

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO



# **Bluetooth based Warning System for Ambient Assisted Living**

**João Pedro Cruz Silva**

Master in Electrical and Computers Engineering

Scientific supervision by:

Hugo Sereno Ferreira, Assistant Professor  
Department of Informatics Engineering

Additional supervision by:

Manuel Monteiro, M.Eng. and Filipe Sousa, M.Eng.  
Fraunhofer Portugal

July 21, 2015



A Dissertação intitulada

“Bluetooth Based Warning System Ambient Assisted Living”

foi aprovada em provas realizadas em 16-07-2015

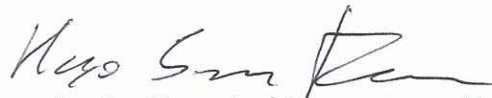
o júri



Presidente Professor Doutor José Nuno Teixeira de Almeida  
Professor Auxiliar do Departamento de Engenharia Eletrotécnica e de Computadores  
da Faculdade de Engenharia da Universidade do Porto



Professor Doutor Ângelo Martins  
Professor Adjunto do Departamento de Engenharia Informática do Instituto Superior  
de Engenharia do Porto



Professor Doutor Hugo José Sereno Lopes Ferreira  
Professor Auxiliar Convidado do Departamento de Engenharia Informática da  
Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projeto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extratos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são corretamente citados.



Autor - João Pedro Cruz Silva



# Resumo

Os idosos são o grupo etário com maior taxa de crescimento, especialmente em países desenvolvidos. Estas pessoas tendem a viver sozinhas, o que, numa situação de emergência, leva a que os serviços de resposta demorem mais tempo a ser notificados da mesma e ainda mais tempo a agir. Seria preferível que estas situações fossem evitadas e não tratadas.

As Redes de Sensores Sem Fios têm estado nos interesses de investigação da comunidade académica já há alguns anos e a Internet das Coisas tem crescido mais depressa que nunca. Estes desenvolvimentos podem ser aplicados em *Ambient Assisted Living* para permitir aos idosos viver no seu ambiente preferido por mais tempo do que o que seria normalmente possível. Isto tem vários impactos positivos: aumenta a qualidade de vida dos idosos, dá-lhes mais independência e alivia a carga dos serviços de emergência.

Neste documento propomos um sistema baseado numa tecnologia sem fios relativamente recente: o *Bluetooth Low Energy*. Esta tecnologia oferece uma largura de banda menor quando comparada com o *Bluetooth* clássico, mas oferece também vida de bateria muito mais longa.

Este sistema funciona com uma arquitectura *Peer-to-Peer* enquanto fornece possibilidades para que seja expandido de formas mais escaláveis. Um pequeno microcontrolador com conectividade BLE correndo um interpretador customizado permite ao utilizador (um idoso com conhecimentos de informática ou seus acompanhantes) configurar regras que, usando sensores e estimando a distância ao utilizador permitem agir em contextos que correspondam a situações de emergência. Por exemplo, se um idoso estiver a cozinhar e se esquecer do fogão ligado, o microcontrolador pode actuar e desligar o fogão se o idoso em questão se afastar demasiado.

Esperamos que a exposição compreensiva de informação neste documento potencie desenvolvimentos futuros nesta área.



# Abstract

The elderly are the fastest growing age group, especially in developed countries. These people tend to live alone, which leads to EMS taking longer to be notified than would be preferred and even longer for them to act. It would be preferable for these emergencies to be prevented rather than treated.

Wireless Sensor Networks have been in the research interests of the academic community for a few years now, and the Internet of Things is growing faster than ever. One can take advantage of these new developments and apply them to Ambient Assisted Living in order to allow the elderly to live in their preferred environment for longer. This has several positive effects: it increases elderly people's quality of life, their independence, and puts less strain on emergency services.

In this paper we present a system based on a relatively new wireless technology: Bluetooth Low Energy. This technology offers us smaller bandwidth performance when compared to its older counterpart (classic Bluetooth) but with greatly increased battery life.

Our system functions in a peer-to-peer architecture while providing a framework for being expanded into more scalable options. A small Microcontroller with BLE capabilities running a custom interpreter allows the user (a tech-savvy elder or his/her caregiver) to set *rules* that, using sensors and estimating distance to a connected device, can contextually act in order to prevent emergency situations: e.g. if an elder is cooking and forgets the stove on, this MCU can act and turn it off if the elder gets too far away.

We hope that this paper's comprehensive presentation of information lays the foundation for future developments in this area.





# Acknowledgements

I'd like to thank:

My girlfriend, for putting up with me, and for being a constant in my life.

My parents, for supporting me, however annoying they may be.

My grandfather, may he rest in peace, for having been a great role model.

My grandmother, for the grandmotherly gifts she gives me.

My godmother and godfather, for their knowledge and support.

My cousin and his wife, Lily, for giving me advice, even though I'm often not willing to take it.

Nuno, because sometimes he can be the only person as deranged as I am.

Jota, for the group assignments together, the long nights, and the rides home.

My dog, for being around for me when I don't want *people* around.


My coaches, for helping to keep me sane when I'm stressed out beyond imagination.

To Hugo for the conversations, ideas and advices; To Manuel for the tough love; To Filipe for the support; And to João Oliveira for the debugging advice.

To Barbosa and to João Lima for being thesis buddies... also for the cereal.

To all those that I can't remember right now but that aren't forgotten.

And, finally, to the giants that came before for lending me their shoulders.

A handwritten signature in black ink, appearing to read 'João Silva'. The signature is stylized and cursive, with the first name 'João' being larger and more prominent than the last name 'Silva'.

(João Silva)



*“Behind every great man is a woman rolling her eyes.”*

Jim Carrey



# Contents

<b>Acronyms and Abbreviations</b>	<b>xv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Problem . . . . .	3
1.3 System Requirements . . . . .	3
1.4 Objectives . . . . .	3
<b>2 Research Problem</b>	<b>5</b>
2.1 The System . . . . .	5
2.2 The Testbed . . . . .	6
<b>3 State of the Art</b>	<b>7</b>
3.1 WSN . . . . .	7
3.1.1 Ubiquitous Computing . . . . .	8
3.1.2 Internet of Things/Everything . . . . .	8
3.1.3 Meshed and Multi-hop Networks . . . . .	9
3.2 MOM . . . . .	11
3.2.1 Protocols . . . . .	12
3.2.2 Applications . . . . .	14
3.2.3 Comparison . . . . .	15
3.3 Wireless Communication Solutions . . . . .	17
3.3.1 What is defined by the <a href="#">IEEE</a> . . . . .	17
3.3.2 Bluetooth . . . . .	19
3.3.3 ZigBee . . . . .	19
3.3.4 Wi-Fi . . . . .	20
3.3.5 nRF24L01+ . . . . .	20
3.3.6 Others . . . . .	21
3.3.7 Comparison . . . . .	21
3.4 Hardware Platforms . . . . .	22
3.4.1 Android Smartphone . . . . .	22
3.4.2 Raspberry Pi . . . . .	22
3.4.3 SheevaPlug . . . . .	23
3.4.4 TelosB . . . . .	23
3.4.5 Texas Instruments ( <a href="#">TI</a> ) CC2540 . . . . .	24
3.4.6 Nordic nRF51822 . . . . .	24
3.5 Indoor Location Solutions . . . . .	24
3.5.1 Bluetooth Low Energy ( <a href="#">BLE</a> ) based . . . . .	25

3.5.2	ZigBee based . . . . .	25
3.5.3	Wi-Fi . . . . .	25
3.6	Applied Research . . . . .	26
3.6.1	Health sensor systems . . . . .	26
3.6.2	Agriculture . . . . .	26
3.6.3	Ambient Assisted Living . . . . .	27
<b>4</b>	<b>Development tools</b>	<b>29</b>
4.1	Fraunhofer Pandlet . . . . .	29
4.2	Development Boards . . . . .	30
4.2.1	PCA10001 . . . . .	30
4.2.2	PCA20006 . . . . .	31
4.3	The Softdevice . . . . .	31
4.4	nRF51 SDK . . . . .	33
4.5	IoT SDK . . . . .	33
4.5.1	The LWIP IP stack . . . . .	34
4.6	Other software . . . . .	34
<b>5</b>	<b>The BlueWarnAAL solution</b>	<b>37</b>
5.1	The solution broken down . . . . .	37
5.2	Components . . . . .	38
5.2.1	FhP Pandlet developments . . . . .	38
5.2.2	Android developments . . . . .	51
5.2.3	Raspberry Pi developments . . . . .	55
<b>6</b>	<b>Testbed and Results</b>	<b>59</b>
6.1	The proposed scenario . . . . .	59
6.1.1	The testbed . . . . .	59
6.2	Functional tests . . . . .	60
6.3	Distance estimation tests . . . . .	61
6.4	Connectivity tests . . . . .	62
<b>7</b>	<b>Conclusion</b>	<b>65</b>
7.1	Achievements . . . . .	65
7.2	Future Work . . . . .	66
7.2.1	Pi application . . . . .	66
7.2.2	MCU . . . . .	66
7.2.3	nRF52 . . . . .	66
7.2.4	Interfacing with the Middleware . . . . .	67
7.2.5	Sensor remote control . . . . .	67
7.2.6	Scaling up . . . . .	68
7.2.7	Calibration . . . . .	68
	<b>References</b>	<b>69</b>

# List of Figures

1.1	The three components that compose our solution . . . . .	4
2.1	Fraunhofer Portugal's Living Lab. . . . .	6
3.1	Quantity computing vs Quality computing . . . . .	9
3.2	Technology roadmap for the evolution into the Internet of Things . . . . .	10
3.3	Basic Network Topologies . . . . .	11
3.4	Fitting two pieces of a puzzle together . . . . .	12
3.5	Comparison of different <b>MOM</b> solutions . . . . .	16
3.6	Another comparison is made with similar results to Fig. 3.5 . . . . .	17
3.7	The <b>OSI</b> model defines several abstraction layers for communication systems. . . . .	18
3.8	The ZigBee protocol stack . . . . .	18
3.9	An example ZigBee network topology. . . . .	20
3.10	A Raspberry Pi Compute Module and a Raspberry Pi model B . . . . .	23
3.11	The Marvell SheevaPlug computer . . . . .	23
3.12	TelosB module with block diagram . . . . .	24
3.13	Proposed system architecture for agriculture . . . . .	26
4.1	Fraunhofer Pandlet CORE with coin for scale . . . . .	29
4.2	The Fraunhofer Pandlet, complete with Memory and Sensing+ modules. . . . .	30
4.3	Nordic's PCA10001 development board . . . . .	30
4.4	Nordic's PCA20006 development board . . . . .	31
4.5	Micro Controller Unit ( <b>MCU</b> ) with full software stack and <b>SD</b> . . . . .	32
4.6	<b>BLE</b> stack as implemented by the <b>SD</b> . . . . .	32
4.7	<b>IoT</b> network with <b>6LoWPAN</b> . . . . .	33
4.8	Memory map generated by linker . . . . .	35
5.1	Block diagram of the solution's three main components . . . . .	37
5.2	Example diagrams of typical situations in which the BlueWarnAAL system would work . . . . .	39
5.3	The different wireless technologies used for node to node communication . . . . .	40
5.4	Block diagram of the Pandlet's three components . . . . .	40
5.5	Exploded view of an AltBeacon advertisement packet . . . . .	41
5.6	First stage of interpreting a rule . . . . .	43
5.7	Flowchart of the different stages of rule processing. . . . .	44
5.8	Second stage of interpreting a rule . . . . .	45
5.9	Third and last stage of interpreting a rule . . . . .	45
5.10	Bad Match Table and calculation process . . . . .	46
5.11	The different stages of the Boyer-Moore algorithm on an example string . . . . .	47

5.12	The naïve algorithm . . . . .	48
5.13	Infix notation vs postfix notation . . . . .	49
5.14	Flowchart of the shunting yard algorithm . . . . .	50
5.15	Step by step solving of an RPN expression . . . . .	51
5.16	Flowchart of the RPN parser algorithm . . . . .	52
5.17	Main switch block of our Reverse Polish Notation (RPN) parser code . . . . .	53
5.18	The Smart Companion Launcher . . . . .	54
5.19	Android application UI flow . . . . .	56
6.1	Schematic of the power switching circuit incorporating a relay . . . . .	60
6.2	The circuit recreated in National Instruments (NI)'s Multisim . . . . .	61
6.3	Oscilloscope plot of voltage at the load . . . . .	62
6.4	Voltage and current for different points in the circuit . . . . .	63
6.5	Open space plot of estimated distance . . . . .	63
6.6	Fraunhofer Portugal's main corridor (first floor) . . . . .	63
6.7	Corridor plot of estimated distance . . . . .	64
6.8	TCPDump utility output during connection with the Nordic MCU . . . . .	64
7.1	The Pimatic UI . . . . .	67



# List of Tables

3.1	Comparison of different qualitative characteristics in the <a href="#">MQTT</a> , <a href="#">STOMP</a> and <a href="#">AMQP</a> protocols . . . . .	16
3.2	Table of wireless technologies and respective <a href="#">IEEE</a> standards . . . . .	18
3.3	Comparison of different wireless technologies' characteristics . . . . .	22
6.1	Circuit bill of materials . . . . .	60



# Acronyms and Abbreviations

**6LoWPAN** IPv6 over Low power Wireless Personal Area Networks

**AAL** Ambient Assisted Living

**AC** Alternating Current

**ADC** Analog-to-Digital Converter

**AES** Advanced Encryption Standard

**AMQP** Advanced Message Queuing Protocol

**AP** Access Point

**API** Application Program Interface

**APIPA** Automatic Private IP Addressing

**APK** Android application PacKage

**ARM** Advanced RISC Machines

**ARP** Address Resolution Protocol

**ART** Android RunTime

**BAN** Body Area Network

**BJT** Bipolar Junction Transistor

**BLE** Bluetooth Low Energy

**BNC** Bayonet Neill–Concelman

**BSD** Berkeley Software Distribution

**BSS** Basic Service Set

**CBC** Cipher Block Chaining

**CoAP** Constrained Application Protocol

**CPU** Central Processing Unit

**CRC** Cyclic Redundancy Check

**DC** Direct Current

**DDR** Double Data Rate

**DHCP** Dynamic Host Configuration Protocol

**DNS** Domain Name Service

**DSP** Digital Signal Processing

**EDR** Enhanced Data Rate

**EMS** Emergency Medical Services

**EMU** Environmental Measurement Unit

**ESS** Enhanced Service Set

**FFD** Full Function Device

**GPIO** General Purpose Input Output

**GPS** Global Positioning System

**GPU** Graphics Processing Unit

**HART** Highway Addressable Remote Transducer protocol

**HS/EDR** High Speed (**HS**)/Enhanced Data Rate (**EDR**)

**HS** High Speed

**HTTP** HyperText Transfer Protocol

**I/O** Input/Output

**IBM** International Business Machines Corporation

**IBSS** Independent Basic Service Set (**BSS**)

**IC** Integrated Circuit

**ICMP** Internet Control Message Protocol

**IDE** Integrated Development Environment

**IEC** International Electrotechnical Commission

**IEEE** Institute of Electrical and Electronics Engineers

**IEEE-SA** **IEEE** - Standards Association

**IETF** Internet Engineering Task Force

**IGMP** Internet Group Management Protocol

**IMU** Inertial Measurement Unit

**IoE** Internet of Everything

**IoT/IoE** Internet of Things (**IoT**)/Internet of Everything (**IoE**)

**IoT** Internet of Things

**IP** Internet Protocol

**IPv4** IP version 4

**IPv6** IP version 6

**ISM** Industrial, Scientific and Medical

**JTAG** Joint Test Action Group

**LAN** Local Area Network

**LwIP** Lightweight IP

**M2M** Machine to Machine

**MAC** Media Access Control

**MAN** Metropolitan Area Network

**MCU** Micro Controller Unit

**MEMS** Microelectromechanical Systems

**MIB** Management Information Base

**MIPS** Microprocessor without Interlocked Pipeline Stages

**MIT** Massachusetts Institute of Technology

**MOM** Message Oriented Middleware

**MPR** Multi-Point Relay

**MQTT** Message Queue Telemetry Transport

**MQTT-SN** MQTT - Sensor Networks

**NI** National Instruments

**NIST** National Institute of Standards and Technology

**NP** Nondeterministic Polynomial

**OASIS** Organization for the Advancement of Structured Information Standards

**OLSR** Optimized Link-State Routing protocol

**OOB** Out Of Band

**OSI** Open Systems Interconnection

**OSPF** Open Shortest Path First

**P2P** Peer to Peer

**PAN** Personal Area Network

- PARC** Palo Alto Research Center
- PC** Personal Computer
- PCB** Printed Circuit Board
- PHY** Physical
- PPI** Programmable Peripheral Interconnect
- PPP** Point-to-Point Protocol
- PPPoE** Point-to-Point Protocol (**PPP**) over Ethernet
- PPPoS** **PPP** over Serial
- QoS** Quality of Service
- RAM** Random Access Memory
- RF** Radio Frequency
- RFD** Reduced Function Device
- RFID** Radio-Frequency IDentification
- RISC** Reduced Instruction Set Computing
- RPN** Reverse Polish Notation
- RSSI** Received Signal Strength Indicator
- RTT** Round Trip Time
- SASL** Simple Authentication and Security Layer
- SD** Soft Device
- SDK** Software Development Kit
- SIG** Special Interest Group
- SNMP** Simple Network Management Protocol
- SoC** System on (a) Chip
- SO-DIMM** Small Outline - Dual In line Memory Module
- SPEC** Standard Performance Evaluation Corporation
- SPI** Serial Peripheral Interface
- STOMP** Simple (or Streaming) Text Orientated Messaging Protocol
- SWD** Serial Wire Debug
- TCP/IP** **TCP**/Internet Protocol
- TCP** Transmission Control Protocol

**TI** Texas Instruments

**TLS** Transport Layer Security

**TX** Transmit

**UART** Universal Asynchronous Receiver/Transmitter

**UDP** User Datagram Protocol

**UI** User Interface

**ULP** Ultra Low Power

**URL** Uniform Resource Locator

**USB** Universal Serial Bus

**USD** United States Dollar

**UTF8** Universal Character Set + Transformation Format — 8-bit

**UWB** Ultra-Wide Band

**VCO** Voltage-Controlled Oscillator

**WEP** Wired Equivalent Privacy

**WPA** Wi-Fi Protected Access

**WPAN** Wireless PAN

**WSN** Wireless Sensor Networks

**XML** eXtensible Markup Language





# Chapter 1

## Introduction

Ambient Assisted Living (AAL) has been a big research interest in current times. Due to the trend of an ever-increasing elderly population measures are needed to allow elderly people to not only cope with their condition but to thrive in their preferred environment. Research projects such as [1] and [2] have been funded by both private and public organizations in the last decade to develop solutions for these problems. In this document we propose a solution by leveraging recent developments in technology. In Chapter 1 we provide an introduction to the document by succinctly explaining our **Motivation**, the **Problem** and the project's **Objectives**. The research problem is further discussed in detail in Chapter 2.

### 1.1 Motivation

The world's elderly population is currently the fastest growing age group, especially in developed countries. Populational trends also lead to these people living alone in single households [3] which obviously constitutes a health hazard since elderly people usually have memory and physical impairments as a side effect of the ageing process. Healthcare costs with elderly people are highest in their final two years of life, since they require further attention by caregivers and healthcare professionals, including Emergency Medical Services (EMS). These factors combined will lead to increased costs. If alternative solutions aren't developed the increased costs will be unsustainable. This will lead to either reduced Quality of Service (QoS) for healthcare or to increased costs for the patients. An integrated AAL solution would allow for the elderly to live largely unassisted in their preferred environment, increasing their autonomy and proactively decreasing healthcare costs by preventing possible problems before they even happen.

Wireless Sensor Networks (WSN) have taken leaps and bounds in the last decade. They have shown great promise due to them being durable, decentralized (in mesh configuration) and being able to be clustered into highly available configurations [4]. Yet, they still haven't been realized to their full potential; new technologies are being developed every day in this area and WSN's are being used in new, interesting applications. These include WSN's for monitoring farms [5] and

also traveling sign-on schemes [6]. This nature makes them ideal for distributed systems that need to be assembled without big costs to infrastructure and also makes them highly flexible.

Computing trends have reversed in the previous two decades. The focus was once to build ever more powerful computers, disregarding their efficiency, both energy and space-wise. Starting with the '90s the focus was on miniaturizing computers: making them smaller and more efficient while also being able to maintain computing power. One only has to look at the common smartphone: usually equipped with several gigabytes of Random Access Memory (RAM) and a multi-core processor, to see real world results of this phenomenon. Yet, computers are all around us, and most are left forgotten, maybe because their computing power pales in comparison to smartphones, but still, even the common microwave oven has at least a microprocessor powering it. Although overlooked, these devices still have some computing power that should not be underestimated. One can take advantage of the small, yet important processing power that these devices provide to process data, instead of using them simply as the dumb middle man that is required to deliver data from where it is produced to where it is processed.

