologies in order to avoid the associated health risks. Special circuit techniques were proposed to overcome the drawbacks of classical ultrasound communications; such as echoes and excess ringing, achieving a measured communication range of 28m with a 50 bits/s data rate and BER of 0.01 for 5 dB SNR. Targeting mainly medical sensor devices, the technology had to be insulated, small size and low power. Utilizing a 40 kHz ultrasound transducer and an 8-pin low power controller, a water resistant, wireless charged, high accuracy remote temperature sensor system with nominal average current consumption of 0.5uA was designed and tested. Multiple subsystems were all merged in a total volume of 12mm diameter and 15mm height, excluding the charging coil. Each of the ultrasound communication, temperature sensor and battery measurement functions do use the same circuit pins with special circuit configurations. Thanks to echoes avoidance, ringing suppression, dynamic detection threshold adjustment techniques along with 3-bit preamble synchronization; the proposed low-power sub-sampling IQ demodulation of OOK bits resulted in high sensitivity robust ultrasound communication system without any alignment requirement for the transducers. The lifetime of the prototype sensor with an 8mAh LiR battery was about 27 months corresponding to sensory data update frequency of 1 sample/minute.

1.3 Thesis Organization

This dissertation is organized as following: The first chapter will introduce the ultrasound transducer, how they work and their characteristics. Then, the second chapter will present the proposed communication scheme, the design and theory behind it for sensing the temperature. In addition, the third chapter will introduce an application based on the communication technique idea, the design and the requirements. Furthermore, the fourth chapter will introduce the results achieved followed by the conclusion and future work.

CHAPTER II

ULTRASOUND IN DATA COMMUNICATION

2.1 Ultrasound Definition

Sound propagates in the form of mechanical energy, a vibrating source is responsible for the production of sound. The sound spectrum can be classified according to the bandwidth range, infra-sound which is below 20Hz, audible sound which is between 20Hz to 20KHz, the ultrasound band which is above 20KHz. The characteristics of ultrasound depend mainly on the medium which sound waves propagate in. It cannot take place in an empty space. A source of ultrasound transfers mechanical disturbances in contact with the medium which consequently initiate vibrations in the particles of the medium.

2.2 Ultrasound Characterization

Ultrasound waves are characterized based on their speed in the medium, intensity, reflections and refraction...etc. The most important characteristics related to data communication are : the velocity, the intensity, reflections and transducer ringing phenomenon.

2.2.1 Ultrasound Generation and Detection

Recalling the mechanical energy form of ultrasound, we can conclude that based on the piezoelectric phenomenon, ultrasound signals can be generated or detected, reverse piezoelectric phenomenon is a process which converts electrical energy into mechanical energy. On the other hand, the piezoelectric effect can be used to detect ultrasound mechanical vibrations, by converting them into an electrical signal. In brief, the generation and detection of ultrasound is done by using crystals of piezoelectric materials. Production of ultrasound relies on the reverse piezoelectric effect, while detection is based on the piezoelectric effect. Because of the reversibility of this phenomenon, it is possible to use the same crystal to produce ultrasound, and meanwhile to detect echoes reflected back to the crystal.

2.2.2 Intensity and Power

Intensity and power are related to each other but they are different physical quantities which are used as an indication to the flow of energy in the medium. When an ultrasound transducer is excited by an electrical voltage, vibrations pass into the medium which means that energy passes from the transducer to the medium. The intensity is defined as the rate of flow of energy through unit area at a certain point in the medium. Intensity is often measured at the focus of the field or within few centimeters of the transducer face. The designers of ultrasound transducers usually measure the intensity over the full depth range for which the transducer will be used. It is very important to characterize the ultrasound source based on how much power it can transmit and how much will be received after certain distance. This is defined by insertion loss. For the transducer used in the project as shown in the Figure 1; this is the measured insertion loss for the transducer used in the proposed research.

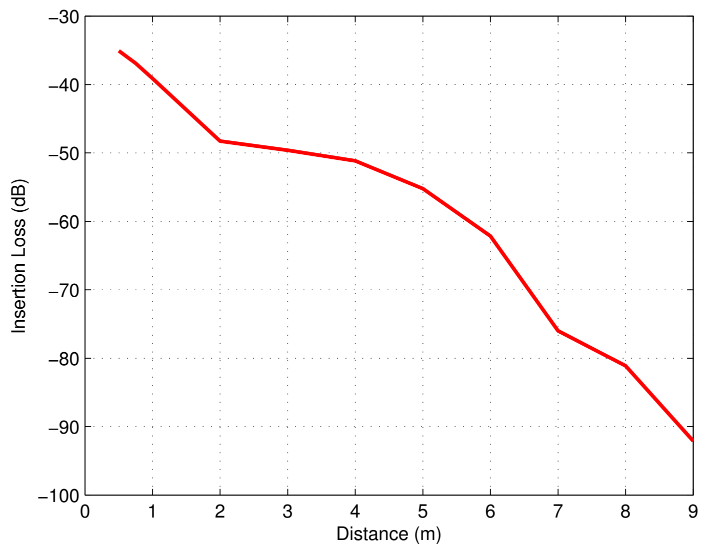

Figure 1: Transducer Insertion Loss.

2.2.3 Wave Velocity

It is defined as the rate at which the ultrasound wave is propagated through a medium. This velocity varies from one medium to another as shown in Table 1, depending on the elastic properties of the material. The velocity changes with the medium temperature. However,the velocity change is quite small for a few degree of temperature shift. Since room temperature is under control and it is not varying suddenly, the effect of temperature on velocity is negligible.

2.2.4 Acoustic Impedance

It is considered to be a physical property of tissue. It is an indication of how much resistance an ultrasound beam encounters as it passes through a tissue. Acoustic impedance depends on the density of the tissue (kg/m^3) and the speed of the sound wave (m/s). The ultrasound speed and associated acoustic impedance are shown in Table 1.

Table 1: Speed of ultrasound and Acoustic impedance.

$Material$	$Speed(m/s)$	$AcousticImpedance(g/cm^2S)$
Water	1480	1.48×10^5
Blood	1570	1.61×10^5
Bone	3500	7.80×10^5
Fat	1450	1.38×10^5
Liver	1550	1.65×10^5
Muscle	1580	1.70×10^5
Polythene	2000	1.84×10^5
Air	330	0.0004×10^5
Soft tissue	1540	1.63×10^5

2.2.5 Environmental Echoes

There are more severe problems in ultrasound communication systems that prevent their use as reliable and widely as RF systems. One of those problems is echoes. As a result of the reflections from the objects around; multiple copies of the transmitted signal arrive at different times to the receiver as shown in Figure 2. What makes this worse is that it is not only environment dependent but also the bit pattern to be transmitted can result in severe interference lasting multiple bit periods. Since the speed of sound is 330 m/s in air, the echoes travel comparable time to the bit period of the modulated signal. Hence, the modulated bit pattern shown in green from the oscilloscope screen-shot of Figure 3, may end up like signal shown in orange at the output of receiver front-end which makes the detection of zeros very hard, if not impossible.

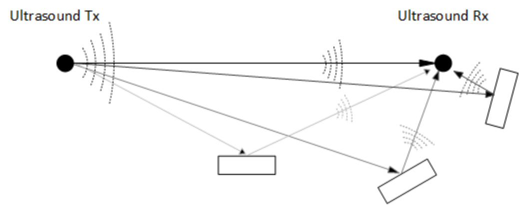

Figure 2: Echoes problem.

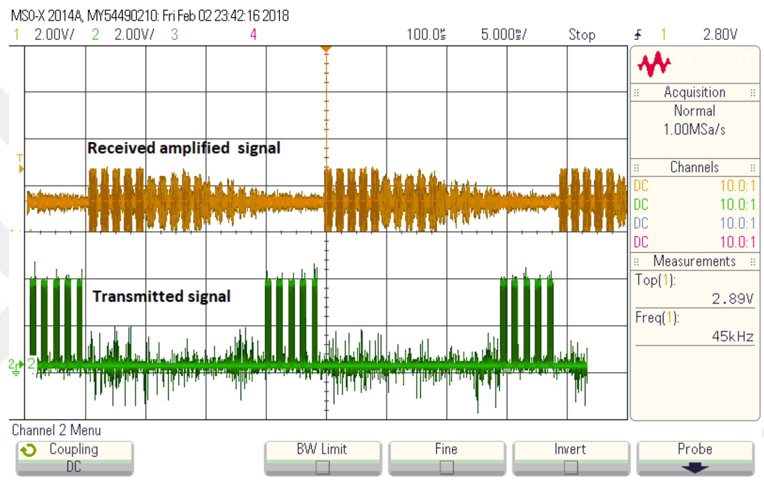

Figure 3: Echoes waveforms.

2.2.6 Transducer Ringing

Ultrasound generation and detection are based on piezoelectric resonance. Hence, the more the membrane will store energy, the more it will take time to release this energy out. This is another reason for ultrasound not being used commonly for short range communication even though it has several advantages. The hardest part is that it is pattern dependent which makes it impossible to predict or take precautions while transmitting or receiving ultrasound digital bits. As shown in Figure 4, the transmitted bits are plotted as a green waveform while the amplified received pulses are plotted as the orange waveform. The received pulses are stretched extensively to the following time slots. If the membrane is driven with less number of pulses, it will ring less. However,the number of pulses is mainly dependent on the bit pattern which is not predictable.

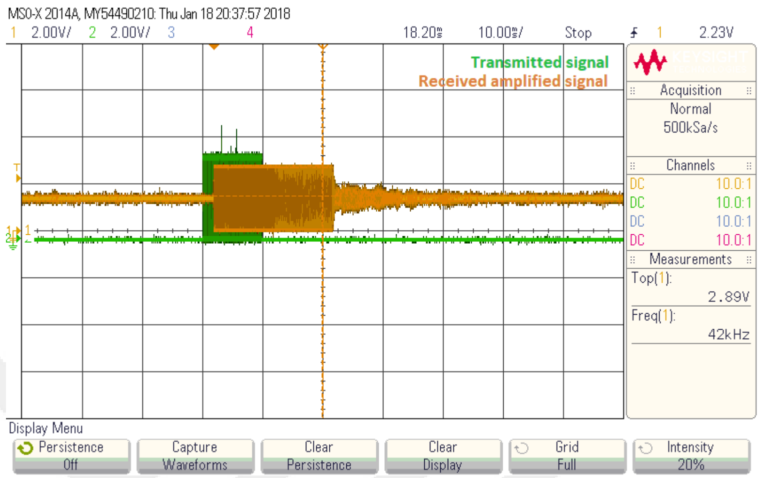

Figure 4: Transducer membrane ringing.

2.2.6.1 Ultrasound transducer modeling

For an air coupled ultrasonic transducer, the measured Butterworth Van-Dyke (BVD) model parameters (characterized using Impedance Analyzer HM 8118), are: R_s =42.6 Ω, L_s =2.6985 mH, C_1=1.9 nF, C_o=5.89 nF. The model shown in Figure 5 was used in circuit design and simulation while optimizing the ultrasound receiver circuit chain.The simulated AC sweep for the equivalent impedance for these parameter is shown in Figure 6. The minimum impedance was at 40 KHz which represents maximum current injected to interface circuit, the model is showing very narrow bandwidth around 40KHz which puts limitation on the modulation type selection.

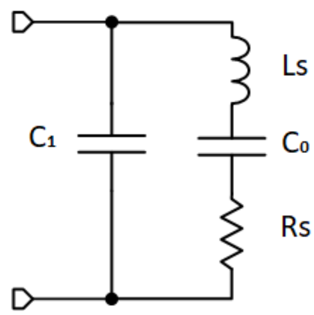

Figure 5: Transducer BVD model.

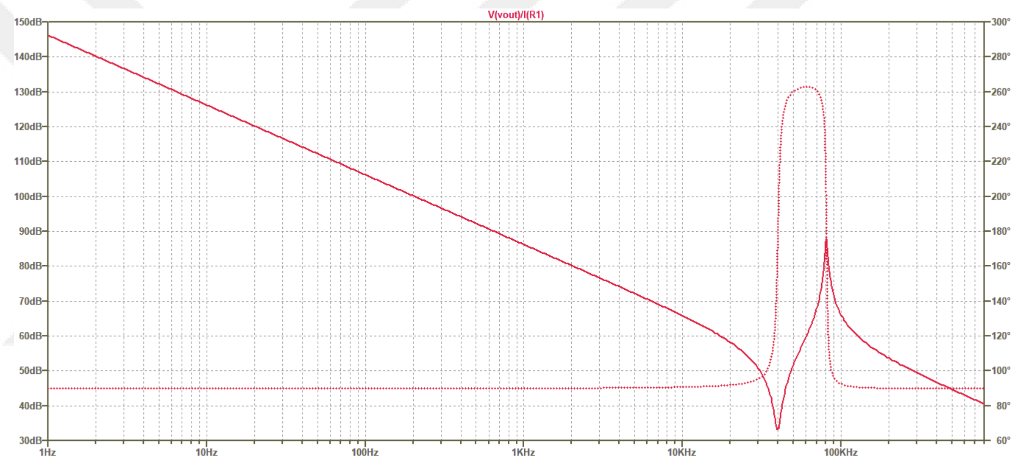

Figure 6: Transducer equivalent impedance.

