Aalto University
School of Science
Degree Programme in Computer, Communication and Information Sciences

Veikko Oittinen

Enabling automatic configuration of cellular data for constrained IoT devices

Master's Thesis
Espoo, April 20, 2018

Supervisor: Assistant Professor Mario Di Francesco
Advisor: Ari Keränen M.Sc. (Tech.)

Aalto University
School of Science
Degree Programme in Computer, Communication and Information Sciences

ABSTRACT OF MASTER'S THESIS

Author:	Veikko Oittinen		
Title:	Enabling automatic configuration of cellular data for constrained IoT devices		
Date:	April 20, 2018	**Pages:**	62
Major:	Computer Science	**Code:**	T-110
Supervisor:	Assistant Professor Mario Di Francesco		
Advisor:	Ari Keränen M.Sc. (Tech.)		

Cellular networks have existed for almost forty years. During the course of their history, they have transformed from wireless voice communication providers to wireless network providers. Nowadays mobile broadband data forms the bulk of the cellular data transfer which was a staggering 14 exabytes per month in year 2017, or 2.9 gigabytes per smartphone per month. The Internet of Things is changing this connectivity landscape by introducing devices in the millions but with scarce individual resources and data usage.

However, there are some challenges related to cellular data connections in constrained IoT devices. This thesis identifies those challenges and proposes solutions to overcome them for enabling simpler cellular data connectivity. We first present the technical challenges and solutions found in today's cellular IoT devices.

We then present a proof of concept prototype that realizes automatic cellular connectivity in a very constrained IoT device. The prototype is capable of connecting to a management system and reporting sensor readings without requiring any user interaction. Besides recognizing important improvements in the next generation of cellular IoT technology, the thesis concludes with suggestions on how to improve the usability of programming interfaces for cellular connectivity.

Keywords:	Internet of Things, IoT, cellular networks, cellular data, constrained devices, automatic configuration
Language:	English

Aalto-yliopisto
Perustieteiden korkeakoulu

Aalto-yliopisto
Perustieteiden korkeakoulu — DIPLOMITYÖN
Tietotekniikan koulutusohjelma — TIIVISTELMÄ

Tekijä: Veikko Oittinen

Työn nimi:
IoT-laitteiden datayhteyden automaattinen määrittely matkapuhelinverkoissa

Päiväys:	20. huhtikuuta 2018	**Sivumäärä:**	62
Pääaine:	Tietotekniikka	**Koodi:**	T-110
Valvoja:	Apulaisprofessori Mario Di Francesco		
Ohjaaja:	Diplomi-insinööri Ari Keränen		

Lähes neljäkymmenvuotisen historiansa aikana matkapuhelinverkot ovat muuttuneet puheen välittäjistä langattomaksi dataverkoksi. Nykyään langaton laajakaista muodostaa suuren osan matkapuhelinverkoissa siirretystä datasta, jota oli 14 exatavua kuukaudessa vuonna 2017. Esineiden Internet tuo verkkoon miljoonia laitteita joiden yksittäinen datansiirron tarve on vähäinen.

Matkapuhelinverkon datayhteyden käyttö ei kuitenkaan ole ongelmatonta rajoittuneissa Esineiden Internetin laitteissa. Tämä diplomityö tunnistaa ja luokittelee näitä teknisiä haasteita ja ehdottaa ratkaisuja niihin.

Esittelemme prototyypin joka toteuttaa automaatisen matkapuhelinverkon datayhteyden luonnin rajoittuneessa laitteessa. Prototyyppi ottaa yhteyden hallintajärjestelmään ja raportoi mittausdataa ilman käyttäjältä vaadittavia toimia. Johtopäätöksenä tämä diplomityö esittää parannuksia tehtäväksi matkapuhelinverkkojen datayhteyksien ohjelmointirajapintoihin niitä käyttävissä laitteissa. Löysimme myös tärkeitä parannuksia joita on jo tehty tulevan sukupolven matkapuhelinverkon määrittelyssä.

Asiasanat: Esineiden Internet, IoT, matkapuhelinverkot

Kieli: Englanti

Acknowledgements

I would like to thank my thesis supervisor Professor Mario Di Francesco for all his help on this project. I would also like to thank my instructor Ari Keränen for his efforts. Thanks to Jan Melen and all the other NomadicLabbers at Ericsson for their contribution on and off the topic.

I am also grateful to my family, friends, and especially to my wife for their support and patience during this long process that finally reaches conclusion with this thesis. Lifelong learning indeed...

And last to my children. Though you were mostly in the way of the actual writing process; everything I do, I do it for you.

Espoo, April 20, 2018

Veikko Oittinen

Abbreviations and Acronyms

3GPP	3rd Generation Partnership Project
3GPP2	3rd Generation Partnership Project 2
5G-NR	5G New Radio
6LoWPAN	IPv6 over Low-power Wireless Personal Area Networks
ABP	Activation By Personalization
APN	Access Point Name
ARP	Address Resolution Protocol
BSS	Base Station
CoAP	Constrained Application Protocol
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DTLS	Datagram Transport Layer Security
eNB	E-UTRAN Node B
EPC	Evolved Packet Core
EPS	Evolved Packet System
ETSI	European Telecommunications Standard Institute
E-UTRAN	Evolved Universal Radio Access Network
GGSN	Gateway GPRS Support Node
GPRS	General Packet Radio Service
GSM	Global System for Mobile communications
GTP	GPRS tunneling protocol

HTTP	Hypertext Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
IMSI	International Mobile Subscriber Identity
IMS	IP Multimedia Subsystem
IoT	Internet of Things
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
LTE	Long Term Evolution
LTE-M	Long Term Evolution category M1
LWM2M	Lightweight Machine-to-Machine
MAC	Media Access Control
MCC	Mobile Country Code
MNC	Mobile Network Code
MSIN	Mobile Subscriber Identification Number
MS	Mobile Station
MTC	Machine Type Communications
NAT	Network Address Translation
NB-IoT	Narrow Band IoT
ND	Neighbor Discovery
NMT	Nordic Mobile Telephony
OTAA	Over-the-Air-Activation
PDN	Packet Data Network
PDP	Packet Data Protocol
PGN	Packet Data Network Gateway
PIN	Personal Identification Number
PoC	Proof of Concept
QoS	Quality of Service
RAM	Random Access Memory
REST	Representational State Transfer
SGN	Serving Gateway

SGSN	Serving GPRS Support Node
SIM	Subscriber Identity Module
SMS	Short Message Service
SSID	Service Set Identifier
SSP	Secure Simple Pairing
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
UE	User Equipment
UICC	Universal Integrated Circuit Card
UMTS	Universal Mobile Telecommunications System
URI	Uniform Resource Identifier
USIM	Universal Subscriber Identity Module
WiMAX	Worldwide Interoperability for Microwave Access
WPS	Wi-Fi Protected Setup

Contents

Abbreviations and Acronyms **5**

1 Introduction **11**

1.1 Problem statement 12

1.2 Structure of the Thesis 13

2 Background **14**

2.1 Cellular Networks 14

2.1.1 Network evolution 15

2.1.2 Packet data connection 17

2.1.3 Subscription identification 19

2.2 IoT connectivity options 19

2.2.1 Ethernet 20

2.2.2 Wi-Fi 20

2.2.3 Bluetooth 21

2.2.4 LoRa 21

2.2.5 Other technologies 22

2.2.6 Cellular subscriptions optimized for IoT 23

2.3 Technologies used in the prototype 24

2.3.1 Mbed OS 24
2.3.2 Constrained Application Protocol 25
2.3.3 Datagram Transport Layer Security 25
2.3.4 OMA Lightweight M2M 26
2.3.5 IPSO Smart Objects 26
2.3.6 Bloom filters 26

3 Challenges **28**

3.1 Overview and methodology 28
3.2 Cellular connection 30
3.2.1 Personal Identification Number 30
3.2.2 Access Point Name 31
3.2.3 Subscription activation 33
3.3 Device constraints 33
3.3.1 Memory limitations 34
3.3.2 Input and output limitations 34
3.4 Management 34
3.5 Development 35
3.6 Security 35

4 Implementation **36**

4.1 Hardware 36
4.2 Software 38
4.3 Reference system 40
4.4 Implemented solutions 41
4.4.1 Personal Identification Number 42

4.4.2 APN database with Bloom filters 42
4.4.3 IP stack offloading 44
4.4.4 Memory usage 44
4.4.5 Device management 45
4.4.6 Security 45
4.4.7 Power consumption 46

5 Evaluation **47**

5.1 Measurements 47
5.1.1 Memory usage 48
5.1.2 Power consumption 50
5.1.3 Connection time 52
5.1.4 Cellular usability 53

6 Conclusions **54**

Chapter 1

Introduction

The Internet of Things (IoT) is predicted to become the next big disruptive technology that changes our lives. In the IoT vision everything around us is connected. Our everyday objects will include sensors and report data to services running locally or in the cloud. Our surroundings will be full of sensors monitoring different physical properties and actuators responding to changes in the environment [10].

The growth of this ecosystem requires connectivity for all the devices. Ericsson predicts in its latest Mobility Report that there will be 29 billion connected devices by 2022, 18 billion of them being IoT devices [19]. The smallest sensors will often operate by using short range radio technologies, but even these will need a gateway with Internet connectivity to report data and receive commands from services in the Internet. In this respect, cellular networks provide a robust, long range, and readily available connection globally.