Location-based Services exploit the user's physical location in order to provide different services [6]. The means used to attain the physical location of the user range from very simple to very complicated and convoluted, using many technologies. The location can be either absolute or relative. An absolute location can be obtained with technologies like Global Positioning System (GPS), beacons and cross-referencing with stored maps or landmarks. The relative location can be calculated (in reference to other nodes) using trilateration, time of arrival, angle of arrival or time difference of arrival. If one knows the absolute location of at least three nodes in the network and the relative locations of all other nodes are known, one can use this information to calculate the absolute location of every node [4].

The integration of these technologies could offer a solution to the increase in the elderly population by providing a safe environment where they could live, autonomously, not requiring constant intervention on the behalf of caregivers or healthcare professionals by proactively preventing problems before they even occur. Throughout this paper the most basic use case we will consider is the following:

“An elderly person, living alone, is cooking supper in the kitchen. For some reason they need to leave the room. They forget the stove is on. The stove is now a potential fire/explosion hazard.”

Our motivation is to leverage these technologies in order to proactively stop this hazard while also providing an extensible framework so that many more use cases can be implemented in the system. This is not as easy as it sounds: It is a widely known fact that the elderly usually struggle with embracing new technologies. Interfaces and systems must be designed maintaining simplicity in mind, and keeping information clear and concise.

## 1.2 Problem

The quality of life of the elderly must either be maintained or increased as the population in that age group increases either maintaining current costs per person or, preferably, decreasing them. In short, one must develop a solution that enables current healthcare solutions to *scale well* with the increasing population.

The technologies that have been mentioned here are naturally heterogeneous, and therefore difficult to integrate. This forces us to develop some kind of *glue* that allows us to abstract the differences in Application Program Interface (API)'s and communications protocols[7].

There are currently no commercial systems that are able to trigger events in the home based on rules this complex. The home automation systems that do exist and provide similar functionalities are expensive and often proprietary, making them very hard if not impossible to be extended by the user [8].

## 1.3 System Requirements

The proposed system, has the following requirements:

- The system must be based around the Fraunhofer Pandlet/Nordic nRF51822 hardware platforms;
- It must use an Android smartphone to both detect the position of the user and to provide a UI for the system;
- Communication between nodes must use Bluetooth Low Energy (BLE) (as it is natively supported by most recent Android smartphones). Other technologies may be used *in addition* to BLE.
- The system must be capable of interpreting rules set by the user, and configured *on-the-fly*;
- The context of a specific situation must be available as an input for rules;
- The system must work in a Peer to Peer (P2P) fashion, i.e. the Pandlet and the smartphone must constitute a working system without additional hardware;
- The system must be scalable, optionally with extra hardware.

## 1.4 Objectives

Our objectives can be divided into two phases:

- **Development phase:** In this first phase we develop the solution. The solution is composed of three components (Fig. 1.1):

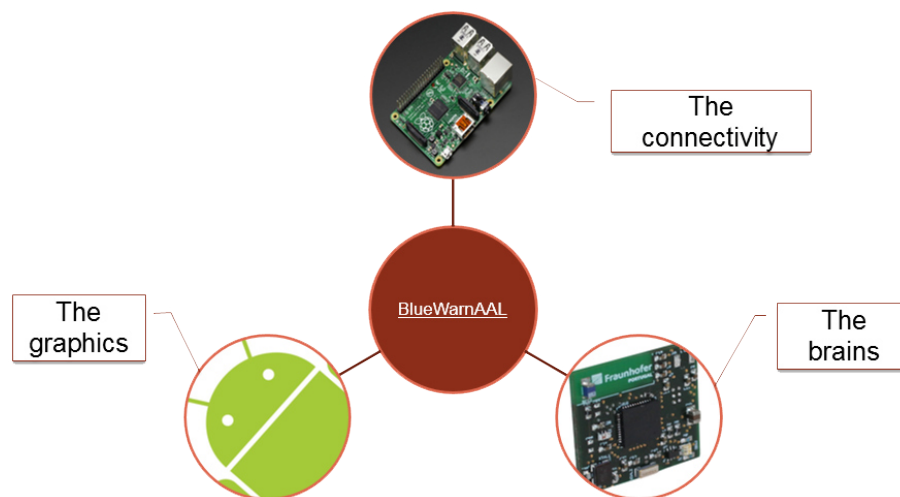


Figure 1.1: The three components that compose our solution: an Android smartphone (graphics), a Nordic **MCU**/Fraunhofer Pandlet (brains) and a Raspberry Pi (connectivity)

- **Nordic application:** This is the **MCU** applicaton. It will have to calculate the distance to the connected smartphone and support custom rules that the user can configure.
- **Android application:** The Android application will work as a frontend for the system, giving the user an interface to access the system and customize its rules.
- **Scaling:** In this phase we give the system the ability to scale from just one node to several.
- **Test phase:** In this phase we test the solution. This will be accomplished by building a testbed and testing the system in a real world situation.

## Chapter 2

# Research Problem

### 2.1 The System

In this section we talk about the problem that we have proposed to address in a more academical sense and discuss its non-triviality.

Developing any [AAL](#) solution is not simple, since the elderly are an age group which is known to be hard to cater to. Solutions developed for the elderly must be simple, but retain functionality. This means that very complicated systems must have simple interfaces and not seem complicated.

The most basic use case that we have proposed to address is mentioned in Section [1.1](#). Although this scenario is simple enough, our main challenge relies in scalability: We want this system to be able to scale from only two devices (the sensor and actuator that are needed in this first scenario) to several tens, hundreds or even thousands of devices. This system should be able to be implemented in every room in an elder's home, or even in a nursing home or perhaps a hospital.

The system must also be easily extensible so that other types of sensors and/or actuators and functionality can be easily added.

On the other hand the system itself is not trivial. Integrating this type of heterogeneous system has been a known challenge for developers. Even though solutions to make communication between applications in distributed systems have been developed (as mentioned in Section [3.2](#)) these systems are inherently complicated.

We are also limited because of power: Most sensors/actuators will not be connected to mains power and must therefore rely on batteries for their operation. For commodity's sake we want the devices to go for as long as possible without battery replacement/recharging. This means that power consumption must be kept to a minimum.

Location with Bluetooth or any other type of generic wireless communication technology is a continuing research effort with no commercial products implemented as of yet. In of itself the location engine that we have proposed to develop can be considered a valid research effort, but in this thesis we propose to not only develop this location engine but use it as a component to our [AAL](#) system, which again, involves a great effort with integration.

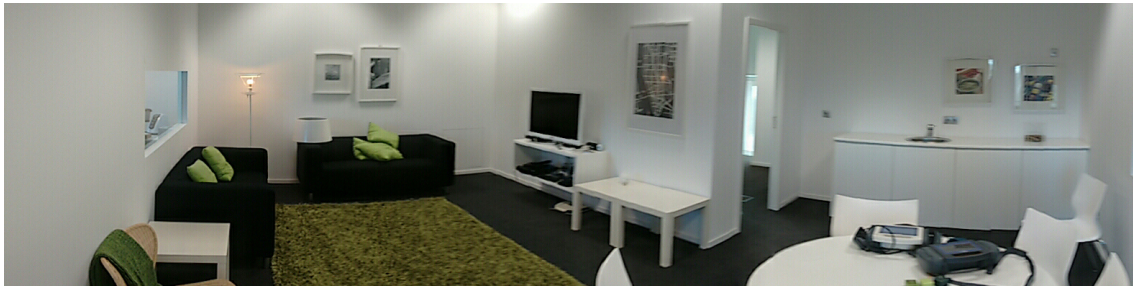


Figure 2.1: Fraunhofer Portugal's Living Lab.

The rule engine that we propose to develop is not without precedent, there have been other efforts in this area, which include [9]. Yet, as far as our state of the art research has been able to ascertain there are no systems similar to ours used in real world situations, especially considering the types of data that we are dealing with and the application.

## 2.2 The Testbed

We propose to develop a life-like testbed to test our system. This means that in a short time frame we need to develop our system and get it to a stage where it is stable enough and bug free so that it can be implemented in an actual testbed.

The testbed will be implemented in Fraunhofer Portugal's Living Lab (Fig. 2.1), which is a reproduction of an average apartment with all usual commodities. The Living Lab consists of a living room/kitchen and a suite (bedroom and bathroom). This will give us a lifelike scenario where test subjects can interact with the system in a plausible situation.

A small-scale tabletop testbed will be built, having power outlets available to where an electric appliance can be connected. These outlets will not, however, be normal outlets. They will be connected to a relay circuit controlled by a [MCU](#). Depending of a smartphone to the testbed, one will see if the system will react as planned: by turning on or off the electric appliance.

# Chapter 3

## State of the Art

### 3.1 Wireless Sensor Networks

Wireless Sensor Networks are the next evolutionary step in computing and automation. *Smart* sensors can be compared in that sense to biological systems. They have exteroceptors, that deal with external stimuli and proprioteptors that deal with information originating in the system.

What constitutes a smart sensor has been standardized by the National Institute of Standards and Technology (NIST) and Institute of Electrical and Electronics Engineers (IEEE) as the IEEE 1451 standard [10]. A *smart* sensor is a sensor that includes, in the same node, software and Digital Signal Processing (DSP) functions as well as standard control protocols and networking interfaces. The point of having a smart sensor is abstracting the output of the transducer (usually a voltage or current value) and the signal processing stage, thereby moving the intelligence closer to the sensor. Wireless sensor networks can be composed of different clusters of sensors, forming different sub-networks. When a clear separation is visible, usually it is between a data collection network and a data processing network.

Since at least some nodes in these networks are not connected to a fixed power supply and must rely instead on batteries to provide power, power conserving and/or generating techniques and ultra-low power node design have become a big focus in this type of research. These techniques include making devices enter a *deep sleep* mode in between transmission/reception in order to conserve power and energy generation including traditional techniques such as as integrating solar panels with the nodes and also innovative ones such as Radio Frequency (RF) energy gathering and the use of Microelectromechanical Systems (MEMS) [11]. Power conserving techniques pertaining to specific wireless transmission technologies will be further discussed in Section 3.3.

Message routing in this specific type of network is usually very different from routing in traditional environments. The network topology is not constant and can be constantly changing as sensors may be mobile. Even traditional routing protocols with a fast conversion time (Open Shortest Path First (OSPF) [12]) can have trouble coping with all the topology changes. The available bandwidth is usually smaller and nodes must be powered down for the most amount of

time in order to conserve power, so any overhead must be minimized [11]. To do this a creative approach is needed and will be discussed in Section 3.1.3.

### 3.1.1 Ubiquitous Computing

While one discusses Wireless Sensor Networks it is important to note the paradigm shift that eventually lead to the coming of age of this technology and its implications, including the huge research interest that revolves around it.

This shift was characterized by a concept that emerged from the mind of a researcher named Mark Weiser at the Xerox Palo Alto Research Center in 1988. Quoting Weiser:

“Ubiquitous computing names the third wave in computing, just now beginning. First were mainframes, each shared by lots of people. Now we are in the personal computing era, person and machine staring uneasily at each other across the desktop. Next comes ubiquitous computing, or the age of calm technology, when technology recedes into the background of our lives [13].”

The future that Weiser envisioned was one where computing was ubiquitous (seeming to be seen everywhere) [14]. Current trends have proven him to be, at least to an extent), right.

Weiser identified three ages of computing:

In the first age many users shared a single Mainframe between them. More powerful computers were necessarily just one bigger, more powerful (and power-hungry) machine.

In the second age, defined by Weiser as the Personal Computer (PC) age, which we are currently experiencing, computing is achieved by means of individual machines. The "One Person, One Computer" paradigm is the norm and computing is achieved by means of these devices. Supercomputers in this day and age are built from thousands of generic off-the-shelf computer components, and their huge performance is essentially attained by exploiting parallelism [15].

Weiser predicted that there would be a third age of computing characterized by huge numbers of small, cheap, and easily-replaceable wirelessly-connected computing nodes. These nodes would fade out of existence, melding imperceptibly with everyday objects. Everything would be connected and computing would be everywhere (see Fig. 3.1). One would use this *Everyware* [16] naturally and fluently without even realising it.

This concept is obviously still just that, but computing is slowly evolving into it. Other similar concepts such as the *Internet of Things/Everything* will naturally come first.

### 3.1.2 Internet of Things/Everything

The Internet of Things is a concept that is closely related to *Ubiquitous Computing* but pertaining more to individual objects rather than ever-present computing. Internet of Things (IoT)/Internet of Everything (IoE) (IoT/IoE) boils down to a scenario where common day objects, people, animals, etcetera, are connected to the Internet and are able to use it to communicate. A *thing* in the IoT is something that can be issued a unique identifier in order for it to communicate without human



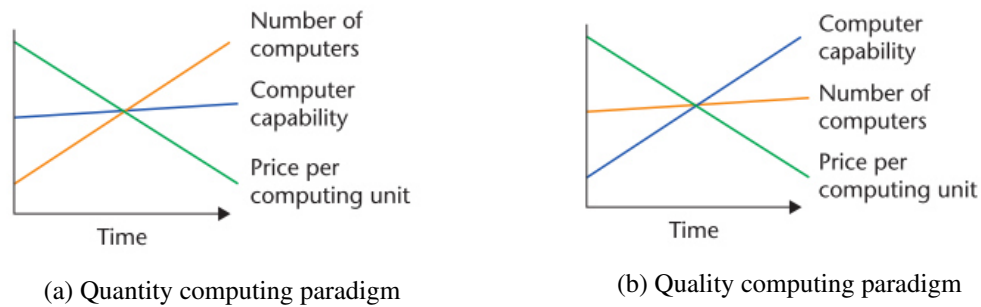


Figure 3.1: If one assumes that the cost of computing will fall over time two outcomes are possible: either (Fig. 3.1a) the number of computers increases or (Fig. 3.1b) their quality increases. If current trends continue the first scenario is more likely and fits into the Ubiquitous Computing concept [13].

interaction. This type of communication is known as Machine to Machine (M2M) communication [17]. In a sample system consisting of a vending machine and the company that restocks the vending machine, a machine connected to the IoT would transmit information about stocks to the company's server and when it would get low on a particular product an automated message would be sent to the employee restocking the machines. This way, the human middle-man is cut out. Products built with this type of capabilities are usually defined as being *smart* (eg: smart label, smart sensor). The concept of IoT was only coined in 1999 by Massachusetts Institute of Technology (MIT)'s Auto-ID Center and its related publications [18], but it had been in the making long before that: The MIT's Computer Science Department runs a Coke machine since the '70s that was the first appliance to have internet connectivity. It allowed users to check the status of the Coke bottles in the machine (quantity and temperature) [19].

As we can see in Fig. 3.2 the first version of the Internet of things came from a necessity for more efficient logistics by using Radio-Frequency Identification (RFID) tags. This first iteration consisted mainly of one-way communication: the tags themselves contained no information beyond an ID code that corresponded to an entry in a database. Eventually, this development will lead to Ubiquitous Computing, where everything is connected.

### 3.1.3 Meshed and Multi-hop Networks

Meshed and Multi-hop Networks are incredibly important, enabling technologies for both Ubiquitous Computing and Internet of Things/Everything. Even though wireless technologies have progressed a lot in the last decade the current networking paradigm is still one of infrastructure: wireless devices access the internet through an Access Point that connects to a wired infrastructure network. This type of web access isn't scalable for huge numbers of ever-present devices: it forces the numbers of Access Points and the size of the wired networks to scale with the number of devices.

One can infer that the future will be in ad-hoc, meshed networking (see Fig. 3.3, bottom-right) [21]. since this type of networking allows for data communication without a wired infrastructure

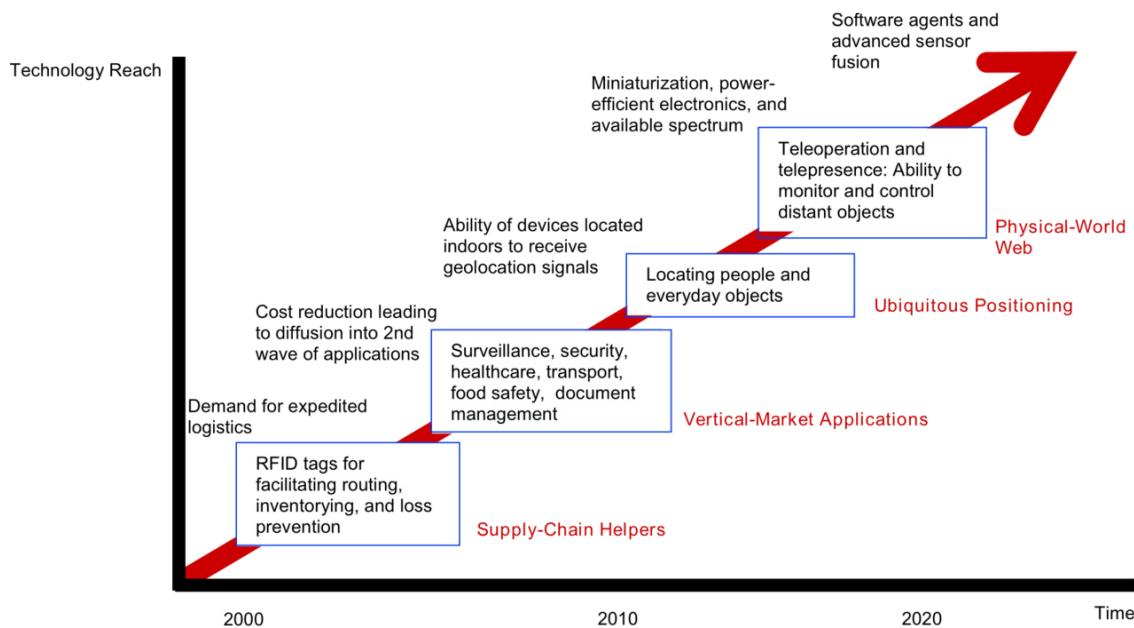


Figure 3.2: Technology roadmap for the evolution into the Internet of Things[20]

network.

Each device in the network can be abstracted as a node: it has computing power and can receive or send messages to other nodes.

Mesh networks generally allow transmission only to a node's nearest neighbour. Multiple paths to a same destination give this network configuration immense versatility and robustness. Nodes are considered identical in the general case, but they need not be. Certain nodes can be designated as "group leaders" of some sort and carry out additional functions besides those of their peers (supernodes). This leads to a network that has a mesh topology and a hierarchy.

As nodes are added to a network, the problem of finding the best path for a message to travel (routing) tends toward **NP** complexity. Breaking up the network into smaller, hierarchical networks helps break up this complex problem into smaller ones that can be solved faster as a whole. In this case, the entry node to a specific cluster is usually designated as the group leader [4].

One can look at the example of the Optimized Link-State Routing protocol (**OLSR**). **OLSR** is a routing protocol developed by the Internet Engineering Task Force (**IETF**) specifically for ad-hoc mobile networks, similar to another Link-State routing protocol: **OSPF**.

The flooding process used by **OSPF** is not appropriate for wireless ad-hoc networks: if every node floods its routing information the media is not used efficiently and any change in network topology (which in the case of wireless networks is much more common than in wired ones) leads to the network being flooded. Also, in the case of mobile networks one wants to keep the number of transmissions to a minimum in order to conserve battery life.

To solve these problems **OLSR** implements the notion of hierarchical nodes: by sending "Hello" messages each node discovers its two hop neighbours and performs a distributed election of a node to be its Multi-Point Relay (**MPR**). The **MPR** acts as the router for a subset of the

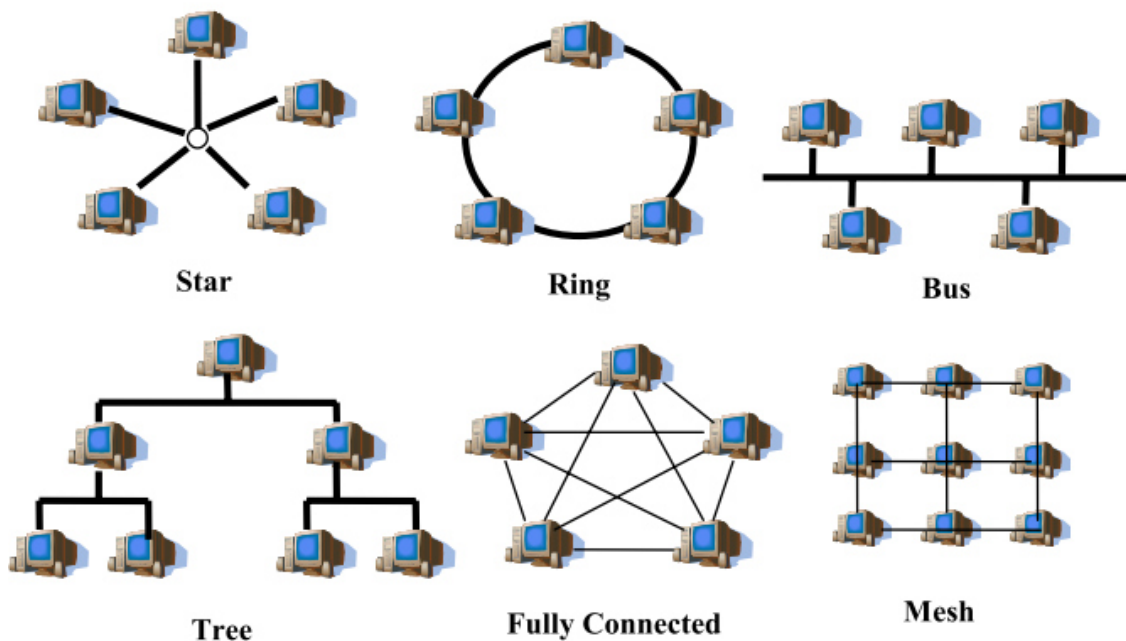


Figure 3.3: Basic Network Topologies[4]

network's nodes and floods the networks with messages pertaining to topology changes on behalf of the other nodes in its group. This means that only the **MPR** has to send messages and greatly reduces the number and frequency of messages flooding the network [22].