2.3 Ultrasound Usage Advantages

Ultrasound offers several benefits which is not offered by radio waves like:

No Electromagnetic Interference Issues: Ultrasound is not like the radio waves, which can electromagnetically interfere with the EMI-sensitive equipment, it is a safe communication medium in strictly EMI-regulated environments like hospitals and nuclear plants. Many of the commercially available tracking systems used in hospitals are based on ultrasound signals.

Low Cost: For the application of wireless sensor networks which requires a large number of sensors to be used and distributed to sense environmental variations,the concern of adding any extra hardware is to be low cost. The used piezoelectric ultrasonic transducers and extra integrated circuits in this research are low cost and commercially available for mass production.

Ultrasound Sonar Applications: Ultrasound sonar modules are extremely available commercially and it's widely used in many applications like robotics and line tracking vehicles. These vehicles can benefit from the low power custom design, light weight of the ultrasonic sonar as the main concern of the application is frequently the weight and the capacity of batteries.

Safe Communication Medium: Ultrasound is relatively safer than the tightly regulated radio spectrum, there are no legally limits on ultrasound transmission. It is considered to be safe for human. For this reason, the ultrasound emission limits recommended by various government and non-government organizations are relatively relaxed compared to radio frequency. For diagnostic ultrasound, the Food and Drug Administration (FDA) recommends ultrasound pressure levels to be below 600 Pa [7]. The Occupational Safety and Health Administration (OSHA) puts limits for frequencies below 8 kHz [8], but there are no limits on ultrasonic frequencies. An American Conference of Governmental Industrial Hygienists (ACGIH), recommends pressure levels below 350 Pa for ultrasound [9].

Most of the recommendations discussed above are for applications that are very different from wireless data communication, but they still give an idea of the relative safety of ultrasound.

2.4 Ultrasound in Commercial Applications

Several suppliers of systems and components are introducing ultrasonics products that have commercial applications in the industry. For example, ultrasound is widely used in medical imaging and it has been investigated to be used for data communication applications.

Medical Imaging Applications: Ultrasound is considered to be the main element in medical imaging applications because of being extensively used in diagnostics for medical imaging. A phased ultrasound waves are swept to form the image of the soft tissues, like muscles and organs, based on the intensity and time delay of the echoed ultrasound [10]. Ultrasound is also used for monitoring heartbeat and blood flow using external devices outside the body [11].

Industrial Applications: Ultrasound is used in a lot of applications in the industry, for example, it is used in detecting the leakage and gas pipes fracture analysis[12], and cracks in bridges [13].

Underwater Data Communication: Ultrasound waves suffer from much more propagation delay than radio frequency waves since the speed of sound(1480 m/s) is way higher than of its value in air(330 m/s) and both of them are extremely slow compared to radio frequency;in [14], data rates of more than 19 kbps have been demonstrated for acoustic modems using 30 kHz carrier.

Acoustic Communication: Using existing microphones in laptops and mobile devices, a device to device/human communications have been presented in [15] and [16]. Recently, this approach has been commercialized for proximity marketing; proximity information is transmitted to the smart phones using sound signals.

Through-metal Communication: Data communication through metal walls has many applications. For example, in nuclear power plants, Radio waves can not be used for wireless transmitting the sensor information through metal walls of gas cylinders because of the metals shielding. Ultrasound has been used for through-metal communication [17] using commercial components by coupling ultrasonic waves into the wall.

Ultrasonic presence detector microsystems: Fully integrated ultrasonic presence detectors have been proposed [18]. However, they mainly depend on high Q MEMS sensors that need expensive fabrication processes. Moreover, due to the high

Q of those sensors, tuning their characteristics is done by heating. As a result; a relatively large power of several milliwatts is being consumed.

CHAPTER III

ULTRASOUND DATA COMMUNICATION THEORY

3.1 Literature Review

In the 1950s, ultrasound transducers had started to be used for biomedical imaging and for therapy purposes. However, in the recent applications, It is used for data communication for short-range applications. In the quest for a remedy of the previously mentioned daily EM exposure problems; ultrasound has been investigated as a viable candidate for short-range wireless communication over the air to be an alternative to radio frequency (RF).

Previous work had achieved ultrasonic data communication via different modulation schemes including on-off keying (OOK), binary frequency shift keying (BFSK), binary phase shift keying (BPSK) and orthogonal frequency division multiplexing (OFDM) using air-coupled ultrasound transducers to achieve different data rates at different distances. For example; in [19][20][21][22], researchers were trying to achieve ultrasonic communication using OFDM modulation scheme. However, the communication range was lower than 9m. Furthermore, this scheme requires multi-resonance ultrasonic transducer or an expensive wideband ultrasonic transducer. Other related research studies in [23][24] proposed that BPSK was the lowest bit error rate for ultrasound communication, the trial modulation schemes were OOK and BFSK in these schemes. However, implementation of BPSK again requires wideband ultrasonic transducers. Some methods were presented in[25] to estimate and detect the received energy and the estimated distance from an ultrasonic sensor was less than 1m. Researchers in [26] have presented a new statistical method to detect and estimate bits in the presence of ringing noise. However, the realization of this method requires large number of random data frames to be practical. Table 2 summarizes the achieved communication rates and communication ranges.

In this work, a novel sub-sampling long range ultrasound communication was designed and tested. A pair of commercially available low-cost 40 kHz ultrasonic transducers was used to achieve a decent data rate over longer distances within the available resonance bandwidth.

Table 2: Literature Review Comparison.

System	*Distance*	*Bitrate(bit/s)*	*BER*	*Frequency*	*Modulation*
[17]	15.26 cm	0.5 Kbit/s	–	1 MHz	–
[19]	1 m	1 Mbps	–	3.5 MHz	Q-PSK
[20]	6 m	180 Kbps	$3 X 10^{-2}$	55KHz - 99KHz	16-QAM
[21]	8 m	30 Kbps	$5 X 10^{-2}$	50KHz to 110KHz	BPSK
[22]	3 m	56 Kbps	$6.25 X 10^{-5}$	28KHz to 78KHz	OFDM-OOK

3.2 Proposed Design

3.2.1 Pulsed Resonance-OOK modulation technique

As mentioned in the previous chapter, after investigating and implementing various phase and frequency modulation schemes, On-Off keying was concluded to be the most appropriate choice for this particular ultrasound communication system. Bit 1 is modulated in a special format with certain duty cycle short duration pulses rather than continuous stream and transmitted as shown in Figure 7. The received signal after the high gain receiver front-end of the system is, though, a full-blown continuous waveform with slight discontinuity disturbances as shown in Figure 7.

The unique feature in this particular method is that it allows the membrane to release its energy before pulsing it again with train of 40 KHz pulses and hence at the transition to bit 0 the ringing dies out quickly without stretching much into the next bit period. This new scheme is called Pulse-Resonance OOK (PR-OOK) Modulation.

It enables the resonant systems like ultrasound to reach higher reliable data rates with reasonable BER and low power consumption. The power saving of the proposed method is not only due to the shortening of overall transmission time and the ringing prone operation, but also due to duty cycled drive of the transducer. If one to pulse the transducer with 50% duty cycle for example resulting in a similar receive signal profile with respect to the full continuous drive, overall TX power consumption can simply be halved even assuming the same bit period for both cases.

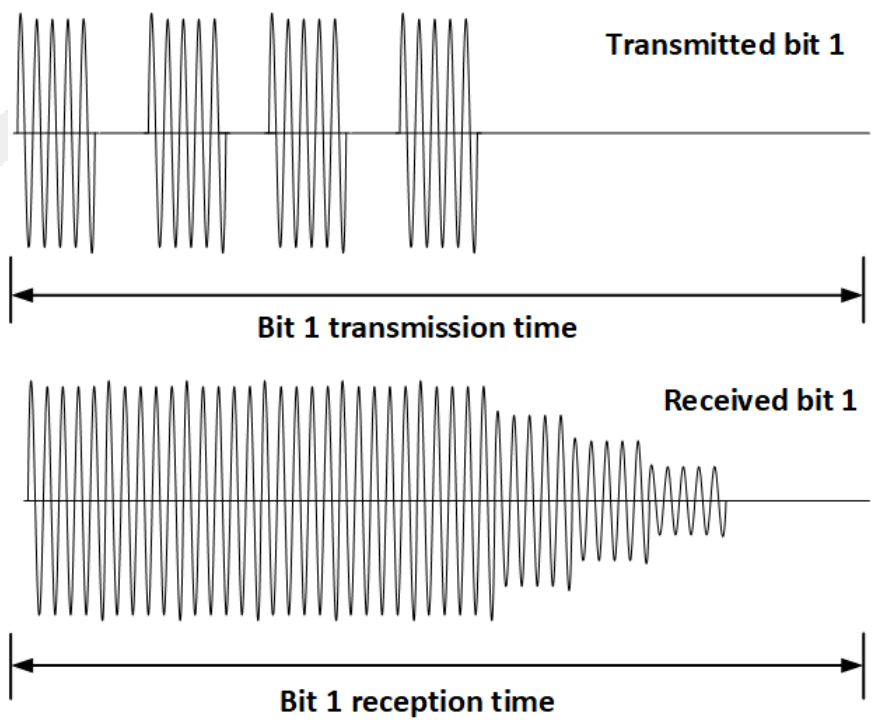

Figure 7: Pulsed Resonance Bit 1.

3.2.2 Sub sampling frequency selection

The received signal, after front-end gain and filtering, reaches the baseband demodulator as shown in Figure 7. The physical implementation of I and Q mixers are based on low power low speed ADC sampling and integration. The ideal full-rate I/O sampling of the received signal would require 160kS/s which is way above the reach of the available ADC in the battery operated low energy system. As a solution, sub-sampling is implemented. The selection of the sampling frequency is based on the following analysis:

$$X(t) = sin(\omega t) \tag{1}$$

$$Samples\ summation = \sum_{i=0}^{n} X(t - nT) \tag{2}$$

Where T is sampling period.

$$4\ samples\ summation = \mid sin(\omega t) - sin(\omega t - 2\omega T) \mid + \mid sin(\omega t - \omega T) - sin(\omega t - 3\omega T) \mid \tag{3}$$

Using trigonometric identities, it can be concluded that:

$$4\ samples\ summation = \mid 4\ X\ cos(\frac{2\pi f}{f_s})\ X\ cos(\frac{\pi f}{f_s})\ X\ sin(\frac{3\pi f}{f_s}) \mid \tag{4}$$

In order to maximize the previous equation;

$$\frac{f}{f_s} = 1.75 - \frac{n}{2} \tag{5}$$

Where n is 0, 1 or 2.

Normally, the integrator can accumulate samples with one of (32 kHz, 53.333 KHz, 160 KHz ...... etc). However, due to ADC speed limitation, this work employed 32 KHz sub-sampling. The unique relation between any of those frequencies and 40 KHz is that; each four samples are composed of different polarity samples of 40 KHz carrier as shown in Figure 8. Figure 9 shows the tested RX bit-period integration result versus the sampling frequency, depicting the best sampling frequencies for the system. Although the sub-sampling degrades the overall SNR by about 6 dB with respect to the 53.33 KHz I/Q sampling, it allows low power low cost realization of the system.

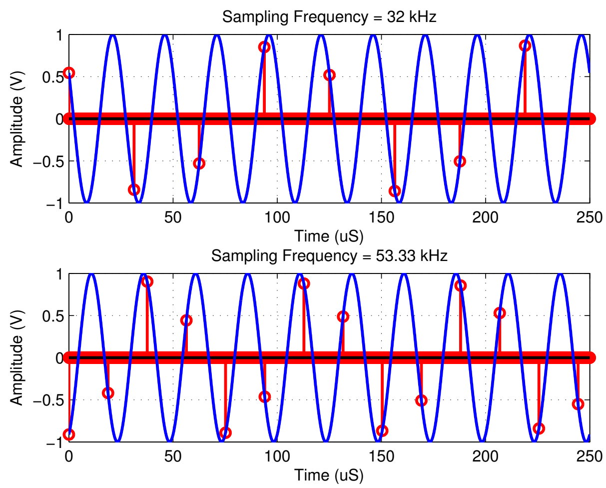

Figure 8: Sampled waveforms.