Cellular connections for IoT have several advantages over other connection methods. The cellular network covers 95 % of the population on the planet, so in most cases no new infrastructure needs to be built. Cellular networks are reliable and provide high-speed data connections when needed. They can satisfy a target quality of service to enable critical operations and more efficient resource usage.

Previous research on IoT device connectivity has focused on bootstrapping the devices to a management server, securing communication, claiming device ownership, and managing devices [46–48]. However, there are some challenges specific to using cellular connections, especially in constrained devices with very limited resources. This thesis identifies these challenges and proposes methods for solving them by focusing on solutions that are practical for implementing IoT device software.

1.1 Problem statement

Some challenges arise when using constrained IoT devices with cellular connectivity. Constrained IoT devices have either very limited or no input capabilities on the device. This makes configuring them difficult during installation [46]. Settings needed for cellular connectivity, such as the Personal Identification Number (PIN) code and Access Point Name (APN) settings, currently require the use of some external input method, e.g., user typing a code with a keyboard.

These devices are also limited in the resources available to them. Memory, computing power, and energy consumption need to be considered when designing IoT devices and systems. The standard Internet protocols, such as Transmission Control Protocol / Internet Protocol (TCP/IP) or Hyper Text Transfer Protocol (HTTP) might be too resource-intensive to be implemented in constrained devices. This requires employing other protocols better suited for constrained environments.

Cellular connection also connects the IoT device to the public Internet instead of a local network. The IP address of the device might be behind Network Address Translation (NAT) and assigned randomly from a pool of addresses. This can make connecting to devices and managing them even more difficult.

1.2 Structure of the Thesis

This thesis is structured as follows. Section 2 gives the necessary background information on the technologies involved. Section 3 describes the challenges that were identified in this research and gives suggestions on solutions. Section 4 introduces the prototype implementation that was built to evaluate feasibility of automatic cellular connection establishment for constrained devices. Section 5 presents and evaluates the findings. Finally Section 6 concludes the thesis.

Chapter 2

Background

This section describes the cellular network as well as IoT device connectivity and provides background information on the technologies used in implementing the proof of concept prototype. It first discusses the evolution and structure of cellular networks. It then explains how the network communicates with the device and how data connections are handled. Next follows review of IoT connectivity technologies. Technologies used in the prototype conclude this chapter.

2.1 Cellular Networks

Cellular networks provide wireless connectivity globally. Large part of inhabited areas in the world have cellular connectivity [19]. This makes cellular networks a compelling choice for device connection. Although cellular networks were originally designed for transmitting speech, they nowadays deliver mostly data. However, some preliminary configuration is still needed for a cellular connection to route user data packets to an external network.

2.1.1 Network evolution

There are several different organizations standardizing cellular network technology. These are the Third Generation Partnership Project (3GPP), the Third Generation Partnership Project 2 (3GPP2), and the Institute of Electrical and Electronics Engineers (IEEE). 3GPP was formed to develop standards for third generation mobile networks based on GSM. 3GPP2, with member organizations from United States and Asia, develops standards based on the cdma2000 specification. IEEE develops WiMAX technology for cellular networks [15]. This thesis mainly focuses on 3GPP networks, presented in Figure 2.1.

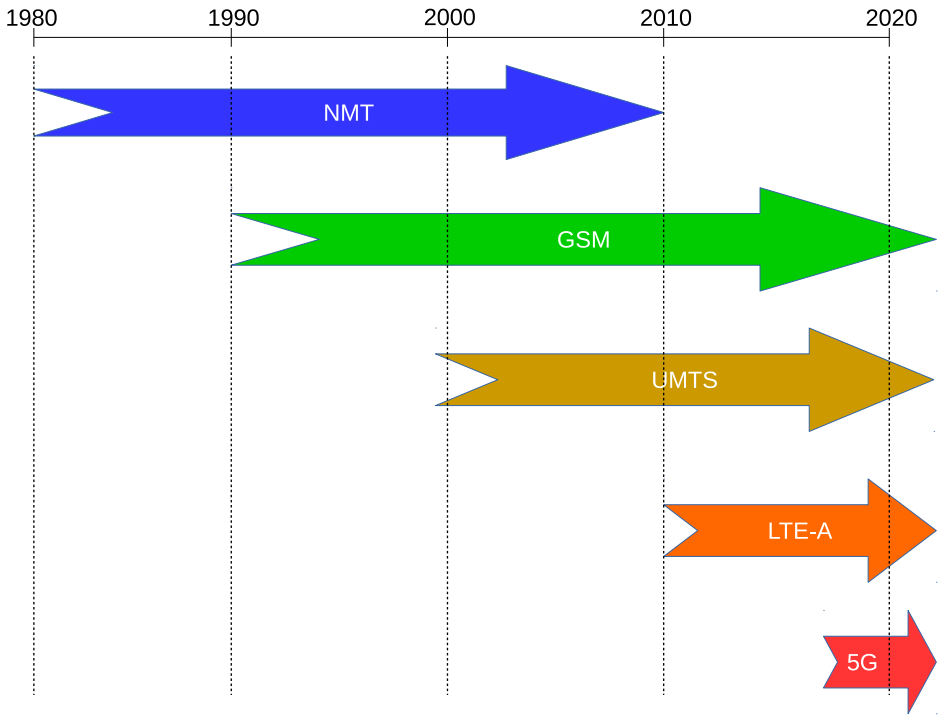

Figure 2.1: Almost 40 years of mobile networks

The first international cellular network was the Nordic Mobile Telephony (NMT) network, developed in the 1980s. It was an analog radio network transmitting speech. The NMT network operated in the 450 MHz frequency band. It was recently discontinued and the frequency was reused for mobile broadband. At the same time with NMT analog cellular telephone networks were also developed in the United States and in Japan.

The second generation (2G) network is the Global System for Mobile communication (GSM). The GSM network is a digital network still in use today. The GSM standards were developed by the European Telecommunications Standard Institute (ETSI). The digital network allowed for more services than the analog. One of the first data services was the Short Message Service (SMS). It became very popular and is still in use. The GSM network also enabled circuit-switched data connections. The first connections operated at a mere 9.6 Kbps speed but proved the demand for data services in mobile devices. In parallel with ETSI development the United States and Japan developed their own digital cellular network solutions.

In the middle of the 1990s General Packet Radio Service (GPRS) was introduced to GSM networks [2]. GPRS enabled packet switched communication in GSM networks. The data connection was best effort, so the quality of the connection varied.

The third generation of mobile network (3G) was developed by an international organization, the Third Generation Partnership Project (3GPP). It has member organizations from Europe, United States, China, Korea and Japan. The 3G network has two domains, the circuit switched and the packet switched. The circuit-switched domain is for the delivery of voice and video services while the packet switched domain is for delivering IP data.

3G networks included new radio access technologies, known as the Universal Mobile Telecommunications System (UMTS). 3G networks also introduced a new system for packet switched network, namely the IP Multimedia Subsystem (IMS). It offered services beyond the capabilities of GPRS. One of the key components is the Quality of Service (QoS), which defines several factors – including maximum bandwidth and maximum delay for a cellular network packet data session [14].

3GPP cellular networks of the fourth generation (4G) are already deployed around the world. The 4G network introduced new radio access schemes and a core network called the Evolved Packet System (EPS). This is a purely IP based system with the Long Term Evolution (LTE) or Evolved Universal Radio Access Network (E-UTRAN) radio access and the Evolved Packet Core (EPC) core network. The 4G is a fully packet switched network. It provides mobile connection speeds comparable to fixed Internet connections. It also provides special features for Machine Type Communications (MTC). These include Long Term Evolution category M1 (LTE-M) [5] and Narrow Band IoT (NB-IoT) [6]. LTE-M and NB-IoT were designed to allow lower modem prices and longer battery lifetime by simplifying the connection and introducing new sleep modes.

Mobile cellular networks of the fifth generation (5G) are being standardized and tested at the time of writing. New radio access schemes and new frequencies are included in the 5G New Radio (5G-NR) specification [7]. First 5G networks should be in commercial use by 2019.

2.1.2 Packet data connection

The first data connections for mobile devices were available in the GSM networks over the circuit switched connection. Packet switched data connections started in mobile networks with the introduction of General Packet Radio Service (GPRS) to GSM networks.

When a Mobile Station (MS) joins the GSM network it connects through its radio network to a base station (BSS). The base station establishes a logical tunnel to a Serving GPRS Support Node (SGSN) using the GPRS tunneling protocol (GTP). This procedure is called a GPRS attach. To allow fast data connections a packet switched connection is needed. The MS requests this connection from the network by sending a Packet Data Protocol (PDP) activation request. This request includes the Access Point Name (APN) the MS wants to connect to. This name identifies an external packet data network, for example the Internet. The SGSN then finds the Gateway GPRS Support Node (GGSN) that handles the data packet connection to the network in question. The SGSN establishes a logical data connection to the GGSN and routes data packets from the MS there [2]. This procedure is same for GSM and UMTS networks.

In LTE networks with the Evolved Packet Core (EPC) the procedure is similar but involves different entities. The User Equipment (UE) connects to the EPC network through the E-UTRAN Node B (eNB). In LTE all the traffic, including phone calls, is packet data. The UE still needs to establish the PDP context to communicate with other packet data networks (PDN) outside EPC. To this end the UE sends the PDP activation request and logical tunnels are formed between eNB and serving gateway (SGN), and between SGN and packet data network gateway (PGN). These tunnels are used to route IP traffic to and from the external network.

