VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Samuel Colley

Software Prototype for NB-IoT Device Debugging using SMS

Technology and Communication

2025

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Information Technology

ABSTRACT

Author Samuel Colley
Title Software Prototype for NB-IoT Device Debugging Using SMS
Year 2025
Language English
Pages 28
Name of Supervisor Smail Menani

This thesis work presents the development of a software prototype for debugging NB-IoT devices using the SMS protocol. The main goal was to enable fallback communication with remote IoT devices when the standard NB-IoT network connection is unavailable. By using SMS, which is a simple and widely used technology, the prototype allows troubleshooting even under difficult network conditions.

The implementation used the Nordic Semiconductor nRF9160-DK board, Onomondo M2M SIM cards, and Zephyr RTOS. AT commands were used to connect with the modem and perform the debugging tasks. In the future, adding features like automatic alerts and supporting other technologies such as LTE-M could improve the system.

Keywords IoT, NB-IoT, SMS, Remote debugging

CONTENTS

ABSTRACT 2
List of Figures and Tables 4
Concepts and Abbreviations 5
1 INTRODUCTION 6
1.1 Background 6
1.2 What is the Problem 6
1.3 Why This Solution 6
1.4 Scope of the Thesis Work 7
2 TECHNOLOGY BACKGROUND 8
2.1 Narrowband IOT/3GPP Standard/LTE-M 8
2.1.1. LTE-M 8
2.1.2 NB-IoT 10
2.2 SMS Protocol in NB-IoT 11
2.3 SIM Cards 12
3 IMPLEMENTATION 14
3.1 Nordic Semiconductor nRF9160-DK 14
3.2 Zephyr RTOS and nRF SDK 15
3.3 Onomondo M2M SIM 16
3.4 AT Commands 17
4 DEVELOPMENT PROCESS 18
4.1 Software Design 18
4.2 Implementation 19
4.3 Code Debugging 22
5 DISCUSSION 23
5.2 Challenges with PIN-less SIM Cards 24
5.3 Next steps in the development 26
5.4 Limitations 26
6 CONCLUSION 27
7 REFERENCES 29

List of Figures and Tables

Figure 1	The nRf9160dk board	Page 14
Figure 2	The nRF Connect app provided the nRF SDK tools.	Page 15
Figure 3	Screenshot of the nRF Connect cellular monitor.	Page 16
Figure 4	The Nordic Semiconductors SDK integrated in the Visual Studio Code environment.	Page 19
Figure 5	Code flow	Page 20
Figure 6	Requesting network status via SMS and sending Stop command via the Onomondo web interface.	Page 22
Figure 7	SMS logs in the Onomondo website.	Page 23
Figure 8	The old Android operating system PIN section that causes the error.	Page 25

Table 1	Comparison of NB-IoT and LTE-M.	Page 8-9
Table 2	Comparison of NB1 and NB2.	Page 10-11
Table 3	The four categories of AT commands and their identifiers.	Page 17

Concepts and Abbreviations

IoT	Internet of Things
NB-IoT	Abbreviation for Narrowband IoT
LPWAN	Low-power Wide Area Network
3GPP	A consortium of companies that creates a blueprint for cellular communication. Includes work on GSM, UMTS, LTE, and 5G NR standards.
LTE	Long Term Evolution, a 4G cellular standard
LTE-M	Long Term Evolution for Machines
GSM	Global System for Mobile Communications, a 2G cellular standard
RTOS	Real-time Operating System
SDK	Software Development Kit
SIM	Subscriber Identity Module
M2M	Machine to Machine
SMS	Short Message Service
SMPP	Short Message Peer-to-Peer
FOTA	Firmware-Over-The-Air, can update software or hardware without needing physical presence.
ICCID	Integrated Circuit Card Identifier, the physical ID for a SIM card
MSISDN	Mobile Station International Subscriber Directory Number, general phone number

1 INTRODUCTION

This section introduces the background of the thesis and describes the problem that is addressed.

1.1 Background

Remotely situated Internet of Things (IoT) devices such as temperature sensors, light sensors, and humidity sensors utilise a radio network connection to provide sensor data measurements to a central server. The 4G Narrow Band IoT (NB-IoT) radio protocol is particularly suited as a connection method as it provides wide coverage and is low power.

1.2 The Problem

A fault may occur where the NB-IoT 4G connection between a remote sensor and the server is lost. This could be due to, for example, device failure or network problems. In this case, the system administrator needs to diagnose and correct the fault condition. The NB-IoT standard includes support for the GSM Short Message Service (SMS). In some fault conditions, when a 4G connection is not available, it may be possible to establish a connection to the remote device using the SMS protocol. The target of this thesis is the development of a diagnostic protocol for NB-IoT 4G devices using the SMS protocol. The work is carried out for the company iProtoxi Ltd. who manufacture NB-IoT products and has identified the need for this solution to strengthen its product offering.

1.3 The Solution

An NB-IoT device can handle SMS if the capability is built into its hardware and firmware. SMS travels on 2G and 3G control channels in addition to newer generations, so it often works in areas where the 4G data connection is weak.

Since SMS packets are small in size, transmission failures due to limited bandwidth are close to none. The packet size of a SMS and a NB-Iot packet are relatively similar. NB-IoT can be deployed on both 2G and 4G, however, once a

4G network is built, then it would not make sense to rebuild it on 2G for a single use case.