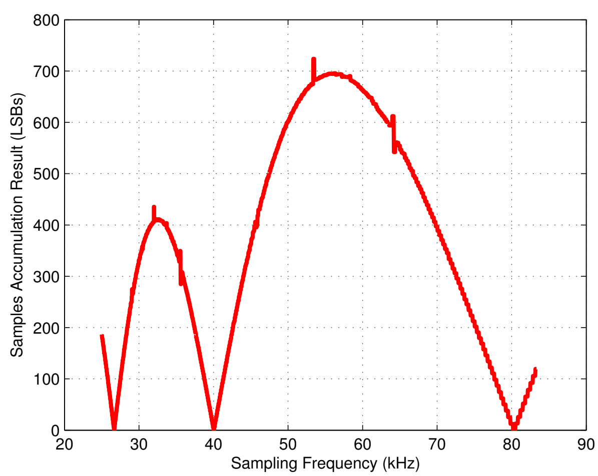

Figure 9: Bit samples integration versus sampling frequency.

3.2.3 Bit width selection

The integration time for any initial sample time interval should be kept reasonable and not to be impacted with that phase/frequency mismatch of the TX and RX. As shown in Figure 10, variance of accumulation due to phase difference should be as minimum as possible. Hence, the minimum bit period should be bigger than 10mS to maintain a good SNR which is already satisfied by the 20 mS to be echo prone detection for the target range of 20m x 20m.

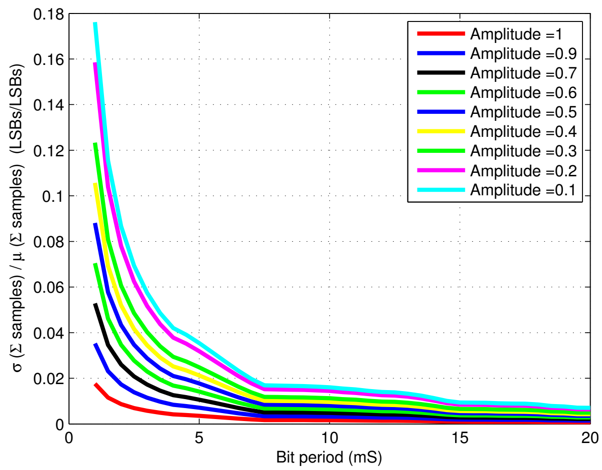

Figure 10: Received accumulation variance with respect to bit period and samples phase differences.

Simple 101 3-bit preamble is needed for precise packet detection, bit period and bit edges detection as well as signal dependent dynamic ringing and echo cancellation at the receive side.

3.2.4 Bit detection and reception

As shown in Figure 11, the integration of samples for the preamble bits is not done over the bit width in one shot then deciding the bit if it is one or zero. The samples integration is done over a bit chip which is very smaller than the decided bit width. By dividing the bit width into smaller bit chips, it allows the reception of bits to be more accurate by estimating the noise levels, implementing bit width detection, ringing suppression and adaptive threshold which will be illustrated in the next sections.

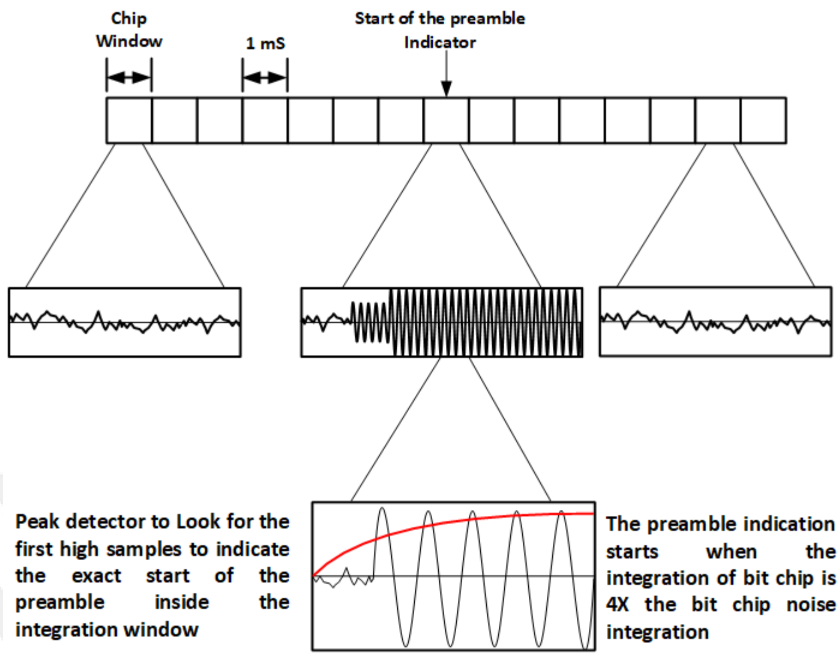

Figure 11: bit detection and reception algorithm.

3.2.4.1 Noise estimation and ringing suppression

By implementing the integration in the case of no data transmitted, this will give an indication for the noise level around the receiver, the integration is for the number of chips needed to give an estimate of the bit width. Now, The receiver has the information for noise per bit chip, integrated noise per total bit width. Those two numbers will be used for the bit detection technique and echoes and ringing suppression. The Echoes/Ringing to Noise Ratio can be decided from the bit zero samples integration and noise information known immediately before reception.

3.2.4.2 Bit width detection

As shown in Figure 11, The receiver then waits for four times larger chip window integration value which indicates that the preamble signal is being received, then by sampling number of the chips that covers the preamble signal. The detection of the bit width and bit edge get relaxed, since the same window edge catching pattern will indicate the change of integration between zero chip or one chip which is used as an indication for exact bit period. Later, the receiver system uses these transition window numbers to allocate the integration intervals accurately at the edges of each bit by software peak detector.

3.2.4.3 Adaptive threshold Implementation

The integration of bit one samples and bit zero samples of the same preamble bits are used to dynamically setting an optimum slicer threshold. This will improve the SNR with respect to the distance. It keeps the slicer level to be almost at the same distance from bit zeros samples integration and bit one samples integration at all the measured distances.

By implementing all these techniques during the bits reception, the system was improved significantly than the regular ultrasound OOK communication system. The results will be shown in Chapter V.

CHAPTER IV

PROPOSED IDEA REALIZATION

4.1 Application definition

The implemented application is a smart thermometer that measures the temperature for the infants. There are plenty of IoT products in the market which uses bluetooth low energy modules(BLE) to communicate with the phone which is dangerous in terms of EM exposure. The proposed application based on the ultrasound data communication will be composed of a transmitter which is connected directly to the sensory node and it transmits the packets to the receiver which includes a Nordic ble (RF device), the receiver gets the bits and send them to the target smart phone. The next section will show the system in details.

4.2 System Design

4.2.1 Transmitter Design

The transmitter as shown in Figure 12 is composed of power management unit which is responsible for charging the battery, 8-pin TI-MSP430G2230 which takes care of the battery measurement and temperature measurement and synchronizing the transmitter to wake up every minute to measure the battery, measure the temperature, transmit the bits and then goes to sleep mode to save the power. If the battery level is not good enough, I mean the battery level should be higher than 2.4v so that the transmitter can transmit the packets. Otherwise, it won't send the packets or do temperature measurement to save the power.

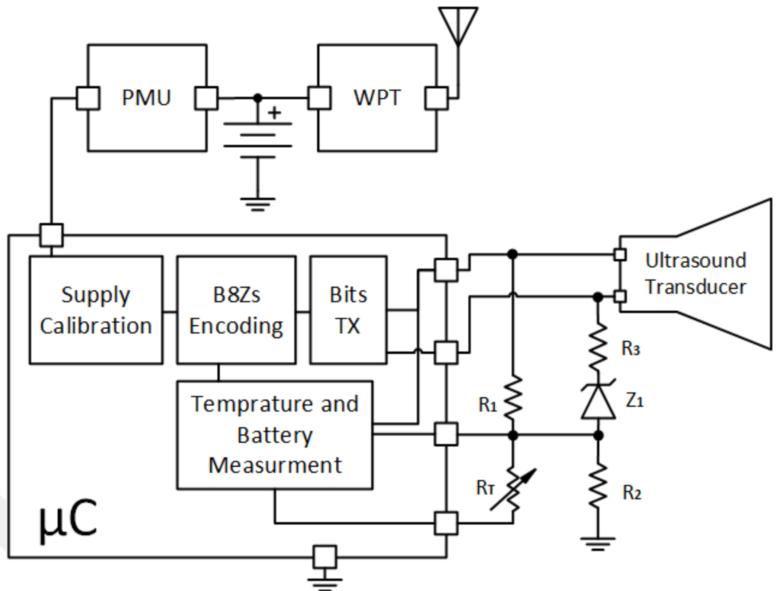

Figure 12: Transmitter system.

4.2.1.1 Battery Measurement

Since the particular controller doesn't have a stable voltage reference embedded, it assumes the supply to be the full-scale reference of the ADC. Hence, the measurement of the battery level can't be possible without having an independent reference. As shown in Figure 13, the proposed design creates a separate reference for the battery by depending on the Zener voltage drop which will be varying according to the logic high voltage value (VDD), and hence the ratio between the measured voltages to the supply as shown in Figure 14 will be used as an indication for the battery level. The current consumption is minimized by using a big value of resistor R_3 to prevent an extra loss path in ultrasound drive mode. In the battery measurement part of the wake-up session, the IOs managing the temperature measurement are driven high impedance (HighZ) which won't disturb the battery measurement.

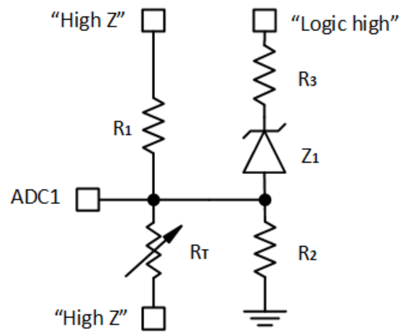

Figure 13: Battery measurement technique.

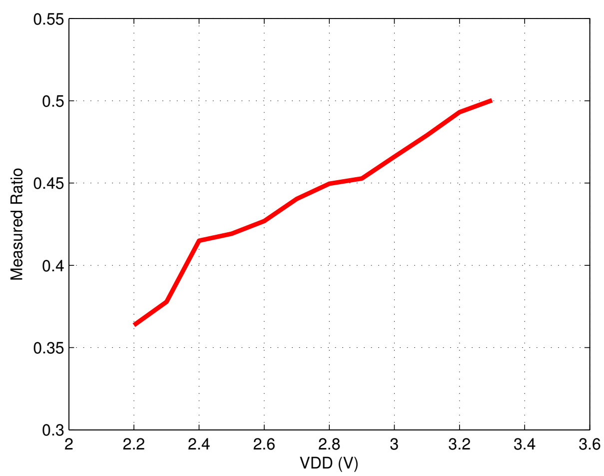

Figure 14: Measured battery ratios.

4.2.1.2 Temperature Measurement

A simple but accurate two-step temperature measurement technique is implemented to measure the temperature again virtually independent of supply variations. The first measurement by the ADC is done by driving one side high and the other side low as shown in Figure 15.A;

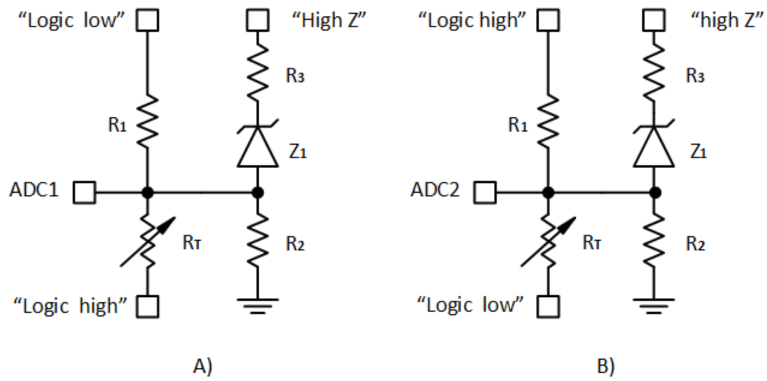

Figure 15: Simple accurate temperature measurement technique.

