TALLINN UNIVERSITY OF TECHNOLOGY
Faculty of Information Technology

IE70LT

Shivalingaiah Venkataramanayya Palya Shivaramaiah
IVEM144711

# INERTIAL MOTION SENSOR PROTOTYPE WITH BLUETOOTH LOW ENERGY CONNECTIVITY

Master's Thesis

Supervisor: Alar Kuusik

PhD

Senior Research Scientist

Tallinn 2018

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

IE70LT

Shivalingaiah Venkataramanayya Palya Shivaramaiah
IVEM144711

# BLUETOOTH LE SIDET KASUTAV INERTSIAALSE LIIKUMISANDURI PROTOTÜÜP

magistritöö

Juhendaja: Alar Kuusik

Tehnikateaduste
doktor
TTÜ vanemteadur

Tallinn 2018

## Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Shivalingaiah Venkataramanayya Palya Shivaramaiah

04.06.2018

# Abstract

The wireless communication can enable swift data transmission, location independent communication and real-time monitoring and control, which are of paramount significance for further health care monitoring purposes. Some of the toughest issues facing portable medical devices are smooth real-time data processing and streaming, especially because of low power consumption requirements.

There was an existing inertial motion sensor (IMS) developed for patient rehabilitation and diagnoses purposes. Existing device had limitation of processing power of sensor data, insufficient amount of memory, difficulties for further software development because of obsolete Bluetooth Low Energy protocol stack. In the thesis core software development of a new sensor prototype solution is described. The prototype is based on

nRF52 development kit and BNO055 motion sensor. Main part of the thesis describes new sensor software solution that is based on FreeRTOS operating system and Bluetooth Low Energy communication stack. Throughput and communication range test results of the system are presented.

This thesis is written in English and is 50 pages long, including 9 chapters, 18 figures, 1equation and 10 tables.

# Annotatsioon

# Bluetooth LE sidet kasutav inertsiaalse liikumisanduri prototüüp

Juhtmevaba side võimaldab kiiret ja asukohast sõltumatut andmevahetust, monitoorimist ja juhtimist reaalajas. Need on esmatähtsad nõuded patsientide tervise jälgimise rakendustes tulevikus. Ühed kõige keerukamad probleemid kantavate meditsiiniseadmete juures on seotud reaalajaliselt sujuva andmetöötluse ja voogedastusega, eriti võttes arvesse madala energiatarbe nõudeid.

Varasemal perioodil oli välja töötatud inertsiaalne liikumisandur patsientide rehabilitatsiooni ja diagnoosi eesmärkidel. Eksisteerinud seadmel olid puudused seoses piiratud arvutusvõimuse ja mälumahuga, edasine tarkvaraarendus oli raskendatud seoses vananenud Bluetooth LE protokollipinuga.

Diplomitöös kirjeldatakse süsteemitarkvara arendust uue sensori prototüübile. Prototüüp põhineb nRF52 arendusmoodulil ja BNO055 liikumisanduril. Töö peamine osa kirjeldab uue sensori tarkvaralahendust, mis põhineb FreeRTOS operatsioonisüsteemil ja Bluetooth LE sidepinul. Esitatakse süsteemi andmevahetuskanali läbilaskevõime ja sidekauguse testide tulemused.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 50 leheküljel, 9 peatükki, 18 joonist, 1 võrrandit ja 10 tabelit.

# Acknowledgement

First and for most, I would like to thank my supervisor Alar Kuusik, Senior Research Scientist of Faculty of Information Technology-Thomas Johan Seebeck Department of Electronics at Tallinn University of Technology. His guidance, encouragement, positive attitude and consultation is very inspiring. I would like to take this chance to thank, Katri Kadakas, Student counsellor, School of information technologies.

I would like to thank all my friends staying in Tallinn. Vinay, Yashwanth, Susu, Chethan, your support is very remarkable in my life.

I can’t forget continuous encouragement and support from my family. I would take this moment to express my very profound gratitude to my family. This accomplishment would not have been possible without my family.

Thank you

Shivalingaiah Venkataramanayya Palya Shivaramaiah

# LIST OF ABBREVIATIONS AND TERMS

| | | |
|---|---|---|
| AES | - | Advanced Encryption Standard |
| API | - | Application Program Interface |
| ARM | - | Advance RISC Machine |
| ASSN | - | Application Specific Sensor Nodes |
| bps | - | bits per second |
| BLE | - | Bluetooth Low Energy |
| BSX | - | Bosch Sensortec sensor fusion software |
| GAP | - | Generic Access Profile |
| GATT | - | Generic Attributes |
| I2C | - | Inter IC Communication |
| ID | - | Identification |
| IDE | - | Integrated Development Environment |
| IMS | - | Inertial Motion Sensor |
| IoT | - | Internet of Things |
| OS | - | Operating System |
| RAM | - | Random Access Memory |
| RF | - | Radio Frequency |
| RISC | - | Reduced Instruction Set Computing |
| ROM | - | Read Only Memory |
| RTOS | - | Real Time Operating System |
| SDK | - | Software Development Kit |
| SiP | - | System in Package |
| SoC | - | System on Chip |
| SPI | - | Serial Peripheral Interface |
| SRAM | - | Static Random Access Memory |
| TI | - | Texas Instruments |
| UART | - | Universal Asynchronous Receiver Transmitter |
| USB | - | Universal Serial Bus |
| UUID | - | Universally Unique Identifier |

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# 1 Introduction

## 1.1 Task description

An inertial motion sensors (IMSs) are widely used for many movement tracking applications like training monitoring, motion pattern recognition, industrial, automotive and aviation motion sensing applications. The custom hardware platforms developed by universities are widely used for motion measurement with help of signal processing and pattern recognition algorithms. For example, motion data could be used for medical rehabilitation and diagnoses of patients to provide proper treatment.

An existing hardware platform used at TTU has certain challenges in terms of existing software improvements. The existing platform software drawbacks are as listed below:

- The existing custom IMS platform is based on 8051 microcontroller of Texas Instruments (TI) [1]; controller is only supporting BLE4.0 stack, processing of BNO055 sensor data simultaneously with communication is performance critical. I observed many times reboot and hang issues;
- Existing BLE4.0 [2] stack is obsolete and there is no more support from the semiconductor vendor; further development of stack or software requires IAR toolset, which is expansive;
- No real-time operating system (RTOS) support to handle peripherals in a simpler way;
- Insufficient amount of memory for the further developments.

To overcome abovementioned software implementation challenges, the task has been defined to find a new hardware and software platform for next generation IMS. New platform should support BLE4.2 or newer version of Bluetooth stack and royalty free software development environment. The platform should have more powerful microcontroller to handle BLE stack, sensor data capture, and application tasks. And, platform should support RTOS [3]. The microcontroller should have enough RAM(>=256kB) and Flash memory(>=512kB) to handle stack, operating system, application program efficiently.

## 1.2 Development objectives

Taking into consideration of software requirement for new IMS hardware platform certain objectives are defined.

- Selection and evaluation of a new microcontroller hardware platform for IMS; analysis of hardware peripherals like RAM, ROM, and communication protocols;
- Surveying of BLE profiles for health monitoring and implementation of relevant connectivity solution;
- Implementation of RTOS based ISM software prototype;
- Testing performance of implemented IMS software.

# 2 IMS System Architecture

## 2.1 System architecture

The IMS system contains of three main components: IMS software (firmware), IMS hardware and an application software on personal computer or mobile device (Fig. 1). According to technical requirement analysis I selected Nordic Semiconductor microcontroller nRF52 [4]. A higher version of nRF52 microcontroller is 32-bit ARM Cortex -M4F runs at 64MHz and have 1Mbit of flash with cache, 512kB of RAM, and many IDE's supported for application development. As by previous IMS device, new one should sample the physical sensor data for every fixed interval (max 20ms) and transmit sensor data via Bluetooth Low Energy to mobile or personal computer, which is running BLE server stack application.

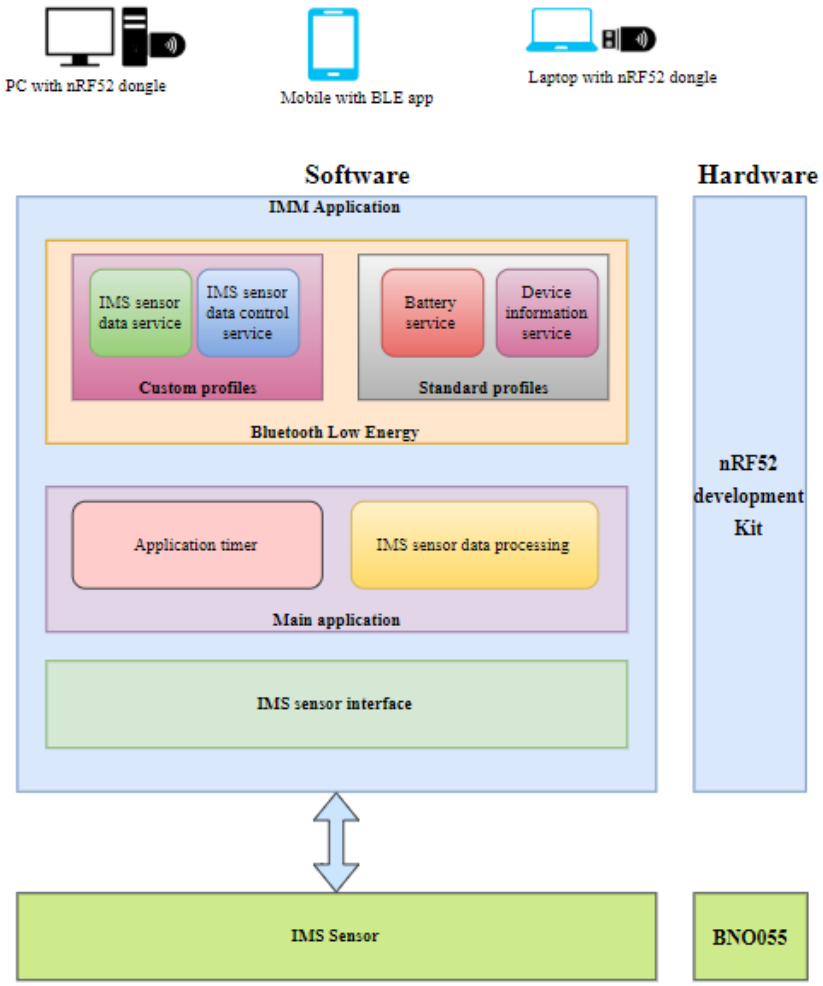


Figure 1: IMS System architecture.