## 3.2 Message Oriented Middleware

With the continuous advances in networking technology and with current networking paradigms creeping ever-closer to the **Internet of Things/Everything** multiple heterogeneous systems that once were regarded as completely independent now have to be integrated with one another. Systems have also been becoming distributed. Although this provides us with many advantages it makes systems inherently more complex. This gives us numerous problems when one wants to integrate these completely different systems. Altering every single component to work correctly with each other is simply not practical.

A solution that has been proposed in [7] and works by acting like a "glue" of some sort between two different systems (as can be seen in Fig. 3.4) is a *Middleware*. The idea behind middleware is that it acts like a translator between two or more systems that allow them to talk to one another without them being altered. Using a middleware allows one to easily integrate normally incompatible and often pre-existing systems without altering them.

Different applications can be linked by transmitting messages between them. Because of this, this type of middleware based on events, or messages, is usually called **Message Oriented Middleware (MOM)**.

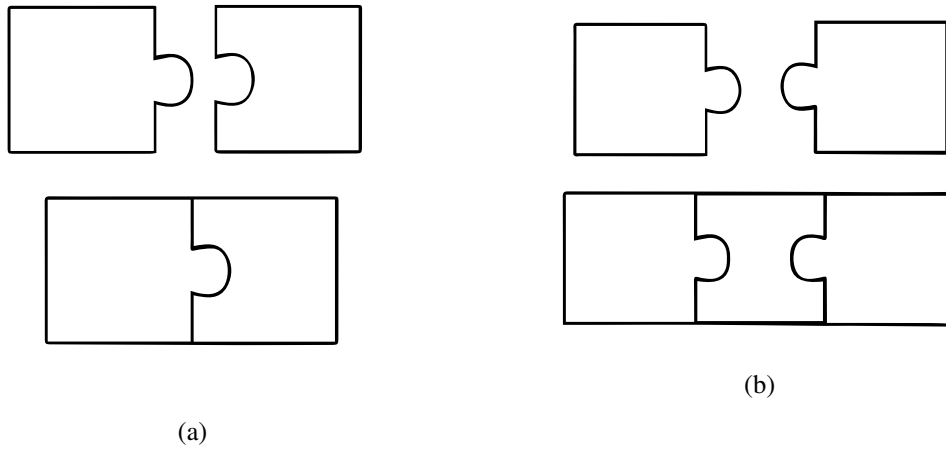


Figure 3.4: In order to fit two pieces of a puzzle together, one can either: Fig. 3.4a alter them individually so that they fit or Fig. 3.4b create an intermediary that fits both.

Some MOM's are based on the Publish/Subscribe model. In this model each client registers either as a publisher or subscriber of messages. Messages have their own *Information Spaces* and can be traded between spaces using *Message Flow Graphs* that specify how the messages are transformed and propagated. This structure can be used to drop obsolete messages, to buffer messages to subscribers that have been off-line and this makes the whole system much more streamlined.

The Publish/Subscribe scheme allows for time, space, and synchronization decoupling. Space decoupling allows two parties to share information between them without actually knowing each other. This is achieved because they both transmit and receive their information using the middleware as a proxy. The subscribers need not know where the information they need is located, they just subscribe to a *topic* or *content type* and the middleware is responsible to get their messages to them. Time decoupling allows those same two parties to not need to be participating in the interaction with the middleware simultaneously in order for the information to be transferred. In this case the middleware acts as a buffer and stores messages for a subscriber and sends them to that subscriber when it is online. Synchronization decoupling means that publishers need not be blocked when producing events and subscribers can get notified in a non-blocking manner when new messages are available for them.

### 3.2.1 Protocols

A number of different protocols have been developed for MOM. In this section we describe three popular ones: *Advanced Message Queuing Protocol (AMQP)*, *Simple (or Streaming) Text Orientated Messaging Protocol (STOMP)* and *Message Queue Telemetry Transport (MQTT)*.

### 3.2.1.1 AMQP

The **AMQP** is an open standard application layer protocol that has been an Organization for the Advancement of Structured Information Standards (**OASIS**) standard since May 2014 [23]. It is a binary, application layer, wire-level protocol and was designed to support many messaging applications. It provides flow control, message delivery guarantees and authentication/encryption. **AMQP** has its own encoding scheme allowing for representation of a wide range of types, while allowing data for being given extra meanings. For example, a string can be appended with additional information for it to be understood as an **URL** [24].

The protocol uses a set of nine frames to create/destroy connections, transfer data and provide control information, these are: "open", "begin", "transfer", "flow", "disposition", "detach", "end" and "close".

Links are the basic connection type in **AMQP**, they are unidirectional. Sessions are comprised of multiple links and can be used for bidirectional transfer of data. A connection between two peers can have multiple sessions. Links are initiated with "attach" and terminated with "detach" frames. Similarly, sessions use the "begin" and "end" frames and connections the "open" and "close" frames. Messages are sent over a link using the "transfer" frame. The "flow" frames are used for flow control, in order ensure **QoS** and prevent deadlocks. The disposition frame is used for the peers to settle on the state of the transfer, for reliability guarantees.

Since **AMQP** is a wire-level protocol that represents data as a stream of octets, it can be used in any application that understands messages in this format. Even though it usually relies on Transmission Control Protocol (**TCP**) for transport purposes it is independent of the **TCP/Internet Protocol (TCP/IP)** protocol stack.

One should note that in this paper we discuss only version 0.9.1 of **AMQP** as it is used in **RabbitMQ**.

### 3.2.1.2 STOMP

The Simple (or Streaming) Text Orientated Messaging Protocol is another open standard messaging protocol that is presented as an alternative to **AMQP** and other implementation specific protocols. It tries to distinguish itself by being simple and lightweight, offering a small but useful messaging API.

Similarly to **AMQP** **STOMP** is frame based, but it is modelled on/inspired by HyperText Transfer Protocol (**HTTP**). Each frame consists of a command and, optionally, headers and a payload. By default, it is text based (Universal Character Set + Transformation Format — 8-bit (**UTF8**)), but supports different encodings, allowing, for example, binary messages to be sent.

Like **AMQP**, **STOMP** uses a reliable 2-way streaming protocol for its transport needs, such as **TCP**, when implemented on the **TCP/IP** stack.

**STOMP** frames follow the structure:

```
COMMAND
header1:value1
```

```
header2:value2
```

```
Body^@
```

Several headers may be sent in a single STOMP message, as long if the keys for each header value are distinct. If there are repeated keys, only the first value is used.

STOMP supports a function named *heart-beating* that allows two peers to test the healthiness of the underlying TCP stream, this is useful for QoS purposes [25].

### 3.2.1.3 MQTT

MQTT is yet another open-source MOM protocol. MQTT was originally developed by IBM and made open-source in the early 2010s.

Similarly to the other protocols presented here MQTT is Publish/Subscribe based, and similarly to STOMP is designed with light weight in mind, but, unlike the other protocols, it is specific to the TCP/IP stack. Unlike STOMP, MQTT's messages are much less verbose and more compact [26].

One specific advantage that MQTT has is a sub-specification: MQTT-SN or MQTT for Sensor Networks. It is an even more trimmed down version of MQTT designed specifically for IoT and Sensor Networking scenarios while being as close to MQTT as possible. It works on non TCP/IP networks such as ZigBee. By using *gateways* and *forwarders* an MQTT network can be extended into these sensor networks.

MQTT-SN also supports an offline keep-alive procedure that is used for supporting *sleeping* clients. Clients that go into a sleep mode have their messages buffered at the gateway and are delivered when they wake up [27].

## 3.2.2 Applications

The protocols that were mentioned in Section 3.2.1 are only that: protocols defined in specifications. In this section we mention three practical MOM implementations that use at least some of these protocols.

### 3.2.2.1 RabbitMQ

RabbitMQ is an open-source MOM message broker written in the Erlang programming language. RabbitMQ uses the AMQP protocol but the project includes gateways for the HTTP, STOMP, and MQTT protocols. It also has a plugin platform that allows one to extend the base functionalities provided.

The RabbitMQ server runs on all major operating systems with official clients written in Erlang, C# and Java, but there is a large community supporting the project offering clients written in a multitude of other languages [28].

One should note that RabbitMQ offers several features for increased reliability and availability, including queue mirroring. RabbitMQ also includes features that make debugging easy, such as a graphical management User Interface (UI) and tracing support.

### 3.2.2.2 Mosquitto

Mosquitto is an open-source MQTT broker written in C. It is a project with a very small footprint aimed at being deployed on low-power and embedded systems. It includes libraries written in C with C++ and Python wrappers [29].

Mosquitto is perfect for being deployed in very low power systems. In [30] the developer mentions that he had tested Mosquitto on a VIA Cyrix III 600 MHz based system processing 1500 messages per second.

### 3.2.2.3 ActiveMQ

ActiveMQ is yet another open-source message broker, but unlike the others presented here it is written in Java. ActiveMQ supports the AMQP, OpenWire, MQTT and STOMP protocols [31].

Since ActiveMQ is written in Java it is supported in all major operating systems without need for porting. With its compatibility with multiple MOM protocols, it supports clients written in basically every language. It also does not require a broker to function and can be implemented in a purely P2P fashion.

However, in benchmarks (such as the ones in Section 3.2.3) ActiveMQ tends to lag behind the competition in performance.

### 3.2.2.4 ZeroMQ

ZeroMQ, often stylized ØMQ is an offshoot project of an AMQP implementation called OpenAMQ. The company that runs the project stopped using AMQP because of concerns that it had become overly complicated.

Unlike other MOM implementations presented in this paper, ZeroMQ is not a message broker, it is in fact a small, fast networking library. It does not require a server running a dedicated message broker in order for it to work. ZeroMQ has a base of tradeoffs: by decreasing complexity it has achieved fantastic performance but at the expense of some commodities.

## 3.2.3 Comparison

Publish/Subscribe MOM is usually intended for very scalable applications. Like everything else, how scalable these middlewares actually are must be quantified.

For this purpose, in 2007 the Standard Performance Evaluation Corporation (SPEC) developed SPECjms2007: an industry-standard performance test for MOM. The SPECjms2007 was

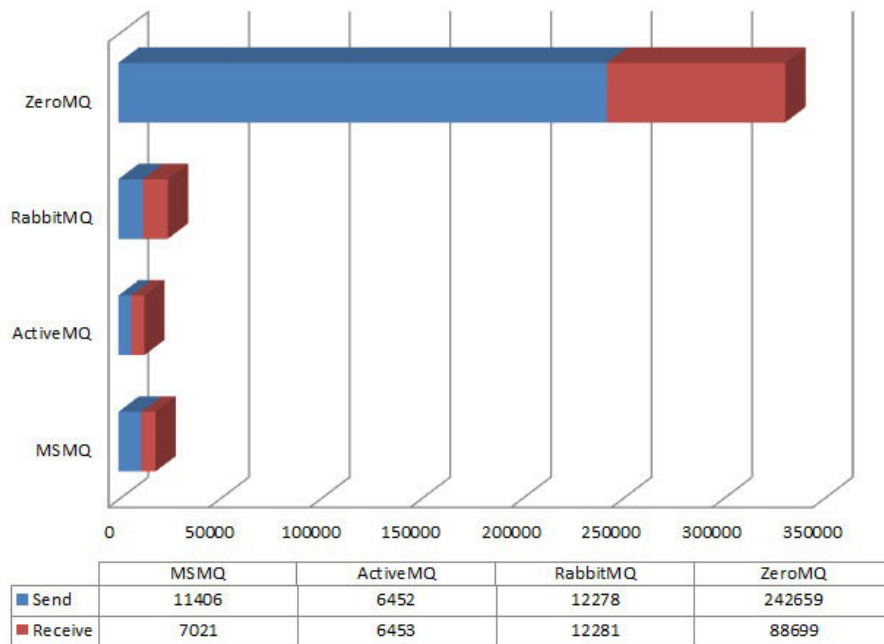


Figure 3.5: In [34] the author compared several different MOM solutions, including ActiveMQ, ZeroMQ and RabbitMQ

extended in 2009 to form the jms2009-PS benchmark specifically for Publish/Subscribe middleware [32, 33]. This benchmark has a real-world scenario behind its thinking and offers a close approximation to how these products should perform in an actual application.

Although these standard testing procedures exist to evaluate the actual performance of the middleware, sometimes performance must be sacrificed for useful features and ease of use. In this section a relative comparison of the studied middlewares is presented.

In terms of raw performance, we can reference these two tests conducted in a non formal fashion [34, 35] (Fig. 3.5 and Fig. 3.6). ZeroMQ is the clear winner, being the fastest with its fast enqueue and dequeue times. RabbitMQ is a close second.

In terms of protocols, AMQP seems to outperform STOMP, which is to be expected, given the latter's increased verbosity.

In [36] another comparison between these protocols is made, but considering their features. In Table 3.1 a similar qualitative comparison is made.

	MQTT	STOMP	AMQP
QoS Levels	3	1	3
Subscription Types	Hierarchical topics	Topics	Exchanges, Queues and bindings
Data Serialization	—	—	AMQP type system or user defined
Standard	OASIS standard	OASIS standard	—
Security	Encryption	—	SASL
	Authentication	Username/Password	—

Table 3.1: Comparison of different qualitative characteristics in the MQTT, STOMP and AMQP protocols



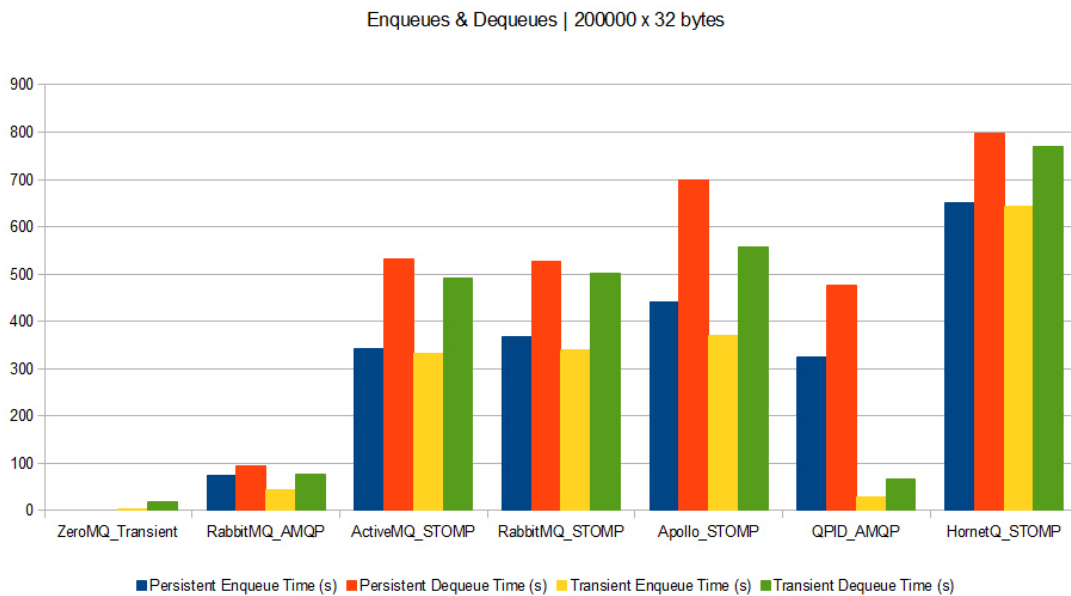


Figure 3.6: In [35] another comparison is made with similar results to Fig. 3.5

### 3.3 Wireless Communication Solutions

In the last three decades a paradigm shift has taken place in computer networking: the advent and popularization of wireless communication technologies. While certain technologies like Bluetooth and Wi-Fi were developed for the consumer market they soon found other applications. In this section we discuss prominent wireless communication technologies such as [Bluetooth](#), [ZigBee](#) and [Wi-Fi](#) and in [Others](#) we mention technologies that, although they aren't especially useful for our needs, we feel are worth mentioning.

The technologies presented here operate in the unlicensed Industrial, Scientific and Medical (ISM) bands. These include the 2.4 GHz band for all technologies presented here, while others can use multiple bands such as 5 GHz for WiFi and 900 MHz for ZigBee.

#### 3.3.1 What is defined by the IEEE

The organization that is responsible for defining and standardizing these wireless technologies is the [IEEE](#) through the [IEEE Standards Association \(IEEE-SA\)](#). All technologies mentioned here have been, to an extent, standardized by this organization under the [IEEE 802](#) standard committee for Local Area Network ([LAN](#))/Metropolitan Area Network ([MAN](#)) connectivity with variable sized packets.

While these technologies are defined by the [IEEE](#), the [IEEE](#) only defines the Physical (PHY) and Media Access Control (MAC) layers as defined by the Open Systems Interconnection (OSI) model (Fig. 3.7), so that these standards can become viable commercial, functioning, products, higher levels of the OSI model must be defined (Fig. 3.8). These are usually defined by company alliances such as the Wi-Fi Alliance and the ZigBee Alliance.

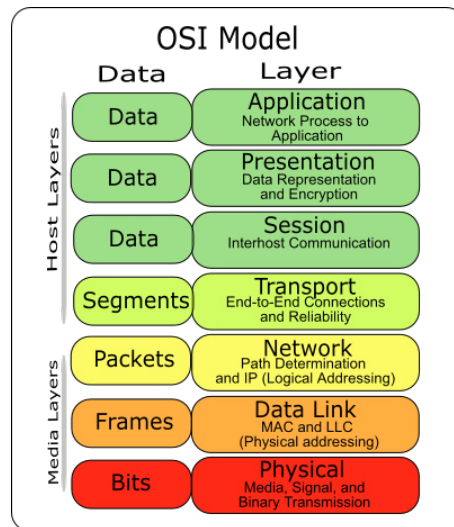


Figure 3.7: The OSI model defines several abstraction layers for communication systems.

IEEE Standard	Technologies
802.15.1 802.15.6	Bluetooth
802.15.4	ZigBee WirelessHART
802.11	Wi-Fi
802.15.3	Ultra-Wide Band (UWB) Radio

Table 3.2: Table of wireless technologies and respective IEEE standards

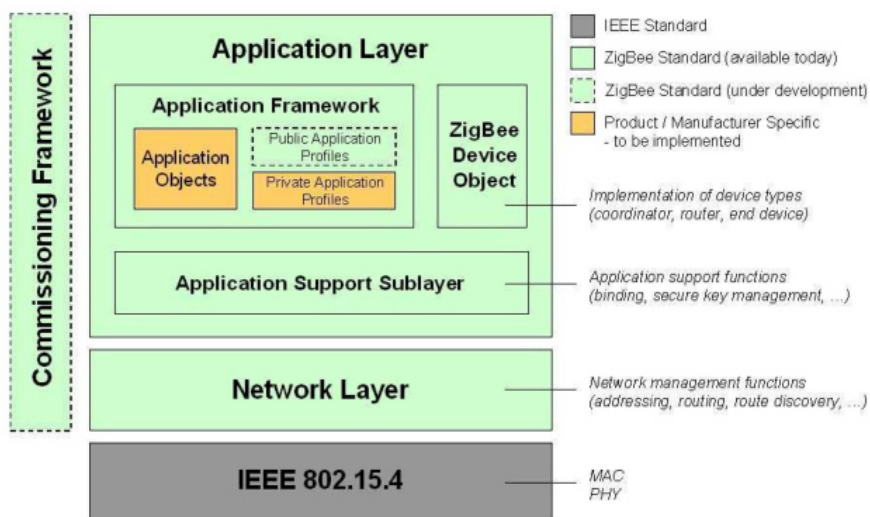


Figure 3.8: The ZigBee protocol stack. Note that it is built on top of the IEEE 802.15.4 MAC and PHY layers[37]

### 3.3.2 Bluetooth

Bluetooth is a technology for **WPANs**. It was originally intended for cable replacement in situations where high data rates were not necessary, such as computer peripherals.

There are two types of networks defined in the Bluetooth standard: the *piconet* and the *scatternet*. In a piconet one device serves as a master and one or more other devices serve as slaves. A device can be a member in several piconets at once thereby promoting a data flow between piconets, forming a scatternet. A Bluetooth device may be a slave in several piconets concurrently but can be master only in one [38, 21].

#### 3.3.2.1 Bluetooth HS/EDR

Bluetooth v2.0 also known as Bluetooth Enhanced Data Rate (**EDR**) was a new version of the Bluetooth Core Specification that was released in 2004. This new version features faster data rates up to 3.0Mb/s with the **EDR** optional feature.