This will end up with the following equation for the first sample:

$$ADC1 = \frac{R_1}{R_1 + R_\mathrm{T}} * VDD \tag{6}$$

The next measurement by the ADC will be done by flipping the polarity of the driving pin as shown in Figure 15.B; this will give the following equation:

$$ADC2 = \frac{R_\mathrm{T}}{R_1 + R_\mathrm{T}} * VDD \tag{7}$$

By dividing equation (6) and (7) at the end user processor, this will yield a supply independent high precision temperature reading which is merely set by the accuracy of the thermistor and the fixed reference resistor. The result will be supply independent and can be written as the ratio of two readouts:

$$X = \frac{ADC2}{ADC1} = \frac{R_\mathrm{T}}{R_1} \tag{8}$$

By picking R1 as a very low temperature variation (25 ppm/C°) stable reference resistor, the thermistor value can be determined accurately.

4.2.1.3 Supply Calibration

Supply calibration is required before modulating the signals and transmitting them. The measured battery levels are used to estimate the clock frequency and hence adjust the delays to ensure that the transmitted waveforms vary minimal with the supply and the output 40 kHz modulating content and the bit period remain to be precise. The precision in the TX is even more critical to get the most out of 40 kHz resonant transducer. More than a kHz or so frequency offsets cause severe loss at both the TX and RX transducers.

The achieved frequency variation range was 39.5 kHz to 41.3 kHz, this means that there will be variation of 5% in the total bit period during battery discharge cycles. These bit width variations although limited, required an additional preamble bit width detection algorithm in the receiver side to improve the overall system SNR and communication range.

4.2.2 Receiver Design

The receiver system is shown in Figure 16. It is composed of an ultrasound transducer. The ultrasound transducer acts as a narrow band resonator at 40 kHz. Hence, it is a narrow bandpass filter, it passes only 2 kHz around 40 kHz and rejects the other frequencies. Hence, the system may allow quite high gain with some additional filtering along the gain stages assuming no significant noise sources on the board to saturate the chain. The ultrasound transducer is followed by 2 filtering gain stages which amplify the received micro voltages to reasonable voltage levels. Finally the BLE controller takes those voltage levels and implements IQ sub sampling.

The receiver wakes up every 1 minute to receive sense the noise level and waits for the preamble bits to estimate the low levels, high levels, and the bit width. Ater doing these fast calculation, it will receive the bits and it will send them in the form of notification to the connected smart phone.

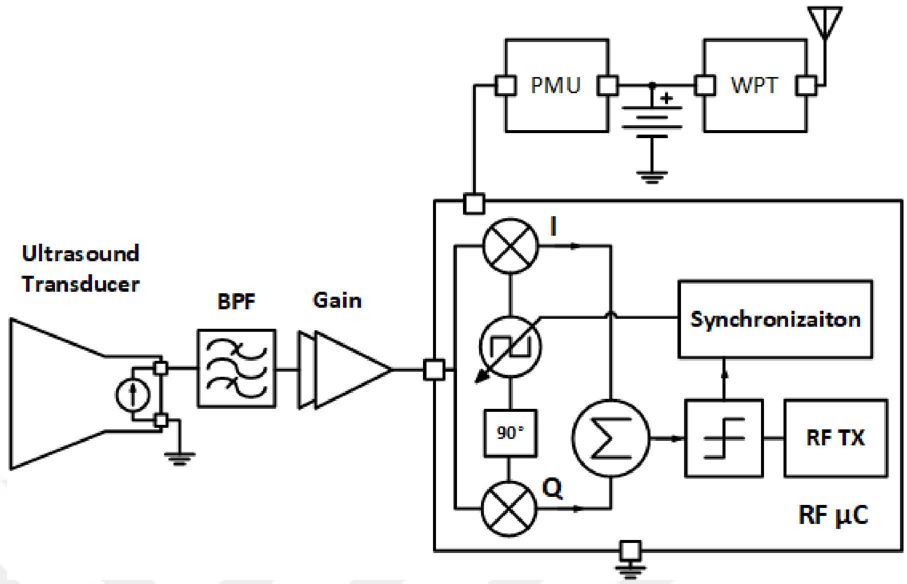

Figure 16: Receiver system.

4.2.2.1 Gain Stage Design

The front-end of the proposed ultrasound receiver system consists of 2 gain and filtering stages as shown in Figure 17. The first low-noise stage is optimized for 30 dB gain without loading the transducer by using pico-Farads capacitor as an input load. The second stage amplifier is also offset free and is designed to yield 46.7 dB gain. The high gain in the second stage required a compensation resistor R_2.

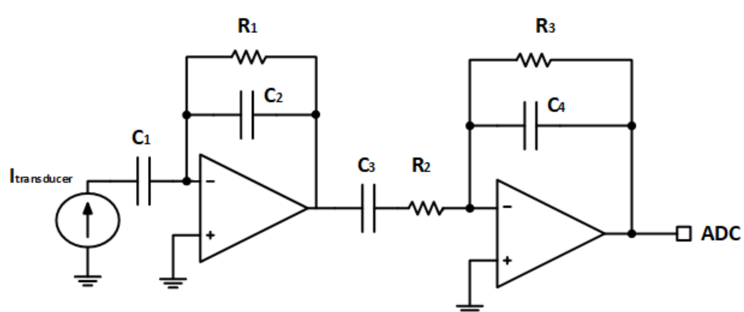

Figure 17: Gain and filtering stages.

After signal amplification, the received peak to peak voltage levels with distance at the ADC input are as shown in Figure 18. The design values are shown in Table 3.

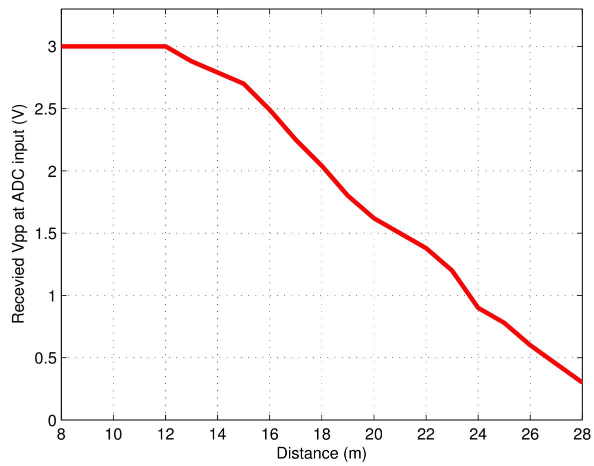

Figure 18: Received signal peak to peak after amplification.

Table 3: Gain Stages R & C values.

ComponentName	*Value*
C_1	180 pf
C_2	0.5 pf
C_3	3.3 nf
C_4	0.5 pf
R_1	300 kΩ
R_2	1 kΩ
R_3	400 kΩ

4.2.2.2 Sub sampling mixer implementation

The ADC of Nordic BLE controller has a maximum sampling frequency of 50 KHz. Hence, the sampling frequency was selected to be 32 KHz in order to get the IQ behavior. The adjustment of delays before and after the ADC samples insures that The sampling frequency is almost equivalent to 32 KHz. The reason of using delays is that the ADC sampling frequency is fixed and the time it takes to finish one sample (25 μS) will be included in the sampling time.

4.2.3 Power Management Units

In order to make the solution water resistant, it has to be fully covered which means that the programming of the ble should be done by firmware over the air bootloading(wireless bootloader). The power management unit should be designed to be wireless as well. Basically, The regular wireless chargers on the market are QI standard. The used one was resonating at 175KHz. So, The receiver antenna was tuned as well to have resonance at that frequency, by converting the received AC signal into DC signal using a full wave rectifier followed by an LDO (low drop-out regulator) to generate the required voltage and currents to charge the battery. Zener diodes were used for over voltage protection. Schottky diode was used to prevent reverse leakage of the battery.

4.2.4 Layout

The layout of the transmitter and receiver is shown in Figure 19, total area of the transmitter is almost 12 mm, both transmitter and receiver are completely covered for being water resistant.

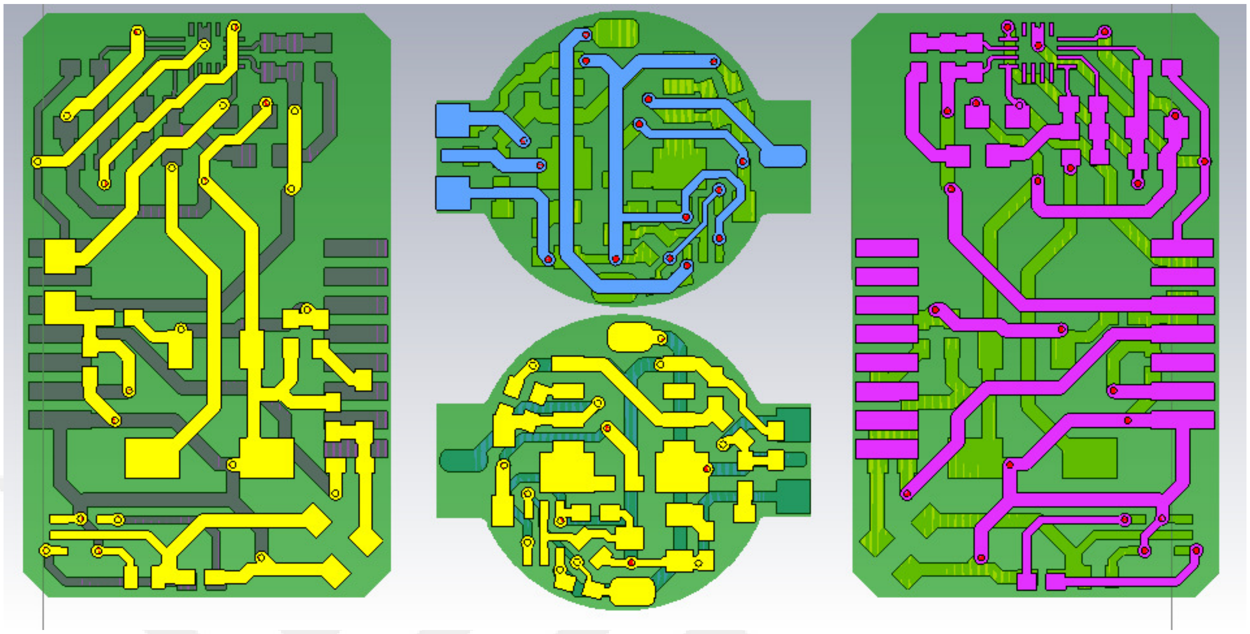

Figure 19: PCB Layout.

CHAPTER V

RESULTS

The proposed techniques effectivenesses were all tested and proven during the sensitivity and range tests. The ringing and echoes impact is improved by 5 times as shown in Figure 20.

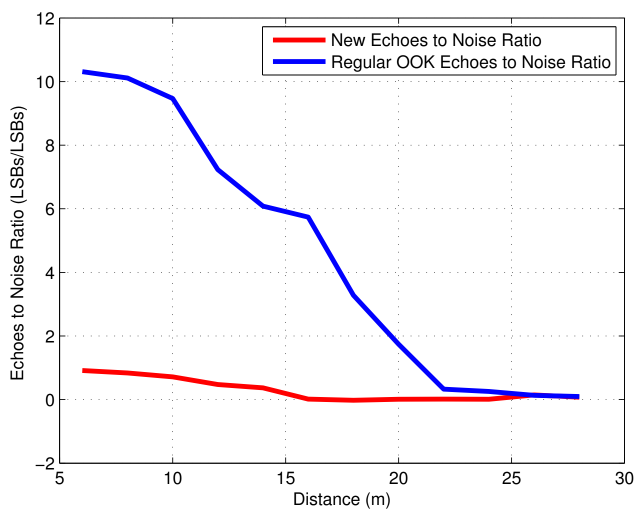

Figure 20: Echoes to Noise Ratio.

Overall system difference between bit "0" and bit "1" is widened by 1.5x ratio as shown in Figure 21.