## 2.2 System architecture components

### 2.2.1 Hardware

The hardware for IMS prototyping used is nRF52 development kit (left in Fig. 2), nRF51 dongle (in the middle) [5] for application testing and BNO055 9DOF motion sensor (right) [6]. More details about hardware components are presented in further chapters.

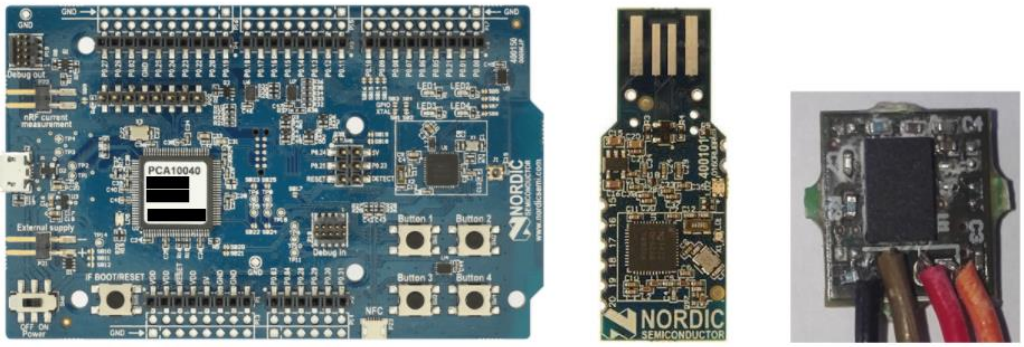


Figure 2: Hardware components used in IMS.

### 2.2.2 Microcontroller software

The IMS application software is structured as three components: (a) main application, (b) Bluetooth Low Energy communication application, (c) BNO055 interface application. The main application controls sensor data sampling using application timer and process the motion sensor data. The BNO055 sensor is interfaced to nRF52 by UART protocol. nRF52 Bluetooth Low Energy stack is used to communicate sensor data to mobile or a laptop. The main application and BLE stack application software development is discussed in further chapters. The main application software also developed for FreeRTOS real-time operating system.

### 2.2.3 Application software

To test application on laptop or personal computer, nRF connect software can be used that is an open source provided by Nordic Semiconductor. To test application on mobile device any Bluetooth Low Energy Android or iOS application can be used. I used nRF Connect application software for testing purpose. BLE Scanner application can be used as well.

# 3 Bluetooth Low Energy

## 3.1 Introduction

Bluetooth Low Energy, also known in the industry as "Bluetooth Smart", is a lightweight subset of classic Bluetooth and was introduced as part of the Bluetooth 4.0 core specification [7]. While there is some overlap with classic Bluetooth, BLE has a completely different lineage and was started by Nokia as an in-house project called 'Wibree' before being adopted by the Bluetooth SIG [2].

Bluetooth Low Energy is designed for low power consumption applications like beacons, fitness devices, home automation, etc. Bluetooth Low Energy devices best suitable for very low data rate and low power consumption applications. BLE is dedicated for devices working from a battery for years. Bluetooth Classic depletes a battery quickly because it was designed to exchange a lot of data in a short amount of time. BLE is intended to provide considerably reduced power consumption, and low cost while maintaining very similar communication range to standard Bluetooth; otherwise known as, radio coverage.

The difference between Bluetooth and BLE is that there is no data throughput because BLE does not support streaming data. After a connection has been established (paired), BLE spends most of the time in sleep mode waiting to send/receive the next set of device status information also known as ‘expose state”, such as the Battery Level. It has a data rate of 1Mbps allowing for quick data transfer of small chunks or data packets (kB), exposing the state of the device to retrieve that information. This status update interval rate delay can be programmed from 7ms up to 4s between data polls. Once the data has been transferred, a few milliseconds, the BLE goes back to sleep to conserve battery; whereas, Bluetooth stays on the entire time regardless if information is being transferred.

There are some differences between Bluetooth Low Energy 4.2 and 4.0, in 4.2 specification [7] induces more security features during connection and data transmission. Many other features similar in both specs.

## 3.2 Bluetooth Low Energy use cases in healthcare

Medicinal services are exorbitant, regularly depending on costly expert hardware and offices, and exceptionally prepared staff. If medicinal services could utilize those human and physical assets all the more proficiently, at that point there is extraordinary potential for cost reserve funds, especially by lessening use on routine assignments. Bluetooth Low Energy is a valuable innovation for executing vitality proficient remote correspondence frameworks for social insurance purpose.

- **Health monitoring at home:** The IoT makes it conceivable to procure gigantic investment funds, increments in proficiency, and upgrades in tolerant solace, by moving noteworthy zones of patient care out of healing facility wards and into the home. It's getting to be conceivable to screen patients' wellbeing wherever they are. Observing gadgets, for example, therapeutic weight scales, heart rate screens and pulse screens can track wellbeing and ready patients, families and guardians to changes in essential signs, or missed prescriptions.

- **Inpatient monitoring:** In hospitals patients are regularly associated with various human services gadgets. Essentially getting rid of wires gives significant advantages, sparing staff time, diminishing danger of blunder, and making patients more agreeable. Electrocardiography (ECG) screens and circulatory strain sensors can exchange crucial sign information remotely to the healing facility's focal observing frameworks.

- **Ambulance monitoring:** As paramedics treat a patient in a rescue vehicle while in transit to healing centre, the defibrillator can utilize BLE to send constant data about the patient's status to a portal inside the vehicle. This data is consequently exchanged to the defibrillator producer's cloud benefit. Healing centres can buy in to this support of be better arranged when the patient touches base at the ER. Clearly, a comparative procedure of continuous data social event can likewise be utilized with other gear in ambulances.

- **Medicine monitoring:** Continuous blood donation centre checking is another appealing application. Blood must be put away inside a specific temperature range, or it may not be ok for utilize. Each blood sack is labelled with a minor re-usable tracer that tracks the temperature by means of a Bluetooth low vitality. The

sensor spends the majority of its life sitting unobtrusively on the rack in rest mode, however is modified to wake up when it recognizes physical development. It demonstrates the legitimacy of the blood by means of a LED and publicizes its quality by means of Bluetooth low vitality.

## 3.3 Bluetooth Low Energy stack

Bluetooth Low Energy is completely redesigned of Bluetooth Classic protocol, they are so different in use case and architecture. Below figure shows architecture of BLE stack [8].

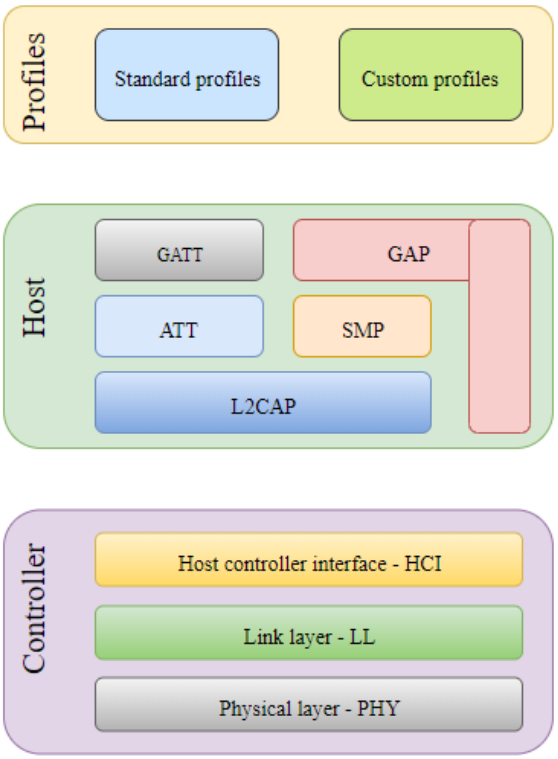


Figure 3: BLE Stack architecture.

### 3.3.1 Physical layer

Bluetooth Low Energy physical layer works in 2.4GHz ISM band and have 40 channels with 3 of them as advertising channels and all other channels are data channels, and each channel bandwidth is 2MHz. so BLE physical layer have 3 advertising channels and 37 data channels. the data is transmitted over that channels are controlled by two events, they are advertising and connection events.

### 3.3.2 Link layer

The BLE link layer have state machine and it has five states. Standby, advertising, scanning and scanning have two sub states active, passive. Initiating, connection and connection have two sub states central, peripheral.

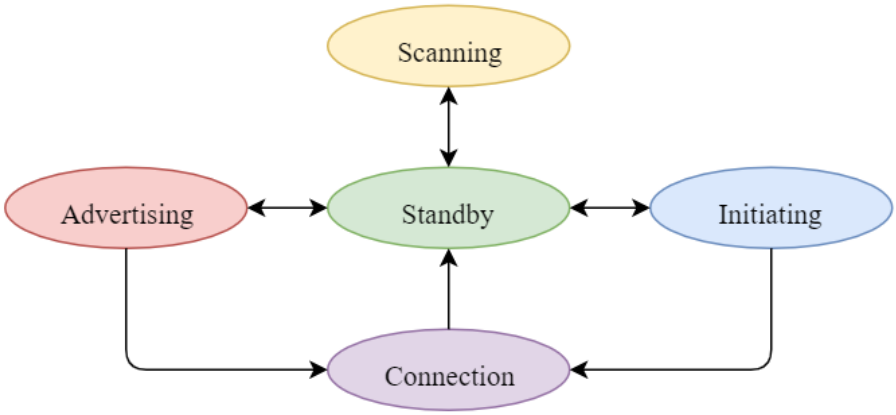


Figure 4: Link layer state machine.