2.1.3 Subscription identification

International Mobile Subscriber Identity (IMSI) is a globally unique identifier assigned to a mobile subscription [1]. IMSI consists of only digits from 0 to 9, has maximum length of 15 digits, and it comprises three parts: Mobile Country Code (MCC), Mobile Network Code (MNC), and Mobile Subscriber Identification Number (MSIN). The MCC is three digits long and identifies the country of the subscription. The MNC is two or three digits long depending on the country. The rest of the IMSI is the MSIN that identifies the subscription inside the operators network as shown in Figure 2.2.

Figure 2.2: IMSI

2.2 IoT connectivity options

Constrained IoT devices have many different connection options available. The best option depends on the particular use case. Some options are physically incompatible with the devices small footprint while others suffer from high energy consumption or do not meet application-specific requirements such as bandwidth. This section presents some of the most common communication interfaces found in constrained devices nowadays.

2.2.1 Ethernet

Ethernet networking was developed already in the 1970s [31]. It was originally based on collision detection on a shared media, but has since evolved. Ethernet can be used over different physical media but most common in IoT devices is the twisted pair wiring. The latest standard from 2017 defines 400 Gb/s speeds for Ethernet [25].

The Ethernet uses Media Access Control (MAC) addresses to deliver packets. Address Resolution Protocol (ARP) connects MAC addresses to IPv4 addresses [42]. IPv6 uses Neighbor Discovery (ND) for the same purpose [35]. IP settings, such as IP address, Domain Name System (DNS) server and routing information can be automatically configured with Dynamic Host Configuration Protocol (DHCP) [18] or with router advertisement in ND. These protocols enable automatic IP connectivity in Ethernet.

2.2.2 Wi-Fi

Wi-Fi is a wireless network protocol suite maintained by the Wi-Fi Alliance. It is based on the IEEE 802.11 family of standards [23]. Wi-Fi operates on several different physical channels. These are shared by different service sets with named identifiers (SSID). Devices can join open Wi-Fi networks automatically by scanning the channels for networks and joining the available open network. Secure Wi-Fi network settings for a device, such as channel and security settings, can be automatically configured using Wi-Fi Protected Setup (WPS) [55]. There are several methods for configuring WPS: in-band configuration, out-of-band configuration, and push button configuration [50]. After association with the Wi-Fi network, IP settings can be automatically retrieved with the same protocols as in Ethernet.

2.2.3 Bluetooth

Bluetooth is a radio communication protocol maintained by the Bluetooth Special Interest Group. It was originally designed for connecting peripherals to mobile phones, but later standards define general networking over Bluetooth [21, 36]. Bluetooth devices need to pair with each other before exchanging IP packets. This can be achieved with the Bluetooth Secure Simple Pairing (SSP). SSP is a standard that defines secure device pairing using elliptic curve Diffie-Hellman public keys. SSP describes several options for pairing: numeric comparison, passkey entry, 'Just Works', and out-of-band. The most commonly used and familiar to users is the passkey entry which is used in mobile phones. Most convenient for constrained devices is the 'Just works' model which is targeted for devices with no display nor keypad. It uses unauthenticated Diffie-Hellman key agreement which leaves it vulnerable to man-in-the-middle attacks. [50].

2.2.4 LoRa

LoRa is a long range low bitrate wireless communication protocol [30]. LoRa devices connect through a radio interface to LoRa gateways by using a proprietary communication protocol. The gateways use the IP protocol to relay data to the network server, which is the main backend component in the LoRa network. It removes duplicate packets received from several gateways, selects a gateway for possible reply, and forwards data to the application server.

LoRa end devices have a unique device identifier in the network and two encryption keys, one for network traffic encryption and one for application data encryption. There are two ways for a device to connect to LoRa network, either by Over-the-Air-Activation (OTAA) or by Activation By Personalization (ABP). With Over-the-Air-Activation handshake messages are sent between the device and the application server. The application server authorizes the device and responds with Join Accept; the end device then derives encryption keys from this message. With Activation By Personalization the device identifier and encryption keys are placed in the device during configuration. This allows the device to join the configured network as it already has the necessary encryption keys, but limits it to a single network [30].

2.2.5 Other technologies

IEEE 802.15.4 is a standard describing low data rate, low complexity short-range radio frequency networks [24]. The standard describes physical layer and media access layer. Several technologies have been developed based on the IEEE 802.15.4. These technologies describe the higher layers of network stack and include 6LoWPAN, Zigbee, and Thread.

IPv6 over Low-power Wireless Personal Area Networks (6LoWPAN) describes the transmission of IPv6 packets over IEEE 802.15.4 networks [34]. This standard can be used to create low power wireless IPv6 networks. 6LoWPAN is supported by many IoT operating systems.

Zigbee is a wireless protocol stack based on the IEEE 802.15.4 standard [56]. It has short range and low cost radio hardware. The low power limits the range to tens of meters, but multi-hop communication allows nodes to forward traffic from other nodes, enabling large networks to be built.

Thread is a network protocol designed for IoT device networks [51]. It is built on the 6LoWPAN standard. It offers low power, low data rate secure communications over mesh networks.

2.2.6 Cellular subscriptions optimized for IoT

Current cellular IoT solutions include systems that can operate globally with reasonable data connection prices. This is done by the network operator that provides the subscription which has roaming agreements with network operators around the world. Although required technology for this functionality has been available for a long time, this is a new business model targeting IoT. One such solution is the Particle Electron shown in Figure 2.3. It has a STM32F205 ARM Cortex M3 microcontroller with 128 KB of RAM and 1 MB of flash. The cellular modem used is a u-blox Sara U270 2G/3G modem. The device connects to the Particle Cloud from where it can be managed and reprogrammed. The data fee for the mobile subscription is $2.99 per month for most countries. This includes 1 MB of data, with additional data priced at $0.99 per megabyte [41].

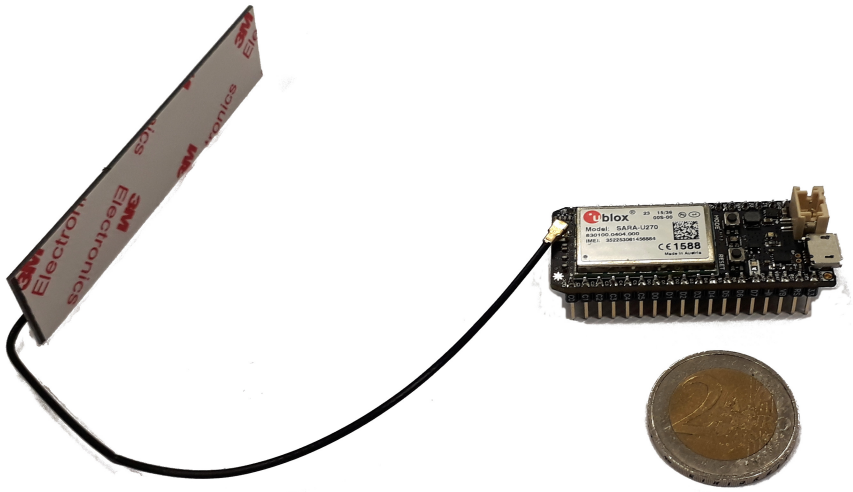

Figure 2.3: The Particle Electron

2.3 Technologies used in the prototype

In this research we implemented a prototype device for experimenting with the automatic cellular connection configuration. This section provides background information about the hardware, software, and methods used in the proof of concept prototype.

2.3.1 Mbed OS

Mbed OS is an IoT operating system from Arm [8]. It supports Arm Cortex-M microcontrollers and currently over one hundred compatible developments boards. C++ programming language is used to write programs for Mbed OS. Mbed OS has a modular library structure that allows only needed features from the operating system to be compiled in to the binary file. Programs can be compiled either with a command line tool supporting several Arm compatible compilers or with the Mbed Online Compiler.

The Mbed OS is an open source operating system with large community support. It is part of the Arm Mbed IoT Device Platform. This platform offers support from device software to cloud and web interfaces. However, the Mbed OS can be used as a standalone operating system for IoT devices without the need to use the Arm platform [9, 43].

2.3.2 Constrained Application Protocol

The constrained application protocol (CoAP) is a web communication protocol designed to be used in constrained IoT devices [49]. CoAP uses REST model similar to HTTP and supports web concepts, such as URIs and Internet media types. CoAP is intended for machine-to-machine applications and it is designed with very low overhead and simplicity.

2.3.3 Datagram Transport Layer Security

Datagram Transport Layer Security (DTLS) is a security protocol for unreliable network connections [44]. It is based on the Transport Layer Security (TLS) protocol [16]. DTLS offers communication confidentiality and integrity. DTLS expands on TLS functionality to allow operation over unreliable UDP connections. These expansions include handshake message reordering and retransmission to handle packet loss during protocol handshake.

2.3.4 OMA Lightweight M2M

Open Mobile Alliance (OMA) Lightweight Machine-to-Machine (LWM2M) is a management protocol for constrained devices [39]. It provides data model to present device capabilities to the server and to manage the device operation. LWM2M was designed to be lightweight and extendable to meet constrained device requirements now and in the future. The protocol enables for example device security profile management, connectivity monitoring, and firmware updates.

2.3.5 IPSO Smart Objects

IPSO Smart Objects provide interoperability in the application level [26]. They provide a common object model to describe resources on a device. IPSO object model is based on the OMA LWM2M model to provide better interoperability. The data model consist of four parts: object representation, data types, operations and content formats. IPSO Alliance provided some basic objects, for instance a temperature sensor and a light object, but users are encouraged to create their own objects.