With SMS it is possible to check the network connection, and in addition, it can send other debugging data to see if the device may be malfunctioning due to other reasons.

1.4 Scope of the Thesis

The content of this thesis work is to implement prototype software using the SMS protocol to communicate with a Nordic semiconductor NB-IoT board. This will then be validated for correct functionality over an NB-IoT network.

2 TECHNOLOGY BACKGROUND

In this section, the technology background of the topic is introduced. This includes:

1. Narrowband IOT/3GPP Standard/LTE-M (NB-IoT and LTE-M)
2. SMS protocol

2.1 Narrowband IOT/3GPP Standard/LTE-M

The two main standards for low power and low bandwidth access are NB-IoT and LTE-M. These two standards use radio signals that are on the licensed spectrum. LPWANs have a lower frequency than LTE, which allows for devices in more inaccessible places, such as underground, to be connected.

2.1.1. LTE-M

LTE-M is a modified LTE system that is designed to be an LPWAN. Since LTE-M is built on the existing LTE technology and systems, the protocols and software are interchangeable. The main problem with LTE-M is that, since it was built upon something that existed beforehand and not built specifically for LPWANs it must maintain compatibility with the original standard.

Table 1. Comparison of NB-IoT and LTE-M (Nordic Semiconductor, n.d.).

Specification	**NB-IoT**	**LTE-M**
Uplink and downlink speeds	Low (Kbit/s). LTE Cat NB2 offers max 159 Kbit/s uplink and 127 Kbit/s downlink. Only uses bandwidth 200kHz	High (Mbits/s). LTE Cat M2 offers max 7 Mbit/s uplink and 4 Mbit/s downlink. Uses bandwidth of 1.4MHz

Coverage/Penetration	Great coverage and great penetration. 15km range	Great coverage and a bit worse penetration. 11km range
Latency	1.5 - 10 s	50 - 100 ms
Global availability	Good availability. Requires new infrastructure. Cheaper to roll-out than LTE.	Great availability where LTE already exists. Built on existing LTE technology.
Roaming capability	Technically possible, but non-existent in reality.	Good roaming availability.
Module costs	Typically cheaper than LTE-M.	Typically more expensive than NB-IoT.
Power saving features	PSM and eDRX support.	PSM and eDRX support.
Battery lifetime estimate	Up to 10 years	Up to 10 years
Mobility	No handover when moving, only suitable for static devices.	Suitable for static and mobile devices.
Freedom to leave	Typically no SMS support. Not good for freedom to leave and eUICC (eSIM).	Supports freedom to leave and eUICC (eSIM).

2.1.2 NB-IoT

Narrowband IoT is a 3GPP standard name for LPWANs for cellular networks (Ground Control, 2024). The NB-IoT communication standard lets IoT devices operate via carrier networks, either within an existing GSM 2G/2.5G communication carrier wave, by using LTE 4G channels or independently.

One of the goals of NB-IoT is to boost the coverage extension beyond what existing cellular technologies offer. To do that, NB-IoT offers transmission repetitions and different bandwidth allocation configurations in uplink transmission.

NB-IoT reduces the power consumption of connected devices by sending less data while increasing system capacity and bandwidth efficiency, particularly in locations that aren't easily covered by traditional cellular technologies. NB-IoT-connected devices can have a battery life of more than 10 years in many use cases. NB-IoT can only be used for stationary IoT applications since it can not handle cellular tower handoffs. Only some of the network operators deploying NB-IoT support SMS.

NB1 is the 13th release of the NB-IoT 3GPP standards, followed by NB2 as the 14th release. NB2 is a significant upgrade of NB1, with upgrades to throughput, latency, mobility support, multicast, and more accessibility to communication options like firmware over the air and connected vehicles (Table 2) (3GPP TS 36.300, 2024). Even though the NB2 is an upgrade to NB1, both releases are still used.

Table 2. Comparison of NB1 and NB2 (Ratasuk et al., 2016).

Specification	**NB1**	**NB2**
MAX Downlink TBS	680 bits	2536 bits
MAX Uplink TBS	1000 bits	2536 bits

Maximum Transmission power	20, 23dBm	14, 20, 23dBm
Channel Bandwidth	180 KHz	180 KHz
Transmission Duplexity	Half	Half
Data Encryption	EPS-AKA	EPS-AKA

Non-terrestrial networks (NTN) NB-IoT is a variant that uses satellite networks instead of cellular networks. NB-IoT devices that use satellite networks are not very common and cost more. There is the development of a dual system, which would allow a device to switch between satellite and cellular networks.

2.2 SMS Protocol in NB-IoT

SMS is a protocol that is part of the GSM standard. A communication standard can be thought of as a set of instructions. Both parties have to use the specified standard to communicate with each other. The SMS protocol requires data associated with the message, including the length of the message, format, time stamp, and destination, to follow the standard. The maximum length of an SMS message is 160 characters. If a message is longer than 160 characters, then the message will be fragmented and sent in multiple SMS messages. Devices take these fragmented pieces and combine them, making a longer-looking SMS message. Another option to be able to send a message longer than 160 characters is to use the Multimedia Messaging Service (MMS), however, MMS is a completely different standard to SMS**.** SMS uses the GSM 7-bit default ASCII alphabet. Since all the data sent is purely binary, if the ASCII version used is different, then the message will be encoded and decoded differently, therefore making a possible miscommunication.