A Bluetooth Low Energy device can be named in one of the following roles according to states of link layer.

- **Scanner**: The device which receive advertising packets on advertising channels without intended to connect are referred to as scanners.
- **Advertiser**: The device which transmit the advertising packets on advertising channels are called as advertisers.
- **Initiator**: the device which need to make a connection to another device listens for connectable advertising packets, are referred as initiators.
- **Connectors**: after establishing connection, the initiator becomes central device and advertising packet becomes peripheral device.

### 3.3.3 Host controller interface

The host controller interface layer should have ability to communicate data without prior knowledge of the data being transferred. Different host controllers been used and the Bluetooth Low Energy specification defined three host controller interfaces, those are USB, UART and SPI.

### 3.3.4 Generic access profile

The generic access profile defines fundamental operations in standard, discovering device and connecting with peer device, establishing secure connections, broadcast packets, etc. also connection parameters are defined here.

### 3.3.5 Generic attribute profile

The generic attribute profile layer defines about data communication. The GATT [9] have two roles, a GATT server and GATT client, they completely independent of the GAP roles. The device which is holding data is GATT server and the device which is accessing data called GATT client. A device can act as both GATT server and GATT client simultaneously. The GATT server organizes data in attribute table and the attributes that contain actual data.

- **Attribute**: The attribute has handle, UUID, and value. Handle controls index in the GATT table for the attribute. The UUID have information of type of data within the attribute. There may be many other attributes in a GATT table within the same UUID.

- **Characteristics**: A characteristic contains at least two attributes, a characteristic declaration and the characteristic which holds attribute value. All data transferred through a GATT service must be mapped to characteristics.

- **Descriptors**: A descriptor is an additional attribute and provides information about characteristics. There is one special feature Client Characteristic Configuration Descriptor this can be added to any characterises and that supports the notify or indicate properties.

- **Service**: A service contains logical collection of related one or more characteristics. Some of the GATT services are defined by SIG group and they called as standard profiles.

- **Profile**: A profile is a collection of one or more services. The profile includes information on services particularly profile as well how the peers communicate data.

## 3.4 Standard and custom services

The Bluetooth special interest group (SIG) defines several profiles, services, Characteristics, and an attribute based on GATT layer of the BLE stack. Anyway, Bluetooth Low Energy all service are depends on requirement and designer, not on the stack. The profiles which is defined by SIG called standard profiles and profiles which is defined by vendors called custom profiles. Some SIG defined standard services are listed below.

| Standard services | Handle |
|---|---|
| Alert notification service | 0x1811 |
| Battery service | 0x180F |
| Blood pressure service | 0x1810 |
| Current time service | 0x1805 |
| Location and navigation service | 0x1816 |
| Device information service | 0x180A |
| Glucose service | 0x1808 |
| Health thermometer service | 0x1809 |
| Heart rate service | 0x180D |

Table 1: BLE standard services.

## 3.5 UUID

A UUID is a 128-bit number and it is globally unique, provides an attribute type information. The SIG defines separation between base UUID and a 16-bit UUID, used to make complete base UUID. A Bluetooth SIG defined UUID have common base 0x0000xxxx-0000-1000-8000-00805F9B. the SIG defined UUID's cannot be used for any custom attributes, services or characteristics.

# 4 Nordic nRF52 Microcontroller

## 4.1 Previous IMS hardware platform

The existing IMS platform uses TI's CC2540 [10] Bluetooth Low Energy wireless microcontroller. It's a System-on-Chip(SoC) processor embedded with industry standard 2.4GHz RF transceiver, In system programmable flash of 128kB/256kB, 8kB SRAM, UART, High speed USB, along with standard 8051 microcontroller. Texas instruments provides Bluetooth Low Energy 4.0 stack for CC2540. IAR embedded workbench is used for stack software development.

## 4.2 New platform selection

Considering requirements, I have chosen Nordic semiconductor nRF52840 advanced multi-protocol SoC microcontroller [12]. nRF52840 is an ultra-low power 2.4 GHz remote framework on chip (SoC) incorporating a multiprotocol 2.4 GHz transceiver, an ARM Cortex-M4F CPU and flash program memory. It is a definitive SoC for any short range remote individual zone system or IPv6-empowered mechanization application.

### 4.2.1 Nordic nRF52840 evaluation kit

The nRF52 Development Kit is a single-board development kit for Bluetooth Smart, ANT and 2.4GHz proprietary applications using the nRF52 Series SoC. This kit supports both development for nRF52832 SoCs [11]. The kit is compatible with the Arduino Uno Revision 3 standard, making it possible to use 3rd-party shield that are compatible to this standard with the kit.

The kit supports the Nordic Software Development Tool-chain using Keil [12], IAR [13] and GCC. The kit likewise bolsters ARM mbed apparatus chain for quick prototyping and advancement utilizing mbed's cloud-based IDE and device chain with a broad scope of open-source programming libraries. Program/Debug alternatives on the pack are Segger J-Link Lite for standard instrument chain and CMSIS-DAP for mbed. The pack offers access to all I/O and interfaces by means of connectors and has 4 LEDs and 4 catches which are client programmable. A scope of programming cases are accessible from the nRF5 SDK to help Bluetooth Smart, ANT, ZigBee and 2.4GHz applications.

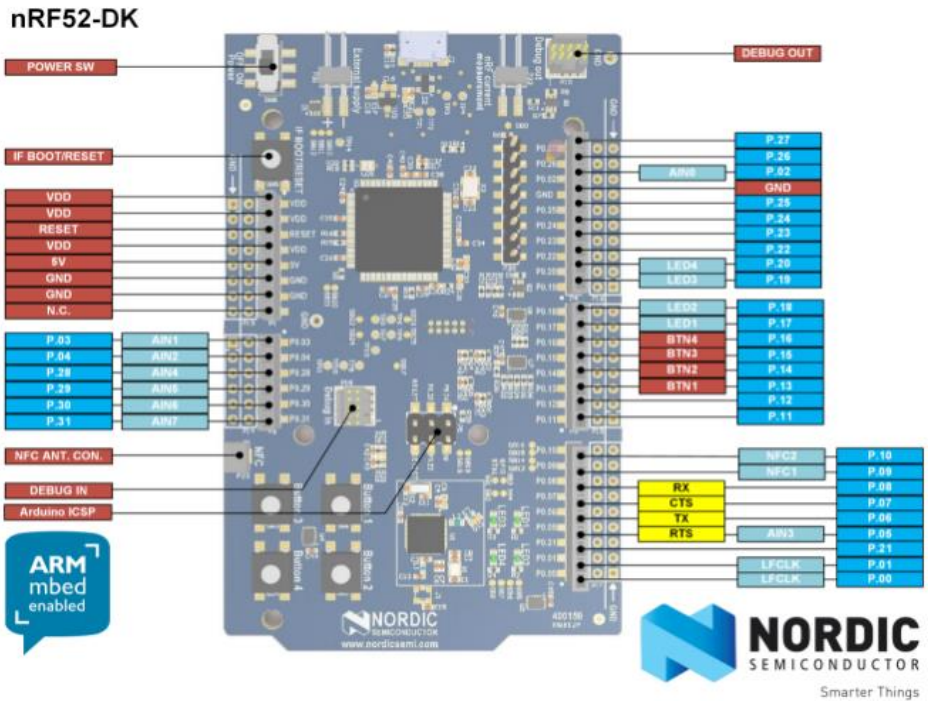


Figure 5: nRF52 development kit [15].

### 4.2.2 Features of PCA10040 development kit

- Hardware prototype board contains nRF52832 System-on-Chip microcontroller and its supports for multiprotocol radio.
- The connectors are compatible with Arduino Uno v3 and all I/O, peripherals and interfaces available over connectors.
- Kit can be programmed by USB drag and drop programming, nRF studio software and using IDE's.
- External pins provided for RF and power measurements.
- Kit can be powered three ways USB, external source(1.8V-3.6V) or a coin cell battery.

To meet hardware requirement, I have chosen nRF52 series microcontroller for Inertial Motion Sensor prototyping. nRF52 series microcontroller very rich in features as compare to old platform designed by CC2540 microcontroller. Very flexible in software development and supported by many IDE's vendors. The comparison of new platform using nRF52 over old platform shown in below table.

| | nRF52840 [11] | CC2540 [10] |
|---|---|---|
| Microcontroller | 32-bit ARM Cortex -M4F @ 64MHz | 8-bit 8051 CPU @32MHz |
| Multiprotocol radio | Bluetooth 5, BLE 4.2, ANT /ANT+, ZigBee, NFC and 2.4GHz proprietary | 2.4GHz Bluetooth low energy |
| Cryptography | 128-bit AES /ECB/CCM/AAR co-processor | AES security coprocessor |
| Over the air programming | Supported by stack | Not supported |
| Memory | 1MB flash with cache, 256kB RAM | 128kB/256kB flash, 8kB SRAM |
| Software stack | Application development independent of protocol stack support | Only BLE 4.0 stack support |
| Software development tools | Keil, IAR, GCC, Med online tool | IAR |
| RTOS | FreeRTOS, Mbed os | No support from vendor |
| Soft Devices | Multiprotocol Soft devices | No Soft devices |

Table 2: Microcontroller comparison.

# 5 Real Time Operating System

## 5.1 Introduction

The real-time operating system (RTOS) [3] is an operating system used to execute time critical operations. There are two different variants of RTOS, classified according to time critical execution. An operating system generally meets a deadline is defined as Soft real-time operating system, for example MP3 player. But, if deadline is deterministic than it is Hard real-time system, for example aircraft control system. There are many RTOS are available in market, the main open source RTOS are FreeRTOS, Mbed OS, Contiki and non-open source RTOS are Vxworks, UCOS-III, etc.