2.3.6 Bloom filters

Bloom filter is a data structure introduced by Burton H. Bloom in 1970 [11]. Bloom originally suggested them for use in hyphenation. Since then Bloom filters have been used in many applications including peer-to-peer networking, locating resources, packet routing, and data summaries for measurement [13]. Recently Bloom filters have been suggested as a replacement for IP routing in packet networks [28].

Bloom filters can be used to test if an element is member of a group. They achieve good spatial compression and provide answer to the test quickly. Bloom filters are a probabilistic data structure. They tell if an element does not belong to a group with certainty or that it belongs to the group with a certain probability. This false positive rate can be controlled by appropriately setting the key parameters associated with the Bloom filter as detailed later.

An empty Bloom filter is just a bit array initialized to zero. Members are added to the filter by computing hashes from the member. These hashes represent bit positions in the filter. When a member is added to the filter, the corresponding bits are set to one. When examining if an element belongs to the group, same hashes are computed from the element and compared to the filter. If all bits are one then the element belongs to the group with probability one minus false positive rate. If any of the bits is zero then the element is not part of the group.

The false positive rate is controlled by the variables in creating the Bloom filter. These variables are the size of the filter m, the number of elements in the filter n, and the number of computed hashes k. The approximate false positive rate can be computed from the formula $(1 - e^{-kn/m})^k$.

Chapter 3

Challenges in automatic configuration of cellular IoT data

This section introduces a set of challenges with cellular connectivity in constrained IoT devices and possible solutions. It first classifies these challenges based on the context they relate to. These challenges and possible solutions were identified from literature research and by implementing a proof of concept prototype.

3.1 Overview and methodology

Related work on the connectivity of IoT devices is mainly focused on the areas of bootstrapping, management and security, which are all part of the prototype implementation.

Sethi et al. [48] propose a method for secure bootstrapping of ubiquitous displays. Their model enables the secure deployment of the devices without user input on the devices or predefined configuration in the device. The method can be applied to any cloud-connected IoT devices with output capabilities, but requires a trust relationship to be established between the deployment network and the cloud management system.

Security is an important aspect of any information system. IoT devices are limited in their resources and therefore traditional security methods might not be applicable [22]. However, Sethi et al. [45] demonstrated that public-key cryptography can be implemented in 8-bit microcontrollers. The method proposed targets IoT devices without any input capabilities.

Device management is one area of research in the IoT. Constrained devices might lack the resources needed to provide an UI for interaction. Automatic configuration of devices is also required as there can be thousands of devices to be deployed. Sethi et al. [47] propose a web based content management system for pervasive displays.

One of the goals of this research was to identify the challenges of using cellular connectivity with constrained IoT devices. Although there are many aspects to the actual decision about the connectivity of a product, such as cost of manufacturing, deployment, usage, and maintenance, this thesis focuses on the technical challenges.

Building and evaluating a proof-of-concept prototype was chosen as the method for researching the challenges with automatic configuration of cellular data. This approach allows discovering the challenges during the implementation. The evaluation of chosen solutions was done by comparing the prototype to a reference system with more resources implementing the same functionality without cellular connection. The prototype was implemented on a constrained device platform, namely a class-2 device by the definition in RFC 7228: Terminology for Constrained-Node Networks [12].

The identified challenges are divided in to five different groups based on the area in question. These five groups are cellular connection, device constraints, device management, software development, and security. Each section contains description of identified challenges in that area and proposes solutions to address them.

3.2 Cellular connection

Cellular connections were originally designed to be used by humans with equipment that has a display as output and keys for input. The main purpose of such connections was voice communication. This causes problems for constrained devices utilizing the connection for data transmission, as data packet connection to Internet is not available by default. The cellular connection also involves other data packet protocols used to carry the IP traffic through the connection. The configuration of these is also challenging for constrained IoT devices.

3.2.1 Personal Identification Number

The Personal Identification Number (PIN) is a security code protecting the Universal Integrated Circuit Card (UICC) against unauthorized access [3]. The UICC contains the Subscriber Identity Module (SIM) for GSM or Universal Subscriber Identity Module (USIM) for 3G networks [4]. The UICC is usually referred to as a SIM card in either case. Most SIM cards come with some default PIN code, such as 0000 or 1234. The user is supposed to change this code to protect the information stored in the SIM and to prevent lost or stolen cards from being used. Because most constrained IoT devices lack any input capabilities, the insertion of the PIN code is rarely possible on the device at runtime. This makes the cellular data connection and other cellular network services unavailable with an arbitrary SIM card.

PIN code query can be disabled in the SIM card. Setting this as the default would allow constrained devices to access the SIM card without the need for the PIN. As most cards come with some generic default PIN code this would not compromise security. The prototype implements a simple method for trying the most common default PIN code. The implementation is described in Section 4.4.1.

3.2.2 Access Point Name

The Access Point Name (APN) defines the gateway which is connected to the external packet data network (PDN) that the cellular device wants to connect with. The gateway may require user identification in the form of username and/or password. Several connections may be active simultaneously on the device to different networks. The APN settings might be different for the same network operator depending on the subscription type. Some operators allow the APN name to be anything in the PDP activation request and in such case use their public Internet gateway as default. APN name can also be used to define a service profile. For example Finnish mobile operator DNA offers a public IP endpoint for the connection as a separate APN [17]. Operators have their own set of APN names mapping to specific gateways. This variation in APN names makes it difficult for a constrained device to configure the cellular packet data connection.

The operator can suggest APN settings in the PDP activation response that is send to the client after the PDP activation request [2]. The client can select one from several options therein. This would allow the operator to always include a default Internet connectivity gateway in the response, but this option does not seem to be widely implemented in operator networks.

The operator can also automatically send correct settings for the device through an SMS message. The problem is that these settings are usually available only for phones or tablets, not IoT devices. Utilizing this functionality would also require an implementation in the device software, increasing memory usage.

The operator can put the APN data in to the SIM card [4]. The USIM application has a dedicated file for network access information. This allows the device to retrieve the information required for Internet access directly. This approach could lead to problems with expired settings on the SIM. Besides, this option might not be available in the cellular modem.

The APN settings for different operators can be stored in a database on the device. This approach is used by many smart phone operating systems [32], including Windows and Android. However, these databases can be up to several hundreds of kilobytes or even megabytes. Constrained devices often lack the necessary space for such a database. The prototype introduces a compressed database suitable for IoT devices described in Section 4.4.2.

The correct APN settings can be retrieved Out-of-Band, for instance, from another device via a Bluetooth connection. This would require the other device to support sending these settings and a compatible communication channel between the devices.

3.2.3 Subscription activation

Mobile network operators sometimes require a separate activation of a mobile subscription. This can take place in many forms, depending on the operator. Possible options include visiting a web site, scanning a QR code, etc. Prepaid subscriptions might come with no value preloaded and require some other method to add value, e.g. sending a text message with activation code or entering the phone number and the code in a web site. This kind of user input involved in the activation process makes it difficult for constrained IoT devices to use the cellular data connection.

3.3 Device constraints

These are challenges related to the limited resources available in the constrained device. The limited resources on the constrained IoT devices include RAM memory, non-volatile storage, CPU processing power, energy resources, and input- and output interfaces. Other problems are related to connecting these devices with standard Internet protocols as they might have very limited memory available. Below are the challenges identified in this thesis.

3.3.1 Memory limitations

Constrained devices have limited amount of both volatile and non-volatile memory available for software. Manufacturing costs and energy consumption limit these to few kilobytes in the most constrained class 0 devices [33]. This limits the viable software solutions and requires careful planning and implementation. Specialized operating systems, network protocols, and applications software allow functionality for these devices. The prototype implements many techniques for reduced memory usage in both flash and RAM. The solutions for memory reduction are described in detail in Section 4.4.4.

3.3.2 Input and output limitations

Constrained devices often lack proper input- and output interfaces for user interaction. Output may be limited to few LEDs and the input interface might not exist. This requires pre-programmed configuration or automatic configuration settings. The prototype provides fully-automated connection establishment to a management server.

3.4 Management

Once the IoT device is able to connect to cellular packet data network it still might need configuration settings and ways to be discoverable. The device might be behind NAT and it possibly gets different IP address every time it connects. The device might also be located in a hard to reach area making configuration changes difficult on site. There might be hundreds or thousands of devices deployed which makes configuring them manually impractical. The proof of concept prototype connects to the Ericsson IoT Accelerator platform [20] by using the Lightweight Machine-to-Machine (LwM2M) management protocol.

3.5 Development

Cellular connectivity poses some challenges for software development. The software APIs for cellular networking might not be as mature as for other network connectivity options. The cellular connection is much harder to debug than, for instance, Ethernet connection where data packets can be easily captured. Cellular connections also connect to the Internet, so all services used by the device in development and testing phase have to be available in a public IP addresses. This could pose a problem for developers using connections behind a Network Address Translation (NAT) or a firewall.

3.6 Security

IoT device security is an important topic that can have serious impact on many related applications and services. The lack of resources on constrained devices also contribute to exacerbate security issues. Encryption algorithms may require significant computational capacity, memory, and network overhead. One problem with encryption in constrained devices is the lack of suitably unpredictable entropy sources for random number generation. This problem can be solved with a hardware random number generator.

Chapter 4

Implementation

The research was carried out by building and analyzing a Proof of Concept prototype. The challenges identified during the implementation were resolved and the results compared to a reference system. This chapter describes the features of the prototype and the implementation.