Short Message Peer-to-Peer (SMPP) is a widely used protocol which is an industry standard protocol designed to provide an interface for the transfer of short

messages between devices. It is a means by which applications can send and receive SMS messages to and from mobile devices. Applications do this using an SMPP connection to a Short Message Service Center (SMSC), SMS gateway, SMPP gateway or hub. The SMSC forms the core infrastructure for SMS transmissions, whereas the SMS gateway acts as a middleman between networks and applications. SMSC, also known as SMS-SC, is an SMS service centre that stores, forwards, converts and delivers SMS messages from one device to another. The purpose of an SMS gateway is to link up and connect two devices so that they can start their SMPP or SMS interactions. The gateway's main functions are to connect and convert protocols like HTTP to SMPP, message queuing and message routing. To send the data safely from one device or application to another, multiple modulation techniques are used. Such techniques include Gaussian minimum shift keying (GMSK) and frequency shift keying (FSK). GMSK is the preferred choice for SMS due to its smoothness and constancy.

When sending an SMS from a mobile device to another mobile device, it is first sent to the SMSC and then to the second mobile device. However, in the case of Machine-to-Machine (M2M) SIMs, most of the time the SMS is only sent to the SMSC and not forwarded to another device. The reason it does not matter is because most networks do not allow roaming SIMs to send SMS messages. This may seem quite a large oversight, however, M2M SIM providers provide a webservice that allows the sent message to the SMSC to be seen, so the destination address does not matter. The destination address or/and source address does not matter, and that is the reason why M2M SIMs do not have phone numbers. When sending the SMS from the SMSC to the M2M SIM, the source address can be a phone number or even a string without any special characters.

2.3 SIM Cards

A SIM card is a type of UICC (universal integrated circuit card). A SIM card has its own circuit, microcontroller and operating system. The Traditional Chip SIMs are designed to have only one purpose, which is decided during production, which

may be a general consumer use SIM card or an IoT-specific SIM card, which cannot be altered later on. For usage in mobile phones, this is not a problem since the SIM card can be physically changed quite easily. However, with cellular IoT devices, changing SIM cards can be a hassle or almost impossible. eSIM cards are designed to fix this problem(University of Helsinki, n.d.). eSIMs are SIM cards that are directly mounted onto the circuit board and have software which can be remotely updated. This makes it possible to add or remove operators without changing the physical SIM card in the device. The eSIM was standardised by the GSMA. An iSIM is an integrated SIM that, instead of being mounted onto the board, is integrated into the device's main processor (Hemus, 2024). This makes it so that the SIM card circuit is not required. This is possible because the iSIM has its own dedicated execution environment on the system-on-chip (SOC). The iSIM works in a completely different environment compared to a SIM or eSIM, and attempts to replicate standard behaviours of the physical SIM cards.

3 IMPLEMENTATION

This section details the specific implementation that has been made as part of this thesis work. The hardware, networking, and development tools used are described.

3.1 Nordic Semiconductor nRF9160-DK

The nRF9160-DK board (Figure 1) is manufactured by Nordic Semiconductors [8]. The nRF9160 DK is an affordable development kit for development on different telecommunication protocols, some including LTE-M, NB-IoT and GNSS. It also includes an nRF52840 board controller, which, for example, can be used to build a Bluetooth Low Energy gateway.

The board has an antenna designed for LTE-M and NB-IoT, which supports a wide range of bands. The LTE bands B1-B5, B8, B12-B14, B17-B20, B25, B26, B28 and B66 have been certified. The board also has a dedicated patch antenna for GNSS and a 2.4 GHz antenna, which can be used with Bluetooth LE. There are additional SWF RF connectors that can be used in tandem with the LTE-M/NB-IoT and 2.4 GHz antennas. There is also a connector which can be used to attach an external GNSS antenna to the board.

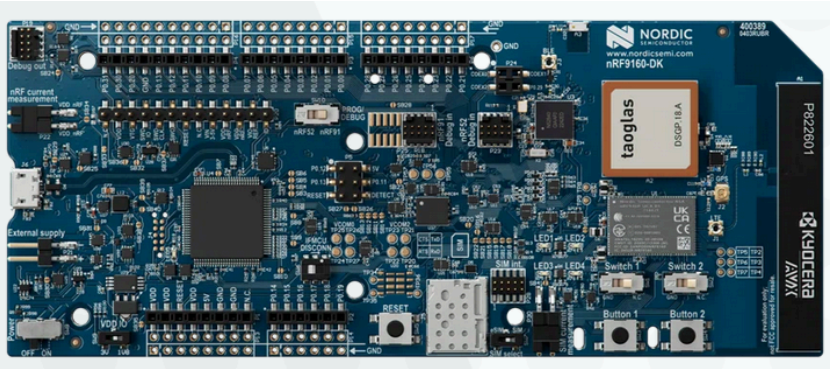

Figure 1. The nRf9160dk board

The Nrf9160SiP incorporates an Arm Cortex-M33 application processor purely for use in applications, an LTE modem, RF Front End (RFFE) and a power

management system. The Nrf9160SiP supports both LTE-M and NB-IoT. NRF9160SiP also has encrypted Firmware Over The Air.