The important features of real-time operating systems are,

- **Reliability**: The system must be reliable, system should operate without any human intervention.
- **Predictability**: The system should complete tasks within time frames. That would define deterministic nature of RTOS
- **Performance**: This characteristic defines overall performance of system in terms of power consumption, processing and throughput, etc.
- **Compactness**: The system should be compact in nature, overall system size, memory, etc.
- **Scalability**: The system should be scalable up or down to meet application specific requirements.

### 5.1.1 nRF52 RTOS support

The nRF52 SDK [11] supports for multiple real-time operating systems, FreeRTOS [14], ARM Mbed OS [15], RTX [16] and provides example project to integrate applications.

## 5.2 FreeRTOS

The FreeRTOS [14] is a widely used real-time operating system in open source community. FreeRTOS is a very small and simple real-time operating system, written in C language and the microkernel consists of just three C files along with small assembly functions. The software is an open source distributed under MIT license.

### 5.2.1 Features

- FreeRTOS scheduler – priority based, pre-emptive and supports for priority inheritance.
- Supports for tickles mode for low power applications.
- FreeRTOS official support for more than 30 embedded system architectures, like ARM, Coldfire, etc.
- Efficient software timers, execution tracing and stack overflow detection options.
- There are no software restrictions on number of real time tasks can be created and task priorities can be used.
- Free development tools for supported architectures, free embedded software source code and royalty free.

### 5.2.2 nRF52 FreeRTOS

The nRF52 SDK supports an implementation of a FreeRTOS port to nRF52 and the provides tickles idle mode. The FreeRTOS is perfectly works with all SoftDevice implementations, that means which never blocks any global interrupt when it has been running with a SoftDevice. The nRF52 uses critical region functions to maintain interrupt compatibility and BASEPRI register used to block interrupts below designed level.

The FreeRTOS port on nRF52 along with SoftDevice have certain limitations.

- The values of configLIBRARY_LOWEST_INTERRUPT_PRIORITY and NRF_APP_PRIORITY_LOW should be same.

- The SYSCALL priority should be set to higher or equal as compare to lowest application priority. to make sure this, set value of configLIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY to equal or higher than NRF_APP_PRIORITY_LOW.

- The behaviour of interrupt is not tested before the FreeRTOS initialization. Interrupts having priority level higher or equal to NRF_APP_PRIORITY_HIGH which uses FreeRTOS functions might break system initialization.

## 5.3 ARM Mbed

ARM Mbed [15] is a platform and operating system for Internet of Things devices. It provides the Real-time operating system, gateway, device management services, and cloud facility. The operating system ported on 32-bit ARM Cortex-M microcontrollers. ARM Mbed is a collaborative project managed by ARM technologies and its partners.

### 5.3.1 Mbed OS

Mbed OS [17] is an open source embedded operating system developed specifically for internet of things devices. The Mbed OS designed for products based on an ARM cortex-M microcontroller, including security, connectivity, and drivers for sensors and I/O devices.

### 5.3.2 Features of Mbed OS

- **IDE and Toolchains:** Development of Mbed OS supports several IDE's and tools available in market, ARM compiler 5, GCC, IAR [13], Keil [12], etc.

- **Devices and RTOS:** The Mbed OS supports for wide range of ARM Cortex -M based devices. An IoT application can prototype easily on low cost developments boards. RTOS core is based open source CMSIS RTOS-RTX [16].

- **Connectivity:** Varies connectivity options available in Mbed OS, supports for WIFI, Bluetooth LE, NFC, Thread, LoRa LPWAN, Ethernet, RFID, Cellular, etc.

- **Modularity:** The software is structured such way that require modules are included automatically and designer just need to focus on application code.

- **Security:** Mbed OS supports for multilayer security that helps to protect IoT solutions. uVisor kernel restrict access to memory and peripherals. The communication security uses SSL and TSL methods to protect data.

- **Open source:** Published under apache 2.0 license, software can be used for commercial and personal projects.

### 5.3.3 nRF52 Mbed OS

The ARM Mbed provide free embedded software for nRF52 and it is royalty free. Also, development tools for software development provided by ARM, which comes for free of cost.

### 5.3.4 Software development tools

The arm Mbed ecosystem provides rich set of software development tools support. They are categorised as arm Mbed Online Compiler and arm Mbed CLI. Depending on development requirements choose software development tools.

### 5.3.5 ARM Mbed online compiler

The ARM Mbed online compiler, which allows to build Mbed os applications without installing a toolchain and it provides facility for editing, compiling source online. The online libraries help to generate optimized code.

## 5.4 RTOS Comparison

| Features | FreeRTOS [14] | Mbed OS [17] |
|---|---|---|
| Developer | Real Time Engineers Ltd | Collaborative project managed by ARM and its partners |
| Language | C, Assembly | C, C++ |
| Platforms | ARM, Atmel AVR, AVR32, Microblaze, Cold fire, Fujitsu, X86, TMS570, etc. | 32-bit ARM Cortex-M |
| Kernel | Microkernel | Hybrid |

Table 3: RTOS comparison.

The nRF52 software development kit supports for different Real Time Operating Systems. I have considered FreeRTOS and Mbed OS for evaluation purpose. Finally, I have chosen FreeRTOS for Inertial Motion Sensor prototyping. It has tickles sleep mode control and supports many other microcontroller architectures other than ARM. The kernel itself micro kernel and require very less flash and RAM memory. According IMS application software using FreeRTOS consumes 27.264kB flash and 9.6kB of RAM memory and have idle task for handling tickles sleep mode. Software is written in C programming language.

# 6 BNO055 Orientation Sensor

## 6.1 Introduction

The BNO055 [18] is the first in a new family of Application Specific Sensor Nodes (ASSN) implementing an intelligent 9-axis Absolute Orientation Sensor, which includes sensors and sensor fusion in a single package. The BNO055 is a System in Package (SiP), integrating a triaxial 14-bit accelerometer, a triaxial 16-bit gyroscope with a range of ±2000 degrees per second, a triaxial geomagnetic sensor and a 32-bit cortex M0+ microcontroller running with BSX FusionLib software. The sensor fused data provides Quaternion, Euler angles, Rotation vector, linear acceleration, Gravity, Heading, etc.

## 6.2 Integral blocks

The BNO055 is an embedded with multiple sensor and fusion software. It has accelerometer, gyroscope, and magnetometer sensor along with cortex M0+ microcontroller controlled by proprietary fusion software.

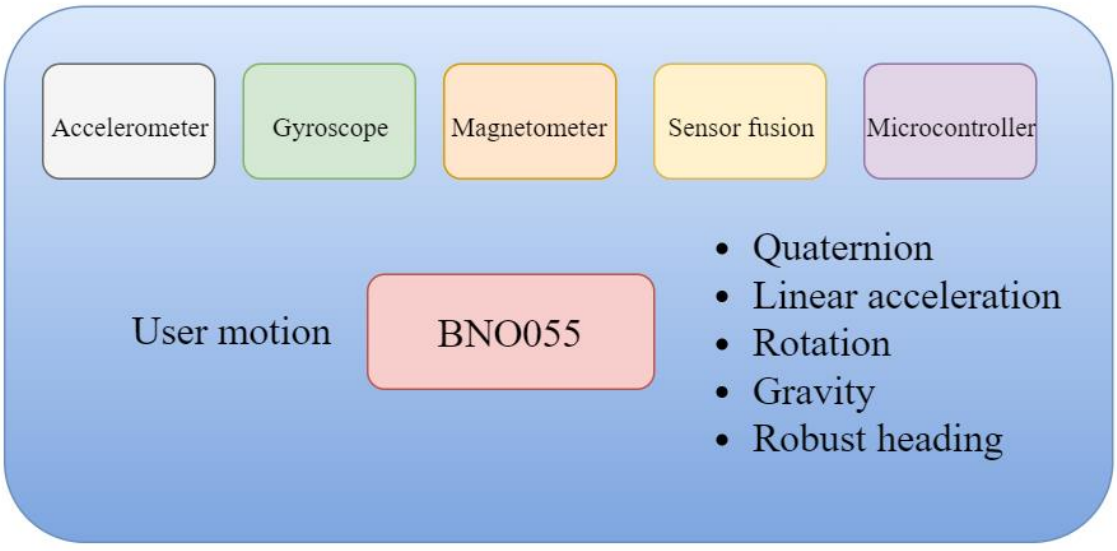


Figure 6: BNO055 internal blocks.

### 6.2.1 Accelerometer sensor

The accelerometer measures acceleration ranges from ±2g/±4g/±8g/±16g and with low pass filter bandwidth of 1kHz to <8kHz. It can operate normal, suspend, low power, standby, and deep suspend mode. Sensor has on chip motion triggered interrupt and configured to any motion detection, slow or no motion detection, and high-g detection.

### 6.2.2 Gyroscope sensor

The gyroscope sensor measure ranges from ±125º/s to ±2000º/s and with low pass filter bandwidth of 523Hz to 12Hz. Sensor operates in five different modes, normal, fast power up, deep suspend, advance power saves. On chip interrupts controller can be set to any motion detection, high rate.

### 6.2.3 Magnetometer sensor

The magnetometer sensor measure ranges from ±1300µT (x and y – axis) to ±2500µT (z – axis) and magnetic field resolution of~0.3µT. sensor operates in four different modes, low power, regular, enhanced regular, high accuracy and with four power modes of operation, normal, sleep, suspend, force.

### 6.2.4 Microcontroller and Software

The BNO055 orientation sensor design with powerful 32bit cortex M0+ microcontroller and running with Bosch Sensortec sensor fusion software and this software not exposes to user programming.

## 6.3 System architecture

As discussed in above section BNO055 sensor embedded with multiple sensors and microcontroller. The below figure shows architecture of BNO055 sensor.