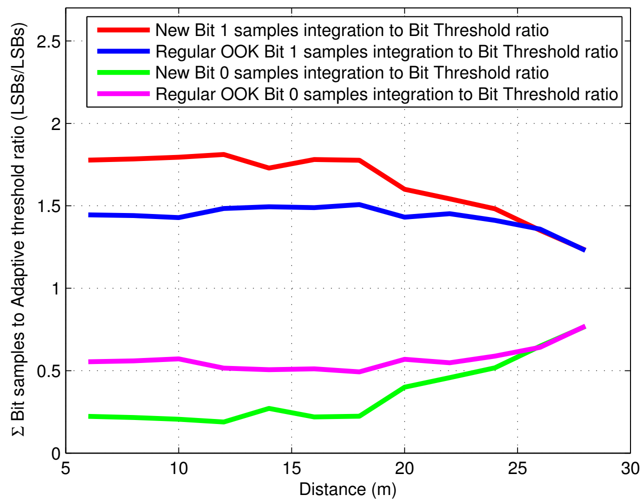

Figure 21: Samples accumulation to Adaptive threshold ratio.

The effect of the dynamic threshold technique is shown in Figure 22. Adaptive threshold placement is applied while detecting the received data to maintain the noise margin high for various distances.

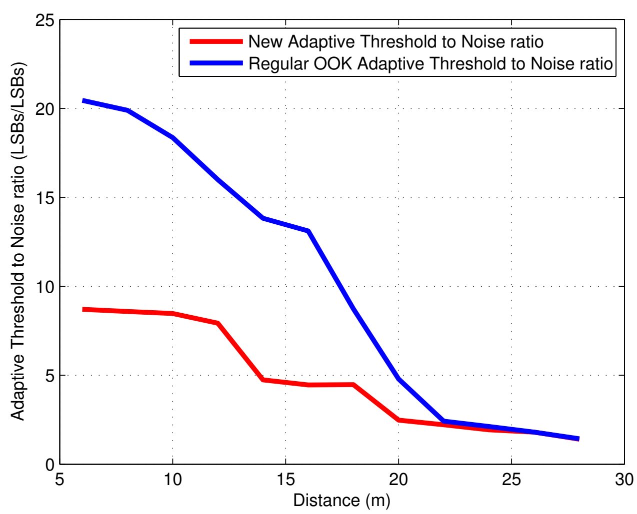

Figure 22: Adaptive Threshold to Noise ratio.

The improvement in maintaining the integration values for the long bits transmission and reception due to bit width detection algorithm is well-explained in Figure 23 and Figure 24.

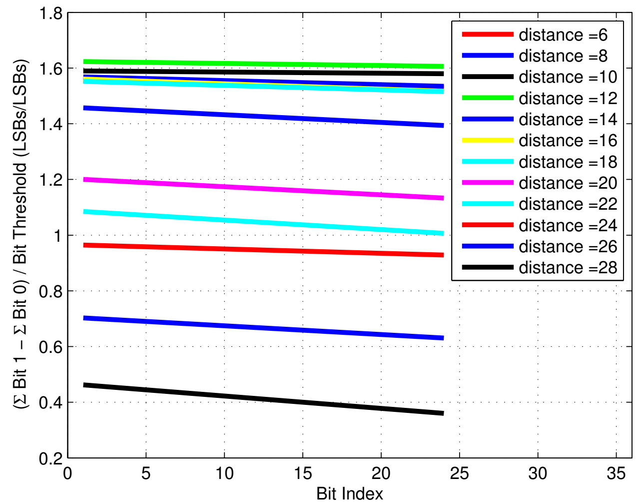

Figure 23: Bit accumulation variance to threshold ratio with bit width detection at different distances.

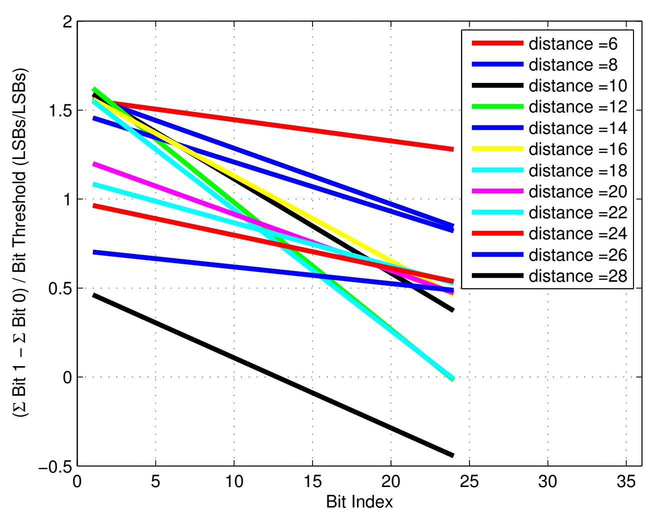

Figure 24: Bit accumulation variance to threshold ratio without bit width detection at different distances.

The achieved bit error rate for the target distance was 0.01 compared to the regular system, it was improved by almost 10x as shown in Figure 25 and Figure 26.

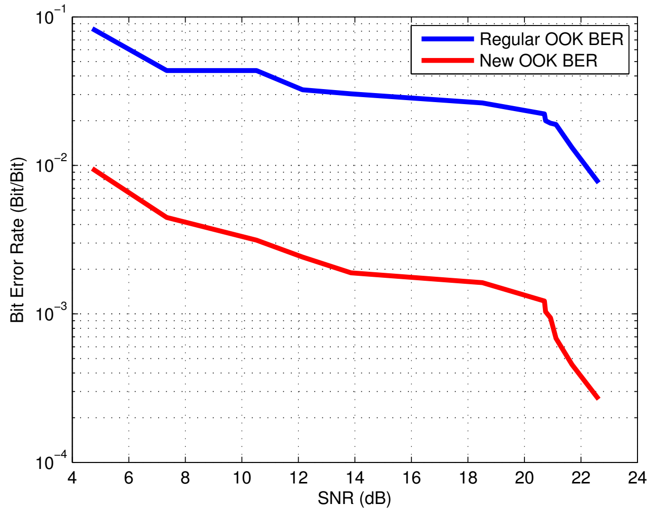

Figure 25: BER vs SNR.

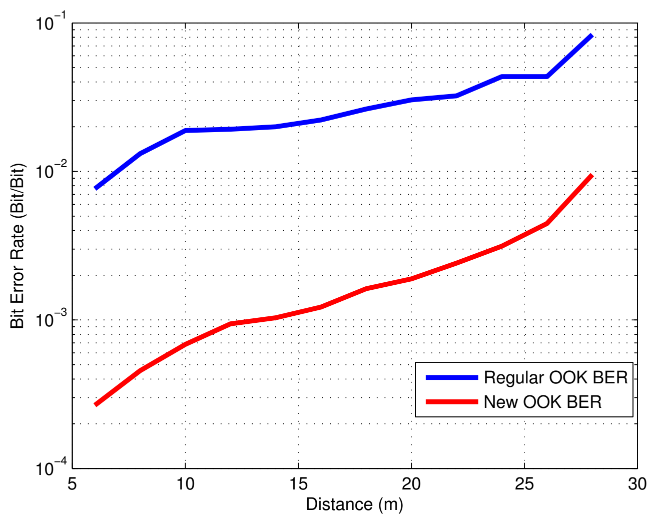

Figure 26: BER vs Distance.

The comparison between the proposed system and the literature is presented in table 4.

Table 4: New System results.

System	*Distance*	*Bitrate(bit/s)*	*BER*	*Frequency*	*Modulation*
[17]	15.26 cm	0.5 Kbit/s	–	1 MHz	–
[19]	1 m	1 Mbps	–	3.5 MHz	Q-PSK
[20]	6 m	180 kbps	3×10^{-2}	55kHz - 99kHz	16-QAM
[21]	8 m	30 kbps	5×10^{-2}	50KHz - 110kHz	BPSK
[22]	3 m	56 Kbps	6.25×10^{-5}	28KHz - 78KHz	OFDM-OOK
This Work	28m	50b/s	1×10^{-2}	39.5KHz - 41.5KHz	OOK

The transmitter power consumption while sleeping is 0.4 uA, and average while transmitting is 400uA, the total average power will be 0.406 uA since it is in the sleep mode most of the time. The receiver power will be regular BLE which is consumed by the regular IoT application. The estimated lifetime for 8 mAH LiR battery is nearly 27 months.

The extra cost in mass production to the regular IoT solution can be shown in Table 5. The resistors and capacitor cost are negligible.

Table 5: Extra Cost.

Component	*PartName*	*Price*
Ultrasound Transducer	—	2 X 0.2 USD
Dual Op-Amp chip	OPA2363AIRSVR	1.37 USD
Micro Controller	TI-MSP430G2230	1.3 USD

CHAPTER VI

CONCLUSION AND FUTURE WORK

In this dissertation, a complete ultrasound communication system along with associated sensor, wireless power transfer and power management interfaces were presented, targeting medical IoT sensor applications. Understanding the system deficiencies and limitations well, some unique approaches were tested, leading to a robust, long range healthy data transmission subsystem for a larger water-proof wireless charging high accuracy temperature sensor. Thanks to the mentioned techniques, air-coupled ultrasound data transmission distance is improved by 3x relative to the existing systems in the literature. The PR-OOK digital modulation technique was the main driver for the improved echoes and ringing performance. The preamble based bit width detection algorithm at the receiver as well had improved noise margin to be the same during the reception of the data, the achieved bit error rate is 0.01 for 50 bit/sec. The dynamic level was starting from 0.3vpp to 1.5Vpp. both transmitter and receiver are powered wireless and the power receivers in both subsystems are tuned to receive power at around 175 kHz from commercially available standard Qi wireless charger systems.

In the first chapter, the most important ultrasound characteristics for data communication application were illustrated. some commercial ultrasound applications were presented. In the second chapter, a literature review on ultrasound in data communication application showed that the maximum achieved communication range was 8m. A new solution for echoes and ringing was proposed. in the third chapter, the realization of the proposed idea was discussed and finally the results showed a significant improvement in terms of communication range and noise margins for ultrasound communication systems.The extra cost and power is considered negligible when it is used in Iot applications.

The transmitter and receiver will be completely enclosed with silicon insulation along with the 12-mm diameter tiny LiR batteries and after that the sensor measurement accuracy will be characterized.

APPENDIX A

SOME ANCILLARY STUFF

A.0.1 Transmitter main code

```
/* Main Function--------------------------------------------*/
void main(void)
{
  ConfigWDT();            //Configuring the WatchDog Timer
  ConfigClocks();         //Configuring the Clock
  ConfigTimerA();         //Configuring Timer A
  ConfigADC10();          //Configuring the ADC
  InitVar();              //Initializing variables
  while(1)
  {
      // battery measurement
      P1DIR |= 0x80;
      P1OUT|=0x80; //Initializing P1.7=1=VDD and P1.2=0=GND
      _delay_cycles(900);
      _delay_cycles(900);
      _delay_cycles(900);
      ADC10CTL0 |= ENC + ADC10SC; //Sampling and conversion start
      _delay_cycles(400);//Giving the ADC time to convert the sample
      adc3=ADC10MEM;
      //Reading the converted sample in the Conversion Memory Register ADC10MEM into adc3 (Battery Measurement reading)
      if( adc3 >= min_bat_level)// min_bat_level
      {
      bat_level = (adc3- min_bat_level) >>3;
      // change the pins to be outputs again and carry on.
```

```
    P1DIR &= 0x00;
    P1DIR |= 0x7F;
    P1OUT &= 0x00; //Initializing P1 to zero
    P1OUT |= 0x40; //Initializing P1.6=1=VDD and P1.2=0=GND
    ADC10CTL0 |= ENC + ADC10SC; //Sampling and conversion start
    _delay_cycles(200);//Giving the ADC time to convert the sample
    adc1=ADC10MEM;
    //Reading the converted sample in the Conversion Memory Register ADC10MEM into adc1 (1st temperature reading)
    P1OUT ^= BIT2;  //Toggling P1.2
    P1OUT ^= BIT6;  //Toggling P1.6
    ADC10CTL0 |= ENC + ADC10SC;    //Sampling and conversion start
    _delay_cycles(200);    //Giving the ADC time to convert
    adc2=ADC10MEM;
    //Reading the converted sample in the Conversion Memory Register ADC10MEM into adc2 (2nd temperature reading)
    r[25]=0x01; //Start bit/flag
    r[26]=0x00;
    r[27]=0x01;
    //Filling the vector r element by element with the 10 bits infant's temperature reading starting by the Most Significant Bit MSB
    bit_num = 0;
    ADC10CTL0 = 0x0000;
    while(bit_num < ((num_of_data_bits - 4)/2) )
    {
        if((adc1 & 0x0200) == 0x00000000)//Comparing the MSB of temp_reading with 1
            r[bit_num+1]=0x00;
        else
            r[bit_num+1]=0x01;
        adc1=adc1 << 1; //Shifting temp_reading to the left by one
        bit_num++;
```

```
    }
    while(bit_num< (num_of_data_bits - 4 ) )
    {
        if((adc2 & 0x0200) == 0x00000000)//Comparing the MSB of temp_reading with 1
            r[bit_num+1]=0x00;//0
        else
            r[bit_num+1]=0x01;
        adc2=adc2 << 1; //Shifting temp_reading to the left by one
        bit_num++;
    }
    while(bit_num< num_of_data_bits )
    {
        if((bat_level & 0x01) == 0x00000000)//Comparing the MSB of Battery_reading with 1
            r[bit_num+1]=0x00;//0
        else
            r[bit_num+1]=0x01;
        bat_level=bat_level >> 1; //Shifting bat_level to the right by one
        bit_num++;
    }
    // transmitting the bits to the ultrasound pins.
    P1DIR |=0x80;
    P1OUT|=0x42;
    // Initializing P1.7 and P1.6 to differentially output the data.
    // pin 2 is initialized to be similar to pin 6 due to power optimization of the board.
    bit_num=27;
    // bits are transmitted from bit 27 to optimize the conditions in the loop to be able to minimize the operating frequency
    while(bit_num)
```

```
        {
            if(r[bit_num])
              {
                for(i=10; i  ;i--)
                {
                  for(count=pulse_time_one; count    ; count--)
                     {
                       P1OUT  = ~P1OUT;
                       _delay_cycles(10);
                     }
                  for(count=pulse_time_zero; count   ; count--)
                     {
                       P1OUT &= ~BIT7 | ~BIT6 | ~ BIT2;
                       _delay_cycles(8);
                     }
                }
                  for(count=pulse_time_half; count   ; count--)
                     {
                       P1OUT &= ~BIT7 | ~BIT6 | ~ BIT2;
                       _delay_cycles(8);
                     }
              }
            else
              {
                for(count=pulse_time; count   ; count--)
                    {
                      P1OUT &= ~BIT7 | ~BIT6 | ~ BIT2;
                      _delay_cycles(8);
                    }
              }
             P1OUT|=0x42;
             bit_num--;
```

```
            }
            P1OUT &= ~BIT7 | ~BIT6 | ~ BIT2;
            bit_num=27;
            P1DIR &= 0x00;
        }
        _bis_SR_register(LPM3_bits + GIE);    //Enter Low Power Mode LPM3 with interrupts

    }
}
```