4.1 Hardware

The hardware used is a u-blox C027-U20-0-03 development board shown in Figure 4.1. It was chosen as a representative of a common class-2 device. It has the required components for cellular connectivity available on board. It is also supported by the Mbed OS that was chosen as the operating system for the prototype. Figure 4.2 describes the components in the C027 board.

The u-blox C027 board is equipped with a GNSS receiver MAX-M8 for global positioning and a LISA-U200 cellular modem for connectivity. It also has Ethernet and CAN interfaces. The board connects to peripherals using GPIO, SPI, I^2C, UART, and I^2S.

The main application processor on the u-blox C027 board is an LPC1768 by NXP Semiconductors. It is a 32-bit ARM Cortex-M3 microcontroller with 64 KB of RAM divided into one 32 KB and two 16 KB blocks. It has 512 KB of flash storage for non-volatile memory. The operating frequency of the processor is 96 MHz.

The C027 development board provides an USB connector that can be used to power the board, program the flash on the device, connect to serial terminal, and use the CMSIS-DAP debug interface, which is an open-source debug firmware from ARM. These functions are provided by a separate microcontroller ARM-Cortex M0 LPC11U35 MCU that handles the connections for the board through the USB interface [52].

The cellular modem on the C027-U20-0-03 board is the LISA-U200-03S-00 from u-blox. It is a 2G/3G modem that provides GSM/EDGE and HSPA+ data connections. It operates in frequency bands of 800 MHz, 850 MHz, 900 MHz, 1700 MHz, 1900 MHz and 2100 MHz [53].

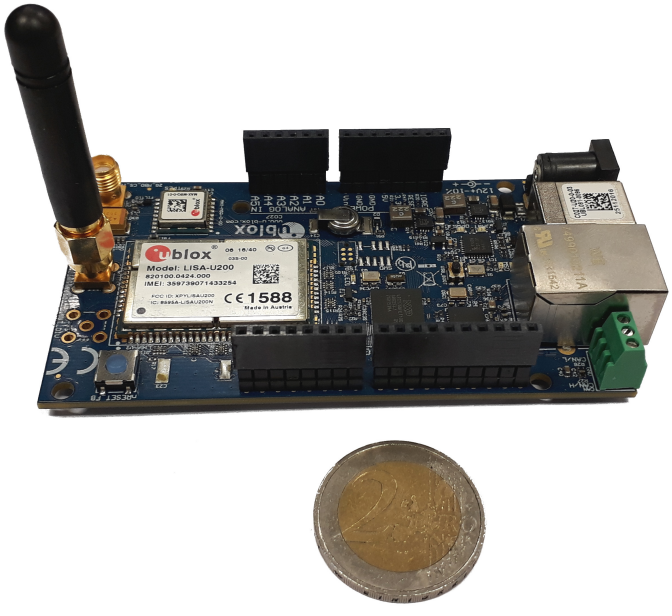

Figure 4.1: The u-blox C027

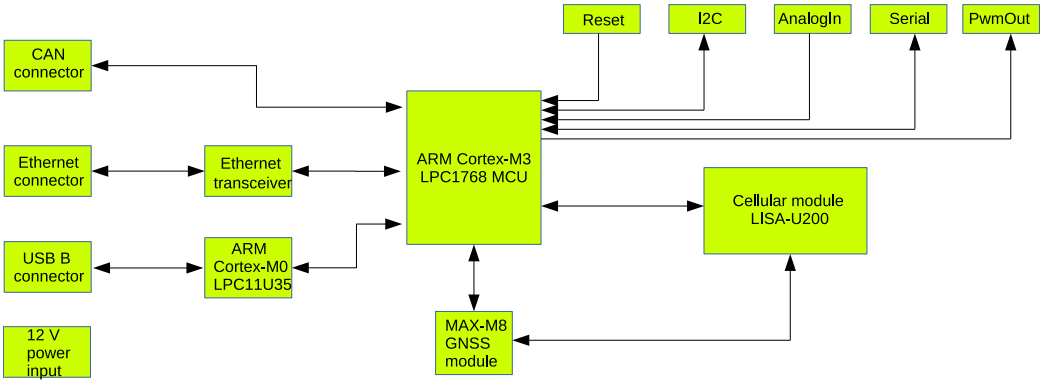

Figure 4.2: C027 block diagram

4.2 Software

The prototype uses the ARM Mbed OS operating system. Mbed OS was chosen for its suitability for constrained devices and support for cellular modems. The Mbed OS is designed for the ARM Cortex-M microcontrollers. It has a modular design that allows small memory footprint by including only needed parts of the code to be compiled to the binary. The mbed client software was used as a LwM2M client for enabling remote device control and monitoring. The client is written in C with some wrappers for C++ that the Mbed OS uses.

The Mbed OS binary code can be compiled in several different ways. The Mbed Command Line Interface (CLI) supports Arm Compiler 5, GNU Arm Embedded Toolchain (GCC) and IAR compiler. ARM also offers an online compiler for developers that uses the Arm Compiler. GCC was chosen for the prototype development as it offers several configuration options and is open source.

The binary code from the compilation differs depending on which compiler was used. There are differences in memory consumption, flash usage and performance of the code. These differences arise from varied reasons, the main ones being the C standard library used and compiler functionality.

The LISA-U200 cellular modem provides the cellular network radio connectivity and cellular specific packet access. The PoC also uses the embedded firmware stack in the modem that provides IP and UDP stacks. This helps reduce the memory footprint in the software and offloads network packet handling of these protocols from the application processor to the modem hardware.

The communication is secured by DTLS encryption. This protocol enables end-to-end security over UDP. The prototype uses the mbed-TLS implementation.

On top of the UDP/IP the prototype uses the Constrained Application Protocol (CoAP). This is a communication protocol designed for constrained devices. It is designed to minimize the communication overhead and provide functionality needed in IoT device communication.

The application level protocol that the PoC uses is Light Weight Machine to Machine (LwM2M) protocol. This protocol allows remote device management and configuration. The prototype also uses IPSO smart objects to expose resources on the device. Figure 4.3 presents the network stack on the device.

The prototype uses the LwM2M protocol to connect to Ericsson IoT Accelerator [20]. IoT Accelerator is an IoT platform that enables centralized control and supervision over users devices. It supports device bootstrapping, device firmware updates, configurable device templates, resource monitoring, alerts, and many other features.

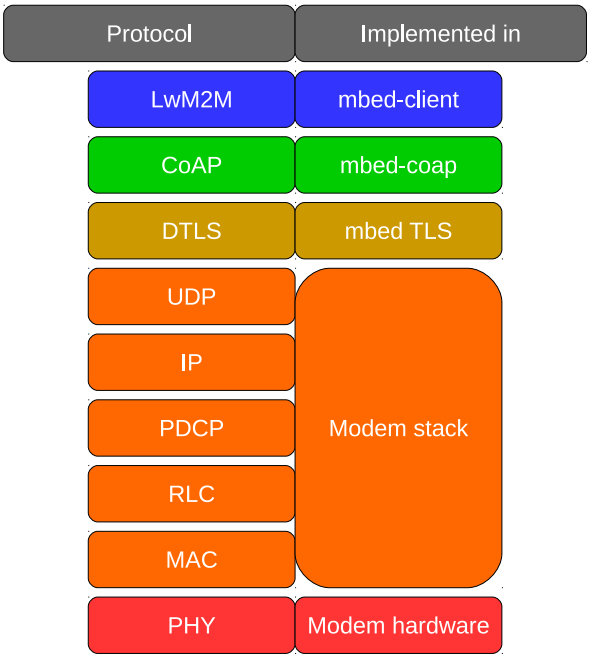

Figure 4.3: Prototype network stack

4.3 Reference system

The results from the research prototype are compared to a reference system running the same software components as the prototype but without any modifications. The board used for the reference system is FRDM-K64F from NXP Semiconductors. The K64F is capable of running the unmodified mbed-client software. The K64F has an ARM Cortex M4 microcontroller that runs at 120 MHz. It has 256 KB RAM and 1 MB of flash storage. Table 4.1 compares the features of the systems.

Feature	**K64F**	**C027**
MCU	Cortex-M4	Cortex-M3
Clock speed	120 MHz	96 MHz
RAM	256 KB	64 KB
Flash	1 MB	512 KB

Table 4.1: System comparison

The reference system is more powerful than the prototype in order to run the unmodified operating system and software installation. The Mbed OS with the mbed client for LWM2M are not officially supported on the C027 board used in the prototype. The reference system also implements Ethernet connection to provide point of comparison for the connection technology.

4.4 Implemented solutions

The prototype implements solutions to several challenges identified either from literature research or encountered during the prototyping. These solutions include setting PIN number and APN settings, reducing the code size for running the code on the constrained device, connecting the device to a management system and securing the data connection.

4.4.1 Personal Identification Number

The prototype has a SIM card that is protected by a PIN number. At first the prototype software tried to open the cellular data connection. Because the PIN number was not set, the SIM card locked. The PUK code was required to unlock the SIM card. The proof of concept implements a simple strategy for entering the PIN code. It tries 0000 and 1234 as these are default PIN codes for SIM cards of many operators. The code in the prototype also checks for remaining attempts on the PIN. The PIN code can be tried only three times. After three failed attempts the SIM card is locked and can only be opened with the Personal Unlocking Key (PUK) code. The implemented code on the prototype only allows PIN insertion if there is more than one attempt remaining. This way the SIM card will not get locked by repeated attempts with a wrong PIN number.