Bluetooth v3.0 also known as Bluetooth High-Speed provides theoretical data rates up to 24 Mbit/s. Unlike v2.0 these data rates are not achieved through Bluetooth but rather Bluetooth is used to send control information while the actual data transfer is done over 802.11. The High-Speed feature in this specification was originally intended for **UWB Radio**[39].

These new specifications also featured other small improvements such as better power control.

#### 3.3.2.2 Bluetooth Smart/BLE

**BLE** marketed as Bluetooth Smart is v4.0 of the Bluetooth standard. It is intended to provide significantly reduced power consumption while maintaining a similar operation range.

This technology was originally introduced in 2006 by Nokia as Wibree [40] and was intended as a low energy competitor for Bluetooth, but it was eventually merged into the new Bluetooth specification.

**BLE** is marketed towards implementations such as beacons, wellness (eg. fitness bands) and home automation [41, 42, 43].

However, **BLE** is not compatible with other versions of the Bluetooth standard, both in terms of protocol as in terms of modulation. Bluetooth devices may implement both versions of the protocol simultaneously in order to ensure compatibility.

### 3.3.3 ZigBee

ZigBee is a specification for low rate **WPANs**. It is intended for simple, low power devices that typically operate in a range of 10m up to 100m depending on transmission power and weather [44].

ZigBee provides, out of the box, self-organized, multi-hop and reliable mesh networking while also maintaining power consumption low. ZigBee specifies two types of network devices: Full Function Device (**FFD**) and Reduced Function Device (**RFD**). **FFDs** may act as Personal Area

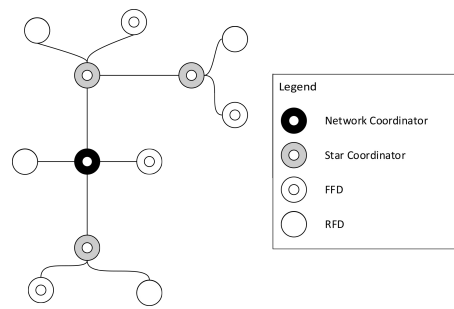


Figure 3.9: An example ZigBee network topology.

Network (PAN) coordinators, coordinators or just devices. FFDs may communicate with all device types while RFDs may only communicate with FFDs. In this way a hierarchical mesh topology is established with FFDs acting as supernodes of sorts (see Section 3.1.3). The RFD implementation is intended for simple devices that only need to send or receive small amounts of information with very little processing involved. It can, therefore, be implemented on very low power, embedded devices. The basic ZigBee cell is the star. In a star network (example topology in Fig. 3.9), one of the FFD devices is chosen to be that star's coordinator. In all the connected stars that make up a ZigBee network, one FFD is chosen to act as overall coordinator [21].

### 3.3.4 Wi-Fi

WiFi was developed, like Bluetooth as a replacement for cables, but, for replacing Ethernet cabling for LANs. It allows for transparent connectivity to upper layers of the protocol stack.

Wi-Fi supports connectivity in two modes: infrastructure and ad-hoc. Infrastructure is the most common: wireless Access Point (AP)s provide wireless access to an underlying cabled infrastructure. In ad-hoc mode, different Wi-Fi enabled clients can connect directly to each other without an intervening AP.

The basic WiFi cell is the Basic Service Set (BSS), which is a set of mobile or fixed stations (eg. a wireless AP and connected clients). On top of this cell two others are defined: the Enhanced Service Set (ESS) and Independent BSS (IBSS). An IBSS is formed when several stations communicate in Ad-Hoc mode without an AP. An ESS is made up of a set of BSSs with an underlying distribution (often wired) network. Essentially the same network is presented via multiple APs [21].

### 3.3.5 nRF24L01+

The Nordic nRF24L01+ is a highly integrated, Ultra Low Power (ULP) 2Mbps RF transceiver Integrated Circuit (IC). It is listed in this paper as a technology because even though it is a hardware solution, it does not conform to any standards imposed by the IEEE for wireless communication and therefore can be considered an independent technology in of itself.

The nRF24L01+ brings some interesting features to the table when compared with other technologies at the cost of mainstream device compatibility.

“The Nordic nRF24L01+ integrates a complete 2.4GHz RF transceiver, RF synthesizer, and baseband logic including the Enhanced ShockBurst™ hardware protocol accelerator supporting a high-speed Serial Peripheral Interface (SPI) interface for the application controller. No external loop filter, resonators, or Voltage-Controlled Oscillator (VCO) varactor diodes are required, only a low cost  $\pm 60$ ppm crystal, matching circuitry, and antenna [45].”

The main advantages of this integrated wireless solution are its ultra low power consumption (approximately 40 mW transmitting (at maximum output power), 50 mW receiving and 10  $\mu$ W in standby), hardware ShockBurst (a Nordic proprietary protocol supporting two-way data packet communication including packet buffering, packet acknowledgement and automatic retransmission of lost packets) permitting the developer to offload some tasks from the MCU.

### 3.3.6 Others

In this section we present other wireless technologies that we feel we should mention because of their characteristics/importance.

#### 3.3.6.1 WirelessHART

WirelessHART is a wireless sensor networking technology that is built on the same IEEE standard as ZigBee but implements the Highway Addressable Remote Transducer protocol (HART) on top. Like HART it is meant as an industrial communication solution. WirelessHART was standardised in 2011 as International Electrotechnical Commission (IEC) 62591, making it the first wireless protocol to become an IEC standard.

#### 3.3.6.2 UWB Radio

UWB radio is a standard for high rate, small range data transmission with a maximum bandwidth of 480 Mbps. With this bandwidth UWB can easily support audio and video streaming and act as a cable replacement for Universal Serial Bus (USB) and IEEE 1394 (FireWire). However due to essentially bureaucratic reasons the standardization process was stalled until the need for the technology diminished and the IEEE 802.15.3 Task Group 3c was put into hibernation in 2009. Still some interest exists in developing this technology further [46, 47, 48, 21].

### 3.3.7 Comparison

In Table 3.3 a comparison of the different wireless technologies is presented.

<sup>1</sup>Maxium TX power is usually only limited by ISM band use legislation and not by the specification. However there may be some implementation specific limitations in some hardware.

<sup>2</sup>No implementation or specification limit imposed, limited only by available resources.

<sup>3</sup>Power consumption depends greatly on implementation and operating conditions. Values presented here are rough estimates from [53, 54, 55] and several datasheets from generic components compatible with these standards and should only serve as a reference relative to one another. These values also refer only to when the device is transmitting.

	Bluetooth	ZigBee	Wi-Fi	nRF24L01+
Standard	IEEE 802.15.1	IEEE 802.15.4	IEEE 802.11	—
Frequency Band(s)	2.4 GHz	2.4 GHz and 780/868/915MHz	2.4 GHz and 5 GHz	2.4 GHz
Maximum Signal Rate	2.1 Mbit/s or 100 kbit/s (for BLE)	250 kbit/s	54 Mbit/s up to 600 Mbit/s	670 kbit/s[49]
Range	10 m	10 m - 100 m	100 m	100 m[50]
Nominal TX power <sup>1</sup>	-100 dBm to 20 dBm	-3 dBm to 20 dBm	0 dBm to 20 dBm	-18 dBm to 0 dBm
Basic cell	Piconet	Star	BSS	—
Extended cell	Scatternet	Mesh	ESS	—
Maximum number of cell nodes	8	65536	— <sup>2</sup>	—
Encryption	E0 stream cipher	AES block cipher	RC4 stream cipher, AES block cipher	—
Authentication	Shared secret [51]	CBC-MAC	WEP/WPA/WPA2	—
Data integrity fields	16 bit CRC	16 bit CRC	32 bit CRC	8 or 16 bit
Average Power Consumption <sup>3</sup>	<30 mW	<60 mW	<2 W	<40 mW

Table 3.3: Comparison of different wireless technologies' characteristics [52, 21]

Since with this system supporting smartphones without additional hardware was a great advantage, BLE was chosen.

### 3.4 Hardware Platforms

In this section we present multiple hardware platforms that can be used to implement the systems that have been mentioned so far, either as a sole solution or used in combination.

#### 3.4.1 Android Smartphone

Android is a mobile operating system based on a customized version of the Linux kernel and developed by Google. It is primarily designed for touchscreen input mobile devices such as smartphones and tablets but also powers other types of devices, such as TVs, watches, game consoles, PCs and others. Android is developed privately by google and is released as open-source code [56].

Android is usually deployed on platforms with the ARM architecture, with x86 and MIPS also being supported. As of Android 5.0 64 bit versions of these architectures are also supported.

Android supports multiple external components, with sensors such as accelerometers, gyroscopes, barometers, proximity sensors being specifically relevant to our interests.

Android applications are usually written in Java (although there exists some limited support for apps written in C), packaged as Android application Package (APK)s and run on top of a virtual machine: Dalvik or Android RunTime (ART) [57].

#### 3.4.2 Raspberry Pi

The Raspberry Pi is a series of cheap credit card-sized single-board computers developed by the Raspberry Pi foundation for teaching purposes. Due to the board's low price (aimed at 35 USD) and versatile hardware it has found numerous other applications from home media to embedded systems. The Pi is based around an ARM System on (a) Chip (SoC) and is capable of running anything that is usually released for that architecture, this includes most Linux operating systems, Android, and as of the Raspberry Pi 2, Windows 10 [58].

The Raspberry Pi has hardware more than capable of running most IoT related software. The B+ model boasts a 700 MHz processor, 512 MB of RAM. It also has a rather powerful dual-core Graphics Processing Unit (GPU), capable of decoding FullHD H.264 encoded video [59, 60]. Unfortunately, Wi-Fi connectivity is only supported via USB dongle.

Recently the Raspberry Pi Compute Module (Fig. 3.10) has been launched. In the form factor of a 200-pin DDR2 SO-DIMM memory module it is intended to be used as a part of other products by providing all the functions of a normal Raspberry Pi in a easy to integrate package [61].

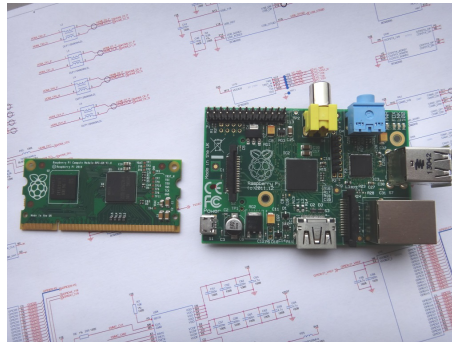


Figure 3.10: A Raspberry Pi Compute Module (left) and a Raspberry Pi model B (right) [61]

### 3.4.3 SheevaPlug

The SheevaPlug (Fig. 3.11) is a plug computer developed by the Marvell Technology Group. The computer itself is in the form factor of a common pluggable AC-DC adapter. SheevaPlug uses an ARM compatible SoC produced by Marvell Semiconductor with a 1.2 GHz Central Processing Unit (CPU), 512 MB of flash memory and 512 MB of DDR2 RAM. Similarly to the Raspberry Pi, the SheevaPlug offers Gigabit Ethernet connectivity and USB 2.0. Yet, it's a bit more costly, with the SheevaPlug Dev kit being sold for 99 USD [62, 63].



Figure 3.11: The Marvell SheevaPlug computer with I/O ports showing [64]

### 3.4.4 TelosB

The TelosB is an Open-Source platform developed by MEMSIC essentially for the research community. It bundles, in a very small package, USB programming capability, an IEEE 802.15.4 radio

with integrated antenna, a low-power **MCU** (8 MHz Texas Instruments (**TI**) MSP430 microcontroller) with extended memory and an optional sensor suite. The TelosB is made to be powered with two AA batteries [65].



Figure 3.12: In Fig. 3.12a the TelosB module is shown, and its respective block diagram in Fig. 3.12b.

### 3.4.5 TI CC2540

The **TI CC2540** is a low cost, low power, **SoC** chip developed for **BLE** applications. It combines an RF transceiver with an 8051 **MCU** in a single package. Integrated in a Printed Circuit Board (**PCB**) with other components and connectors it can prove a very good solution for low power sensors and actuators that are in sleep mode most of the time [66].

### 3.4.6 Nordic nRF51822

The Nordic nRF51822 is a powerful and flexible **SoC** by Nordic Semiconductor for **BLE** solutions built around an ARM Cortex-M0 **MCU** and featuring a 2.4 GHz transceiver designed for the **ISM** band. It supports several analogue and digital peripherals though 31 General Purpose Input Output (**GPIO**)s. These peripherals can communicate with each other without **CPU** intervention though the use of a Programmable Peripheral Interconnect (**PPI**) system. A specific version of the **MCU** that is available to us, the nRF51822AA has a total of 256 kB of flash and 16 kB of **RAM**. This **SoC** is compatible with Nordic's Gazelle protocol, ANT protocol and the Bluetooth Smart protocol [67].

## 3.5 Indoor Location Solutions

Indoor location has been a topic of discussion in academia for quite some time without a conclusion being made as to a good solution for the problem. Global location systems such as **GPS** aren't accurate enough or simply not practical in most situations (**GPS** requires line-of-site to work), so, new specific solutions have to be developed. There is also an interest that these new solutions rely



on existing, and commonplace systems, so that new investment to obtain new functionality is kept to a minimum. In this section we discuss several proposed solutions for this problem.

### 3.5.1 BLE based

Although Bluetooth was originally intended for low bitrate communication it can be applied for localization purposes.

The Bluetooth specification forces each Bluetooth-enabled device to report Received Signal Strength Indicator (**RSSI**) to upper protocol layers for power-control purposes, but this information can be used for location through trilateration. This process has two main disadvantages:

1. So that the **RSSI** value can be used for trilateration, each node participating in the location process needs to be transmitting at full power. This has two more implicit disadvantages:
  - (a) For the node to be constantly transmitting at full power it must be in discovery mode, and therefore cannot be simultaneously used to transmit data.
  - (b) Increased power consumption.
2. Since there is no clock synch mechanism between Bluetooth nodes, measurements used for trilateration may not have been made at the same time. This introduces errors in the measurement, especially if the target is moving [68]. The main advantage of this technology is that it can be found in most consumer-grade electronics equipment that is capable of communication.

In [6] the use of this technology is explored for a location-aware sign on system, in [69] a guide application was developed based on this technology and in [70] an application for cat tracking is developed to a certain degree of success.

### 3.5.2 ZigBee based

The ZigBee technology is similar to Bluetooth and most location solutions hover around the same processes proposed in Section 3.5.1, and use the **RSSI** indicator.

However, some different interesting approaches have been developed for this technology. In [71] and in [72] two different algorithms have been proposed that increase effectiveness in location systems. In [71] the tree network topology for ZigBee is exploited to allow for a statistical maximum-likelihood location approximation. In [72] the implemented algorithm divides the space in between different ZigBee beacons in different zones that can, themselves, be divided into sub-zones. The algorithm predicts in what zone the node is more probable to be located. Location errors obtained were generally less than 2 m.

### 3.5.3 Wi-Fi

Similarly to the other alternatives presented here, Wi-Fi location techniques revolve around the same method of trilateration based on **RSSI** and the placing of beacons. However, due to the

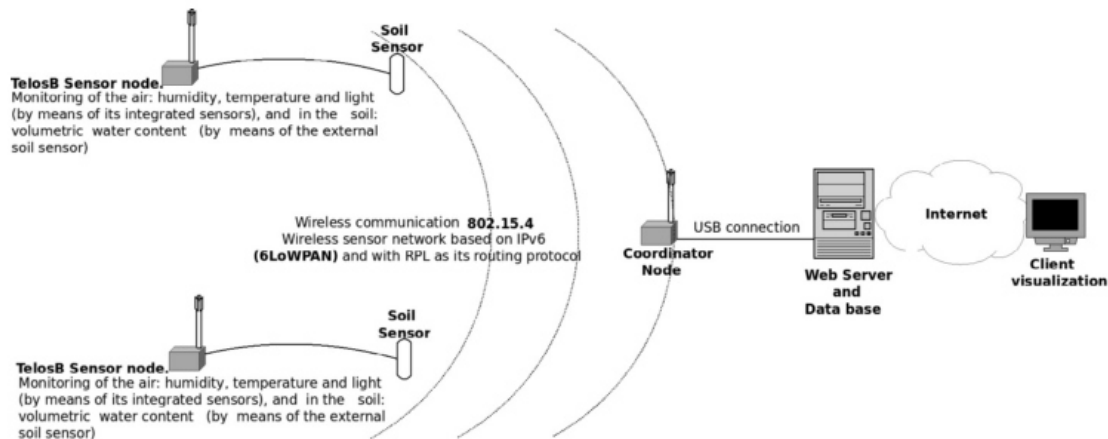


Figure 3.13: System architecture proposed in [5]

nature of the technology and its higher transmission power (typically 20 dBm) it is more prone to be affected by multipath propagation and reflection, so huge errors are typically associated with the use of Wi-Fi for location. Therefore, typically Wi-Fi location systems resort to some kind of geotagging system: when the client connects to a different AP it is assumed to be in a certain place [73].

## 3.6 Applied Research

In this section we present some practical, working, systems that were implemented in real-world scenarios leveraging some of the technologies that were exposed in this chapter.

### 3.6.1 Health sensor systems

This type of system has been applied for health purposes to some extent. In [9] a system to integrate multiple health sensors by creating a rule engine was proposed. In [74] a health monitoring system is fabricated by integrating sensors and transducers into textiles, offering a less intrusive way to collect this type of data.

### 3.6.2 Agriculture

In [5] (see system architecture in Fig. 3.13) a sensor network was developed using the ZigBee protocol along with the Sheevaplug and TelosB hardware platforms. The purpose of this product was wireless monitoring of greenhouses for agricultural purposes. The authors were able to provide an interesting and effective solution, being able to squeeze up to 53 weeks (one year) of battery powered node operation. The companion web and mobile apps that were developed offered a simple interface to the farmers that ran these greenhouses, helping with their management.

### 3.6.3 Ambient Assisted Living

There has been a lot of research interest in the EU, in part due to the Ambient Assisted Living Joint Programme: an ongoing funding effort running since 2006. Some notable projects include:

eCAALYX, a project to enable a commercially viable solution to permit remote monitoring of elderly by healthcare professionals and caretakers [1, 75].

The Cockwork project used a number of hardware and software platforms, including a WSN with sensors and actuators and was targeted mainly at shift workers that were greatly affected by chronodisruption. The project's objective was to help users maintain a healthy day-night cycle by manipulating their environment [76].

ALMA was a project with the objective of providing a modular, low-cost, integrated system to support autonomous mobility and orientation for elders [77].



## Chapter 4

# Development tools

In this section we present the various tools used to develop our proposed solution.

### 4.1 Fraunhofer Pandlet

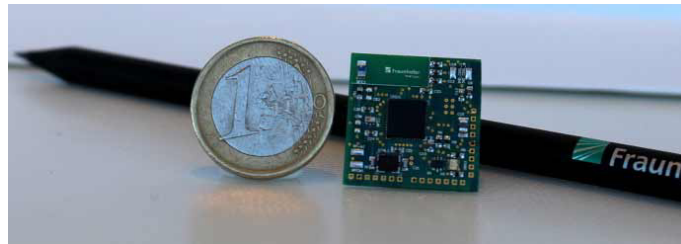


Figure 4.1: The Fraunhofer Pandlet CORE (right, square, 28.4 mm wide). 1€ coin (left) for reference.

The Fraunhofer Pandlet is an integrated sensor and processing platform still in active development at Fraunhofer Portugal, based around the [Nordic nRF51822](#).

The modules created with the pandlets are composed of several building blocks, that can be interconnected in order to provide for extra hardware functionality. The base building block is called the pandlet CORE and includes, in a small package (Fig. 4.1) a Nordic nRF51822 SoC with a Bluetooth 4.0 interface and 16 MHz ARM M0+ CPU; support for Qi wireless charging and Inertial (IMU) and Environmental (EMU) Measurement Units. The IMU contains an accelerometer, a gyroscope and a magnetometer. The EMU contains a humidity sensor, an air pressure sensor and a temperature sensor.

In addition to the CORE module, currently there are two more modules available: the Memory module and the Sensing+ module (Fig. 4.2). These modules extend the capabilities of the original CORE module. The Memory module adds a Micro USB port for wired charging and a Micro SD card slot to allow for large quantities of local, persistent storage. The Sensing+ module allows for the connection of extra external sensors through various ports, including two BNC connectors.

The sensor's outputs can be connected directly to one of the **MCU's GPIOs** for direct reading or for **I<sup>2</sup>C** communication or to an external **Analog-to-Digital Converter (ADC)**.

With further development of the platform, new form factors and blocks may be developed.

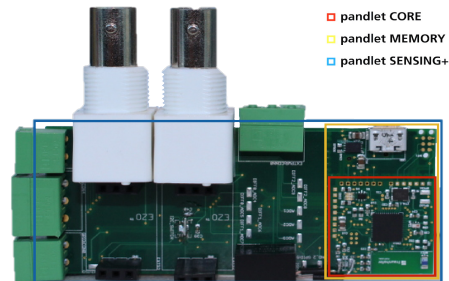


Figure 4.2: The Fraunhofer Pandlet, complete with Memory and Sensing+ modules.