A.0.2 Receiver BLE Service

```
#include "ble_Temp.h"
#include <string.h>
#include "nordic_common.h"
#include "ble_srv_common.h"
#include "app_util.h"
#include "nrf_gpio.h"


#define INVALID_temp_LEVEL 255


/**@brief Function for handling the Connect event.
 *
 * @param[in]   p_temp       temp Service structure.
 * @param[in]   p_ble_evt    Event received from the BLE stack.
 */
static void on_connect(ble_temp_t * p_temp, ble_evt_t * p_ble_evt)
{
    p_temp->conn_handle = p_ble_evt->evt.gap_evt.conn_handle;
}

```

```

/**@brief Function for handling the Disconnect event.
 *
 * @param[in]   p_temp       temp Service structure.
 * @param[in]   p_ble_evt   Event received from the BLE stack.
 */
static void on_disconnect(ble_temp_t * p_temp, ble_evt_t * p_ble_evt)
{
    UNUSED_PARAMETER(p_ble_evt);
    p_temp->conn_handle = BLE_CONN_HANDLE_INVALID;
}


/**@brief Function for handling the Write event.
 *
 * @param[in]   p_temp       temp Service structure.
 * @param[in]   p_ble_evt   Event received from the BLE stack.
 */
static void on_write(ble_temp_t * p_temp, ble_evt_t * p_ble_evt)
{
    if (p_temp->is_notification_supported)
    {
        ble_gatts_evt_write_t * p_evt_write = &p_ble_evt->evt.gatts_evt.params.write;

        if (
            (p_evt_write->handle == p_temp->temp_level_handles.cccd_handle)
            &&
            (p_evt_write->len == 2)
           )
        {
```

```
            // CCCD written, call application event handler
            if (p_temp->evt_handler != NULL)
            {
                ble_temp_evt_t evt;

                if (ble_srv_is_notification_enabled(p_evt_write->data))
                {
                    evt.evt_type = BLE_temp_EVT_NOTIFICATION_ENABLED;
                }
                else
                {
                    evt.evt_type = BLE_temp_EVT_NOTIFICATION_DISABLED;
                }

                p_temp->evt_handler(p_temp, &evt);
            }
        }
    }
}


void ble_temp_on_ble_evt(ble_temp_t * p_temp, ble_evt_t * p_ble_evt)
{
    if (p_temp == NULL || p_ble_evt == NULL)
    {
        return;
    }

    switch (p_ble_evt->header.evt_id)
    {
        case BLE_GAP_EVT_CONNECTED:
            on_connect(p_temp, p_ble_evt);
```

```
            break;

        case BLE_GAP_EVT_DISCONNECTED:
            on_disconnect(p_temp, p_ble_evt);
            break;

        case BLE_GATTS_EVT_WRITE:
            on_write(p_temp, p_ble_evt);
            break;

        default:
            // No implementation needed.
            break;
    }
}


/**@brief Function for adding the temp Level characteristic.
 *
 * @param[in]   p_temp        temp Service structure.
 * @param[in]   p_temp_init   Information needed to initialize the
    service.
 *
 * @return      NRF_SUCCESS on success, otherwise an error code.
 */
static uint32_t temp_level_char_add(ble_temp_t * p_temp, const
    ble_temp_init_t * p_temp_init)
{
    uint32_t            err_code;
    ble_gatts_char_md_t char_md;
    ble_gatts_attr_md_t cccd_md;
    ble_gatts_attr_t    attr_char_value;
```

```
    ble_uuid_t           ble_uuid;
    ble_gatts_attr_md_t  attr_md;
    uint8_t              initial_temp_level;
    uint8_t              encoded_report_ref[BLE_SRV_ENCODED_REPORT_REF_LEN];
    uint8_t              init_len;

    // Add temp Level characteristic
    if (p_temp->is_notification_supported)
    {
        memset(&cccd_md, 0, sizeof(cccd_md));

        // According to temp_SPEC_V10, the read operation on cccd should be possible without
        // authentication.
        BLE_GAP_CONN_SEC_MODE_SET_OPEN(&cccd_md.read_perm);
        cccd_md.write_perm = p_temp_init->temp_level_char_attr_md.cccd_write_perm;
        cccd_md.vloc       = BLE_GATTS_VLOC_STACK;
    }

    memset(&char_md, 0, sizeof(char_md));

    char_md.char_props.read   = 1;
    char_md.char_props.notify = (p_temp->is_notification_supported) ? 1 : 0;
    char_md.p_char_user_desc  = NULL;
    char_md.p_char_pf         = NULL;
    char_md.p_user_desc_md    = NULL;
    char_md.p_cccd_md         = (p_temp->is_notification_supported) ? &cccd_md : NULL;
    char_md.p_sccd_md         = NULL;
```

```
    ble_uuid.type = p_temp->uuid_type;
    ble_uuid.uuid = UUID_TEMP_CHAR;

    memset(&attr_md, 0, sizeof(attr_md));

    attr_md.read_perm  = p_temp_init->temp_level_char_attr_md.read_perm;
    attr_md.write_perm = p_temp_init->temp_level_char_attr_md.write_perm;
    attr_md.vloc       = BLE_GATTS_VLOC_STACK;
    attr_md.rd_auth    = 0;
    attr_md.wr_auth    = 0;
    attr_md.vlen       = 0;

    initial_temp_level = p_temp_init->initial_temp_level;

    memset(&attr_char_value, 0, sizeof(attr_char_value));

    attr_char_value.p_uuid    = &ble_uuid;
    attr_char_value.p_attr_md = &attr_md;
    attr_char_value.init_len  = 5*sizeof(uint8_t);
    attr_char_value.init_offs = 0;
    attr_char_value.max_len   = 5*sizeof(uint8_t);
    attr_char_value.p_value   = &initial_temp_level;

    err_code = sd_ble_gatts_characteristic_add(p_temp->service_handle, &char_md,
                                               &attr_char_value,
                                               &p_temp->temp_level_handles);
    if (err_code != NRF_SUCCESS)
    {
```

```
        return err_code;
    }

    if (p_temp_init->p_report_ref != NULL)
    {
        // Add Report Reference descriptor
        BLE_UUID_BLE_ASSIGN(ble_uuid, BLE_UUID_REPORT_REF_DESCR);

        memset(&attr_md, 0, sizeof(attr_md));

        attr_md.read_perm = p_temp_init->temp_level_report_read_perm;
        BLE_GAP_CONN_SEC_MODE_SET_NO_ACCESS(&attr_md.write_perm);

        attr_md.vloc    = BLE_GATTS_VLOC_STACK;
        attr_md.rd_auth = 0;
        attr_md.wr_auth = 0;
        attr_md.vlen    = 0;

        init_len = ble_srv_report_ref_encode(encoded_report_ref,
    p_temp_init->p_report_ref);

        memset(&attr_char_value, 0, sizeof(attr_char_value));

        attr_char_value.p_uuid    = &ble_uuid;
        attr_char_value.p_attr_md = &attr_md;
        attr_char_value.init_len  = init_len;
        attr_char_value.init_offs = 0;
        attr_char_value.max_len   = attr_char_value.init_len;
        attr_char_value.p_value   = encoded_report_ref;

        err_code = sd_ble_gatts_descriptor_add(p_temp->
    temp_level_handles.value_handle,
```

```
                                                        &attr_char_value,
                                                        &p_temp->
    report_ref_handle);
        if (err_code != NRF_SUCCESS)
        {
            return err_code;
        }
    }
    else
    {
        p_temp->report_ref_handle = BLE_GATT_HANDLE_INVALID;
    }

    return NRF_SUCCESS;
}


uint32_t ble_temp_init(ble_temp_t * p_temp, const ble_temp_init_t *
    p_temp_init)
{
    if (p_temp == NULL || p_temp_init == NULL)
    {
        return NRF_ERROR_NULL;
    }

    uint32_t   err_code;
    ble_uuid_t ble_uuid;

    // Initialize service structure
    p_temp->evt_handler                 = p_temp_init->evt_handler;
    p_temp->conn_handle                 = BLE_CONN_HANDLE_INVALID;
```

```
    p_temp->is_notification_supported = 1;//p_temp_init->
        support_notification;
    p_temp->temp_level_last[0]              = INVALID_temp_LEVEL;
    p_temp->temp_level_last[1]              = INVALID_temp_LEVEL;
    p_temp->temp_level_last[2]              = INVALID_temp_LEVEL;
    p_temp->temp_level_last[3]              = INVALID_temp_LEVEL;
    p_temp->temp_level_last[4]              = INVALID_temp_LEVEL;

    ble_uuid128_t base_uuid = {TEMP_UUID_BASE};
    err_code = sd_ble_uuid_vs_add(&base_uuid, &p_temp->uuid_type);

    ble_uuid.type = p_temp->uuid_type;
    ble_uuid.uuid = TEMP_SERVICES_UUID;

    err_code = sd_ble_gatts_service_add(BLE_GATTS_SRVC_TYPE_PRIMARY, &
        ble_uuid, &p_temp->service_handle);
    if (err_code != NRF_SUCCESS)
    {
        return err_code;
    }
    // Add temp level characteristic
    return temp_level_char_add(p_temp, p_temp_init);
}


uint32_t ble_temp_temp_level_update(ble_temp_t * p_temp, uint8_t
    temp_level[])
{
    if (p_temp == NULL)
    {
        return NRF_ERROR_NULL;
    }
```

```
    uint32_t err_code = NRF_SUCCESS;
    ble_gatts_value_t gatts_value;

    if (true)//temp_level != p_temp->temp_level_last)
    {
        // Initialize value struct.
        memset(&gatts_value, 0, sizeof(gatts_value));

        gatts_value.len     = 5*sizeof(uint8_t);
        gatts_value.offset  = 0;
        gatts_value.p_value = temp_level;

        // Update datatempe.
        err_code = sd_ble_gatts_value_set(p_temp->conn_handle,
                                          p_temp->temp_level_handles.value_handle,
                                          &gatts_value);
        if (err_code == NRF_SUCCESS)
        {
            // Save new temp value.
            p_temp->temp_level_last[0] = temp_level[0];
            p_temp->temp_level_last[1] = temp_level[1];
            p_temp->temp_level_last[2] = temp_level[2];
            p_temp->temp_level_last[3] = temp_level[3];
            p_temp->temp_level_last[4] = temp_level[4];

        }
        else
        {
            return err_code;
        }
```

```
        }

        // Send value if connected and notifying.
        if ((p_temp->conn_handle != BLE_CONN_HANDLE_INVALID) && p_temp->is_notification_supported)
        {
            ble_gatts_hvx_params_t hvx_params;

            memset(&hvx_params, 0, sizeof(hvx_params));

            hvx_params.handle = p_temp->temp_level_handles.value_handle;
            hvx_params.type   = BLE_GATT_HVX_NOTIFICATION;
            hvx_params.offset = gatts_value.offset;
            hvx_params.p_len  = &gatts_value.len;
            hvx_params.p_data = gatts_value.p_value;

            err_code = sd_ble_gatts_hvx(p_temp->conn_handle, &hvx_params);
        }
        else
        {
            err_code = NRF_ERROR_INVALID_STATE;
        }
   // }  //notify if level changes only

    return err_code;
}
```