4.4.2 APN database with Bloom filters

The prototype implements an APN database by using Bloom filters. Bloom filters allow high compression rate and efficient search of APN settings in the prototype. The Bloom filter used is a 512 bit (or 64 byte) bit array. This array contains 86 mobile network operator identifiers, in this implementation the mobile country code and the mobile network code, hashed into the filter. The implementation uses the SHA256 hash function to hash the operator identifier. The resulting 32 byte long hash is then divided into four parts. After that the integer values of the 8-bit bytes in each part are summed together and then a modulo 512 is taken from these results. The four resulting values in the range from 0 to 511 correspond to bit positions in the filter and uniquely identify the operator. Figure 4.4 shows this process.

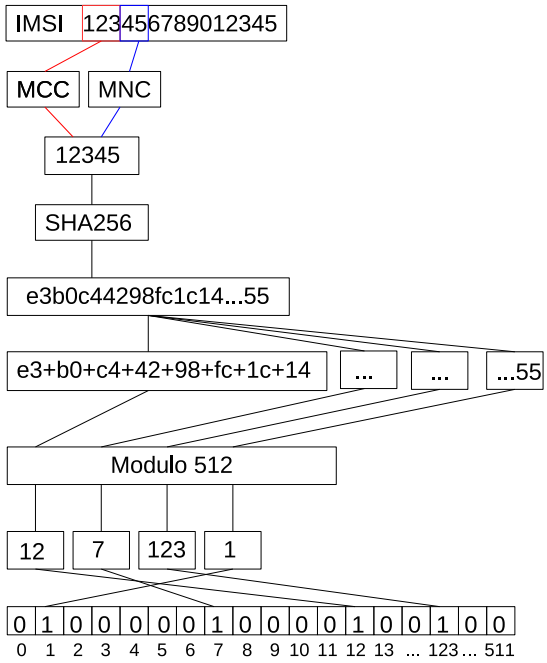

Figure 4.4: APN Bloom filter

The operator identifier used as input in this process is the first five digits (MCC + MNC) of the IMSI retrieved from the SIM card. If the database was built without compression, the resulting size would be 1,118 bytes (5 bytes for each MCC and MCN; and for each setting the string "Internet" 8 bytes. This combination is repeated 86 times). With Bloom filter the data takes only 72 bytes (64 for filter + 8 for "Internet").

The size and false positive rate of a Bloom filter depend on the variables chosen. The filter size is based on number of objects in the filter, the number of hashes, and the desired false positive rate. The Bloom filter configuration variables we have chosen to use are 512 bit filter with 86 elements and with 4 hashes. This combination gives a false positive rate of under 6 %, which is acceptable since APN settings can be retried if connection creation fails.

SHA256 is a slow hash algorithm for Bloom filter use, but in the prototype it has the advantage of being already loaded to memory for the secure communications used by the Mbed TLS software. Because the hash needs to be calculated only once the no memory overhead for hashing outweighs the SHA256 computational overhead compared to a simpler hash algorithm.

4.4.3 IP stack offloading

The LISA-U200 modem includes an embedded IP, TCP, UDP, HTTP, and TLS stack. We can remove the LwIP library, which is the UDP/IP stack used in Mbed OS, by using the UDP/IP stack in the modem exposing a socket to the operating system. This reduces memory usage by approximately 8 KB in RAM and 50 KB in flash. The offloading also reduces resource usage on the application processor as it does not have to handle UDP or IP packets.

4.4.4 Memory usage

The default settings in different parts of the system allowed much improvement in memory usage. Operating system stack reservation, thread memory reservation of the RTOS, and I/O buffer sizes were reduced to save memory. Unnecessary functions for the operation, such as debugging, were removed from the client. Dynamic runtime memory allocations were replaced with static buffers when ever possible. Development proved difficult with such small memory margins. For example with all the prototype functionality there was not enough memory left for debugging dynamic memory allocation calls.

4.4.5 Device management

The prototype system has limited usability even after it has managed to set up a connection to the Internet. It has a packet data connection, but with cellular connectivity this endpoint is a random address from a pool of IPs. It is most likely also behind a network address translation (NAT). These constraints make it difficult to connect to the device. To solve these issues the prototype connects to the Ericsson device and data management system, which uses CoAP protocol with LwM2M to manage the device and its resources [20]. The prototype device resources, including a thermometer, are exposed as IPSO objects. This allows the management system to discover the available resources on the device instead of requiring pre-configured settings.

4.4.6 Security

The prototype implements DTLS encryption on the connection to the management interface to secure the communication. The used security model is pre-shared key (PSK). In PSK both parties share the same secret key used for the encryption. The problem with the security implementation on the prototype is the lack of a hardware random number generator (HRNG) on the device. The session key generation for the encryption needs random feed to be unguessable. With no HRNG available, the session encryption is not cryptographically strong.

The lack of good entropy sources and security hardware is a common problem for constrained IoT devices. The encryption techniques for the communication encryption often depend on generating random key for the session. This key must be properly random in order to keep the encryption secure. With no HRNG or good entropy sources the system can not create safe encryption keys. The FRDM-K64F reference system contains a hardware random number generator.

4.4.7 Power consumption

The prototypes power consumption was measured using KCX-017 USB power meter [27]. The consumption was compared to the power consumption of the same functionality in the FRDM-K64F. Both boards were powered through the micro USB port on the board. The client was configured to send updates of its status to the server every ten seconds. This rate might be considerably lower in a real world application but it allowed to better understand the power required for the communication in the prototype. No power saving optimization was done in the prototype or in the reference platform. The same operating system and software versions were used where applicable. The prototype used a cellular connection and the K64F used an Ethernet connection.

Chapter 5

Evaluation

This chapter describes the results obtained from a performance evaluation of the prototype. Measurable quantities such as memory usage and power consumption are reported. There is also a qualitative evaluation of other properties related to usability of cellular connectivity, for example, ease of configuration and operation.

5.1 Measurements

This section presents measurement results from the prototype. A comparison is made to the reference system where applicable. The measured properties include flash memory usage, RAM memory usage, power consumption of the development board during operation, and connection time from power on to the management system.

5.1.1 Memory usage

The memory usage of the application was considerably reduced in the prototype. With the original program the GCC ARM compiler reported memory usage was 224,566 bytes of flash and 28,500 bytes of static RAM. This would not run on the C027 board because of the dynamic memory reserved by the system on runtime. The prototype implementation uses 220,997 bytes of flash and only 13,412 bytes of static RAM. The software still does some dynamic memory reservations at runtime but the memory usage optimizations allow the implementation to run on the C027 board. Figure 5.1 presents the difference in Flash usage and Figure 5.2 shows differences in RAM usage.