## 4.2 Development Boards

### 4.2.1 PCA10001

The PCA10001 (Fig. 4.3) is a MBED-enabled development board distributed by Nordic Semiconductor. MBED is a platform and operating system designed for fast development and prototyping of **MCU** platforms, namely Cortex-M 32 bit **MCUs** such as the **Nordic nRF51822**. MBED provides **APIs** to abstract differences between supported **MCUs** to make sure that most things work between platforms out of the box. Although the preferred way to program for MBED platforms is with an online Integrated Development Environment (**IDE**) and compiler provided by MBED, it is also compatible with offline toolchains. We will use the GNU ARM GCC toolchain to develop our application for this board. After the application is successfully compiled, one can simply connect the PCA10001 board to a computer and it will be identified as a thumb drive. The **MCU** can then be programmed by dragging the resulting binary file into the thumb drive. This is an advantage over traditional programming since no debugger/flasher is necessary, however, despite supporting communication with the **MCU** with Universal Asynchronous Receiver/Transmitter (**UART**) over **USB**, MBED does not support on-chip code debugging [78, 79].



Figure 4.3: Nordic's PCA10001 development board [79]

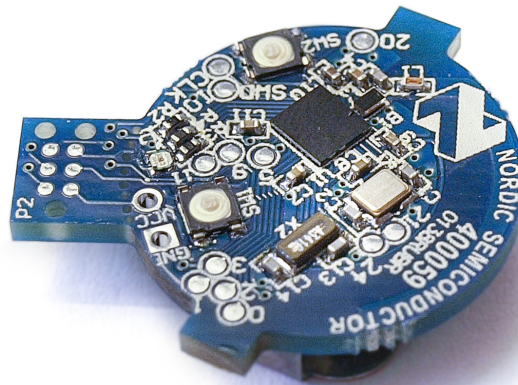


Figure 4.4: Nordic’s PCA20006 development board

#### 4.2.2 PCA20006

The PCA20006 (Fig. 4.4) is a development board that, similarly to the PCA10001, is built around the Nordic nRF51822. However, it is much smaller, and is not MBED-enabled. As such, it must be programmed using a Joint Test Action Group (JTAG)/Serial Wire Debug (SWD) emulator such as, and in our case, the SEGGER J-Link EDU.

### 4.3 The Softdevice

The Nordic protocol stacks are called *Soft Devices*.

Nordic SD are pieces of pre-compiled, pre-linked software that implement the required protocol stacks, in this case Bluetooth Low Energy. They give the developer the important ability to not have to implement a full protocol stack and interact with proprietary technology. The SD implements the full Bluetooth stack, so that the developer doesn’t have to. The SD’s functions are accessed through a pure C API. The SD is flashed onto the microcontroller alongside the application and at compile time the linker generates jumps to positions in memory where SD instructions are located.

This SD does have its disadvantages. Firstly, it is proprietary, so a developer that relies on it is completely reliant on Nordic Semi for support. Secondly, since it requires the use of pre-compiled binary files, it forces the developer to use the same toolchain that Nordic initially used to produce the binary files, or one will experience linker errors and warnings. Using the SD also severely limits the available resources that the developer has available. In our case, the SD requires half of all the available flash and memory. 1500 B are also used by the SD for its call stack in the RAM region addition to 500 B more used by the application call stack. This means that by using the SD one effectively gets restricted to 128 kB of memory and about 4 kB of RAM [80, 81]. Despite these disadvantages we feel that using the SD is more than worth the risk.

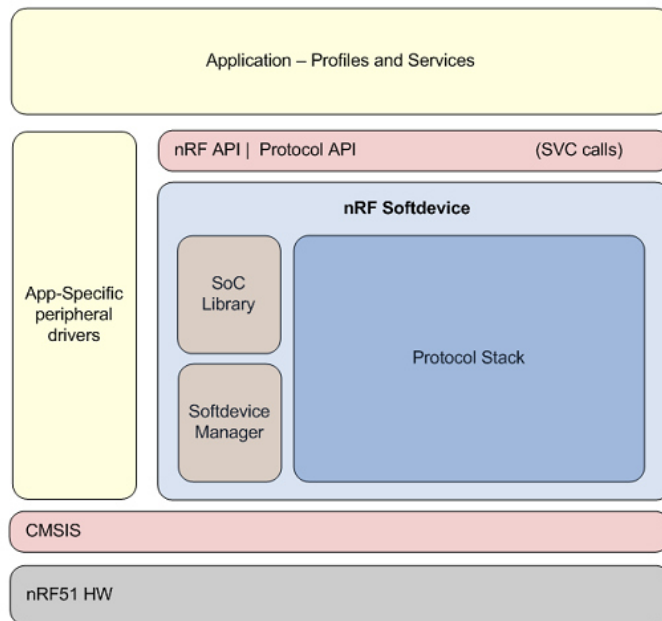


Figure 4.5: An example of the full software stack running on the **MCU**, featuring the **SD** programmed alongside the application [80]

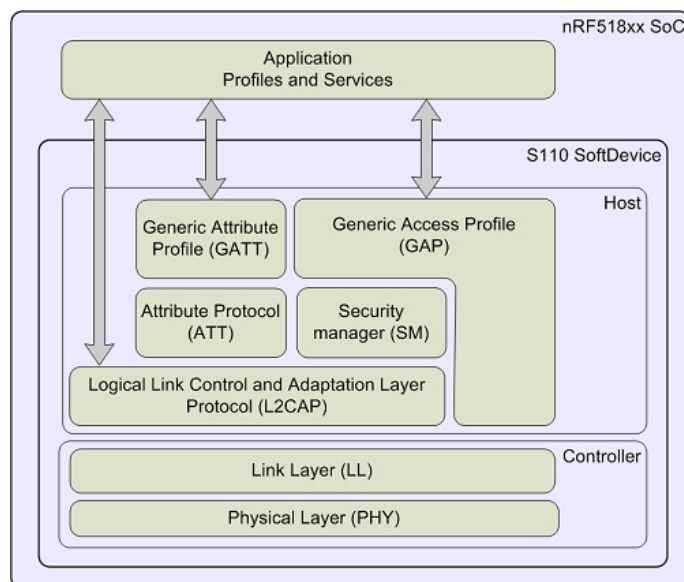


Figure 4.6: The Bluetooth Smart stack that is implemented by the **SD** [80]





Figure 4.7: An example of an IoT network communicating by using BLE links with 6LoWPAN support [82]

Nordic's BLE SDs come in several flavors: The S100 series and the S300 series SDs. The S100 series SDs only implement a BLE stack; The S110 SD can communicate in a peripheral role and the S120 SD can communicate in a central role, while the S130 SD can switch between those two modes of operation. The S300 series SDs, containing only the S310 SD implement BLE in both modes of operation and also the ANT protocol (proprietary) concurrently.

For our specific needs, the S110 SD was chosen, due to its reliability and simplicity.

## 4.4 nRF51 SDK

A Software Development Kit (SDK) is a set of tools that allow for the development of applications for a certain platform. The nRF51 SDK is one such set of tools that is provided by Nordic Semiconductor to developers using their series of nRF51 SoCs. Besides containing all the header files and such that are required for application development for that specific platform using BLE, it also contains relevant SDs, documentation and examples for that specific technology and also other protocols that can be used with this platform that are proprietary to Nordic, mainly featuring low power, simple types of communication.

## 4.5 IoT SDK

The IoT SDK is a prototype (not yet production ready), modified, version of the nRF51 SDK that is capable of communicating via IP version 6 (IPv6) by implementing the 6LoWPAN standard proposed by the IETF. The 6LoWPAN standard proposes encapsulation and header compression mechanisms that allow IPv6 packets to be transferred in IEEE 802.15.4 networks such as BLE and ZigBee. The purpose of this standard is to allow the most low powered devices to be addressable by IPv6, giving them access to the broader Internet and to the Cloud [83]. The IoT SDK also includes a complete Internet Protocol Suite including Internet Control Message Protocol (ICMP), User Datagram Protocol (UDP), TCP, Constrained Application Protocol (CoAP) and MQTT protocols, and examples on how to develop applications using them. Devices using this SDK implement a Bluetooth profile designed specifically for this type of communication, and it is through this profile that packets are transferred between the IoT device and another, WPAN-enabled device, connected to the Internet, that will act as an IPv6 access point and/or router. The SDK also

contains instructions on how to set-up a Raspberry Pi to act as a router, and a pre-compiled kernel that has to be enabled on the Pi to enable [6LoWPAN](#) support [84].

The [IoT SDK](#) available when development started for this solution was v0.7.0, and due to numerous [API](#) and structure differences between versions, it was retained even after Nordic released v0.8.0 of the [SDK](#).

#### 4.5.1 The LWIP IP stack

The Lightweight IP ([LwIP](#)) protocol stack is a very small [TCP/IP](#) stack aimed at being deployed in ultra low power embedded systems originally developed by Adam Dunkels and now maintained as an open source project with a modified Berkeley Software Distribution ([BSD](#)) license. Nordic ported this stack for the [nrf51822 MCU](#) and distributed it with the [IoT SDK](#).

[LwIP](#)'s main features include [85]:

- **Protocols:** Internet Protocol ([IP](#)), [ICMP](#), [UDP](#), [TCP](#), Internet Group Management Protocol ([IGMP](#)), Address Resolution Protocol ([ARP](#)), Point-to-Point Protocol ([PPP](#)) over Serial ([PPPoS](#)), [PPP](#) over Ethernet ([PPPoE](#)), Dynamic Host Configuration Protocol ([DHCP](#)) client, Domain Name Service ([DNS](#)) client, AutoIP/Automatic Private IP Addressing ([APIPA](#)) (Zeroconf), Simple Network Management Protocol ([SNMP](#)) agent (private Management Information Base ([MIB](#)) support)
- **APIs:** specialized [APIs](#) for enhanced performance, optional Berkeley-alike socket [API](#)
- **Extended features:** [IP](#) forwarding over multiple network interfaces, [TCP](#) congestion control, Round Trip Time ([RTT](#)) estimation and fast recovery and retransmission

As is mentioned in their website [85] and as one can deduce by the list of features, one of the main objectives of [LwIP](#) is to provide a *complete* [TCP/IP](#) stack while making it as resource-efficient as possible.

The [LwIP](#) stack is modular and features can be added/removed as required. The first version of the [IoT SDK](#) released by Nordic did not support, for example, [DNS](#) and [SNMP](#). These features were, however, added in a subsequent release.

Use of these functions in the [MCU](#) requires one to part with use of about 60 kB of flash memory (Fig. 4.8).

## 4.6 Other software

In this section we mention software that, though not especially fundamental to this solution, was nonetheless used.

The fundamental programming (and scripting) languages used were C (gnu99), Java (Android) and Python. Bash scripting was fundamental in automating repetitive tasks.

The [IDEs](#) used were Eclipse by the Eclipse Foundation, Android Studio and Android [SDK](#) by Google and PyCharm Community Edition by JetBrains.

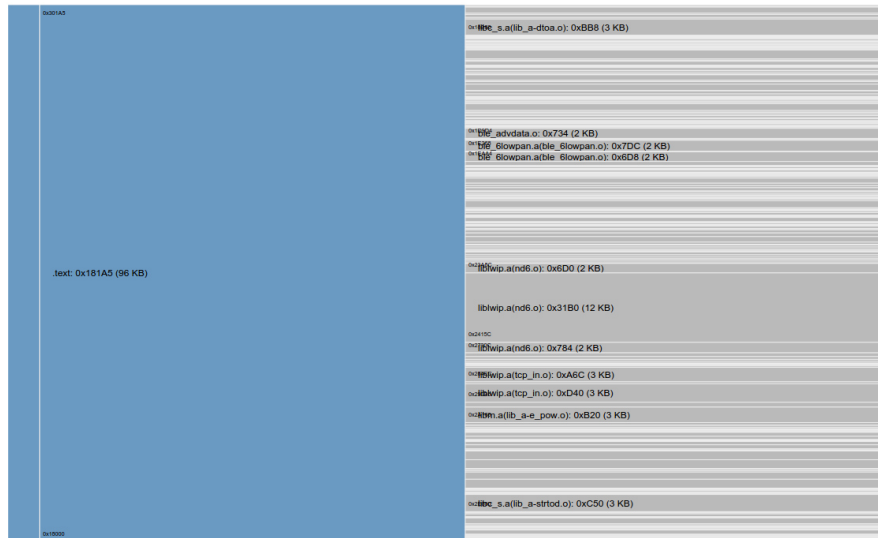


Figure 4.8: A graphical representation of a memory map file generated by [86] from a map file produced by the linker used in our solution. As one can see, the biggest slices of memory being used belong to **LwIP** (e.g. liblwp.a, 12 kB).

SEGGER’s J-Link EDU debugger was used for **SWD** on-chip software debugging.

GNU make and the GNU arm toolchain were essential for compiling and flashing the application onto the **MCU**.

RabbitMQ was run on a Raspberry Pi computer, with its **MQTT** plugin to provide an **AMQP** and **MQTT** broker, along with the BlueZ Bluetooth stack and a Linux kernel with **6LoWPAN** support provided by Nordic to enable the **MCU** to communicate with the Raspberry Pi over **BLE**.

An Android app by Nordic Semiconductor, nRF Master Control panel, was used to connect to the Pandlet and debug **BLE** characteristics and services.



## Chapter 5

# The BlueWarnAAL solution

In this chapter we present the BlueWarnAAL solution. In Section 5.1 the whole designed system is broken down into its building blocks and each one of them explained. In Section 5.2 each of the solution's components are explained more in-depth.

This solution allows for a user to load rules onto a computing node based on the Fraunhofer Pandlet, and for those rules to be evaluated and processed dynamically. Different actions can be taken depending on the defined rule and on the current context.

### 5.1 The solution broken down

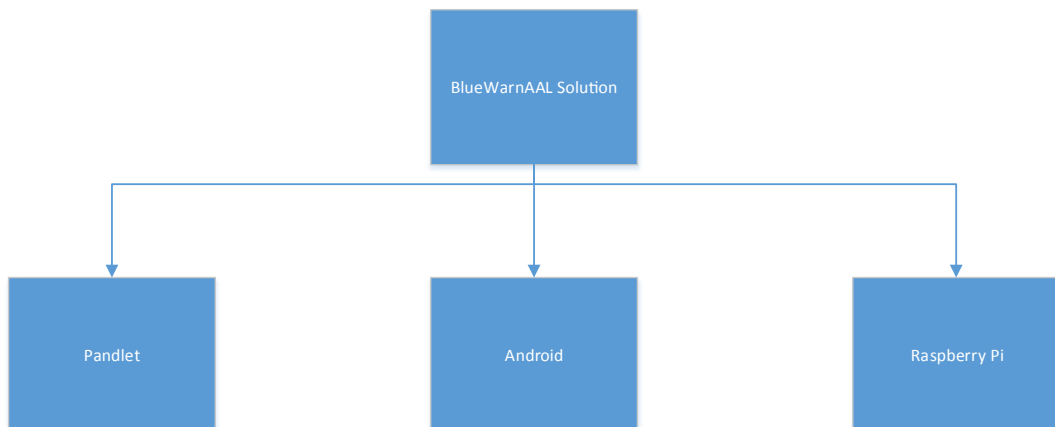


Figure 5.1: Block diagram of the solution's three main components

The solution can be broken down into three macro components: the Pandlet application, the Android application and the Raspberry Pi application that are explained in more detail in Section 5.2.

The Pandlet is the base of the system. It is the part of the system that contains the rules, processes them and then applies the results to the outputs. It also estimates the distance to a client based on the **RSSI** from the **BLE** link and offers this distance as an input to the rule engine.

The Android application enables the smartphone to connect to the Pandlet and start the ranging process. It also presents a graphical frontend for configuration by the user. It uses Fraunhofer Portugal's Smart Companion library for design to make the application more intuitive and the UI more accessible for older people [87]. The Raspberry Pi application is basically a script that searches for BlueWarnAAL Pandlet nodes and instructs the BlueZ stack to connect to them. This brings up a Bluetooth interface in the Raspberry's IP stack and enables IP communication.

The described system can best be visualised by the examples provided in Fig. 5.2; Here, an example rule is configured, and is triggered when the user gets too far away from the stove. The context of the situation is inferred from the distance information.

The wireless technologies used between network nodes are represented in **fig:wirelessconnections.png!** (**fig:wirelessconnections.png!**).

## 5.2 Components

### 5.2.1 FhP Pandlet developments

The Pandlet application is also made up of three different components (Fig. 5.4): The rule engine, the BLE stack services and characteristics, and the RSSI distance engine.

#### 5.2.1.1 BLE Stack

**Characteristics** In order to logically group all the characteristics that were required for our solution a new BLE service was created with a custom UUID to differentiate it from Bluetooth Special Interest Group (SIG)'s reserved UUID.

In our service there are four types of characteristics defined. For each of these characteristics only one example characteristic was created in our prototype. A description of what the characteristic signifies is stored in the "Characteristic User Description" field. These characteristics expose device functionality to the Android application.

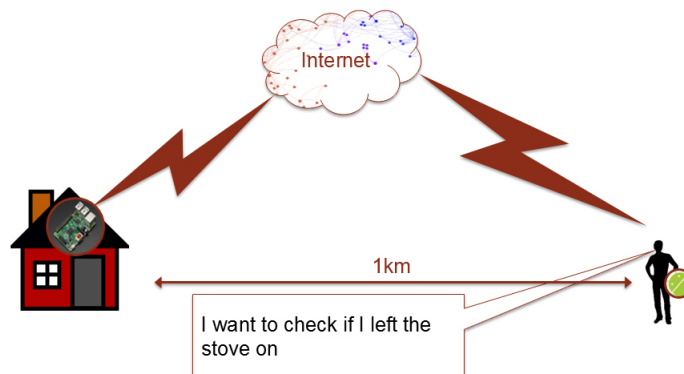
- **User Configuration Characteristic:** This type of characteristic is used to store values that are used in the configuration of the device. In our case, the characteristic selected was named "Device\_Name" and its value is used as the device name in advertisements.
- **Input Characteristic:** These characteristics represent inputs. These inputs can be received directly from external sensors (e.g. a voltage level from a photodiode) or can be from processed values. In our case, the characteristic is named "Distance" and its value represents the distance to the currently connected device. If no device is currently connected the characteristic has the value 255 (maximum value that can be represented in one byte).
- **Output Characteristic:** These characteristics are similar to Input Characteristics, but are tied directly to outputs. Our characteristic is named "Actuator\_Value" and represents the current (boolean) value of a GPIO output on the development board. That GPIO is connected



(a) The user is 5 meters away from the stove, the actuator is off



(b) The user is 21 meters away from the stove, the output from the rule is now false and the stove is turned off



(c) The user is away from home and wants to check the status of the system

Figure 5.2: Example diagrams of typical situations in which the BlueWarnAAL system would work, in Fig. 5.2a and Fig. 5.2a the configured rule is  $Distance < 20$

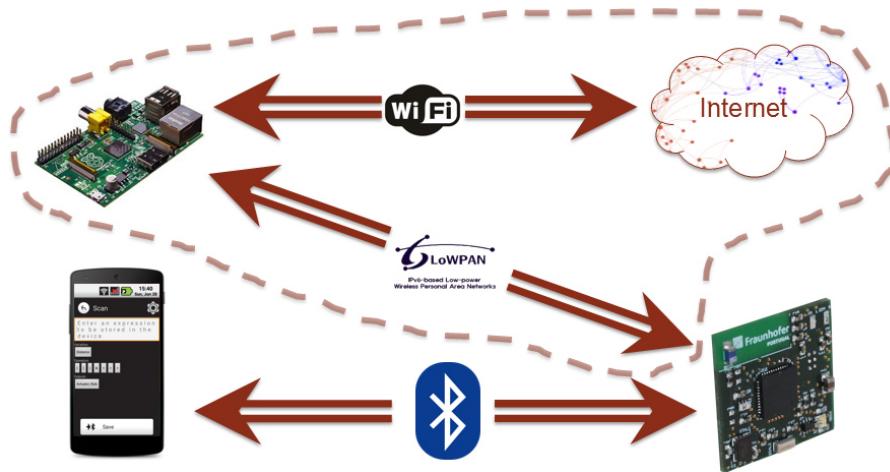


Figure 5.3: The different wireless technologies used for node to node communication. Between the Pandlet and the smartphone, BLE is used, for the connection between the Pandlet and the Raspberry Pi 6LoWPAN and for the Raspberry Pi connection to the internet, Wi-Fi is used.