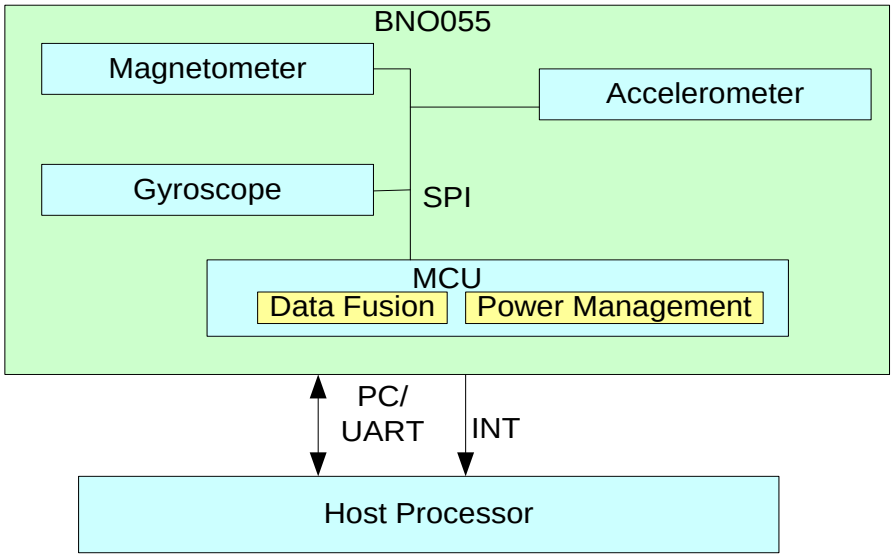


Figure 7: BNO055 system architecture.

The BNO055 microcontroller controls interfaces of internal and external devices. Accelerometer, Gyroscope and Magnetometer sensor are embedded in System in Package(SiP) and uses serial peripheral interface(SPI) for communication. And Controller manages sensor data read/write and power management according to register settings. BNO055 can be interface with external processor using I2C or UART communication protocol.

The below picture is BNO055 sensor supported by UART protocol and this sensor is used IMS prototyping purpose. The sensor board was developed for TUT.

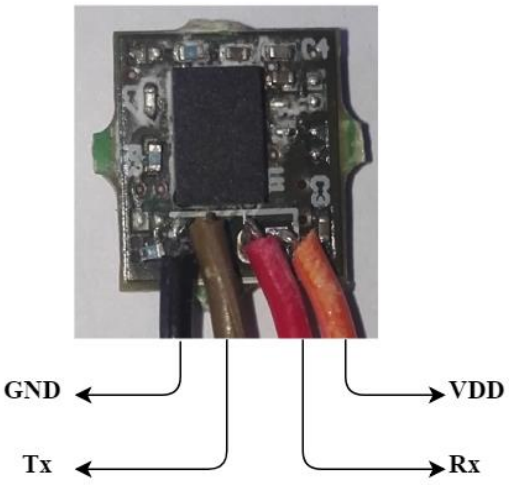


Figure 8: IMS prototype sensor.

### 6.3.1 Power management

The BNO055 sensor can be configured to run in three different modes. Normal mode, low power mode, and suspend mode. The power mode selected through writing to the PWR_MODE register as defined in below table.

| Parameter | Value | [Reg Addr]: Reg value |
|---|---|---|
| Power mode | Normal mode | [PWR_MODE]: xxxxxx00b |
| | Low power mode | [PWR_MODE]: xxxxxx01b |
| | Suspend mode | [PWR_MODE]: xxxxxx10b |

Table 4: BNO055 power management.

In normal mode all embedded sensors are switched ON. The register map and internal peripherals are operated in this mode. If there no activity sensor can be configure to low power mode. So that, power is saved. In this mode accelerometer is active. Once motion detected, the system is woken up and entered normal mode. The interrupt pins can also be configured to inform host controller about mode change. In suspend mode system is paused and kept in sleep mode. Values are not updated into mapped registers. To exit from suspend mode changed by writing to the PWR_MODE register.

### 6.3.2 Operation modes

The BNO055 provides a variety of operation modes, mainly Fusion and non-fusion mode. the appropriate operation mode selected based on requirement. The default mode after power on reset is CONFIGMODE. According to operation mode sensor are powered on. While the sensors whose signals not required kept in suspend mode.

# 7 Implementation of Prototype Solution

## 7.1 IMS interface to nRF52

### 7.1.1 Hardware

The IMS board is connected to nRF52 development kit through UART interface [19]. The nRF52 transmitter line connected to receiver of IMS and nRF52 receiver line connected to transmitter of IMS. Power line(Vdd) of 3.3V and ground lines also connected to get power up IMS board. The below block diagram shows connections between nRF52 development kit and IMS.

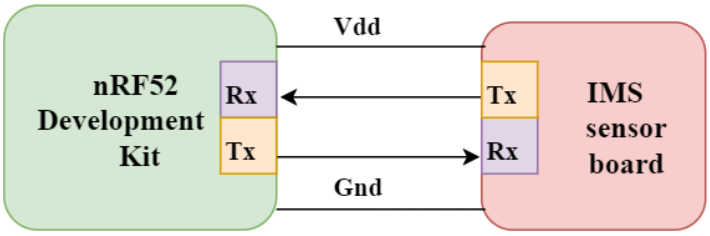


Figure 9: nRF52 and IMS board interface.

The below picture shows real interface of nRF52 development kit and IMS board.

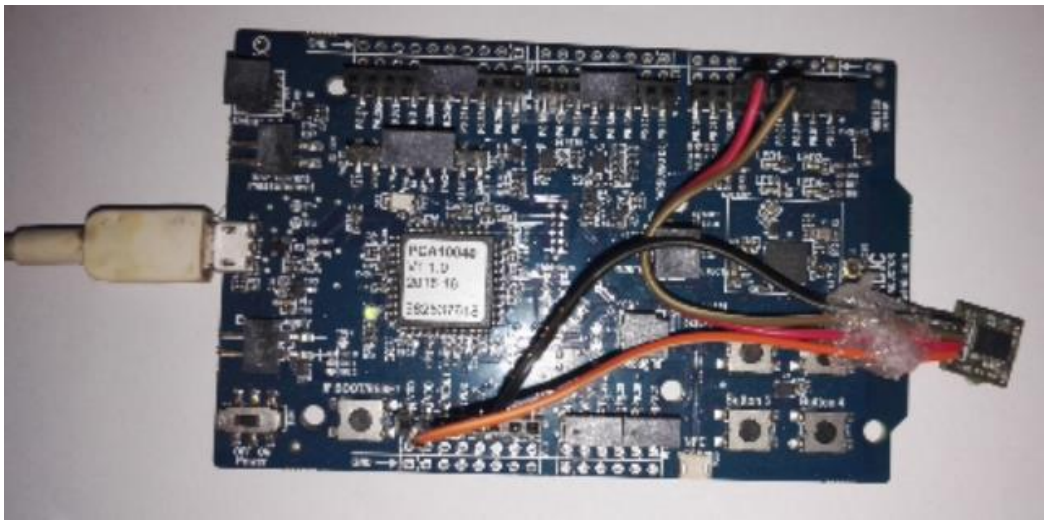

Figure 10: nRF dev kit and IMS board connection.

### 7.1.2 Software

The IMS data communication is controlled by nRF52 microcontroller. So, software for IMS is initialized after nRF52 bootup. very first, UART protocol configured to

115200bps, 8bit data, no parity and one stop bit [20]. Once the communication protocol initialized IMS is configured. prior configure sensor verify chip responds or not. So, to verify sensor, read chip ID and it should be 0xA0. After receiving valid chip ID select page0 address of IMS sensor configure NDOF mode [21] for sensor operation. The flow chart shows initialization of IMS sensor in software.

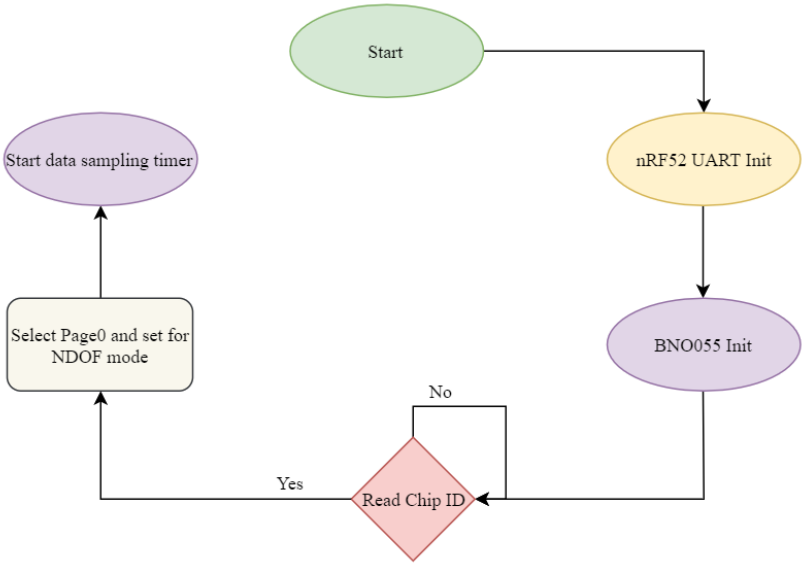


Figure 11: IMS sensor initialization.

The function `imm_sensor_application_init ()` is used for IMS sensor initialization in software.

```
void imm_sensor_application_init (void)
{
      imm_sensor_uart_init (); // Configures UART protocol
      imm_sensor_init ();      // IMM sensor configuring
      imm_sensor_timers_init (); // Starting sampling timer
}
```

## 7.2 IMS communication packets

According to implementation, IMS system follows communication packet structures [22]. They are sensor command packet, sensor response and ble application packet.