3.2 Zephyr RTOS and nRF SDK

Zephyr RTOS is an open–source RTOS that the Nordic Semiconductor boards use. Other companies that use Zephyr in their systems are Intel, Texas Semiconductor, and more. Zephyr RTOS can read and compile both C and C++. Zephyr RTOS has its own command line functions and can be used in different environments. In the case of Nordic boards, the nRF SDK and other provided nRF applications attempt to streamline and simplify the process.

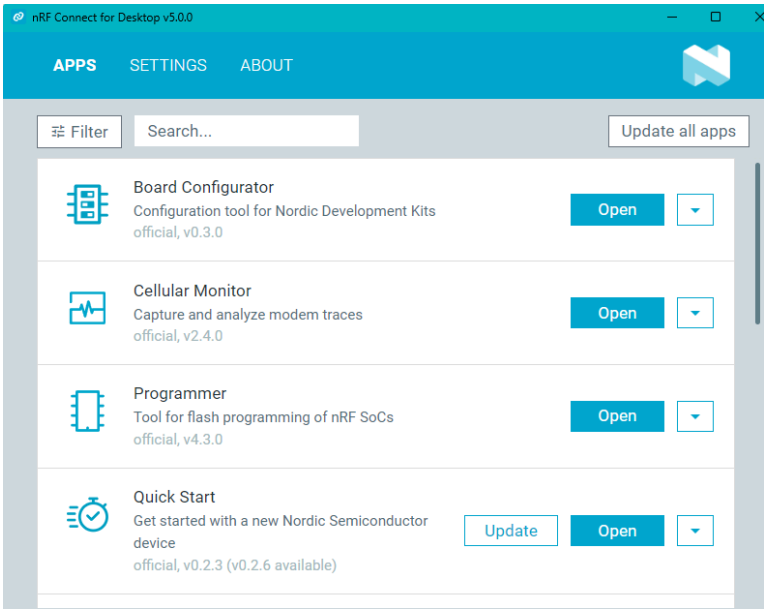

Figure 2. The nRF Connect app provided the nRF SDK tools.

The nRF Connect application (Figure 2) contains necessary tools like a serial terminal and cellular monitor (Figure 3).

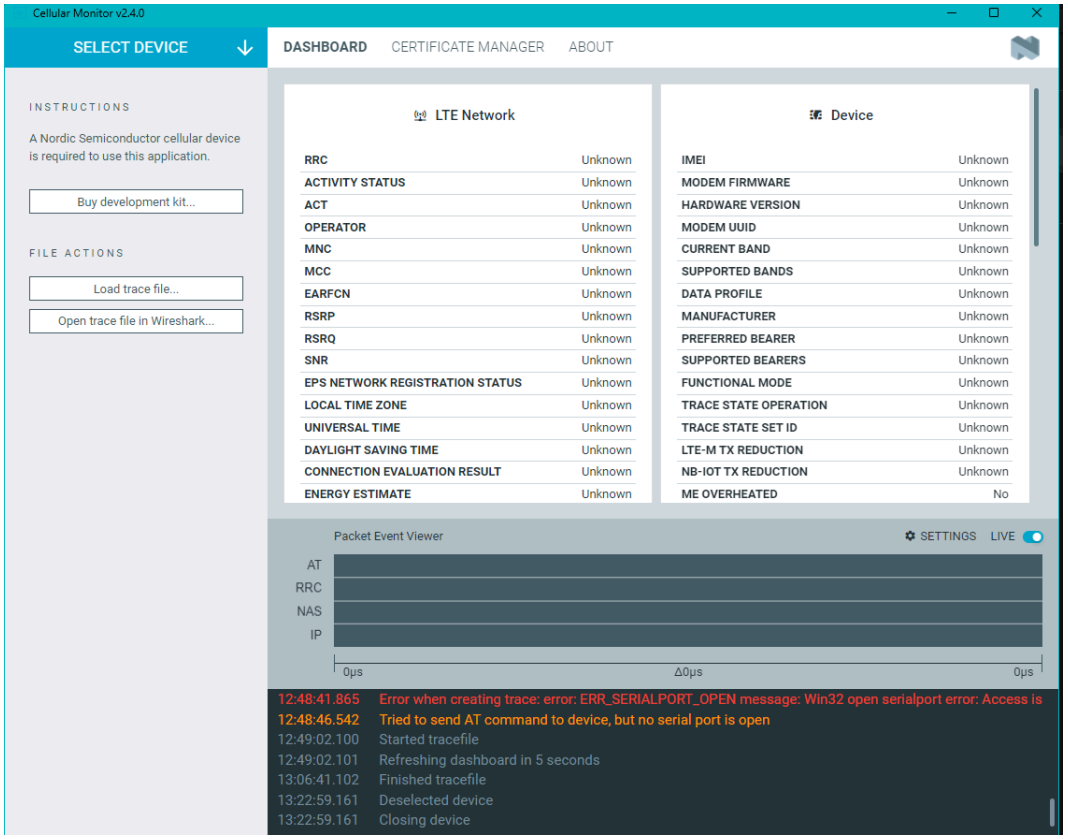

Figure 3. Screenshot of the nRF Connect cellular monitor.