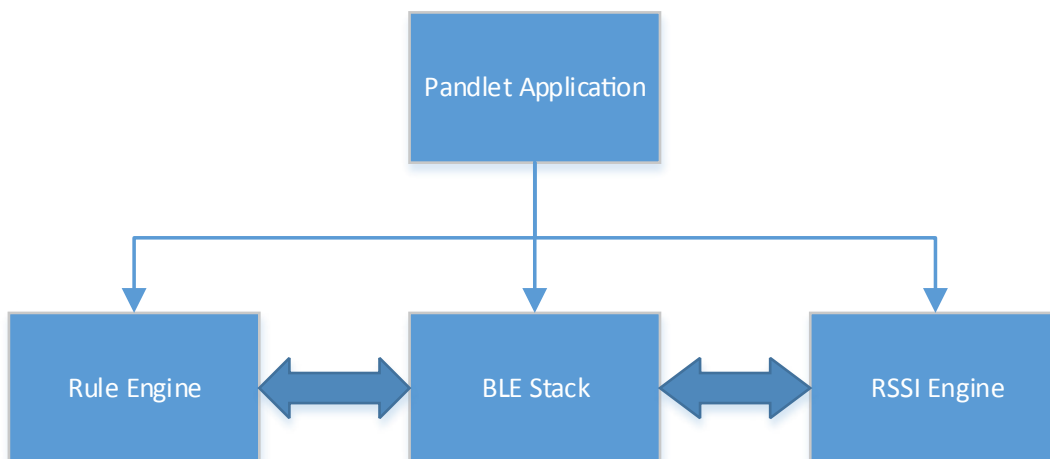


Figure 5.4: Block diagram of the Pandlet's three components



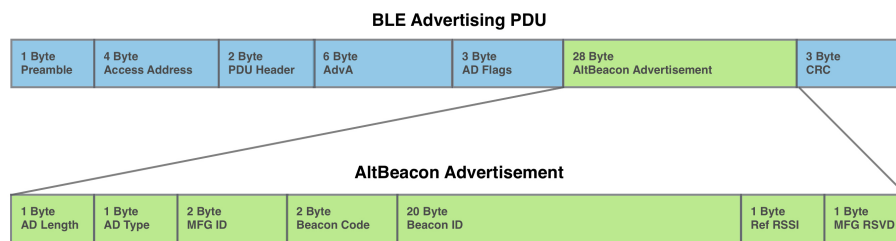


Figure 5.5: Exploded view showing the structure of an AltBeacon advertisement packet [89]

to an LED to provide a visual representation of the characteristic value. These characteristics can also be used as Input Characteristics, fostering composition. These Characteristic User Descriptions must be a name and suffixed with "\_Value" (e.g. Actuator\_Value or LED\_Value).

- Rule Characteristic:** This is where the actual rules are stored. The rules are stored as UTF8 strings and must be evaluable to boolean values. Rule characteristic's Characteristic User Descriptions must follow the following convention: the user Description of the Output Characteristic corresponding to this rule must be suffixed with "\_Rule" (e.g. TV\_Rule or Chandelier\_Rule). Our characteristic was named "Actuator\_Rule" and by default contained the value "Distance < 10".

**Advertising** In order to enable for future use of the Pandlet as a Beacon, for ranging purposes from a smartphone, its advertisement packets were made to conform to the AltBeacon specification. The structure of an advertisement packet can be seen in Fig. 5.5. Since the fields required by the specification are 31 B one cannot send any more information in this advertisement packet[88]. It was opted to include in the advertisement the device's name, so that the user could configure friendlier and meaningful names (e.g. "Living Room Pandlet") and have them appear in Bluetooth device searches. For this, the Scan Response, a packet of extra advertisement data sent to the scanner after a scan attempt, was used.

### 5.2.1.2 Estimating Distance

The Bluetooth standard calls for the RSSI to be passed to the application from the Bluetooth stack. In our solution this information is used for ranging. According to [90], RSSI varies with distance with a relationship expressed by 5.1, where  $d$  is the distance,  $n$  is the signal propagation constant (free space value is 2) and  $A$  is the average RSSI as measured at 1 m from the emitter (−61 dBm for the Nordic nRF51822).

$$RSSI_{(d)} = -(10 \cdot n \cdot \log_{10}(d) + A) \quad (5.1)$$

By solving 5.1 for  $d$ , we get 5.2. The IoT SDK allows the developer to implement a handler function that is called every time that the RSSI value in the current link changes. We leveraged

this function to estimate a new distance value when the **RSSI** value changed on the current link. However, we found that this **RSSI** value changed much too often and with values too large to be used as a reliable measure of distance all by itself: for example, the distance value estimated from this **RSSI** would often, in the space of just 10 s change in excess of 20 m.

$$d_{(RSSI)} = 10^{\left(\frac{A-R}{10n}\right)} \quad (5.2)$$

The chosen solution was to implement a rolling average: Whenever a new **RSSI** value was read, it was inserted in a circular buffer large enough to hold about 50 samples. The average value of all these samples was then used to estimate the distance. This meant that sudden variations in reported **RSSI** took longer (a few seconds) to produce visible effects in the distance estimation, however, sudden changes in **RSSI** were effectively smoothed out.

### 5.2.1.3 Implementing a Rule Engine

One of the challenges of this solution was to allow rules present on the Pandlet to be modified at-will by the user. While there is no lack of solutions for this problem that work in average, modern systems, in our specific case this is non-trivial. For one to evaluate rules in run-time, one has to be able to receive a mathematical expression in string form at run time (e.g.  $2 > 1$ ) and evaluate it into a result (e.g. *true*), this is beyond the standard capabilities of the C programming language. One solution to overcome this problem is usually embedding another programming language in our program. Languages such as Lua and Tcl provide for these situations by offering **APIs**, written in C, to allow for programmers to embed them. When these languages are embedded, they essentially act as virtual machines, running on top of the memory and resources allocated to the underlying application. This would allow us to simply pass an expression to be evaluated to the embedded language's interpreter. However, either these languages were easily embeddable and their requirements surpassed the nRF51822's capabilities, or the process of porting them was far too complex. The chosen solution was to use a small number of efficient, tiny libraries that, put together, constituted a, somewhat limited, but functional, interpreter.

Having the "intelligence" of the system in the **MCU** gives us several advantages. The increased battery life that comes with using **BLE** for communication mainly stems from the fact that it usually communicates at low power and for limited amounts of time. **BLE** devices can go for weeks or even months without establishing a connection, which is the most taxing in terms of energy use when compared with undirected advertisements. By having all of the processing done locally on the **MCU** we are, in fact, conserving battery life, since the Android smartphone needs only to be connected for configuration and ranging purposes. This also allows the system to work "offline". Since the smartphone doesn't need to be connected for data processing to happen, this allows the **MCU** to do its job without having a connection active either to a central server or to the smartphone. This allows for a "fire and forget" kind of operation, where one leaves rules and simply allows the **MCU** to apply them. This, together with **IP** connectivity, gives this system great flexibility.

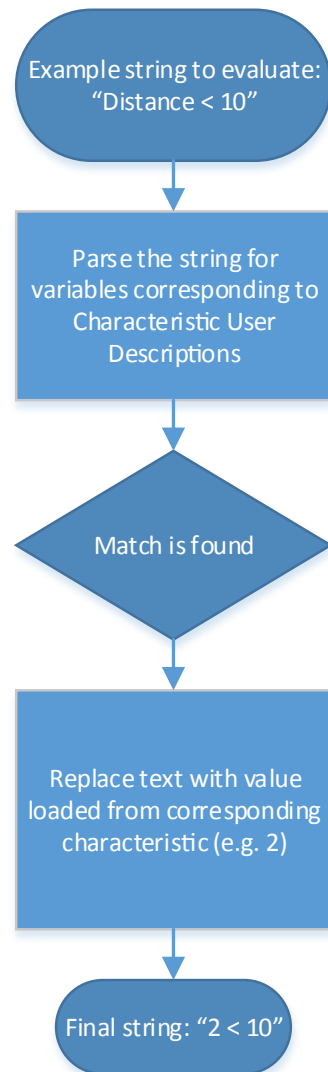


Figure 5.6: First stage of interpreting a rule

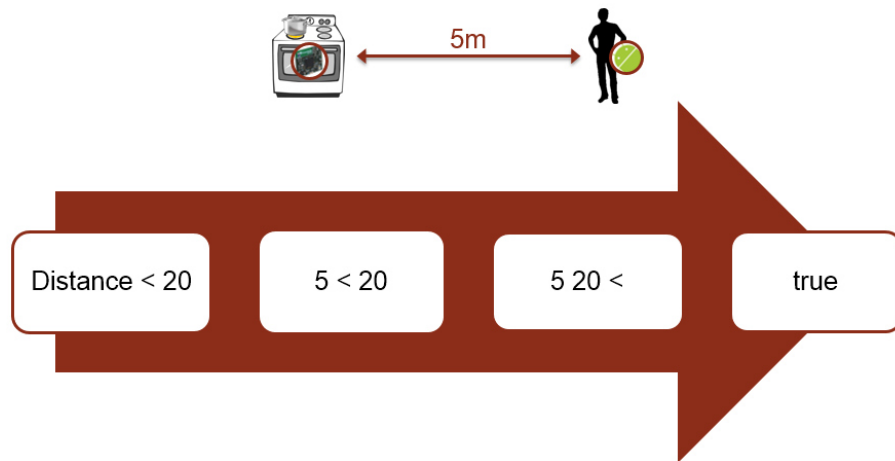


Figure 5.7: Flowchart of the different stages of rule processing for an example situation where the user is 5 m away from the Pandlet.

The different steps of rule processing are presented in Fig. 5.7. The first stage of processing a rule (e.g.  $Distance < 10$ ), is to find variables, i.e. substrings, that match Characteristic User Descriptions, and replace those substrings with the current values of the corresponding characteristics. A diagram corresponding to this process can be consulted in Fig. 5.6 and the matter is discussed more in-depth in Section 5.2.1.4. The second stage is to convert the resulting mathematical expression from infix notation (common arithmetic expression notation) into postfix notation or Reverse Polish Notation (RPN). This is necessary because it is far less complex to evaluate expressions in that notation, simply leveraging a basic stack as the underlying data structure. In Fig. 5.8 the example string  $2 < 10$  is converted into its RPN equivalent,  $2\ 10 <$ . This is further discussed in Section 5.2.1.5.

The final stage is to parse the resulting RPN expression into a final result. This is also accomplished with a stack, and is further discussed in Section 5.2.1.6. Once we have the final result, *true* or *false* one simply loads this value into the corresponding Output Characteristic and sets the corresponding actuator value, as can be seen in Fig. 5.9.

#### 5.2.1.4 String Replacement Algorithms

There is no standard C function to search a string for a certain substring and replace it (akin to Java's *String.replace*). This is mainly due to the fact that replacing a substring with another substring may or may not involve memory reallocation depending on the size of the new substring in relation to the old substring. Nevertheless, this problem is left to be solved by the programmer.

In our case, the [Boyer-Moore string search algorithm](#) was initially used to search for substring occurrences and return a pointer to the instance of the substring, being the string subsequently replaced, with memory kept, expanded or freed depending on what was necessary. This approach, however efficient CPU-wise, proved not to be very efficient in terms of used-up RAM. Since this was our main bottleneck (having only about 4 kB free), we opted for a more RAM efficient, custom, alternative.

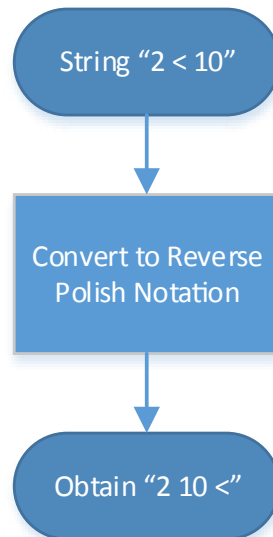


Figure 5.8: Second stage of interpreting a rule

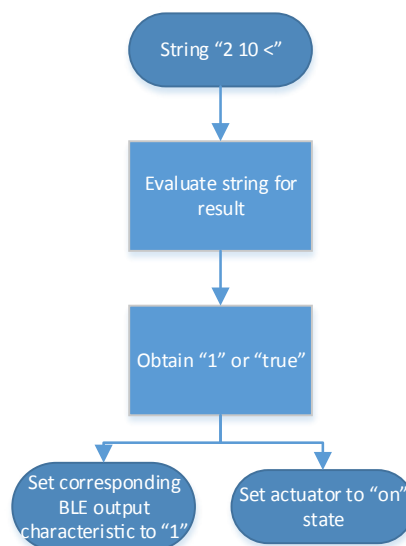


Figure 5.9: Third and last stage of interpreting a rule

**Boyer-Moore string search algorithm** The Boyer-Moore string search algorithm was developed by Robert S. Boyer and J Strother Moore, both members of faculty of the University of Texas at Austin. It is a fast algorithm and is considered the standard benchmark for performance in string searching algorithms. The Boyer-Moore algorithm requires pre-computation of a so-called "Bad Match Table", from the pattern that is to be searched. The Bad Match Table is calculated by assigning a value to each unique letter present in the pattern calculated by  $\max(1, \text{patternlength} - \text{index} - 1)$ , only being stored the value for the last occurrence of the letter, the table is then suffixed with the character \*, representing "every other letter" and having a value equal to the length of the pattern. A special case is the last letter in the word, that either keeps its value if it has already been defined (having other occurrences), or the length of the pattern is attributed as its value. We will take the pattern "tooth" in the string "trusthardtoothbrushes" as an example, taking inspiration from [91]. The Bad Match Table for "tooth" can be consulted in Fig. 5.10.

	T	O	O	T	H
	0	1	2	3	4

Letter	T	O	H	*
Value	4 1	3 2	5	5

1. **T**:  $5 - 0 - 1 = 4$
2. **O**:  $5 - 1 - 1 = 3$
3. **O**:  $5 - 2 - 1 = 2$
4. **T**:  $5 - 3 - 1 = 1$
5. **H**:  $length = 5$

Figure 5.10: Bad Match Table and calculation process

One essentially begins the process by aligning the pattern with the string and matching the last character in the pattern with the character in the string (Fig. 5.11a), since there is no match, one looks up the letter from the string in the Bad Match Table. In this case, the letter against which there was no match was the letter T, that has a value 1 in Fig. 5.10, so one advances the pattern in relation to the string by 1, this method aligns the T in the text with the last T in the pattern. If the letter isn't present explicitly in the table, it will match against the asterisk and have a value of 5.

After the first iteration, a match is found for the letter H, as can be seen in Fig. 5.11b.

One starts to match the pattern against the string backwards. In this case, the pattern matches until we reach the letter S in the string (Fig. 5.11c). When the mismatch is found, one looks up the next "jump" by looking up the value for H, since this was the first letter that was matched.

As can be seen in Fig. 5.11d, after the jump the letter H is mismatched against the letter O. Looking up O in the Bad Match Table, we get a next jump of 2, resulting in Fig. 5.11f with yet another mismatch at T. So, we advance the pattern by 1.

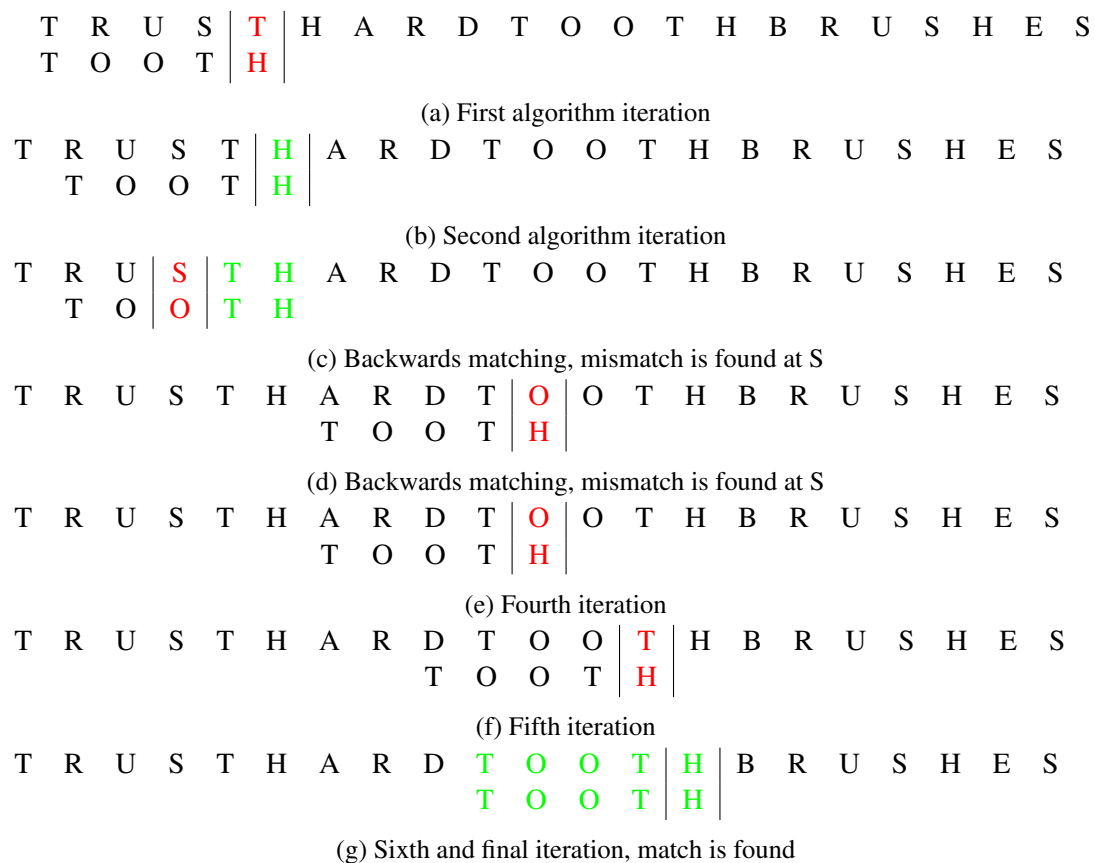


Figure 5.11: The different stages of the Boyer-Moore algorithm on an example string

In the algorithm's sixth iteration (Fig. 5.11g) a match is found for the pattern. The last index of the matched substring can then be returned, and its first index corresponds to  $lastindex - length$ . The algorithm can continue until the end of the string is found, thereby returning all occurrences of a particular substring.

This algorithm has a good performance for most string searching applications, however it has a worst case  $\mathcal{O}(n^2)$  complexity [92, 93].

**Custom Alternative** The custom alternative we present here is based off the, so called, naïve string search algorithm. One simply iterates over the string's characters for the first character of a pattern. One then continues iterating over the string for the remaining characters in the pattern. If all characters are successfully matched, a match for the pattern is found, if not, one simply continues iterating over the string. Despite its obvious disadvantages, including an average  $\mathcal{O}(n^2)$  complexity, we found it to be the most simple in ease of implementation, in code footprint and on, already strained, **MCU** resources. A flowchart for this algorithm can be consulted in Fig. 5.12.

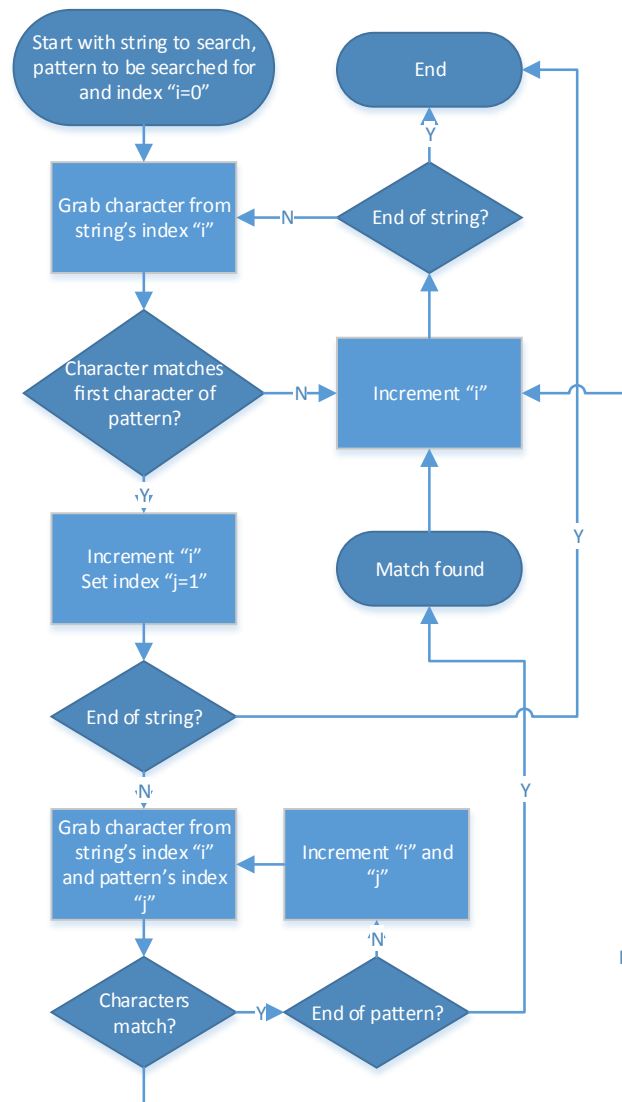


Figure 5.12: The naïve algorithm



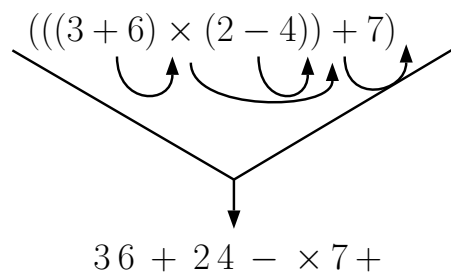


Figure 5.13: Arithmetic expression in common infix notation (above) and its equivalent expression in postfix notation (below) [94]

### 5.2.1.5 Transforming a mathematical expression into Reverse Polish Notation

So that one is able to efficiently parse an expression in a computer, one has to first convert it from infix notation to postfix, or Reverse Polish, notation (see Fig. 5.13 for an example). In this type of notation no parentheses are used to establish operator precedence, it is solely specified by the order in which the members are ordered. Evaluation of expressions of this type is explained in Section 5.2.1.6.