### 7.2.1 Command packet

To communicate with IMS sensor, the nRF52 should follow UART packet structure defined by BNO055 sensor [22]. The command packet sent by nRF52 defined as follows,

the packet contains 4bytes. The first byte indicates start command(0xAA), second byte is read command(0x01), third byte defines data reading address and fourth byte indicates number of data to be read.

| No of bytes | Byte1 | Byte2 | Byte3 | Byte4 |
|---|---|---|---|---|
| Details | Start | Read | Address | length |
| Data | 0xAA | 0x01 | 0x08 | 0x12 |

Table 5: Sensor command packet.

### 7.2.2 Response packet

As a response for sensor command packet, the IMS sensor transmits response packet. The typical time to acknowledge for command packet is 1ms and maximum is 4ms. In case of successful, the IMS sensor will respond following message.

| No of bytes | Byte1 | Byte2 | Byte3 to Byte8 | Byte9 to Byte14 | Byte15 to Byte20 |
|---|---|---|---|---|---|
| Details | Response | Length | Accelerometer | Magnetometer | Gyroscope |
| Data | 0xBB | 0x14 | 0xXX | 0xXX | 0xXX |

Table 6: Sensor response packet.

In case of an error, the IMS sensor will respond with an error message and a status information.

| No of bytes | Byte1 | Byte2 | Comments |
|---|---|---|---|
| Details | Response | Status | |
| Data | 0xEE | 0x01 | Write success |
| | | 0x03 | Write fail |
| | | 0x04 | Register map invalid address |
| | | 0x06 | Wrong start byte |
| | | 0x07 | Bus over run error |
| | | 0x08 | Maximum length error |
| | | 0x09 | Minimum length error |
| | | 0x0A | Receive character timeout |

Table 7: Sensor response error packet.

### 7.2.3 BLE application packet

The final packet turns over after receiving sensor data packet. The BLE application packet contains two major parts one is packet count and sensor data. The data will be sent out if

IMS data service indication flag is enabled in Bluetooth Low Energy stack. Below table details ble application packet.

| No of bytes | Byte1 to Byte2 | Byte3 to Byte8 | Byte9 to Byte14 | Byte15 to Byte20 |
|---|---|---|---|---|
| Details Data | Packet count 0xXX | Accelerometer 0xXX | Magnetometer 0xXX | Gyroscope 0xXX |

Table 8: BLE application packet.

In case of an error, the ble application packet will respond with an error message and a status information.

| No of bytes | Byte1 to Byte2 | Byte3 |
|---|---|---|
| Details Data | Packet count 0xXX | Error status 0xXX |

Table 9: BLE application error status packet.

## 7.3 IMS application using nRF52 SDK

The IMS applications developed using nRF52 software development kit [23]. Different version of nRF52 SDK available in following link [23]. I have chosen nRF5 SDK v11.0.0 for IMS application development. The SDK comes with BLE stack, example code and SoftDevice hex files. SoftDevice is a proprietary software of Nordic semiconductor and only API's exposed to application development. The application software can be developed using following IDE's Keil(up to 32kb free), Segger studio, GCC and its freely available.

A standalone IMS application is implemented on top of UART example code provided by nRF5 SDK. The sequence of software execution is as shown in below flow chart. Software initialization starts after nRF52 bootup, the application software configures IMS sensor and Bluetooth Low Energy stack of nRF52. IMS sampling timer starts after all peripheral initialized, default sampling period is 100ms. Sampling period could be changed in run time using BLE IMS sampling service in multiples of 100ms. The sampled IMS data is transferred using BLE IMS data service.

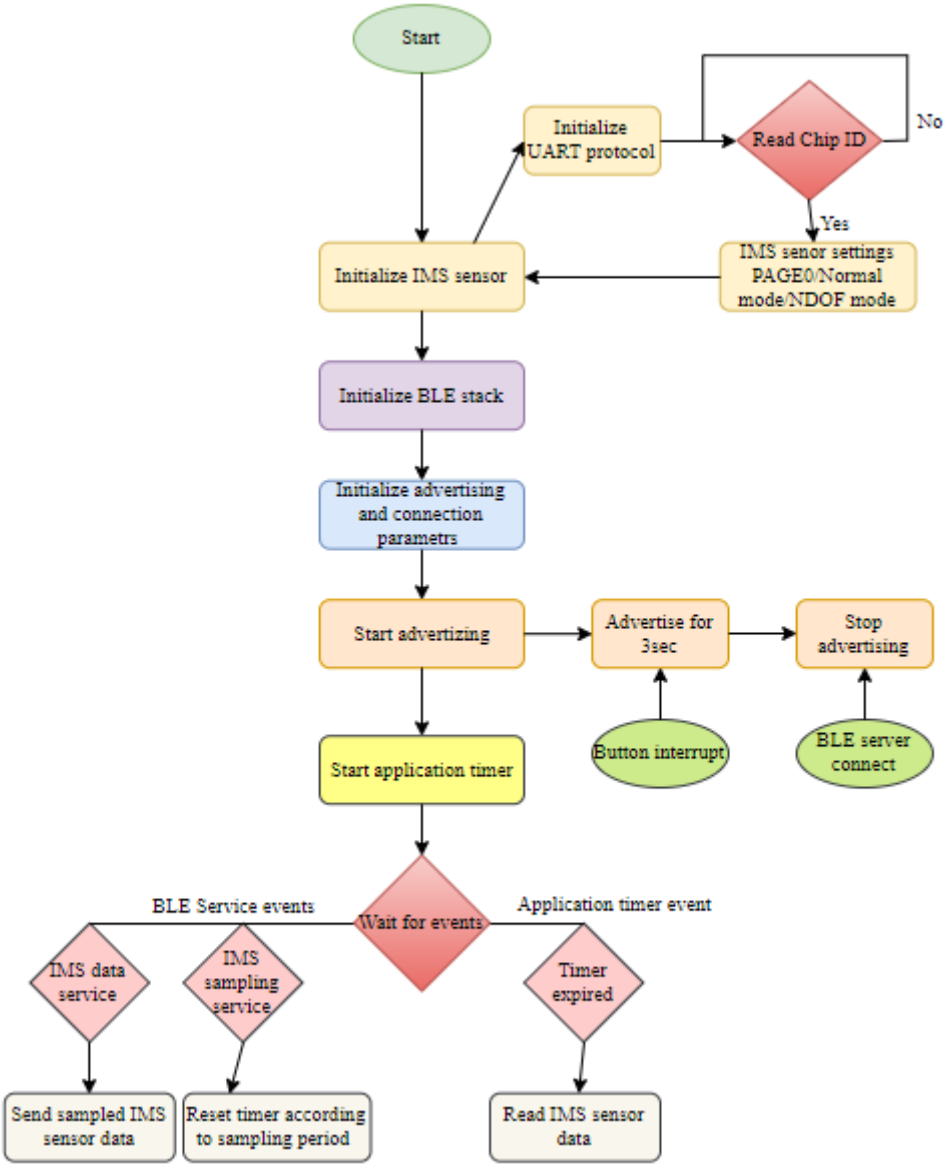


Figure 12: IMS application using nRF52 SDK.

Two custom profiles are created in standalone IMS application, one is for sensor data communication and other one is for sensor data sampling.

IMS data custom service UUID: 6E400027-B5A3-F393-E0A9-E50E24DCCA9E

IMS sampling custom service UUID: 6E400026-B5A3-F393-E0A9-E50E24DCCA9E

Default data sampling rate: 100ms

## 7.4 IMS application using FreeRTOS

The nRF52 SDK also supports for FreeRTOS application development. I have developed IMS application software using FreeRTOS provided by nRF52 SDK. The nRF52 ported FreeRTOS provides many functionalities like tickles idle mode, semaphores, timers, device drivers, etc. To make use of FreeRTOS functionalities in IMS application two tasks are created, one is for handling Bluetooth Low Energy stack and other one is for handling nRF52 sleep mode. In this application, four BLE services are available. Two custom profiles and two standard profiles.

IMS data custom service UUID is 6E400027-B5A3-F393-E0A9-E50E24DCCA9E and notification property is activated.

IMS sampling custom service UUID is 6E400026-B5A3-F393-E0A9-E50E24DCCA9E and write property is enabled.

Battery standard service UUID is 0x180F and notification property is enabled, simulated battery samples are notifying for every one second.

Device information standard service UUID is 0x180A and read only property is enabled.

The below flow chart shows software execution sequence of IMS application using FreeRTOS.

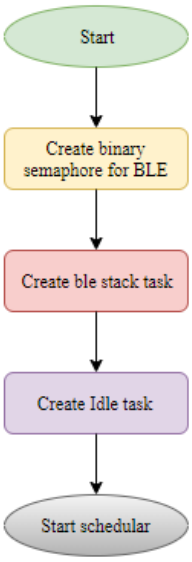


Figure 13: IMS application using FreeRTOS.

### 7.4.1 Idle task

One of the core feature of FreeRTOS is tickles sleep mode. In IMS application software idle task handles tickles sleeping of nRF52 microcontroller. The below flow chart shows idle task execution sequence.

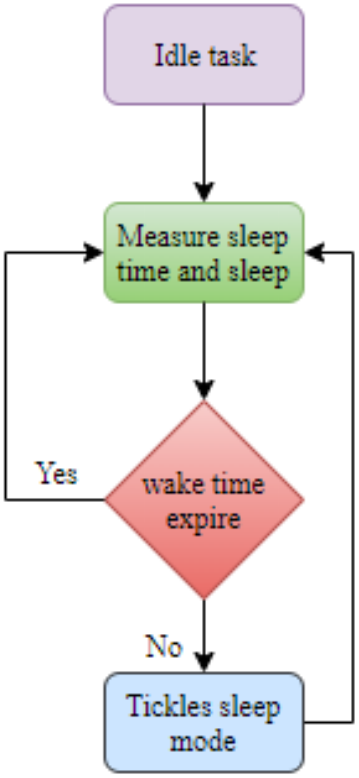


Figure 14: IMS application idle task.