There is also an nRF cloud IoT service for device and SIM card management. The nRF cloud service only supports ibasis SIM cards, which does not allow for SMS services. Therefore, this project used Onomondo M2M SIMs instead.

In the Zephyr files, testing programs can be found, for example, for cmock and NAT routing tests. Other tools integrated into Zephyr RTOS, such as LVGL GUI builder, are out of the scope of this work.

3.3 Onomondo M2M SIM

For the network connectivity, Onomondo was selected as the service provider. Onomondo is a virtual mobile network operator that provides global mobile connectivity tailored for Internet of Things (IoT) devices that operate on a single network layer instead of roaming agreements between operators. This allows IoT devices to operate seamlessly across borders without needing multiple SIM cards

or separate contracts with various operators. Onomondo removes the need for public networks by using VPN tunnels, TLS encryption, and private IPs.

3.4 AT Commands

To interface to the radio modem in the selected hardware, one option is to use AT commands. AT commands, also known as Hayes command set or attention commands, are a standardised command language used in connecting and configuring cellular connections. The ETSI GSM 07.05 (3GPP TS 27.005) specifies AT-style commands for managing the Short Message Service (SMS) features of GSM. They can also be used to get modem and connection status information. For example, AT+CMGS is the command to send an SMS, and AT+CMGF is the command to receive an SMS (Dev & Sunny, 2024). AT commands follow the following structure:

AT<COMMAND><SUFFIX><DATA><CARRIER RETURN>

There are four categories of AT commands: read, set, test, and execute. Each category has its own special character in the command sequence that indicates which category it is a part of (see Table 3).

Table 3. The four categories of AT commands and their identifiers.

Action	Description	Example
RUN	Indicated with =	AT+CLCK="SC",1,"0000"
READ	Indicated with ?	AT+CPIN?
TEST	Indicated with =?	AT+CPIN=?
EXECUTE	Indicated with nothing	AT+CSQ

4 DEVELOPMENT PROCESS

This section explains the development process

4.1 Software Design

Pseudocode for the software is as follows:

```
BEGIN
  // Initialize device to listen for incoming SMS
  Initialize_NB_IoT_Device()
  Enable_SMS_Receiving()

  //Register listener
    sms_register_listener(sms_callback, NULL)

  WHILE (True) DO
    // Wait for incoming SMS

    IF (SMS_Received()) THEN
      // Retrieve sender information
      sender = Get_SMS_Sender()
      // Get the content of the SMS
      sms_content = Get_SMS_Payload()
      // Decode the content of the SMS
      decoded_command = Decode_SMS_Content(sms_content)
      // Based on decoded content, execute relevant
action

        IF (decoded_command == "REQUEST_STATUS") THEN
        // Prepare the status report to send back
        status_report = Get_Device_Status()
        // Encode the response into an SMS format
        response_msg = Encode_SMS_Response(status_report)
        // Send SMS back to the sender
        Send_SMS(sender, response_msg)

        ELSE IF (decoded_command == "REQUEST_CONFIG")
THEN
        // Get the device configuration
        device_config = Get_Device_Config()
        // Encode the response into SMS format
        response_msg = Encode_SMS_Response(device_config)
        // Send SMS with the device configuration
        Send_SMS(sender, response_msg)

       ELSE
        // Handle unknown commands
        Send_SMS(sender, "Unknown Command")
```

```
        END IF
    END IF
  END WHILE
END
```

It is necessary to first register the device as a listener for communication to be possible. The listen state is the device waiting for incoming messages; listening to hear anything rather than listening to a specific message. When the message is received, it sets off a callback function. Inside this callback function is where the decoding and reading of everything in regards to the SMS message happens. Then another callback function is activated and compares the message to the instruction list. If the message is the same as an instruction, then it sets off an if statement, which then does the specific instruction, which in the case of this program reads and sends specific sensor or configuration data.

4.2 Implementation

Nordic Semiconductors suggests using Visual Studio Code as the environment. The SDK and toolchain become integrated with Visual Studio Code.

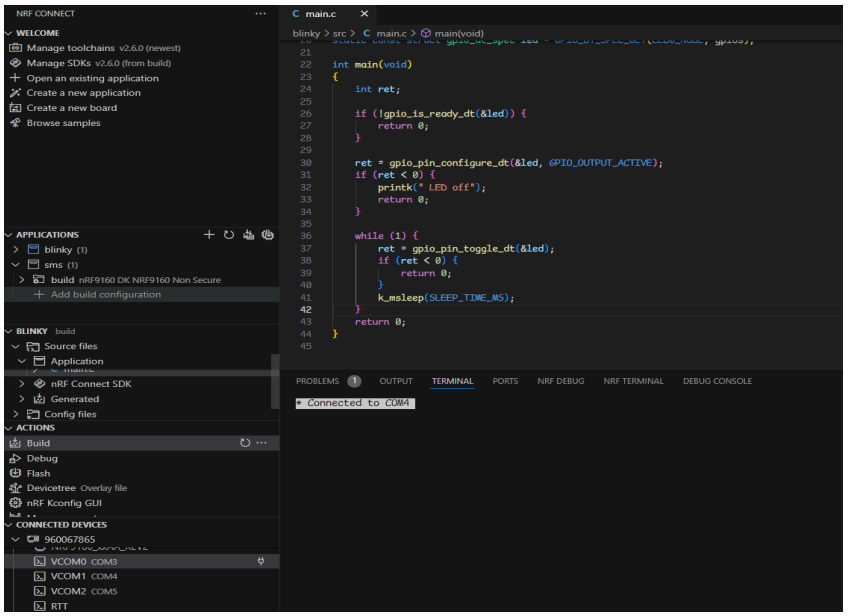

Figure 4. The Nordic Semiconductors SDK integrated in the Visual Studio Code environment.