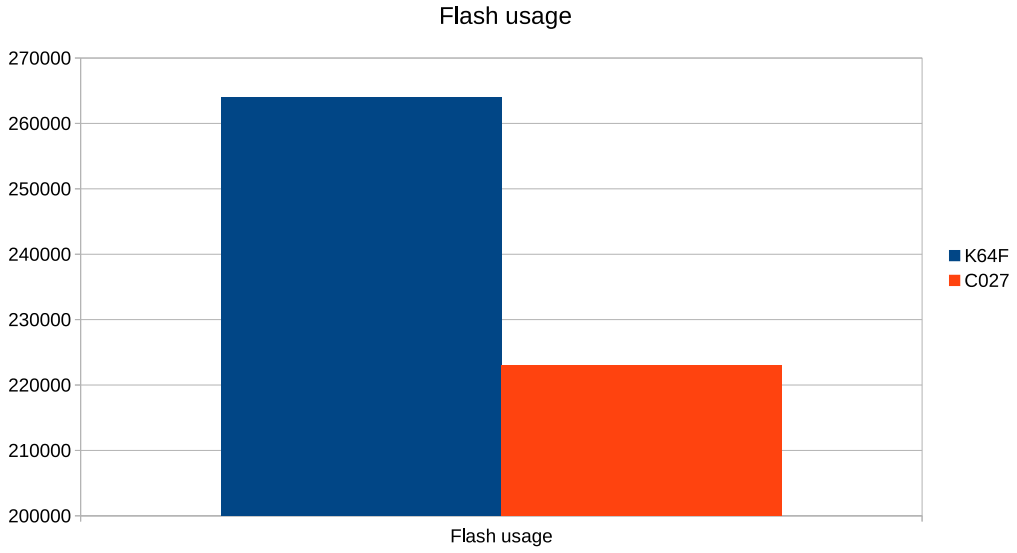

Figure 5.1: Flash usage

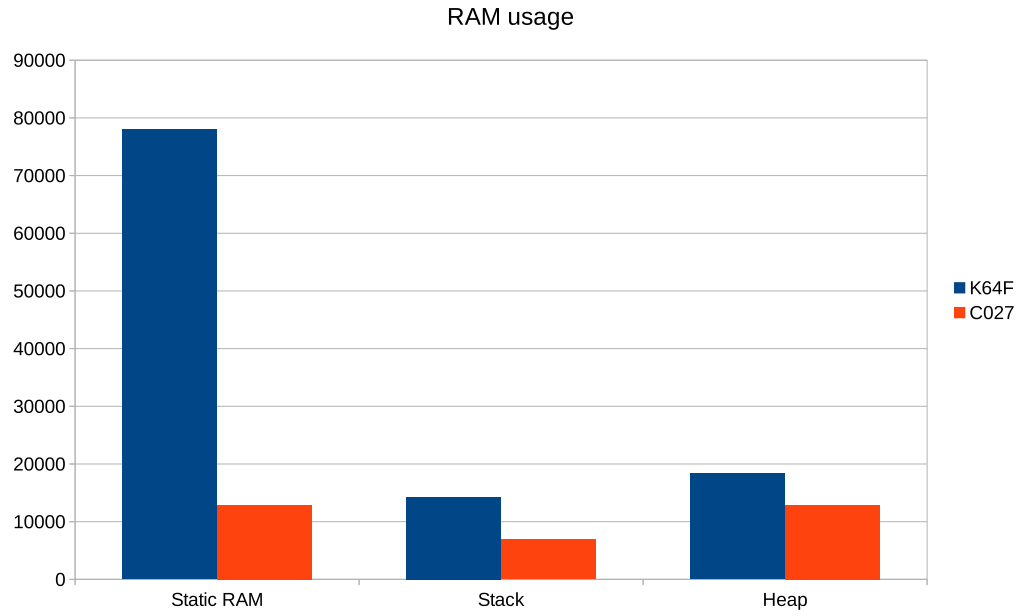

Figure 5.2: RAM usage

Stack and buffer sizes were reduced by experimenting with different sizes. The Mbed OS 5 includes the Cortex Microcontroller Software Interface Standard (CMSIS) Real Time Operating System (RTOS). The RTOS has its own stacks that need to be configured separately. Table 5.1 presents the RAM memory reductions done in stacks and buffers of the prototype system. These values were obtained experimentally by reducing the sizes until the program did not run anymore. However, these modifications are platform- and application-dependent so they must be adjusted for each specific application.

The prototype was compared against a reference system. This used the same software with default settings, and Ethernet as a connection. The flash usage of mbed-client in the reference board was 263,992 bytes. The difference in the RAM usage is substantial, as the reference system used 78,008 bytes of static RAM, 14,256 bytes of stack and 16,060 bytes of heap memory in operation.

Component	Default	Prototype
Main stack	4,096	2,048
Default thread stack	2,048	768
RTOS event loop	6,144	2,048
OS idle thread	512	256
Serial tx/rx buffers	256	64

Table 5.1: Memory usage without and with optimization

5.1.2 Power consumption

Figure 5.3 shows the power consumption of the prototype compared to the reference system. The K64F was drawing power steadily at 760 mW. The C027 was using 600 mW when idle and peaked to 1,000 mW during data transmission. The results are for a data update interval of 10 seconds. Table 5.2 shows the energy consumption with the difference within the error margin of the power meter. With longer update intervals the lower idle power drain of the C027 would result in energy savings compared to the K64F.

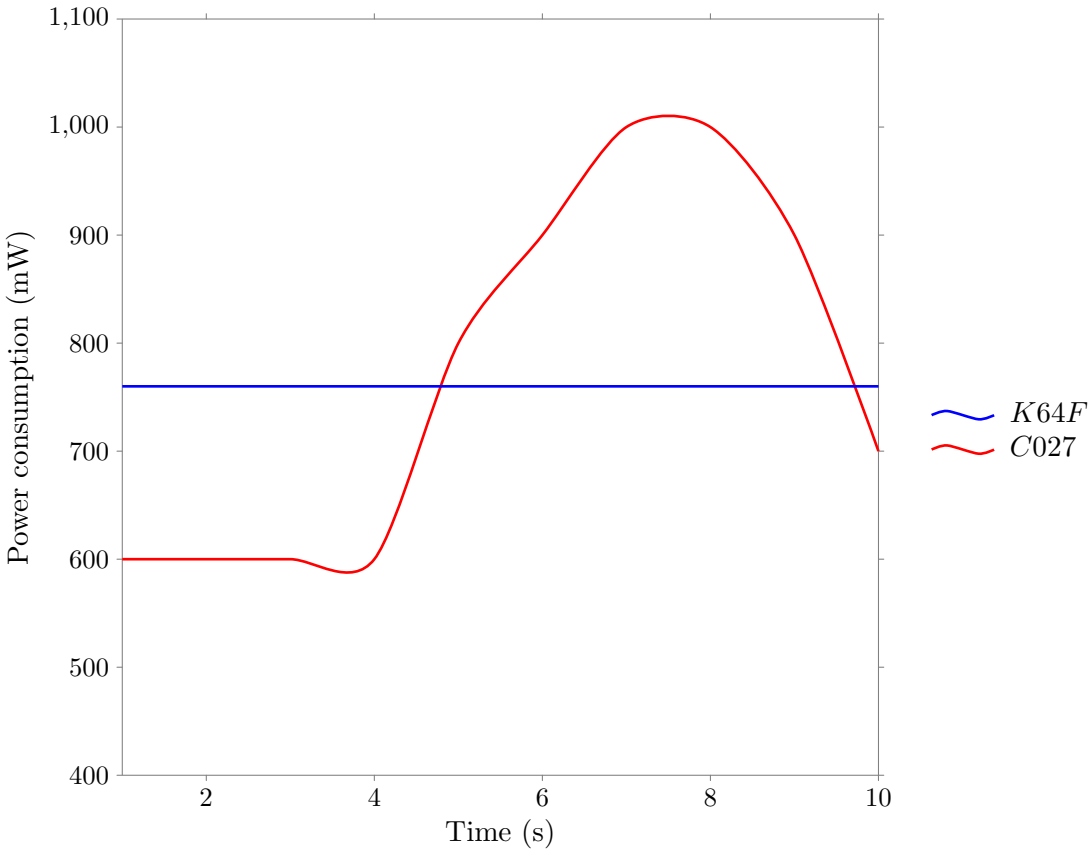

Figure 5.3: Power consumption

These results are in line with the given values for the respective microcontrollers and peripherals [37, 38]. The power consumption results are quite high considering that a Raspberry Pi Zero W running an application with active Bluetooth communication draws only 450 mW. However, no power optimization was done in the prototype or reference system. Turning off unnecessary peripherals, lowering the clock speed of the microcontrollers, and utilizing sleep modes on the microcontrollers might result in notably lower power consumption. The LPC1768 MCU data sheet claims deep power off state power consumption of only few microWatts [38].

Time (minutes)	**Energy consumption (Wh) K64F**	**Energy consumption (Wh) C027**
10	0.13	0.14
20	0.25	0.26
30	0.38	0.39
40	0.50	0.51
50	0.63	0.64
60	0.76	0.77
70	0.89	0.90
80	1.00	1.02
90	1.14	1.16
100	1.27	1.29

Table 5.2: Energy consumption comparison

5.1.3 Connection time

We also measured the initial connection time from powering on the device to the first reading appearing in the IoT Accelerator user interface. This was 20 seconds for the prototype using cellular connectivity and only 8 seconds for the reference board. Both the prototype and the reference board used several seconds to boot the device and load the operating system. A significant amount of the time difference between the two platforms was due to the setting up of the cellular data connection. This startup time can be even longer for the first time the device connects to the cellular network. Liberg et al. [29] calculate that an initial connection creation for a cellular IoT device can take up to 10 minutes.

5.1.4 Cellular usability

We encountered many problems during the prototype implementation concerning the usability of cellular data connection on an IoT device. Cellular modem support is not presently available in all IoT operating systems. When operating system level support is available, it may be limited to a serial connection to the modem. This requires the developer to implement functionality by using AT commands to control the modem [54]. The added configuration needed for setting up the connection, namely PIN and APN settings, make cellular data today less usable then some of the alternative connectivity methods.

Chapter 6

Conclusions

The proof of concept prototype implements a secure IoT device management connection over cellular network. It demonstrates that class-2 devices can be used in secure remote managed applications with automated cellular connectivity. A careful choice of selected functionality could probably allow even class-1 devices to operate with these requirements. Many of the used libraries could be even further optimized for constrained device usage.

The power consumption of the prototype reveals that the cellular modem consumes a large share of the energy. However, it must be noted that the prototype employs a 3G modem. The power consumption of Narrow Band Internet of Things (NB-IoT) modems is considerably lower. Liberg et al. [29] provide power calculations for NB-IoT modems claiming ten years of operation time with two AA batteries. These modems should also be more moderately priced as they are simpler in design. The introduction of NB-IoT networks and modems makes cellular connection a viable option even for low-end sensor devices.

The usability of the cellular connection and especially the programming APIs were a concern in many conversations on IoT device connectivity when presenting the prototype implementation. Better programming APIs and connectivity management should be provided to support cellular connection adoption in IoT systems.

The ongoing standardization of 5G networks provides many interesting research areas for cellular connectivity on IoT devices. Many of the connectivity issues encountered in this thesis could be addressed already on the network protocol level. The power consumption of new modems is always an interesting subject because of its impact on the battery lifetime of the devices.

With the rising number of devices connected to Internet, new improved methods are needed to enable connectivity. 5G networks provide better support for device communication. Palattella et al. [40] analyze the technological and financial advantages and disadvantages of 5G networks as IoT connectivity enabler. With improved radio resource usage, lower power consumption, and decreased cost cellular modems are capable of meeting future IoT connectivity requirements.

Bibliography

[1] 3GPP. TS 23.003: Numbering, addressing and identification, Dec 1999.

[2] 3GPP. TS 23.060: General Packet Radio Service (GPRS); Service description; Stage 2, Dec 1999.

[3] 3GPP. TS 31.101: UICC-terminal interface; Physical and logical characteristics, Oct 1999.

[4] 3GPP. TS 31.102: Characteristics of the Universal Subscriber Identity Module (USIM) application, Oct 1999.