The conversion from infix notation to postfix notation can be done using the Shunting-Yard algorithm. This stack-based algorithm was invented by Edsger Dijkstra, renown computer scientist, in the '60s, and first described in [95]. It gets its name from its operation resemblance to the method in which railroad cars are sorted in some railroad yards called "shunting-yards". In addition to supporting unary and binary operations this algorithm also supports functions, making it extremely versatile.

The algorithm requires two variables: the input and the output, in addition to an auxiliary stack in which to hold operators that haven't been added to the output. It also requires a table in which operators are stored and that stores the operator type (unary, binary, or function) and its associativity (left or right).

This algorithm has  $\mathcal{O}(n)$  running time complexity [96].

The operation of this algorithm is described in Fig. 5.14.

The code that was used in our solution is a modified version of this [97] C Shunting-Yard implementation.

### 5.2.1.6 The Recursive Descent Parser

Having the expression to be evaluated in the correct notation, it is necessary to do the actual evaluating. This is done with a Recursive Descent Parser.

Expressions in this notation are evaluated from left to right. One essentially iterates over the expression until one finds an operator. That operator is applied to the two operands (numbers) before it. One goes back to the beginning of the expression and starts over until only one operand is left (the final result) and the expression is successfully evaluated. A step by step evaluation of an equation can be consulted in Fig. 5.15 [98] and the flowchart for this algorithm can be consulted in Fig. 5.16.

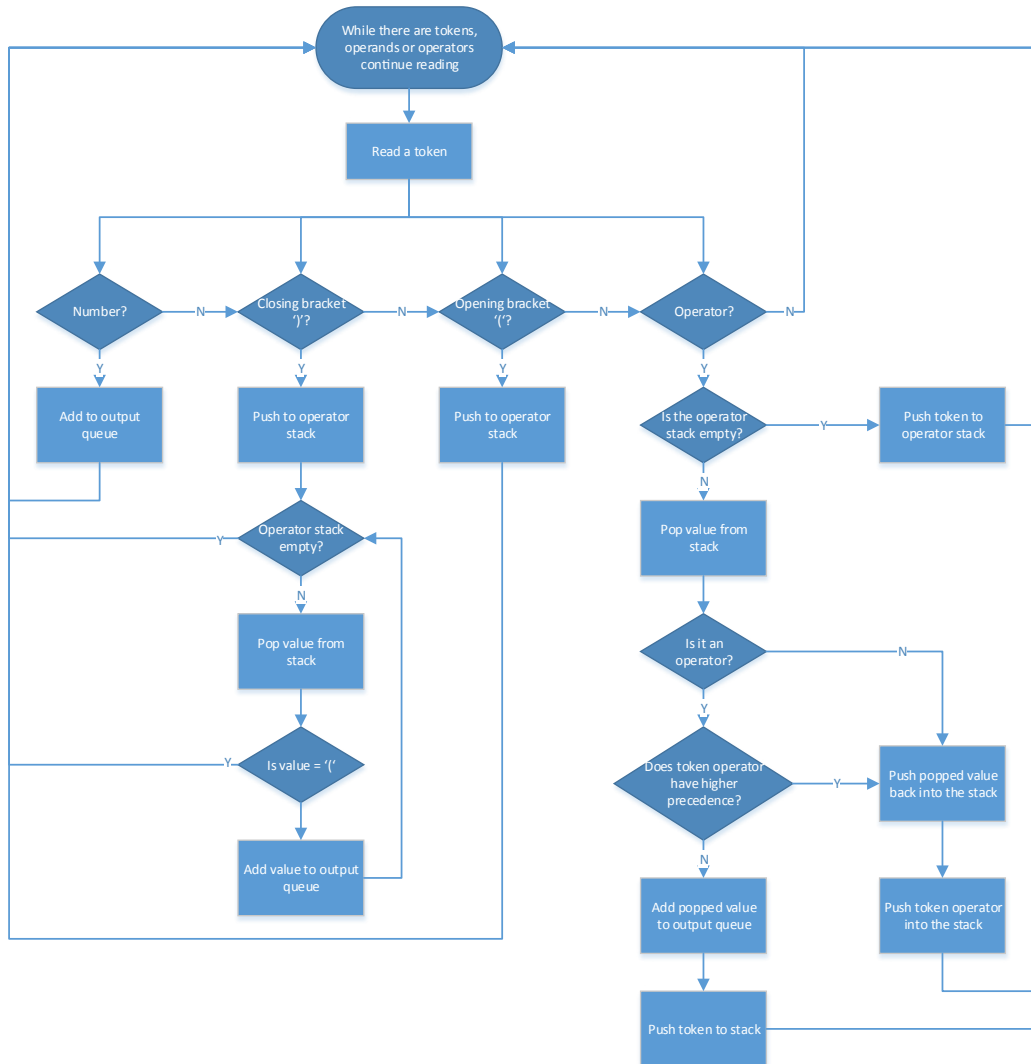


Figure 5.14: Flowchart of the shunting yard algorithm

In this system the RPN parser is implemented using code adapted from [99].

$$\begin{array}{r}
 7 \quad \underline{10 \ 5 \ /} \quad + \quad 6 \ 2 \ * \ + \\
 7 \quad \mathbf{2} \ + \ 6 \ 2 \ * \ + \\
 \\
 \underline{7 \ 2 \ +} \ 6 \ 2 \ * \ + \\
 \mathbf{9} \ 6 \ 2 \ * \ + \\
 \\
 9 \quad \underline{6 \ 2 \ *} \ + \\
 9 \quad \mathbf{12} \ + \\
 \\
 \underline{9 \ 12 \ +} \\
 \mathbf{21}
 \end{array}$$

Figure 5.15: Step by step solving of an RPN expression

The main switch block in the code (as can be seen in Fig. 5.17) is where the actual parsing stage happens. When one pops an operator from the stack it is applied to both previous operands ( $a$  and  $b$ ) and the result is pushed back onto the stack. As can be inferred from the code, our RPN parser supports arithmetic operations: *addition*, *subtraction*, *multiplication*, *division* and *exponentiation* (with *ipow*, an integer only version of the *pow* function) and relational operations: *greater than*, *less than* and *equal to*. This allows for a certain degree of freedom in rule construction, certainly providing far more liberty than most users would need. However, this RPN parser can be easily extended to support boolean operators *and* and *or*.

The current limitation of this parser is that unary operations are not supported, at least without the use of a *dummy* operand, e.g. to evaluate  $\text{not}(1)$  expressed as  $0 \ ! \ 1$  so that  $\text{push}(a \ ! \ b)$  would yield the correct result. However, with some code refactoring (a small decision block before the switch block, where one would test if the operator was binary or unary and re-push unneeded operands onto the stack) this parser can easily be made to support almost any operator.

## 5.2.2 Android developments

In this section we discuss the different developments on Android.

### 5.2.2.1 The Android BLE Stack

The Android BLE stack took longer than typical to develop for Google. Although some third parties like Samsung and Motorola supported BLE with their own APIs, native Android support for BLE only came with the Android 4.3 release [100]. Although BLE was subsequently natively supported, Android's BLE stack was known to be buggy. The BLE stack is known to be far more

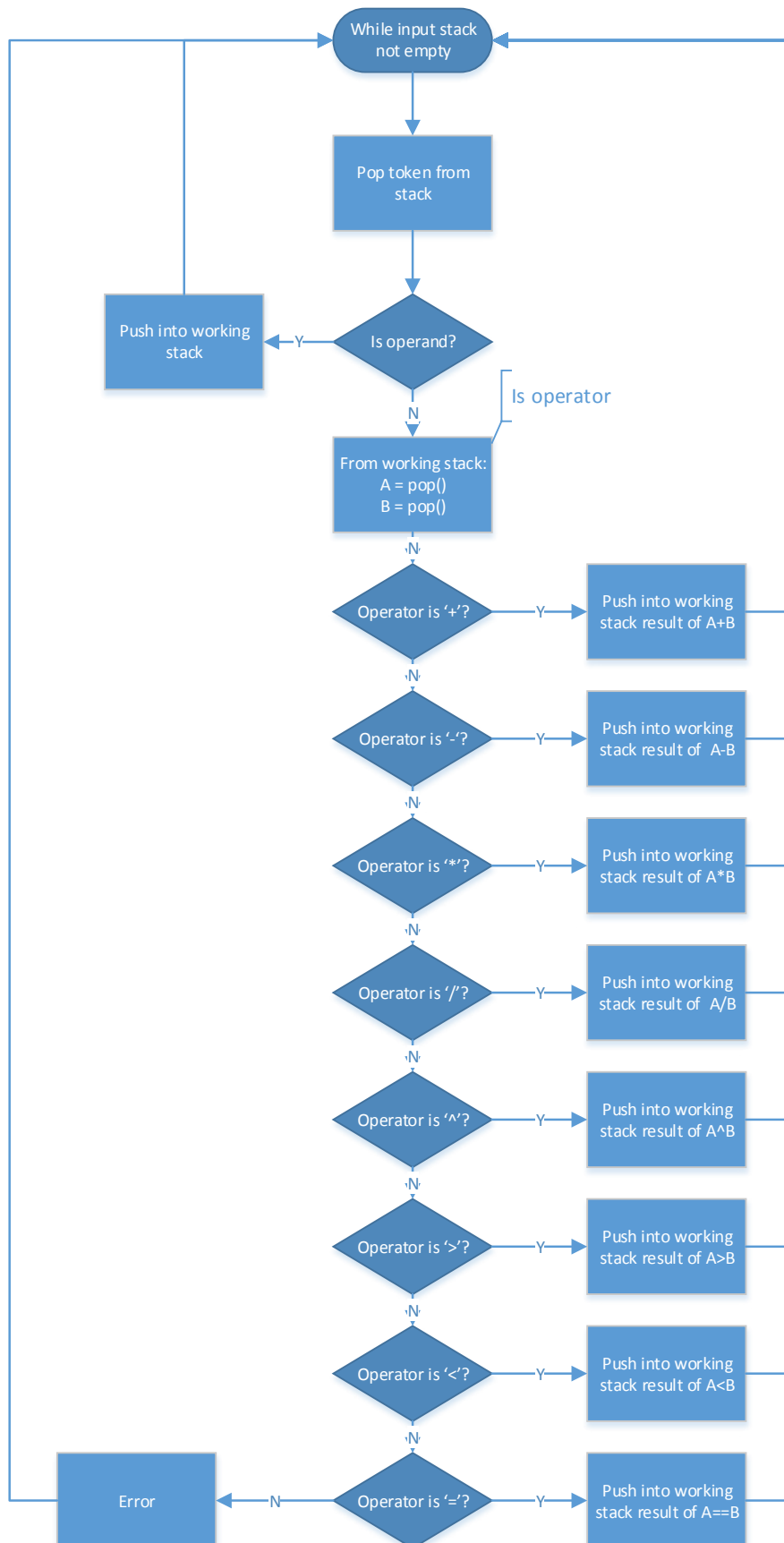


Figure 5.16: Flowchart of the RPN parser algorithm

```
switch(*s){
    case '+':
        push(a + b);
        break;
    case '-':
        push(a - b);
        break;
    case '*':
        push(a * b);
        break;
    case '/':
        push(a / b);
        break;
    case '^':
        push(ipow(a, b));
        break;
    case '>':
        push(a > b);
        break;
    case '<':
        push(a < b);
        break;
    case '=':
        push(a == b);
        break;
    default:
        //found an unknown operator
        *err_code = RPN_UNKOWN_OPERATOR;
        return 0;
        break;
}
```

Figure 5.17: Main switch block of our [RPN](#) parser code

stable in Android 5.0 (API level 21), so that is the version we chose to support. It should be noted that this release added the capability for an Android device to act as a Peripheral device (to send advertisement packets), however, few are the devices that support this mode of operation [101], so we chose to work around this limitation. Despite these improvements, the stack remains, in our opinion, buggy and far from perfect.

Usually one would create a BLE service to run in the background with applications of this type. But for ease of implementation we opted to have all BLE related code in the main application Activity and have extra required configuration windows pop-up at the user via Dialogs. This is further discussed in Section 5.2.2.3.

### 5.2.2.2 The Smart Companion graphical library

The Smart Companion is a project by Fraunhofer Portugal to build a smartphone app (Fig. 5.18) launcher that is a constant companion to both elderly and their caregivers and relatives. The Smart Companion's UI was developed with test groups and usability tests in order to be user-friendly to the elderly while maintaining functionality. The Smart Companion team then released a set of rules and guidelines on how to develop applications according to this model, and released an Android library to help develop applications that fit into this launcher and provide ease of use to the elderly [87].

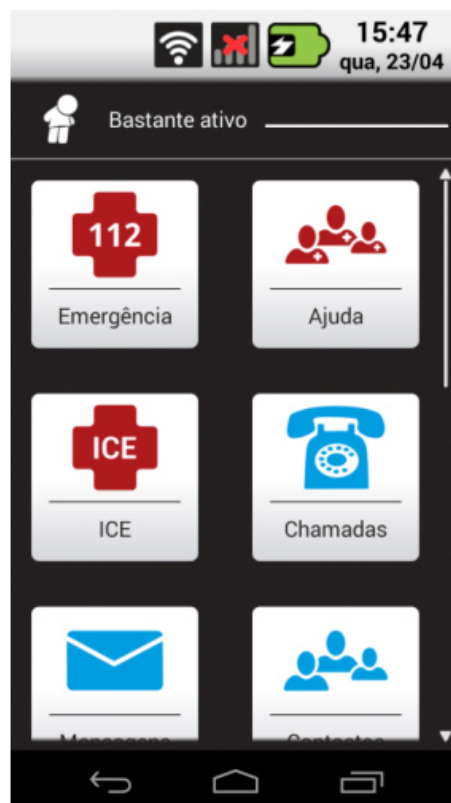


Figure 5.18: The Smart Companion Launcher [87]

Main features of using this library include a fixed status bar on top of the main application featuring relevant status info, such as Wi-Fi connection status, cellular connection status and battery level, more visual navigation menus such as an *always-on* backwards navigation bar answering common questions asked by the elderly such as "Where am I?", "What does this screen show me?" and "What can I do from here?". Font types and sizes are also specified by the library and are stored in eXtensible Markup Language (XML) files, that can be referenced from inside the Android application.

The base visual of the application can easily be made to conform to the Smart Companion visual standard by editing just a couple lines of code: first, one must add the library either as a *.jar* or an *.aar* file to the Android project's dependencies and one must define the application's theme as being of the type "SAActivityStyle". After these two steps are done, for the whole of the application one just needs to add a couple snippets of code per Android Activity to conform to the Smart Companion standard.

### 5.2.2.3 The Application

The application itself is just a frontend for the configuration of the Nordic MCU since all of the processing of data happens there. The Android frontend just exposes the ability to alter the values of BLE characteristics present in the MCU thereby modifying its behaviour.

The application consists of two Activities: a loading activity and a main activity; and extra settings are configured via Android Dialogs (pop-ups). A flowchart of all the actions available to the user can be consulted in Fig. 5.19.

## 5.2.3 Raspberry Pi developments

Unfortunately the Raspberry Pi was the least developed platform with which we worked. Despite this, developments can be consulted in this section: Middleware service can be consulted in Section 5.2.3.1 and the python application that was developed can be consulted in Section 5.2.3.2.

One should note that for 6LoWPAN to work with the Raspberry Pi one has to install a kernel different from those usually distributed for the Raspberry Pi. This is due to the relatively recent nature of the support for this feature in the Linux kernel. Compiling such a kernel is an overnight affair when one is compiling directly on the PI, luckily one can either cross-compile on a common x86 system using the GNU ARM Toolchain or use the pre-compiled kernel provided by Nordic Semi with its IoT SDK. We opted for the latter.

One should note that for the Nordic client to obtain a globally routable IPv6 address, one has to correctly install and configure the Router Advertisement Daemon (radvd) on the Raspberry Pi.

### 5.2.3.1 RabbitMQ

The first step in Pi development consisted on the installation of the RabbitMQ package (with a simple *apt-get* command) and the activation of RabbitMQ's MQTT plugin. The whole affair is

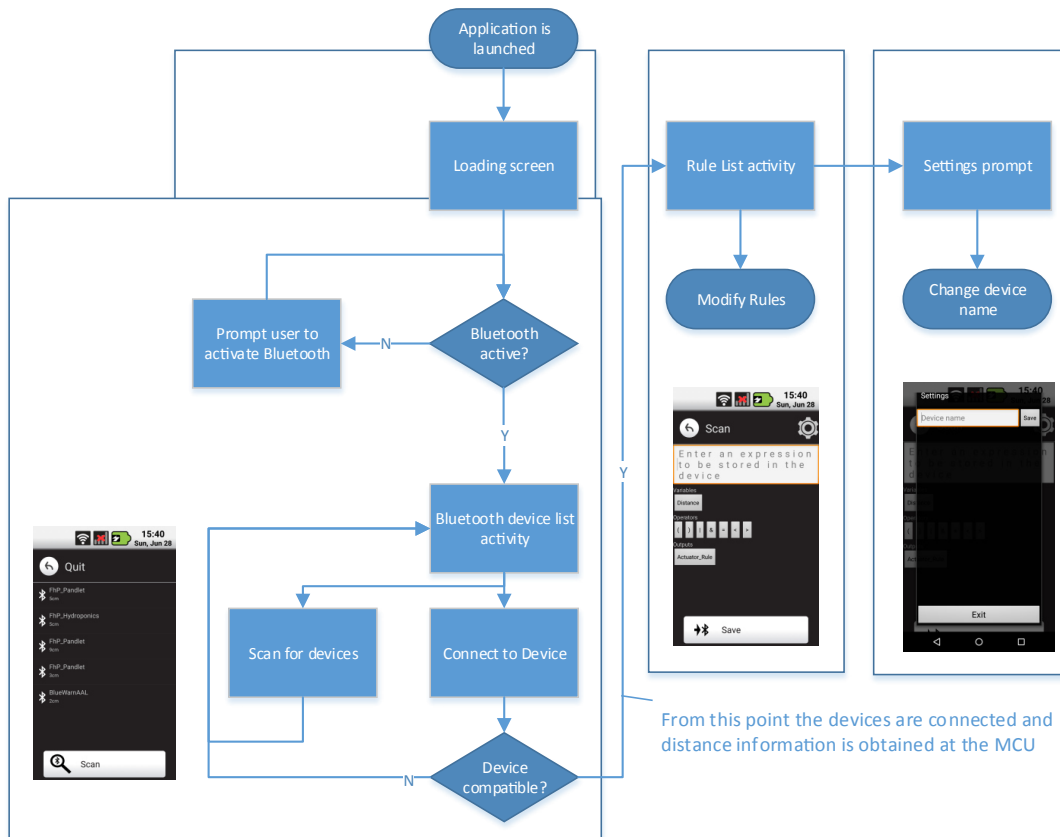


Figure 5.19: Android application UI flow

pretty straightforward. After this, the RabbitMQ server is ready to accept connections either using the [AMQP](#) protocol or the [MQTT](#) protocol.

### 5.2.3.2 Python application

Since support for this kind of connectivity is still very much recent, connecting to [6LoWPAN](#) enabled devices has to be instructed to the Linux kernel running on the Raspberry Pi via commands echoed to the `/sys/kernel/debug/bluetooth/6lowpan_psm` file, as can be consulted in [102], it was chosen to develop a Python-based application to control to which devices the Raspberry Pi connected and subsequently offered a [6LoWPAN](#) connection to the Internet.

This application leverages the `hcidump` and `hctool lescan` shell utilities to obtain a dump of scanned [BLE](#) advertisement packets, and then parses them to figure out what packets are pertaining to our application and which packets can be simply discarded. It then calls for the Linux kernel to connect to these selected devices and to allow for them to have an Internet connection for about 30 s. It then promptly disconnects and iterates over to the next [BlueWarnAAL](#) device that it has found from advertisements. It repeats this process until all nodes have had 30 s to talk to the Middleware, it then sleeps for about 30 min and repeats this process. This allows for all nodes to be able to talk to the Middleware in regular intervals. Since the [MQTT](#) specification calls for every



node to be able to store up to a limited number of messages [103] this introduces some latency in the system, but allows for automated rotation of Internet access for all nodes without any type of user intervention. This is necessary since this connection has to be initiated by the Raspberry Pi.



# Chapter 6

## Testbed and Results

In this section we present the process that was undertaken to test out the BlueWarnAAL system.

### 6.1 The proposed scenario

The main use case scenario, as presented in Section 1.1, is considered for testing. To simulate a real-world environment, a testbed was developed. The development process can be consulted in Section 6.1.1.

#### 6.1.1 The testbed

The testbed was planned to simulate, on a small scale, real world functioning of the system. Essentially, this testbed is a small table with AC power connections, a power switching circuit and the [Fraunhofer Pandlet](#).