To build more complicated things, the Cmake, prj.conf, and the Kconfig files have to be altered. The header files should not be altered, and instead use the prj.conf and Kconfig files to change the values. A general problem with the integration of the toolchain is that it does not remove or take over VS code's build, run, debug settings and buttons. This means that building it the usual way in VS Code is it is building for vanilla VS code and not the toolchain. There are 3 VCOM ports which can be assigned to VS Code, or, for example, to the cellular monitor in nRF Connect (Figure 3).

Due to NB-IoT not being fully designed with SMS in mind, some LTE-M options may have to be used at certain points. Nordic has an AT command called %XSYSTEMMODE which gives the option to prioritise NB-IoT, LTE-M, or GNSS. If the device has to be swapped from NB-IoT to LTE-M then the device has to be disconnected or put into flight mode and then connected using the other protocol. There are few benefits to swapping between NB-IoT and LTE-M because the standards are different, and swapping may not proceed smoothly. Fortunately, in this project, it is not required to swap from between the standards..

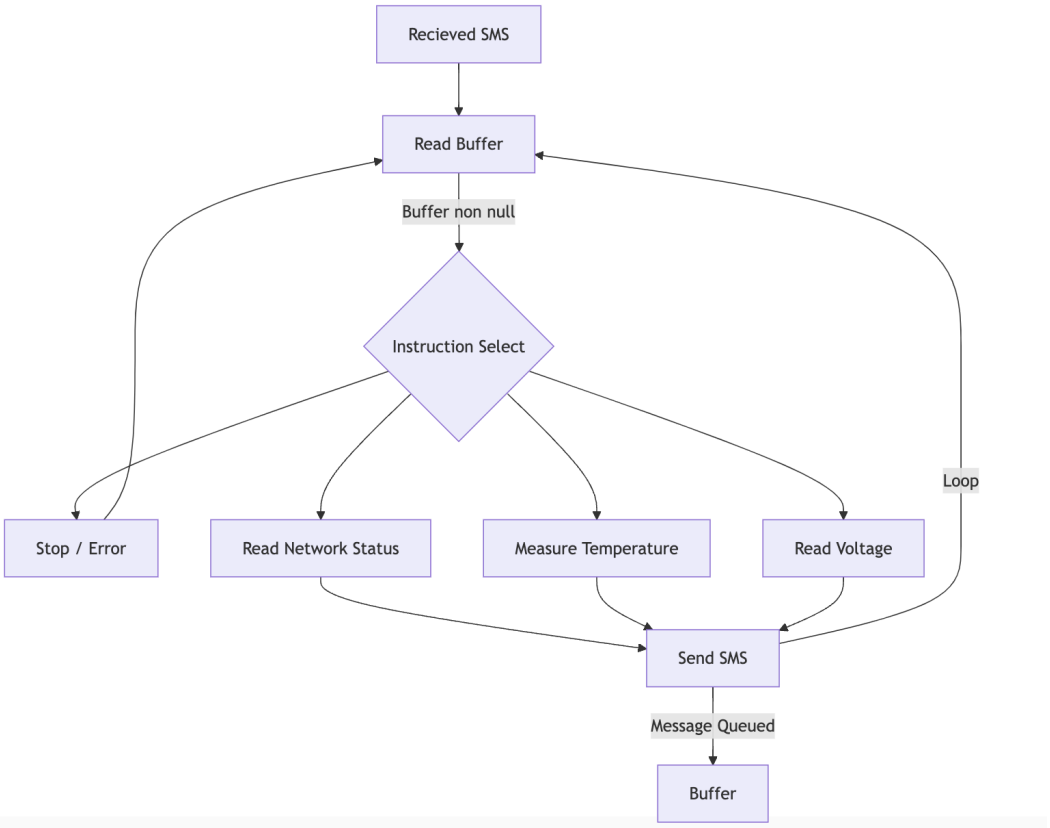

Figure 5. Code flow

The general flow of the code is shown in Figure 5. An infinite loop, also known as a super loop, without cleaning the buffer at the beginning or end of the code, is used, enabling the sending of multiple instances of SMS messages. The SMS messages are purely used for receiving sensor data, and having only a single instance of sensor data is insufficient to conclude what the behaviour of the device is. The problem with this workflow of the code is that since it is an infinite loop, it will continuously send SMS messages, unless told otherwise. A single SMS message costs around 1 Euro cent, which would add up over time if an SMS message that requests data is sent for every bit of sensor data.