A.0.3 Receiver BLE main Functions

```
#define num_bit                 24
#define number_samples          680
#define number_samples_noise    640
#define number_samples_min_2m   18*32
```

```
#define number_samples_div     32  // wtih 32k frequency this gives 1m of window
#define threshold              0x70
#define threshold_bit          0x70

uint16_t k, Sample_I, Sample_Q,delay, thres_I, thres_Q , bit_width, Sample_I_noise, Sample_Q_noise;
uint8_t  sample_I_temp[number_samples];
uint8_t  j , bit_decision[num_bit];
uint16_t Sample_I_bits[num_bit], Sample_Q_bits[num_bit],Sample_I_bitd[2], Sample_Q_bitd[2];
uint8_t  flag, data2send[4];
uint32_t Temp_readout;
/**@brief Function for initializing temp Service.
 */
static void temp_init(void)
{
    uint32_t       err_code;
    ble_temp_init_t temp_init_obj;

    memset(&temp_init_obj, 0, sizeof(temp_init_obj));

    temp_init_obj.evt_handler          = NULL;
    temp_init_obj.support_notification = true;
    temp_init_obj.p_report_ref         = NULL;
    temp_init_obj.initial_temp_level   = 0;

    BLE_GAP_CONN_SEC_MODE_SET_OPEN(&temp_init_obj.temp_level_char_attr_md.cccd_write_perm);
    BLE_GAP_CONN_SEC_MODE_SET_OPEN(&temp_init_obj.temp_level_char_attr_md.read_perm);
```

```
    BLE_GAP_CONN_SEC_MODE_SET_OPEN(&temp_init_obj.temp_level_char_attr_md.write_perm);

    BLE_GAP_CONN_SEC_MODE_SET_OPEN(&temp_init_obj.temp_level_report_read_perm);

    err_code = ble_temp_init(&m_temp, &temp_init_obj);

    APP_ERROR_CHECK(err_code);
}
static void services_init(void)
{
  uint32_t err_code;
  dis_init();
  temp_init();
  #ifdef BLE_DFU_APP_SUPPORT
      ble_dfu_init_t   dfus_init;
      // Initialize the Device Firmware Update Service.
      memset(&dfus_init, 0, sizeof(dfus_init));
      dfus_init.evt_handler   = dfu_app_on_dfu_evt;
      dfus_init.error_handler = NULL;
      dfus_init.evt_handler   = dfu_app_on_dfu_evt;
      dfus_init.revision      = DFU_REVISION;
      err_code = ble_dfu_init(&m_dfus, &dfus_init);
      APP_ERROR_CHECK(err_code);

      dfu_app_reset_prepare_set(reset_prepare);
      dfu_app_dm_appl_instance_set(m_app_handle);
  #endif
}
```

```
void noise_estimate()
{
        Sample_I=0; Sample_Q=0;
        NRF_ADC->CONFIG     = (ADC_CONFIG_RES_8bit
 << ADC_CONFIG_RES_Pos)        |
                              (
 ADC_CONFIG_INPSEL_AnalogInputOneThirdPrescaling <<
 ADC_CONFIG_INPSEL_Pos)   |
                              (ADC_CONFIG_REFSEL_SupplyOneThirdPrescaling <<
  ADC_CONFIG_REFSEL_Pos)   |
                              (ADC_CONFIG_PSEL_AnalogInput4              <<
  ADC_CONFIG_PSEL_Pos)      |
                              (ADC_CONFIG_EXTREFSEL_None                 <<
  ADC_CONFIG_EXTREFSEL_Pos);
        NRF_ADC->EVENTS_END = 0;
        NRF_ADC->ENABLE       = ADC_ENABLE_ENABLE_Enabled;

        for (k=0; k < number_samples_noise ; k++)
        {
        //  NRF_ADC->EVENTS_END  = 0;     // Stop any running conversions
 .
          NRF_ADC->TASKS_START = 1;
          while (!NRF_ADC->EVENTS_END)
          {
          }
          sample_I_temp[k] = NRF_ADC->RESULT;
          NRF_ADC->EVENTS_END       = 0;
          NRF_ADC->TASKS_STOP       = 1;
          nrf_delay_us(7);
        }

        for (k=0; k < number_samples_noise ; k = k+4)
```

```
        {
                if( sample_I_temp[k] >= sample_I_temp[k+2])
                  Sample_I =Sample_I + sample_I_temp[k] - sample_I_temp[k+2];
                else
                  Sample_I =Sample_I + sample_I_temp[k+2] - sample_I_temp[k];

                if( sample_I_temp[k+1] >= sample_I_temp[k+3])
                  Sample_Q =Sample_Q + sample_I_temp[k+1] - sample_I_temp[k+3];
                else
                  Sample_Q =Sample_Q + sample_I_temp[k+3] - sample_I_temp[k+1];

        }
        Sample_I_noise=Sample_I;
        Sample_Q_noise=Sample_Q;
}


void bit_detect()
{
    uint32_t err_code;
    do{
  //     nrf_gpio_pin_toggle(0x05);
        for (k=0; k < number_samples_div ; k++)
        {
          //NRF_ADC->EVENTS_END = 0;     // Stop any running conversions.
          NRF_ADC->TASKS_START = 1;
          while (!NRF_ADC->EVENTS_END)
```

```
            {
            }
            sample_I_temp[k] = NRF_ADC->RESULT;
            NRF_ADC->EVENTS_END     = 0;
            NRF_ADC->TASKS_STOP     = 1;
            nrf_delay_us(7);
        }
        Sample_I = 0; Sample_Q = 0;
        for (k=0; k < number_samples_div ; k = k+4)
        {
                if( sample_I_temp[k] >= sample_I_temp[k+2])
                  Sample_I =Sample_I + sample_I_temp[k] - sample_I_temp[k+2];
                else
                  Sample_I =Sample_I + sample_I_temp[k+2] - sample_I_temp[k];

                if( sample_I_temp[k+1] >= sample_I_temp[k+3])
                  Sample_Q =Sample_Q + sample_I_temp[k+1] - sample_I_temp[k+3];
                else
                  Sample_Q =Sample_Q + sample_I_temp[k+3] - sample_I_temp[k+1];
        }
    }while(Sample_Q  < threshold && Sample_I < threshold);
    // estimate of the peak
    for(k=0; sample_I_temp[k] < 0x88 ; k++); // || sample_I_temp[k] > 0x75
    //j=k;

    nrf_delay_us(1000 + (k -3) * 31);
 //  nrf_delay_ms(18);// change that number
```

```
}


void calc_thres()
{
    thres_Q=0; thres_I=0;
    for (j = 0; j < 2 ; j++)
    {
  //    nrf_gpio_pin_toggle(0x05);
      for (k=0; k < number_samples_min_2m  ; k++)
      {
      //  NRF_ADC->EVENTS_END  = 0;     // Stop any running conversions.
        NRF_ADC->TASKS_START = 1;
        while (!NRF_ADC->EVENTS_END)
        {
        }
        sample_I_temp[k] = NRF_ADC->RESULT;
        NRF_ADC->EVENTS_END      = 0;
        NRF_ADC->TASKS_STOP      = 1;
        nrf_delay_us(7);
        }
      Sample_I=0; Sample_Q =0;
      for (k=0; k < number_samples_min_2m ; k = k+4)
      {
            if( sample_I_temp[k] >= sample_I_temp[k+2])
              Sample_I =Sample_I + sample_I_temp[k] - sample_I_temp[k+2];
            else
              Sample_I =Sample_I + sample_I_temp[k+2] - sample_I_temp[k];

```

```
                if( sample_I_temp[k+1] >= sample_I_temp[k+3])
                  Sample_Q =Sample_Q + sample_I_temp[k+1] - sample_I_temp[k+3];
                else
                  Sample_Q =Sample_Q + sample_I_temp[k+3] - sample_I_temp[k+1];

        }
        Sample_I_bitd[j]=Sample_I;
        Sample_Q_bitd[j]=Sample_Q;
  }
        thres_I=(Sample_I_bitd[0] + Sample_I_bitd[1])>>1;
        thres_Q=(Sample_Q_bitd[0] + Sample_Q_bitd[1])>>1;;

}

  void calc_bit_width()
  {
      flag =0x01; j=0;
      do{
//          nrf_gpio_pin_toggle(0x05);
        for (k=0; k < number_samples_div ; k++)
        {
          //NRF_ADC->EVENTS_END  = 0;      // Stop any running conversions.
          NRF_ADC->TASKS_START = 1;
          while (!NRF_ADC->EVENTS_END)
          {
          }
          sample_I_temp[k] = NRF_ADC->RESULT;
          NRF_ADC->EVENTS_END      = 0;
          NRF_ADC->TASKS_STOP      = 1;
```

```
            nrf_delay_us(7);
        }
        Sample_I = 0; Sample_Q = 0; j++;
        for (k=0; k < number_samples_div ; k = k+4)
        {
                if( sample_I_temp[k] >= sample_I_temp[k+2])
                  Sample_I =Sample_I + sample_I_temp[k] - sample_I_temp[k+2];
                else
                  Sample_I =Sample_I + sample_I_temp[k+2] - sample_I_temp[k];

                if( sample_I_temp[k+1] >= sample_I_temp[k+3])
                  Sample_Q =Sample_Q + sample_I_temp[k+1] - sample_I_temp[k+3];
                else
                  Sample_Q =Sample_Q + sample_I_temp[k+3] - sample_I_temp[k+1];
        }
      }while(Sample_Q  < threshold && Sample_I < threshold);
      // estimate of the peak
      for(k=0; sample_I_temp[k] < 0x88 ; k++); // || sample_I_temp[k] > 0x75
      //j=k;

      bit_width= (38000+ (1000*j) + ((number_samples_div-k)*31) + (k>>2))>>6;
      bit_width= bit_width + 4; // 4 is for trimming
  //    bit_width= 612;
//   nrf_gpio_pin_toggle(0x05);

      nrf_delay_us((1000*(j-1)) + (k*31));
```

```
    nrf_delay_us(18000);// change that number
}




void reported_data()
{
    uint32_t err_code;
    // received bits
    err_code = ble_temp_temp_level_update(&m_temp, data2send);
    if ((err_code != NRF_SUCCESS) &&
        (err_code != NRF_ERROR_INVALID_STATE) &&
        (err_code != BLE_ERROR_NO_TX_BUFFERS) &&
        (err_code != BLE_ERROR_GATTS_SYS_ATTR_MISSING)
      )
      {
          APP_ERROR_HANDLER(err_code);
      }
    // thresholds
    nrf_delay_ms(10);
    data2send[0]=thres_I>>8;
    data2send[1]=thres_I>>0;
    data2send[2]=thres_Q>>8;
    data2send[3]=thres_Q>>0;
    err_code = ble_temp_temp_level_update(&m_temp, data2send);
    if ((err_code != NRF_SUCCESS) &&
        (err_code != NRF_ERROR_INVALID_STATE) &&
        (err_code != BLE_ERROR_NO_TX_BUFFERS) &&
        (err_code != BLE_ERROR_GATTS_SYS_ATTR_MISSING)
      )
      {
```

```
          APP_ERROR_HANDLER(err_code);
      }
    // the first one calculation
    nrf_delay_ms(10);
    data2send[0]=Sample_I_bitd[0]>>8;
    data2send[1]=Sample_I_bitd[0]>>0;
    data2send[2]=Sample_Q_bitd[0]>>8;
    data2send[3]=Sample_Q_bitd[0]>>0;
    err_code = ble_temp_temp_level_update(&m_temp, data2send);
    if ((err_code != NRF_SUCCESS) &&
        (err_code != NRF_ERROR_INVALID_STATE) &&
        (err_code != BLE_ERROR_NO_TX_BUFFERS) &&
        (err_code != BLE_ERROR_GATTS_SYS_ATTR_MISSING)
      )
      {
          APP_ERROR_HANDLER(err_code);
      }
    // the first zero calcuation
    nrf_delay_ms(10);
    data2send[0]=Sample_I_bitd[1]>>8;
    data2send[1]=Sample_I_bitd[1]>>0;
    data2send[2]=Sample_Q_bitd[1]>>8;
    data2send[3]=Sample_Q_bitd[1]>>0;
    err_code = ble_temp_temp_level_update(&m_temp, data2send);
    if ((err_code != NRF_SUCCESS) &&
        (err_code != NRF_ERROR_INVALID_STATE) &&
        (err_code != BLE_ERROR_NO_TX_BUFFERS) &&
        (err_code != BLE_ERROR_GATTS_SYS_ATTR_MISSING)
      )
      {
          APP_ERROR_HANDLER(err_code);
      }
```

```
    // noise estimation
    nrf_delay_ms(10);
    data2send[0]=Sample_I_noise>>8;
    data2send[1]=Sample_I_noise>>0;
    data2send[2]=Sample_Q_noise>>8;
    data2send[3]=Sample_Q_noise>>0;
    err_code = ble_temp_temp_level_update(&m_temp, data2send);
    if ((err_code != NRF_SUCCESS) &&
        (err_code != NRF_ERROR_INVALID_STATE) &&
        (err_code != BLE_ERROR_NO_TX_BUFFERS) &&
        (err_code != BLE_ERROR_GATTS_SYS_ATTR_MISSING)
      )
      {
          APP_ERROR_HANDLER(err_code);
      }
    // last bit 1 calculation, "estimate the zero"
    nrf_delay_ms(10);
    data2send[0]=Sample_I>>8;
    data2send[1]=Sample_I>>0;
    data2send[2]=Sample_Q>>8;
    data2send[3]=Sample_Q>>0;
    err_code = ble_temp_temp_level_update(&m_temp, data2send);
    if ((err_code != NRF_SUCCESS) &&
        (err_code != NRF_ERROR_INVALID_STATE) &&
        (err_code != BLE_ERROR_NO_TX_BUFFERS) &&
        (err_code != BLE_ERROR_GATTS_SYS_ATTR_MISSING)
      )
      {
          APP_ERROR_HANDLER(err_code);
      }
}