### 7.4.2 BLE stack task

The BLE stack task handles BLE services and advertising BLE packets for connection. The below flow chart shows execution of BLE stack task.

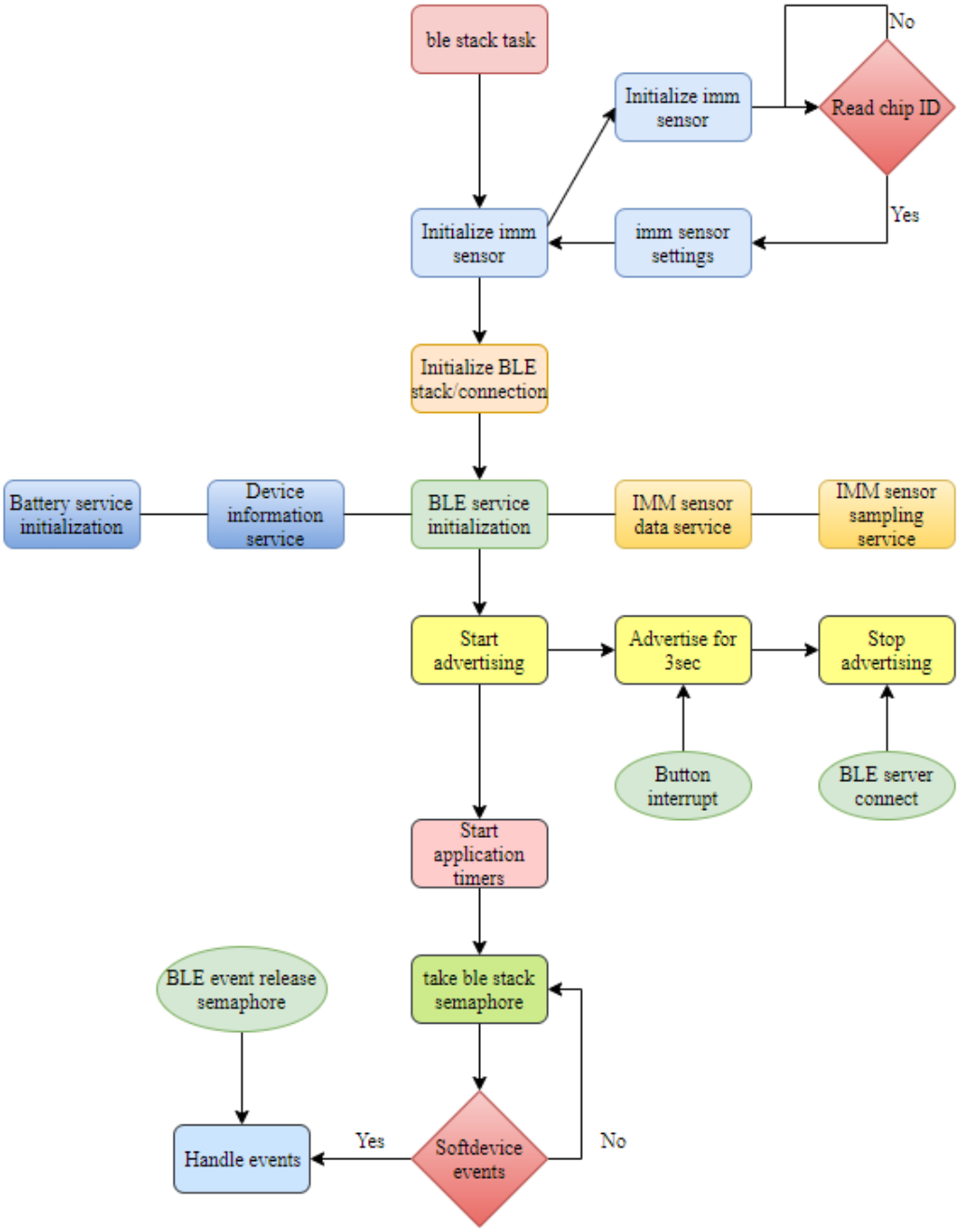


Figure 15: IMS application BLE stack task.

# 8 IMS Performance Assessment

## 8.1 Throughput

The maximum achievable throughput is depending on different factors. The major factors are listed here, selection of connection interval which is part of BLE connection parameter. According to nRF52 SDK the minimum connection interval is 16ms and maximum connection interval is 60ms. The second major factor is operating system which is handling BLE stack call back functions and notification or indication responses. Sometimes transmitted or received data from BLE require an acknowledgement, if that not handled on time. This would significantly reduce data rate. Third factor is number of packets transmitted per connection event. The nRF52 can both send or receive up to 6 packets per interval. Some of the android devices supports ~4 packets per interval and iOS devices supports up to 6 packets per interval. Other minor factors also influence on throughput, such as interference.

Now throughput can be calculated as below, number of packets per interval n and T is a connection interval.

$$Throurghput = n * 20B * \frac{1}{T} = 6 * 20B * \frac{1}{0.016s} = \frac{7.5kB}{s} = 60\ kbps$$

Equation 1: Throughput calculation.

## 8.2 Range test

The nRF52 ble stack supports for different transmitter power settings, according application requirement transmitter power can be set by using SoftDevice function call sd_ble_gap_tx_power_set(). The below table shows different transmitter power and achieved distance in meters.

| Transmitter power | Distance achieved in meters |
|---|---|
| −40 dBm | ≤ 1 |
| −30 dBm | ≤ 1.5 |
| −20 dBm | 2 |
| −16 dBm | 3 |
| −12 dBm | 5 |
| −8 dBm | 7 |
| −4 dBm | 12 |
| 0 dBm | 16 |
| 4 dBm | 18 |
| 8 dBm | Not supported by nRF52832 |

Table 10: Range test result.

## 8.3 IMS application test

### 8.3.1 IMS application test using Mobile

Hardware required: nRF52 development kit(PCA10040) [24], IMS sensor board, Mobile with Bluetooth Low Energy supported for example Honor 7 [25].

Software required: IMS standalone application software or IMS application software using FreeRTOS, nRF connect android mobile application [26].

Procedure: Install nRF connect android mobile application on mobile. Load IMS application hex file to nRF52 development kit using nRFgo studio software [27]. Open nRF connect application on mobile look for IMS application advertising device and connect to that. The below pictures show sequence of IMS application software sending sensor data to mobile.

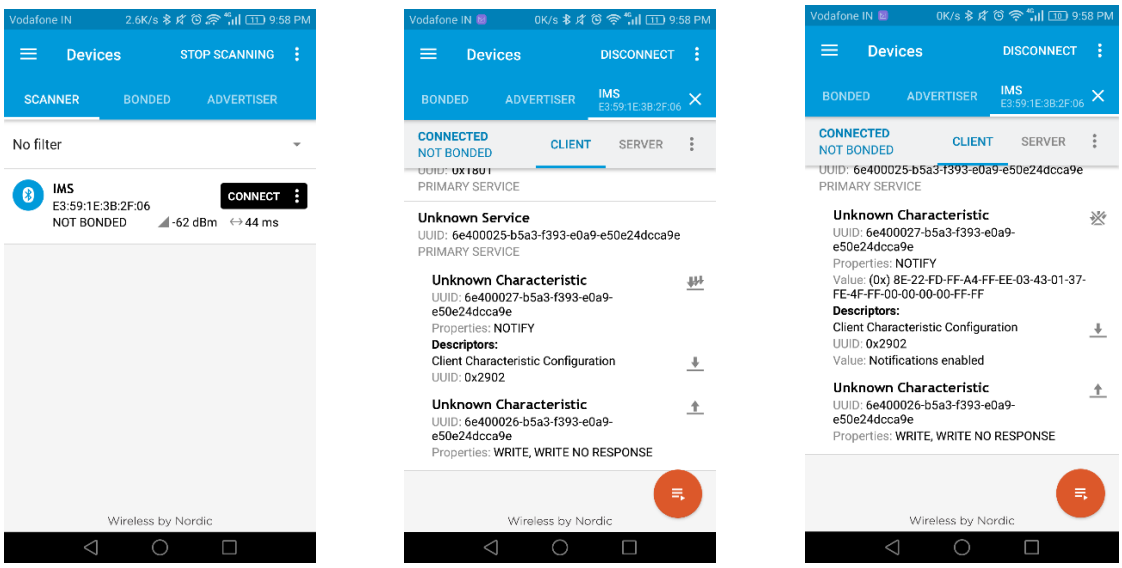


Figure 16: BLE mobile app connection sequence.

The IMS application transmits data for every 100ms interval using IMS data service (0x0027 handle) and interval can be modified using IMS data sampling service (0x0026 handle). Data sampling interval should be multiples of 100ms.

### 8.3.2 IMS application test using Laptop

Hardware required: nRF52 development kit(PCA10040) [24], nRF51 USB dongle, IMS sensor board.

Software required: IMS standalone application software or IMS application software using FreeRTOS, nRF connect for desktop software [28].

Procedure: Connect nRF51 USB dongle to laptop and run nRF connect desktop software, select respective serial port for dongle and set for 115200 baudrate. Scan for BLE devices and connect to IMS application device. After connecting BLE services are displayed on application software. Enable notification of IMS data service and nRF connect software start receiving sensor data for every 100ms interval and data is displayed on console window. using IMS sampling service sensor data sampling interval could be changed in multiples of 100ms. The below picture shows setup for IMS application test setup using laptop.