In an application that works in a super loop, a common way of breaking out of the loop is by using an interrupt. An interrupt is a hardware signal that preempts the processor's current task and transfers control to a higher priority service routine. Since an interrupt is a hardware action, it does not understand or listen

to the software, which can cause unwanted behaviour. An interrupt does not know which thread it is interrupting, it instead interrupts the whole CPU. Interrupts are usually kept as short as possible due to their high priority and effect on the whole workflow of the device. Hence, in this project, it is better to keep the code running in the super loop and have a stop function that does not send SMS messages but continues in the infinite loop. In the case of connectivity being bad enough that the 2G SMS connection is also affected, having the device constantly attempt to send a message is better than dropping the messages entirely. Also, the project's client gave instructions not to use interrupts.

Figure 6: Requesting network status via SMS and sending Stop command via the Onomondo web interface.

4.3 Code Debugging

With the Nrf9160dk an iBasis SIM was provided, which was used first in an attempt to build the basics before paying for the Onomondo SIM. However, the iBasis SIM does not have SMS capabilities, so it can not be used for the project.

As explained in section 2.2, if an SMS is sent to a phone number, it will not send it to that phone number, but will be sent to the Onomondo SMS server, which then sends it to the Onomondo web console. A problem with the Onomondo web

console is that it must be constantly open to see the SMS logs. If the web console has been closed, a "device sent message" prompt for earlier messages can only be seen, as shown in Figure 7.

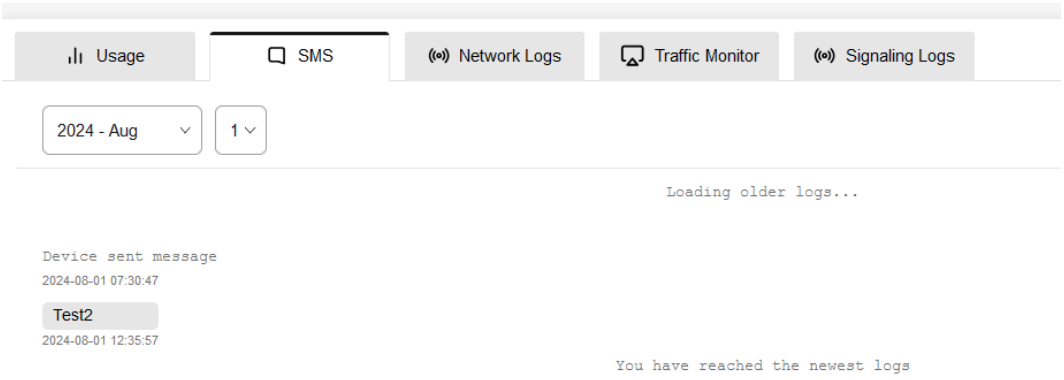

Figure 7. SMS logs in the Onomondo website.

This problem can be solved with a webhook. A webhook is a custom HTTP callback, which is usually triggered by an event. When this event occurs, the original site sends an HTTP request to the URL configured for the webhook. In the case of this project, when the SMS data is received by the Onomondo website it will automatically trigger an event flag, and the data will be automatically sent to the webhook destination.

5 DISCUSSION

Some challenges were encountered during the development process, which are discussed below. Also, some ideas for the next steps in developing the solution are given.

5.1 Trade-offs in System Design

The SMS-based prototype uses a “super loop” approach, an infinite loop that continuously processes incoming SMS messages and sends out responses. This approach simplifies the control flow and avoids reliance on interrupts, which are not used due to hardware constraints and reliability concerns.

One trade-off is the cost. Each SMS message has a small monetary cost (about €0.01), and repeated status checks or sensor queries can accumulate over time. To reduce cost, the device sends only the start and end requests, which makes the program loop on an instruction that sends SMS messages periodically until a new SMS is received. Another trade-off is between responsiveness and power use: while continuous polling ensures timely responses, it increases energy consumption compared to event-driven or interrupt-based systems.

Interrupts were explicitly avoided in this implementation due to their tendency to disrupt ongoing processes. In embedded systems, improperly managed interrupts can lead to undefined states, especially when dealing with modem-level operations.

5.2 Challenges with PIN-less SIM Cards

Standard consumer SIM cards for mobile phone use all have PIN codes given by the service provider. Typically, these PIN codes are initially set to a default PIN code of 0000 or 1234. To change the PIN code through AT commands, the following command is used:

AT+CLCK="SC",1,"1234"

This is only possible once the SIM card is unlocked for obvious security reasons. However, with PINless SIM cards, the SIM card is always unlocked. In fact, all SIM cards have PIN codes and PUK codes, even if they are PINless. The main manner of communication with a SIM via telecommunications is by using AT commands. The difference between a PINless SIM card and a normal SIM card is that the AT commands that have to do with PIN codes end up giving errors. An error in this case is not considered a failed attempt to unlock the SIM through a PIN code. This is due to PINless SIM cards not having the AT commands related to PIN codes

in their instruction set. The purpose of PINless SIM cards is that they allow a programmer to interact with them without handling the security requirements that PINs provide. However, this could be used nefariously by taking a PINless SIM card and sending it unwanted AT commands. This problem is easily stopped by not allowing the SIM card to receive or accept AT commands other than the specific ones that are required.