[5] 3GPP. TR 36.888: Study on provision of low-cost Machine-Type Communications (MTC) User Equipments (UEs) based on LTE. Tech. rep., 3GPP, 2013.

[6] 3GPP. TR 45.820: Cellular system support for ultra-low complexity and low throughput Internet of Things (CIoT). Tech. rep., 3GPP, 2013.

[7] 3GPP. TS 38.211: NR: Physical channels and modulation, Dec 2017.

[8] ARM. Developer platform for devices, Mbed, 2018. `https://os.mbed.com`. Accessed 9 Jan 2018.

[9] ARM. Mbed home, 2018. `https://www.mbed.com`. Accessed 9 Jan 2018.

[10] Atzori, L., Iera, A., and Morabito, G. The Internet of Things: A survey. *Computer Networks 54*, 15 (2010), 2787 – 2805.

[11] Bloom, B. H. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Commun. ACM 13*, 7 (July 1970), 422–426.

[12] Bormann, C., Ersue, M., and Keranen, A. RFC 7228: Terminology for Constrained-Node Networks. Tech. rep., IETF, 2014.

[13] Broder, A., and Mitzenmacher, M. Network applications of bloom filters: A survey. *Internet mathematics 1*, 4 (2004), 485–509.

[14] Camarillo, G., and Garcia-Martin, M.-A. *The 3G IP multimedia subsystem (IMS): merging the Internet and the cellular worlds*. John Wiley & Sons, 2007.

[15] Dahlman, E., Parkvall, S., Skold, J., and Beming, P. *3G evolution: HSPA and LTE for mobile broadband*. Academic press, 2007.

[16] Dierks, T., and Rescorla, E. RFC 5246: The transport layer security (TLS) protocol. Tech. rep., IETF, 2008.

[17] DNA. *Mokkuloiden ja päätelaitteiden APN-asetukset*. DNA, 2016. `https://www.dna.fi/documents/753910/853450/DNA_Mokkuloiden_ja_paatelaitteiden_APN-asetukset.pdf/70d0801a-27c1-a63c-67b5-5698ddefbbde`. Accessed 10 Jan 2018.

[18] Droms, R. RFC 2131: Dynamic Host Configuration Protocol. Tech. rep., IETF, 1997.

[19] Ericsson. The Ericsson Mobility Report, 2017. `https://www.ericsson.com/en/mobility-report`. Accessed 19 Dec 2017.

[20] Ericsson. IoT Accelerator, 2018. `https://www.ericsson.com/en/internet-of-things/solutions/iot-platform`. Accessed 9 Jan 2018.

[21] Gomez, C., Darroudi, S., and Savolainen, T. Internet-Draft: IPv6 Mesh over BLUETOOTH(R) Low Energy using IPSP, 2017.

[22] Heer, T., Garcia-Morchon, O., Hummen, R., Keoh, S. L., Kumar, S. S., and Wehrle, K. Security challenges in the ip-based internet of things. *Wireless Personal Communications 61*, 3 (2011), 527–542.

[23] IEEE. IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications. *IEEE Std 802.11-1997* (Nov 1997), 1–445.

[24] IEEE. IEEE Standard for Low-Rate Wireless Networks. *IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011)* (April 2016), 1–709.

[25] IEEE. IEEE Standard for Ethernet - Amendment 10: Media Access Control Parameters, Physical Layers, and Management Parameters for 200 Gb/s and 400 Gb/s Operation. *IEEE Std 802.3bs-2017* (Dec 2017), 1–372.

[26] Jimenez, J., Koster, M., and Tschofenig, H. IPSO smart objects. In *Position paper for the IOT Semantic Interoperability Workshop* (2016).

[27] KCX. *KCX-017 USB Terminal Power Adapter Voltage / Current /Capacity Tester*. KCX, 2016. `https://cdn.solarbotics.com/products/datasheets/usb%20terminal%20power%20adapter%20-%20kcx-017.pdf`. Accessed 10 Jan 2018.

[28] Laari, P. *State-Efficient Forwarding with In-packet Bloom Filters*. PhD thesis, Aalto University, 2017.

[29] Liberg, O., Sundberg, M., Wang, E., Bergman, J., and Sachs, J. *Cellular Internet of Things: Technologies, Standards, and Performance*. Academic Press, 2017.

[30] LoRa. *LoRaWAN 101: A Technical Introduction.* LoRa Alliance. https://docs.wixstatic.com/ugd/eccc1a_20fe760334f84a9788c5b11820281bd0.pdf. Accessed 10 Jan 2018.

[31] Metcalfe, R. M., and Boggs, D. R. Ethernet: Distributed packet switching for local computer networks. *Communications of the ACM 19*, 7 (1976), 395–404.

[32] Microsoft. COSA overview, 2017. https://docs.microsoft.com/en-us/windows-hardware/drivers/mobilebroadband/cosa-overview. Accessed 10 Jan 2018.

[33] Minges, M. L., et al. *Electronic materials handbook: packaging*, vol. 1. Asm International, 1989.

[34] Montenegro, G., Kushalnagar, N., Hui, J., and Culler, D. RFC 4944, Transmission of IPv6 Packets over IEEE 802.15.4 Networks. Tech. rep., IETF, 2007.

[35] Narten, T., Simpson, W. A., Nordmark, E., and Soliman, H. RFC 4861: Neighbor discovery for IP version 6 (IPv6). Tech. rep., IETF, 2007.

[36] Nieminen, J., Savolainen, T., Isomäki, M., Patil, B., Shelby, Z., and Gomez, C. RFC 7668: IPv6 over BLUETOOTH(R) Low Energy. Tech. rep., IETF, 2015.

[37] NXP. *Kinetis K64F Sub-Family Data Sheet.* NXP Semiconductors. https://www.nxp.com/docs/en/data-sheet/K64P144M120SF5.pdf. Accessed 10 Jan 2018.

[38] NXP. *LPC1769/68/67/66/65/64/63 Product data sheet.* NXP Semiconductors. https://www.nxp.com/docs/en/data-sheet/LPC1769_68_67_66_65_64_63.pdf. Accessed 10 Jan 2018.

[39] Open Mobile Alliance. LWM2M Specification 1.0. Tech. rep., Open Mobile Alliance: San Diego, CA, USA, 2017.

[40] Palattella, M. R., Dohler, M., Grieco, A., Rizzo, G., Torsner, J., Engel, T., and Ladid, L. Internet of Things in the 5G Era: Enablers, Architecture, and Business Models. *IEEE Journal on Selected Areas in Communications 34*, 3 (March 2016), 510–527.

[41] Particle. Particle Electron (2G/3G) Cellular Development Kits, 2016. `https://www.particle.io/products/hardware/electron-cellular-dev-kit`. Accessed 9 Jan 2018.

[42] Plummer, D. RFC 826: Ethernet Address Resolution Protocol. Tech. rep., IETF, 1982.

[43] Rahman, L. F., Ozcelebi, T., and Lukkien, J. J. Choosing your IoT programming framework: Architectural aspects. In *Future Internet of Things and Cloud (FiCloud), 2016 IEEE 4th International Conference on* (2016), IEEE, pp. 293–300.

[44] Rescorla, E., and Modadugu, N. RFC 6347, Datagram Transport Layer Security Version 1.2. Tech. rep., IETF, 2012.

[45] Sethi, M., Arkko, J., and Keränen, A. End-to-end security for sleepy smart object networks. In *Local Computer Networks Workshops (LCN Workshops), 2012 IEEE 37th Conference on* (2012), IEEE, pp. 964–972.

[46] Sethi, M., Kortoci, P., Di Francesco, M., and Aura, T. Secure and low-power authentication for resource-constrained devices. In *Internet of Things (IOT), 2015 5th International Conference on the* (2015), IEEE, pp. 30–36.

[47] Sethi, M., Lijding, M., Di Francesco, M., and Aura, T. Flexible management of cloud-connected digital signage. In *Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom), 2015 IEEE 12th Intl Conf on* (2015), IEEE, pp. 205–212.

[48] Sethi, M., Oat, E., Di Francesco, M., and Aura, T. Secure bootstrapping of cloud-managed ubiquitous displays. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (2014), ACM, pp. 739–750.

[49] Shelby, Z., Hartke, K., and Bormann, C. RFC 7252: The Constrained Application Protocol (CoAP). Tech. rep., IETF, 2014.

[50] Suomalainen, J., Valkonen, J., and Asokan, N. Security associations in personal networks: A comparative analysis. In *European Workshop on Security in Ad-hoc and Sensor Networks* (2007), Springer, pp. 43–57.

[51] Thread Group. Thread 1.1 Specification, 2017. `https://www.threadgroup.org/ThreadSpec`. Accessed 3 Jan 2018.

[52] u blox. *C027-C20/U20/G35 mbed enabled Internet of Things (IoT) starter kit User Guide.* u-blox, 11 2015.

[53] u blox. *LISA-U2 series 3.75G HSPA+ Cellular Modules Data Sheet.* u-blox, 10 2016.

[54] u-blox. *u-blox Cellular Modules AT Commands Manual.* u-blox, 2018. `https://www.u-blox.com/sites/default/files/u-blox-CEL_ATCommands_(UBX-13002752).pdf`. Accessed 10 Apr 2018.

[55] Wi-Fi Alliance. Wi-fi protected setup specification. *WiFi Alliance Document 23* (2007).

[56] Zigbee Alliance. Zigbee, 2017. `http://www.zigbee.org`. Accessed 3 Jan 2018.