```

```
static void temp_level_meas_timeout_handler(void * p_context)
{
    uint32_t err_code ;

    noise_estimate();
    bit_detect();
    calc_thres();
    calc_bit_width();

    for (j = 0; j < num_bit ; j++)
    {
      nrf_gpio_pin_toggle(0x05);
      for (k=0; k < bit_width ; k++) //number_samples
      {
      //  NRF_ADC->EVENTS_END  = 0;     // Stop any running conversions.
        NRF_ADC->TASKS_START = 1;
        while (!NRF_ADC->EVENTS_END)
        {
        }
        sample_I_temp[k] = NRF_ADC->RESULT;
        NRF_ADC->EVENTS_END      = 0;
        NRF_ADC->TASKS_STOP      = 1;
        nrf_delay_us(7);
        }
      Sample_I=0; Sample_Q =0;  bit_decision[j] = 0x00;
      for (k=0; k < bit_width ; k = k+4)
      {
            if( sample_I_temp[k] >= sample_I_temp[k+2])
              Sample_I =Sample_I + sample_I_temp[k] - sample_I_temp[k+2];
            else
```

```
                Sample_I =Sample_I + sample_I_temp[k+2] - sample_I_temp[
k];

              if( sample_I_temp[k+1] >= sample_I_temp[k+3])
                Sample_Q =Sample_Q + sample_I_temp[k+1] - sample_I_temp[
k+3];
              else
                Sample_Q =Sample_Q + sample_I_temp[k+3] - sample_I_temp[
k+1];

        }
        Sample_I_bits[j]=Sample_I;
        Sample_Q_bits[j]=Sample_Q;

        if(j<= 7)
        {
          if ( Sample_I + Sample_Q >=  thres_I + thres_Q )
            data2send[0] = data2send[0]  <<1 | 0x01 ;
          else
            data2send[0] = data2send[0]  <<1 | 0x00 ;
        }
        if(j<= 15)
        {
          if ( Sample_I + Sample_Q >=  thres_I + thres_Q )
            data2send[1] = data2send[1]  <<1 | 0x01 ;
          else
            data2send[1] = data2send[1]  <<1 | 0x00 ;
        }
        if(j<= 24)
        {
          if ( Sample_I + Sample_Q >=  thres_I + thres_Q )
            data2send[2] = data2send[2]  <<1 | 0x01 ;
```

```
          else
            data2send[2] = data2send[2] <<1 | 0x00 ;
        }
      }
    reported_data();

    Sample_I_bitd[1]=Sample_I_bitd[0] =0;
    Sample_Q_bitd[1]=Sample_Q_bitd[0] =0;
    thres_I= thres_Q=0;
    Sample_I_noise=0;
    Sample_Q_noise=0;
  }

  uint8_t temp_level_get(void)
  {
      temp_level_meas_timeout_handler(NULL);
  }

  /**@brief Function for dispatching a BLE stack event to all modules
     with a BLE stack event handler.
   *
   * @details This function is called from the scheduler in the main
     loop after a BLE stack
   *           event has been received.
   * @param[in]   p_ble_evt   Bluetooth stack event.
   */
  static void ble_evt_dispatch(ble_evt_t * p_ble_evt)
    {
    //  dm_ble_evt_handler(p_ble_evt);
      on_ble_evt(p_ble_evt);
      ble_db_discovery_on_ble_evt(&m_ble_db_discovery, p_ble_evt);
      ble_conn_params_on_ble_evt(p_ble_evt);
```

```
        ble_advertising_on_ble_evt(p_ble_evt);
        ble_temp_on_ble_evt(&m_temp, p_ble_evt);
    #ifdef BLE_DFU_APP_SUPPORT
        /** @snippet [Propagating BLE Stack events to DFU Service] */
        ble_dfu_on_ble_evt(&m_dfus, p_ble_evt);
        /** @snippet [Propagating BLE Stack events to DFU Service] */
    #endif // BLE_DFU_APP_SUPPORT
}
```

Bibliography

[1] F. H. Sambo, Yusuf Abdulrahman and M. A. Imran., "A survey and tutorial of electromagnetic radiation and reduction in mobile communication systems," *IEEE Communications Surveys & Tutorials*, vol. 17.2, no. 4, pp. 790–802, 2015.

[2] e. a. Ahlbom, A., *Guidelines for limiting exposure to time-varying electric, magnetic, and electromagnetic fields (up to 300 GHz)*. ICNIRP, Oct. 1998.

[3] R. H. M. M. R. Luria, I. Eliyahu and N. Meiran, "Cognitive effects of radiation emitted by cellular phones: The influence of exposure side and time, bioelectromagnetics," *Bioelectromagnetics*, vol. 30, no. 3, no. 4, p. 198204, 2009.

[4] e. a. McKinlay, A. F., "Statement by nrpb: Restrictions on human exposure to static and time varying electromagnetic fields and radiation," Oct. 1993.

[5] "Restrictions on human exposure to static and time varying electromagnetic fields and radiation: Scientific basis and recommendations for the implementation of the boards statement," 1993.

[6] "Electromagnetic fields and the risk of cancer: Supplementary report by the advisory group on non-ionising radiation," 1994.

[7] "Information for manufacturers seeking marketing clearance of diagnostic ultrasonic systems and transducers," 2008.

[8] "Occupational safety and health standards 1910.95," 1986.

[9] "Threshold limit values for chemical substances and physical agents and biological exposure indices," 2002.

[10] J. C. B. C. R. Hill and G. R. ter Haar, *Physical Principles of Medical Ultrason.* Wiley, 2002.

[11] K. S. S. Bower and S. Campbell, "Doppler ultrasound screening as part of routine antenatal scanning: prediction of pre-eclampsia and intrauterine growth retar-dation," *An International Journal of Obstetrics and Gynaecology*, vol. 100, no. 11, p. 989994, 1993.

[12] H. C. S. Sheen and A. Raptis, "Ultrasonic techniques for detecting helium leaks," *Sensors & Actuators: B. Chemical*, vol. 71, no. 3, pp. 197–202, 2000.

[13] K. Wong, "Instrumentation and health monitoring of cable-supported bridges,," *structural Control and Health Monitoring*, vol. 11, no. 2, pp. 91–124, 2004.

[14] M. Stojanovic, "Recent advances in high-speed underwater acoustic communications," *IEEE Journal of Oceanic Engineering*, vol. 21, no. 2, pp. 125–136, 1996.

[15] C. Lopes and P. Aguiar, "Acoustic modems for ubiquitous computing," *IEEE Pervasive Computing*, pp. 62–71, 2003.

[16] M. Stojanovic, "Things that talk: Using sound for device-to-device and device-to-human communication," *IBM Systems Journal*, vol. 39, no. 3, pp. 530–546, 2000.

[17] G. Saulnier, H. Scarton and P. Das, "P1g-4 through-wall communication of low-rate digital data using ultrasound," *Ultrasonics Symposium, 2006. IEEE*, pp. 1385–1389, 2006.

[18] C. Kuratli and Q. Huang, "A fully integrated self-calibrating transmitter/receive ic for an ultrasound presence detector microsystem," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 6, pp. 832–841, 1998.

[19] J. v. d. B. Carlson, Johan E. and M. Mohamad, "Mbit/second communication through a rock bolt using ultrasound," *IEEE International Ultrasonics Symposium (IUS)*, pp. 1–4, 2017.

[20] W. Jiang and W. M. Wright, "Indoor wireless communication using airborne ultrasound and ofdm methods," *IEEE International Ultrasonics Symposium (IUS)*, 2016.

[21] W. Jiang and W. M. Wright, "Multichannel ultrasonic data communications in air using range-dependent modulation schemes," *IEEE transactions on ultrasonics, ferroelectrics, and frequency control*, vol. 66, no. 1, pp. 147–155, 2016.

[22] W. Jiang and W. M. Wright, "Ultrasonic wireless communication in air using ofdm-ook modulation," *IEEE International Ultrasonics Symposium (IUS)*, 2014.

[23] H. D. A. . G. R. J. Li, C., "Short-range ultrasonic digital communications in air," *IEEE transactions on ultrasonics, ferroelectrics, and frequency control*, vol. 55, no. 4, pp. 908–918, 2008.

[24] D. A. H. Li, Chuan and R. J. Green, "Short-range ultrasonic communications in air using quadrature modulation," *IEEE transactions on ultrasonics, ferroelectrics, and frequency control*, vol. 56, no. 10, 2009.

[25] K. Herman and W. Fernandez, "A comparative study of selected methods for ultrasonic signals energy estimation for target strength and distance evaluation," *CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*, vol. 1109, no. 10, pp. 1–3, 2017.

[26] B. Jameson and R. Manduchi, "Estimation and detection in the presence of ringing noise," *IEEE International Ultrasonics Symposium (IUS)*, 2010.

VITA

He was born in 1992 in Cairo, Egypt. He got his B.Sc. in 2014, major of Electronics and Communication Engineering from Cairo University with Excellence grade with degree of honor.

He worked as a firmware developer and hardware design engineer in WaveWorks INC. for more than two years.

He pursued his M.Sc. degree in Electrical and Electronics Engineering in Ozyegin University, Istanbul, Turkey under supervision of Prof. Ahmet Tekin, where he focused in Class-D audio power amplifier, Wireless power transfer, Mechanical solenoid control circuitry and Ultrasound communication.

Furthermore, he could attend the IEEE 50^{th} ISCAS Conference, Baltimore, Maryland, USA in 2017 and presented his paper from his work in wireless power transfer. Also, He could get a patent in Arm wearable support and submit a journal article to Mechatronics transactions.

he is going to submit a journal article from his master thesis to circuits and systems transactions. His research interests are: Analog mixed signal circuit design, RF circuit, Power Electronics and Biomedical circuits.

In addition, he was a teaching assistant for Analog Electronics and Circuit Analysis courses in Ozyegin University for two years.

He got a visiting researcher student opportunity in Stanford University, Electrical Engineering department where he may pursue his PhD degree there.