When attempting to use a PINless SIM card through certain versions of the Android operating system, some strange behaviour can happen, one of which was found was that the Android device, which was using a kernel version 3.18.14, asks for the PIN even though it is considered PINless. Even if a blank PIN code is inserted, the code will still be considered an incorrect PIN code. The PIN or PUK that the manufacturer puts on the specific PINless SIM card has to be used, however, most manufacturers do not document to the consumer what these codes are. Most SIM cards have a limit of three attempts until the SIM card becomes locked and a PUK is required. Of course, the number of allowed attempts can be changed so this error does not occur, however, it is not something that is thought about in regards to a PINless SIM card. Therefore, the Android operating system's communication with the PINless SIM card causes some security faults. This was found to be the case with older Android operating systems, mainly ones with an "ok" button in the PIN code entry section. With newer versions of the operating system, this has been fixed. In this iteration of entering PIN codes seems to only check if the sent code is similar to the PIN, however, it does not check if the sent code is in acceptable form, for example, if it is 4 digits long or not. As can be seen in Figure 8, the phone even gives options for inserting alphabetical characters in the PIN code.

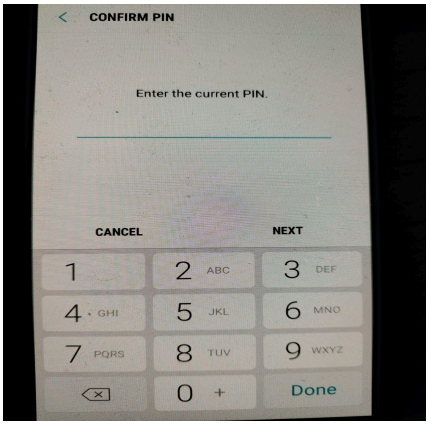

Figure 8. The old Android operating system PIN section that causes the error.

5.3 Next steps in the development

Future developments for this project would be making custom AT commands so that the SMS connection would be utilised by the program to do something other than read and send sensor data. The system could also include features like automatic event alerts using webhooks, with custom server logging and better error handling to make it more reliable. Adding support for other connection options like RTT(Real Time Transfer) or LTE-M could make the system more adaptable, since some regions of the world do not have NB-IoT infrastructure. As this project was designed with one device, there could be a difference in implementation when implementing it to multiple devices at the same time.

5.4 Limitations

This prototype relies on the SMS protocol, which introduces a few constraints. Firstly, SMS messages are limited to 160 characters. Although message concatenation is possible, this adds overhead and complexity. Secondly, the system requires stable cellular coverage, which may not be available in remote locations. Thirdly, the SMS-based implementation does not support real-time or bidirectional communication. Messages are queued and delivered without a guarantee of precise timing or receiving acknowledgement. These limitations

make SMS a fallback solution rather than a full replacement for active debugging connections.

6 CONCLUSION

This thesis created a software prototype for NB-IoT device debugging using the SMS protocol. The key achievement was the creation of a fallback communication method that remains functional even when standard NB-IoT data connectivity is unavailable. By using SMS, which is a widely available protocol, the solution prototype provides a lightweight and accessible mechanism to request device status, configuration, or trigger specific actions remotely.

The prototype was successfully implemented using the Nordic Semiconductor nRF9160-DK board, Onomondo M2M SIM cards, and Zephyr RTOS. Using AT commands, it was possible to connect to the modem and perform debugging tasks in real time.

This thesis highlights a practical approach to increasing the resilience of IoT systems, especially in cases where remote access is critical and the infrastructure is not reliable. While there are clear limitations, for example, message length, latency, and real-time communication, the solution offers a valuable backup option for system administrators and developers. As IoT systems become more widespread, maintaining communication connections, even under degraded conditions, will be increasingly important.

7 REFERENCES

3GPP TS 36.300 V17.5.0 (2024-03). Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description; Stage 2. Retrieved December 25th, 2024, from https://www.3gpp.org/DynaReport/36300.htm

3GPP TS 27.005 V17.0.0 (2023-12). Use of Data Terminal Equipment - Terminal Adaptation Functions for Short Message Service (SMS). Retrieved December 25th, 2024, from https://www.3gpp.org/DynaReport/27005.htm

Dev, B., Sunny, S. (2024). AT tutorial. Retrieved December 25, 2024, from https://www.cavliwireless.com/blog/nerdiest-of-things/an-introduction-to-cellular-at-commands.html.

Ground Control (5th June 2024). Satellite-Enabled LPWAN: What Network, When, and Why. Retrieved December 25th, 2024, from https://www.groundcontrol.com/blog/satellite-enabled-lpwan-what-network-when-why/.

Hemus, B. (18th November 2024). SIM vs eSIM vs iSIM - Everything you need to know. Retrieved December 25th, 2024, from https://www.emnify.com/iot-glossary/sim-vs-esim-vs-isim.

Nordic Semiconductor (n.d.). nRF9160-DK Development Kit. Retrieved December 25, 2024, from https://www.nordicsemi.com/Products/Development-hardware/nRF9160-DK.

Ratasuk, R., Mangalvedhe, N., Zhang, Y., Robert, M., & Koskinen, J. P. (2016, October). Overview of narrowband IoT in LTE Rel-13. In 2016 IEEE conference on standards for communications and networking (CSCN) (pp. 1-7). IEEE.

University of Helsinki (n.d). SIM Cards and Roaming. Core 5G and beyond (Online course). Retrieved December 25th, 2024, from

https://courses.mooc.fi/org/uh-cs/courses/5g-mooc/chapter-2/sim-cards-and-roaming.