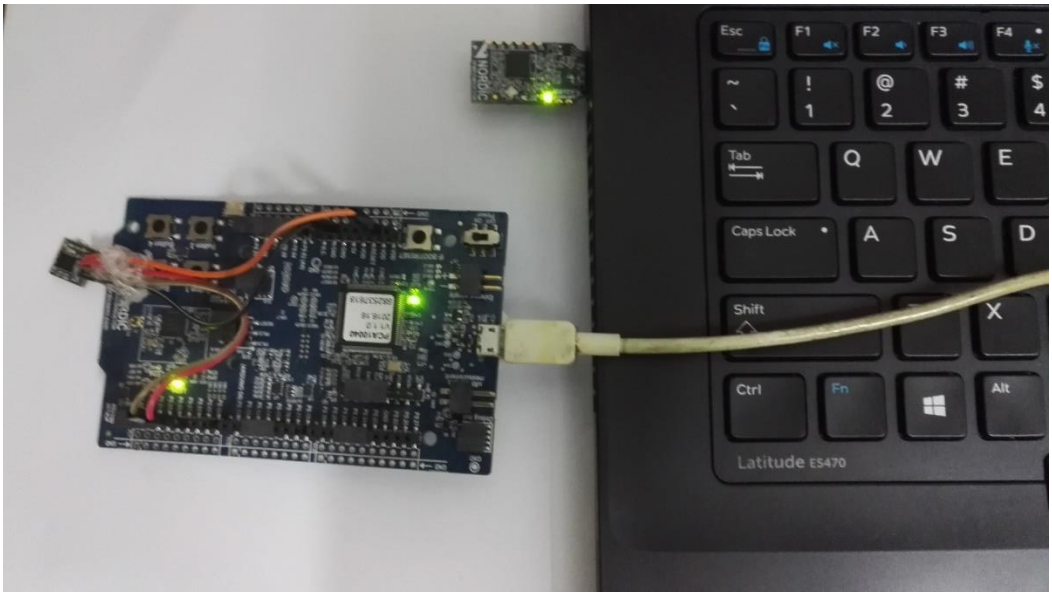


Figure 17: IMS application test setup using laptop.

The below picture is nRF connect desktop software screen shot and could be able to see IMS application device and sensor data.

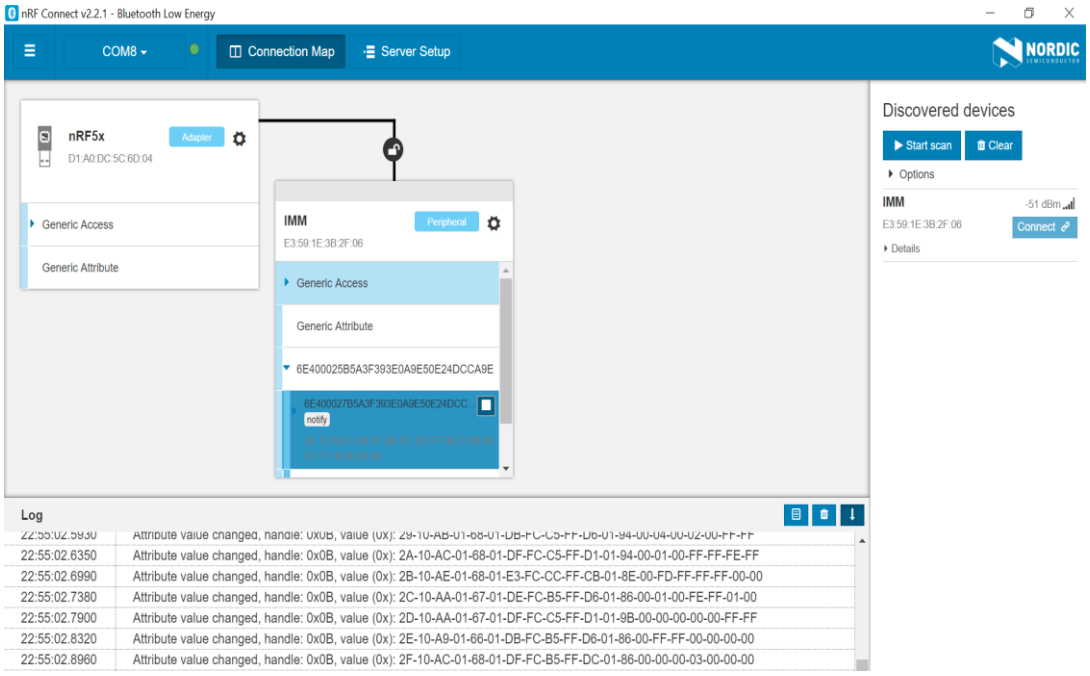


Figure 18: nRF connect desktop software data.

# 9 Conclusion

The powerful and energy efficient device nRF52840 wireless microcontroller from Nordic Semiconductor was selected for IMS prototype work. So, new IMS prototype was implemented using nRF52 series development board. The sensor software prototype was implemented on nRF52840 PCA10040 development board that supports different real-time Operating Systems. I have considered FreeRTOS and Mbed OS for evaluation purpose. Finally, FreeRTOS was selected for IMS prototype. FreeRTOS has energy efficient tickles sleep mode control, small footprint micro kernel and it supports other microcontroller architectures besides of ARM as well.

I implemented IMS application software based on FreeRTOS on nRF52840 evaluation board. Software uses only 27.264kB flash and 9.6kB of RAM memory and supports tickles sleep mode. Software was written in C programming language.

According to range test results nRF52 PCA10040 hardware development kit could be able to send IMS sensor data 18 meters at 4 dBm of transmitter power. Range could be increased by using another version of nRF52 chip. However, 10-15 meters of communication range is well enough for the body area health applications.

The default IMS data sampling interval is set for 100ms. For every 100ms microcontroller collect data from sensor and send it over BLE IMS data service. Data sampling interval can be changed using by calling write property of IMS data sampling service. The sampling rate only accepts multiples of 100ms. I have tested up to 20ms sampling interval and able to receive without data lose.

# References


[1] M. A. Mazidi, The 8051 Microcontroller and Embedded Systems: Using Assembly and C, Pearson.

[2] B. S. I. Group, *Core_V4.0.pdf,* https://www.bluetooth.com/.

[3] Wikipedia, "Real time operating systems," pp. https://en.wikipedia.org/wiki/Real-time_operating_system.

[4] N. Semiconductors, nRF52, [Online]. Available: http://www.nordicsemi.com/eng/Products/nRF52-Series-SoC.

[5] N. Semiconductors, nRF51 USB dongle, [Online]. Available: https://www.nordicsemi.com/eng/Products/nRF51-Dongle.

[6] B. Sensortec, BNO055, [Online]. Available: https://www.bosch-sensortec.com/bst/products/all_products/bno055.

[7] Blutooth, "Bluetooth 4.0 Core specifications," [Online]. Available: https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=229737.

[8] B. SIG, "Core specifications," [Online]. Available: https://www.bluetooth.com/specifications/bluetooth-core-specification.

[9] N. semiconductors, "Bluetooth low energy stack," [Online]. Available: http://infocenter.nordicsemi.com/index.jsp.

[10] B. S. i. g. -. SIG, "GATT Specifications," [Online]. Available: https://www.bluetooth.com/specifications/gatt.

[11] T. instruments, "CC2540 datasheet," [Online]. Available: http://www.ti.com/lit/ds/symlink/cc2540.pdf.

[12] N. Semiconductor, "nRF52840 Product Specification," [Online]. Available: http://infocenter.nordicsemi.com/pdf/nRF52840_PS_v1.0.pdf.

[13] N. semiconductor, "nRF52832 development kit," [Online]. Available: http://infocenter.nordicsemi.com/index.jsp.

[14] K. software, "Keil software for ARM architecture microcontrollers," [Online]. Available: http://www.keil.com/.

[15] I. Systems, "IAR Embedded Workbench," [Online]. Available: https://www.iar.com/.

[16] A. Mbed, "Nordic nRF52-DK," [Online]. Available: https://os.mbed.com/platforms/Nordic-nRF52-DK/.

[17] FreeRTOS, "The FreeRTOS™ Kernel - Real time operating system," [Online]. Available: https://www.freertos.org/.

[18] A. Mbed, "Arm Mbed IoT Device Platform," [Online]. Available: https://www.mbed.com/en/.

[19] A. Keil, "RTX Real-Time Operating System," [Online]. Available: http://www.keil.com/arm/rl-arm/kernel.asp.

[20] A. M. OS, "Mbed OS 5 - Real time operating system," [Online]. Available: https://os.mbed.com/.

[21] B. Sensortec, "BNO055 Intelligent 9-axis absolute orientation sensor," [Online]. Available: https://ae-bst.resource.bosch.com/.

[22] BOSCH, "BNO055 UART interface," [Online]. Available: https://ae-bst.resource.bosch.com/media/_tech/media/application_notes/BST-BNO055-AN012-00.pdf.

[23] BOSCH, *Application note - BNO055 UART interface, BNO055 UART protocol*, BOSCH.

[24] BOSCH, "BST_BNO055_DS000_14 Datasheet," in *BNO055 Intelligent 9-axis absolute orientation sensor*, BOSCH, pp. 21, 23.

[25] BOSCH, "BST-BNO055-AN012-00 - UART interface," in *BNO055 UART interface*, BOSCH, p. 3.

[26] N. Semiconductor, *nRF52 Software development kit*, http://infocenter.nordicsemi.com/index.jsp.

[27] N. Semiconductor, "nRF52832 Development Kit v1.1.x User Guide," [Online]. Available: http://infocenter.nordicsemi.com/pdf/nRF52_DK_User_Guide_v1.2.pdf.

[28] gsmarena, *Honor 7 specifications*, https://www.gsmarena.com/huawei_honor_7-7269.php.

[29] N. Semiconductor, *nRF Connect for Mobile*, https://www.nordicsemi.com/eng/Products/Nordic-mobile-Apps/nRF-Connect-for-Mobile.

[30] N. Semiconductor, *nRFgo Studio-Win64*, https://www.nordicsemi.com/eng/nordic/Products/nRFgo-Studio/nRFgo-Studio-Win64/14964.

[31] N. Semiconductor, "nRF Connect for desktop," [Online]. Available: https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF-Connect-for-desktop.

# APPENDIX A **Hardware**

## Typical UART connection to BNO055