Any type of ordinary home appliance can be connected to the AC power outlet and work exactly like it would normally, with one crucial exception: AC power can be switched on or off via the power circuit, that is connected to the [Fraunhofer Pandlet](#), depending on the rule configured on the Pandlet by the user.

##### 6.1.1.1 Planning

Planning consisted of two phases: designing the power circuit and designing the table.

**The power circuit** The objective of this circuit is to allow for the MCU, with the help of a relay, to switch the appliance on or off. The schematic for the circuit can be seen in Fig. 6.1. It is very simple and includes only 5 components: a resistor, a NPN Bipolar Junction Transistor (BJT), a diode, a Zenner diode and a relay. All the components besides the relay are only there for protection against current spikes that are common when relays switch on or off.

After the circuit's design was decided on, it was simulated using National Instruments (NI)'s Multisim software (Fig. 6.2). The 10  $\Omega$  resistor is meant to simulate a 2.2 kW load. Current

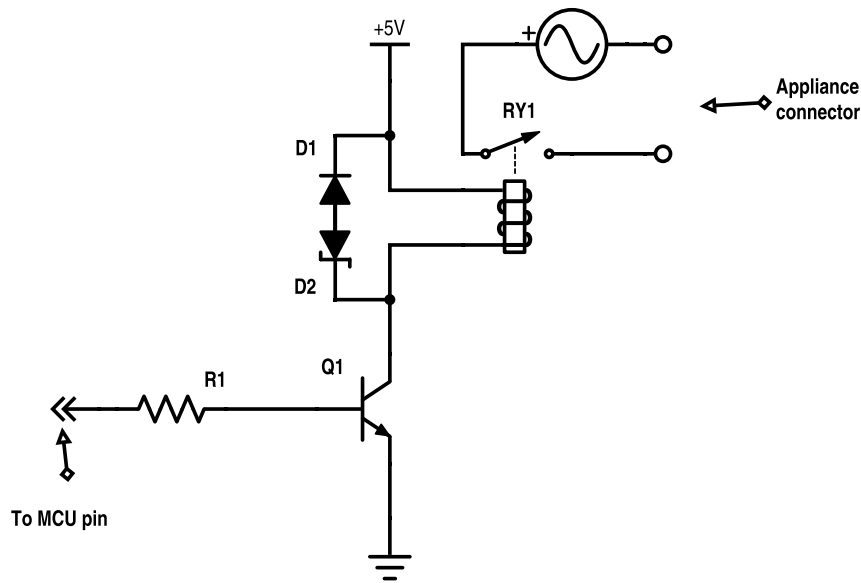


Figure 6.1: Schematic of the power switching circuit incorporating a relay

probes (virtual devices that convert amperage to voltage, so that one is able to measure current in an oscilloscope) were attached in places deemed critical.

The simulation confirmed that the circuit worked as expected. In Fig. 6.3 one can see the output waveform as measured at the load, having the switch *SI* been toggled. As expected voltage drops to 0 when the switch is turned off.

As can be seen in Fig. 6.4, the current in the diode branch spikes (negatively due to inverted polarity from the oscilloscope referential) when the circuit is switched. However, there are no current spikes in the BJT, it can therefore be concluded that the circuit is working as expected.

Finally, having the circuit been validated via simulation, the required components were ordered. The bill of materials is available in Table 6.1.

Description	Product
Zener Diode	Central Semiconductor CZ5344B TR
BJT	Central Semiconductor 2N3904
Diode	NXP Semiconductors BYV29FX-600,127
Relay	TE Connectivity T9AS1D12-5

Table 6.1: Circuit bill of materials

## 6.2 Functional tests

The system was tested with the following procedure:

1. The system was turned on (with the default rule of *Distance* < 10) and with an led connected to the actuator output;

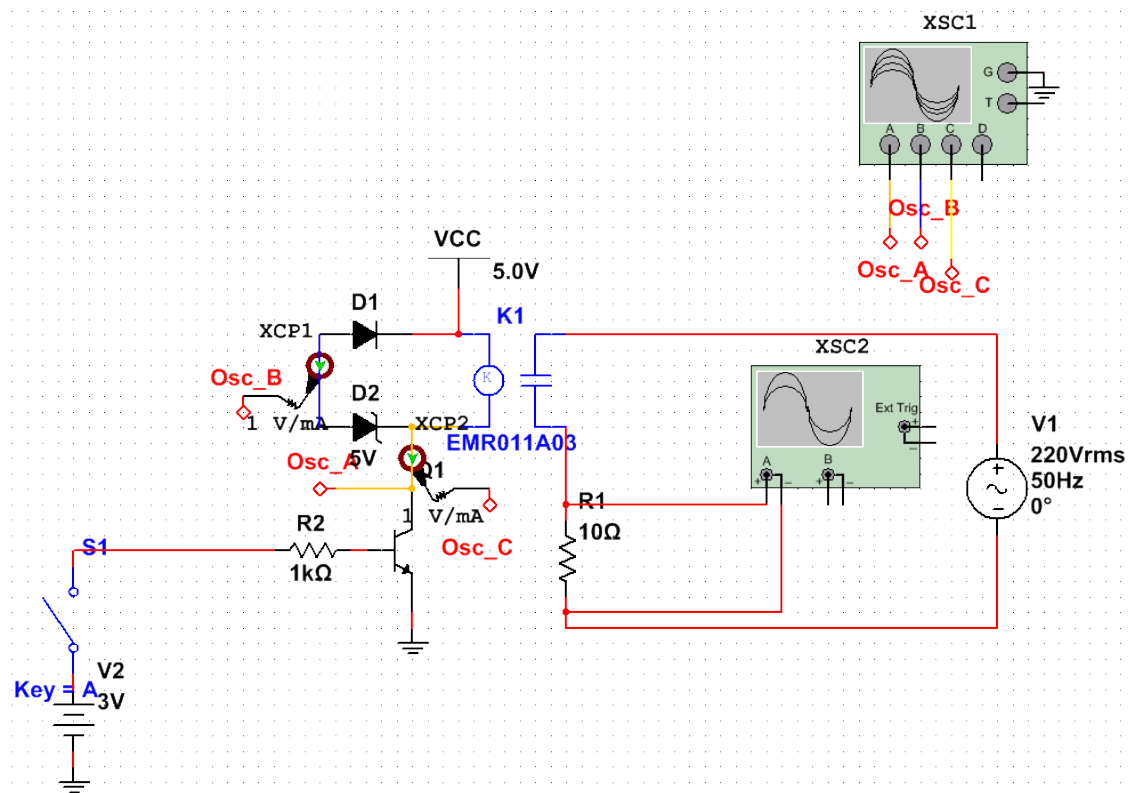


Figure 6.2: The circuit recreated in NI's Multisim

2. One stood close to the **MCU** and progressively stepped back until the LED turned off. One would then measure the distance to the **MCU**;
3. The distance rule in the **MCU** was changed, and the procedure repeated.

The system was deemed accurate with an error of about 2 m, up to a distance of about 70 m with a clear line of sight. Without line of sight the error rate after a distance of 20 m makes the system unusable. However, for our purposes, the system is reliable enough.

### 6.3 Distance estimation tests

Two tests were conducted to ascertain the accuracy of the distance estimation algorithm: one was conducted in open space (Fig. 6.5), and one, in a 70 m long corridor (Fig. 6.7). The open space results were satisfactory, with a small error in relation to the real distance. The corridor tests showed a behaviour that was expected: Bluetooth location based on **RSSI** is prone to errors due to reflexions and refractions. The corridor in which the measurements were taken (Fig. 6.6) acts as a wave guide, and leads to stronger **RSSI** measurements than would be expected in open space.

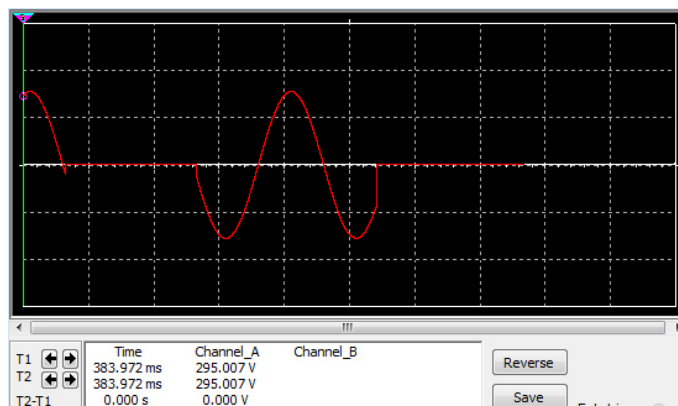


Figure 6.3: Oscilloscope plot of voltage at the load

## 6.4 Connectivity tests

The connectivity tests that were conducted essentially consisted on running the Raspberry Pi app and verifying whether there was IPv6 connectivity. As can be seen from Fig. 6.8, this was, effectively the case, since IPv6 Neighbour Solicitations and Router Announcements were traded between the Raspberry Pi and the Norcic MCU.

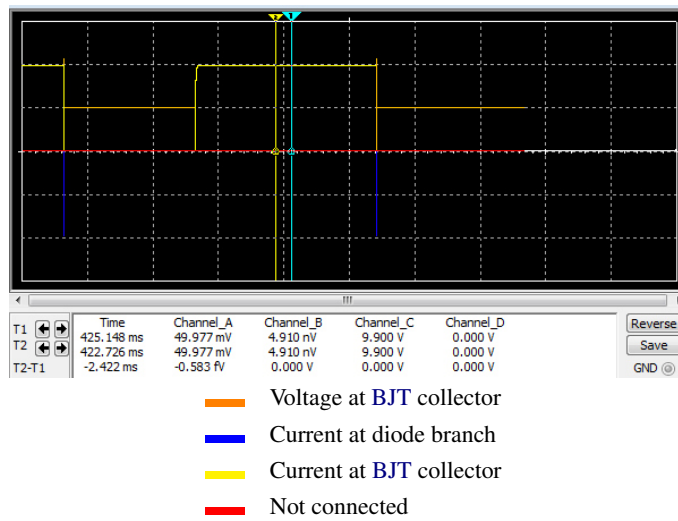


Figure 6.4: Voltage and current for different points in the circuit, being the relay switched through-out. For current measurements 1 mA = 1 V



Figure 6.5: Open space plot of estimated distance



Figure 6.6: Fraunhofer Portugal's main corridor (first floor)

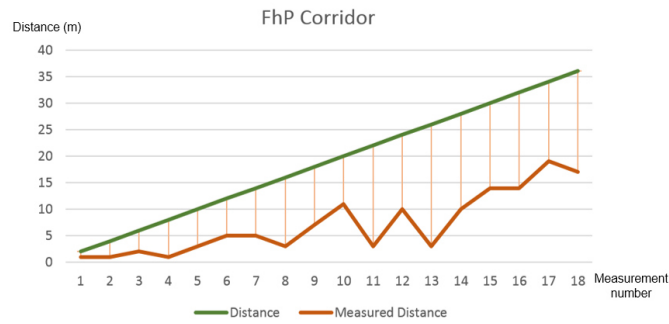


Figure 6.7: Corridor plot of estimated distance

```

root@raspberrypi:~# tcpdump -i bt0
tcpdump: WARNING: bt0: no IPv4 address assigned
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on bt0, link-type LINUX_SLL (Linux cooked), capture size 65535 bytes
15 25.30 821899 IP6 :: > ff02::1:ff66:b52: ICMP6, neighbor solicitation, who has fe80::5ef3:70ff:fe66:b52, length 32
15 25.30 889638 IP6 :: > ff02::1:ff28:e008: HBH ICMP6, multicast listener report, tx resp delay: 0, addr: ff02::1:ff28:e008, length 24
15 25.31 024892 IP6 :: > ff02::1:ff28:e008: ICMP6, neighbor solicitation, who has fe80::2d2:59ff:fe28:e008, length 32
15 25.32 847868 IP6 fe80::2d2:59ff:fe28:e008 > ff02::1:ff28:e008: HBH ICMP6, multicast listener report, tx resp delay: 0, addr: ff02::1:ff28:e008, length 24
15 25.36 475548 IP6 truncated-arp - 34292 bytes missing 4000:708::304:4060 > 1:5180:0:3840::fd1b:96d2: ip-proto-250 34304
tcpdump: pcap_loop: The interface went down
5 packets captured
6 packets received by filter
0 packets dropped by kernel

```

Figure 6.8: TCPDump utility output during connection with the Nordic MCU



# Chapter 7

## Conclusion

In this dissertation we were able to develop a complete, peer to peer, assisted living solution to help the elderly in their daily lives. As was discussed in Chapter 5, this solution was developed in three fronts: on the Nordic nRF51822, on the Raspberry Pi and on an Android smartphone. Although these systems are very much different, the BLE technology was instrumental on allowing these systems to talk to each other and to convey state information while keeping energy consumption down. The Smart Companion library allowed us to develop a powerful, yet extremely simple and usable application without having to worry about the tried and tested graphical design. Bleeding edge technology such as the Nordic's IoT stack and the Linux kernel's support for 6LoWPAN, allowed us to pave the way for future scalability.

Our main achievements are exposed in Section 7.1 and future work in Section 7.2.

### 7.1 Achievements

- **Nordic:**
  - **Distance:** We were able to accurately estimate distance from a Bluetooth client to our MCU based only on RSSI;
  - **Rule engine:** A completely functional rule engine was developed that was capable of evaluating rules customizable by the user;
  - **IP stack:** The LwIP IP stack was set up correctly and is able to send and receive packets;
- **Android:** A functional UI for interacting with the system was developed;
- **Raspberry Pi:** An application that is able to automate the process of connecting to various nodes was developed;
- **Scaling:** IPv6 connectivity was assured.

## 7.2 Future Work

Throughout the development work several factors contributed for us not to be able to complete all the initial objectives that we proposed ourselves to complete. Also, new ways in which this system can be further developed were thought up. All these tasks are rounded up in this section.

### 7.2.1 Pi application

The Raspberry Pi application needs to be refined, it is a very simple utility and could easily be expanded into a full fledged command line utility and daemon, since there are no utilities available that allow for this type of functionality. An addition that would require intervention also on the Nordic side would be to add another field to the Scan Response that would be set to *true* whenever the **MCU** had data it needed to send via the **IPv6** interface. This would allow the Pi to initiate a connection only when necessary, allowing the Nordic to sleep for larger periods of time and conserving power, while also avoiding the requirement for the Pi to connect to each **MCU** and subscribe for notifications (in order to use **BLE**'s native notification system).

### 7.2.2 MCU

As it stands, each second the **MCU** wakes up to gather data and update rule results. This could, and should, be changed into an event driven approach in order to allow the **MCU** to sleep for longer. Currently there is also a bug in the neighbour discovery section of the **LwIP** stack that isn't allowing the **MCU** to obtain a Global Address and crash. This means that while it can contact hosts in its subnet and use them as proxies it currently cannot communicate directly with hosts on the Internet. The cause of the bug is still uncertain, but the two main candidates are: a bug in the **IoT SDK** or lack of available **RAM**. Since Nordic announced in [104] that the next version of the **IoT SDK** would be released for the **nRF52**, if the cause was a bug in the **SDK**, the solution would be to migrate to the **nRF52**. Another way to get around this problem would be to port another, even lighter **IP** stack to the **nRF51**, such as  $\mu$ **IP** ([105]). However despite the effort of porting the **IP** stack, we would be giving up the vast functionality provided by **LwIP**.

### 7.2.3 nRF52

While nearing the conclusion of this dissertation, Nordic Semiconductor announced the **nRF52**, the successor to the **nRF51**. Main features include: 32-bit ARM Cortex-M4F Processor, 512 kB flash + 64 kB **RAM**, On-chip NFC tag for Out Of Band (**OOB**) pairing and Advanced Encryption Standard (**AES**) hardware encryption and lower power consumption. Since the new **MCU** is, for the most part, compatible with the previous series, migrating would fix both the problems mentioned in Section 7.2.2 and provide us with a more robust and powerful platform to work on.

## 7.2.4 Interfacing with the Middleware

As priorities changed during the dissertation this initial objective got pushed back. Since the **IoT SDK** has native support for **MQTT** and **IPv6** connectivity is already assured, this is just a matter of implementing a few more functions on the **MCU**.

## 7.2.5 Sensor remote control

One great way to expand on the developed functionality would be to allow rules to be altered from a web interface, or to override actuator states remotely. One can also introduce the concept of *local* rule and *remote* rule. The first would be evaluated in the **MCU** and would only have the context that is available to it; The second would be processed on the remote server and only the result would be sent to the different nodes in the system. Priorities could be established for these rules in a node by node basis.

### 7.2.5.1 Pimatic

This functionality could be implemented using Pimatic: an open source home automation framework running on node.js. Since it supports external scripting via an **API**, one would need to write a small plugin to interface with the middleware, and one could then control all the system's functionalities via the polished Pimatic **UI** (Fig. 7.1), while interfacing this system with other home automation systems [106].

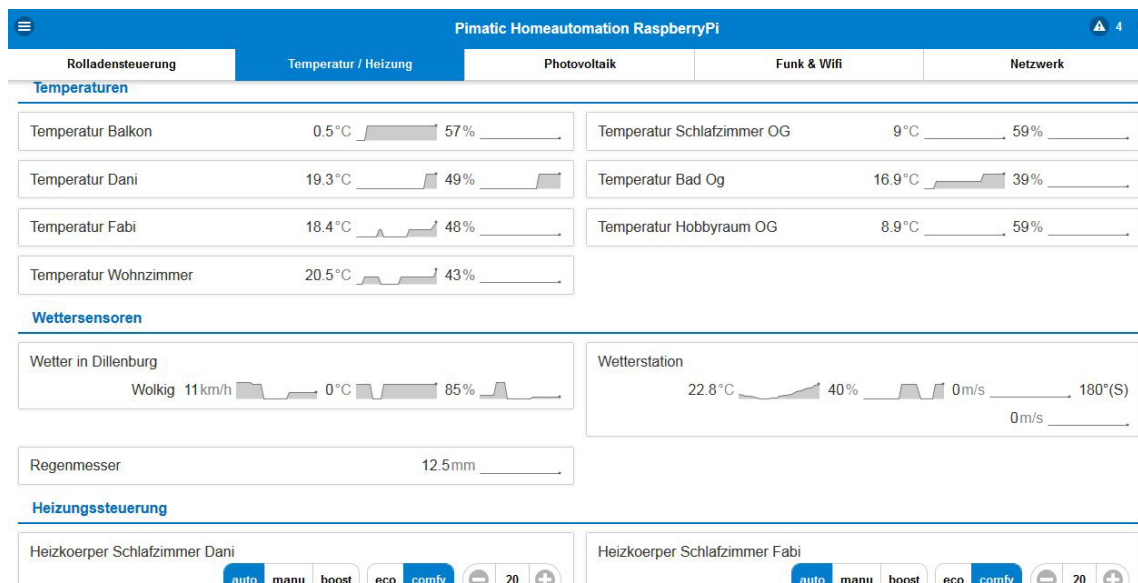


Figure 7.1: The Pimatic UI

### 7.2.6 Scaling up

It would be interesting and more powerful, especially from an IoT perspective, to explore RabbitMQ's message aggregation and queue mirroring capabilities to expand this system into a hierarchical one. This way one could have a RabbitMQ server that processes messages from a household, and, for example, another that processes messages from various households or even a city. This would allow this system to be applied in far more complex situations while retaining its flexibility.

### 7.2.7 Calibration

Since this system is meant to be installed in an elder's home a calibration procedure that could be run when the system was first set up would greatly improve the distance results obtained in Section 6.3. Even a small corrective gain factor could bring our distance estimation results very close to the real distance.

# References

- [1] S. Prescher, A. K. Bourke, F. Koehler, A. Martins, H. Sereno Ferreira, T. Boldt Sousa, R. N. Castro, A. Santos, M. Torrent, S. Gomis, M. Hospedales, and J. Nelson, “Ubiquitous ambient assisted living solution to promote safer independent living in older adults suffering from co-morbidity.”, *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Annual Conference*, vol. 2012, pp. 5118–21, Jan. 2012, ISSN: 1557-170X. DOI: [10.1109/EMBC.2012.6347145](https://doi.org/10.1109/EMBC.2012.6347145). [Online]. Available: <http://www.ncbi.nlm.nih.gov/pubmed/23367080>.
- [2] P. Rashidi and A. Mihailidis, “A survey on ambient-assisted living tools for older adults”, *IEEE Journal of Biomedical and Health Informatics*, vol. 17, pp. 579–590, 2013, ISSN: 21682194. DOI: [10.1109/JBHI.2012.2234129](https://doi.org/10.1109/JBHI.2012.2234129).
- [3] Kleinberger Thomas, M. Becker, E. Ras, A. Holzinger, and P. Müller, “Ambient Intelligence in Assisted Living: Enable Elderly People to Handle Future Interfaces”, *Universal Access in Human-Computer Interaction. Ambient Interaction*, pp. 103–112, 2007, ISSN: 03029743. DOI: [10.1007/978-3-540-73281-5\\_11](https://doi.org/10.1007/978-3-540-73281-5_11). [Online]. Available: [http://user.medunigraz.at/andreas.holzinger/holzinger/papers%20en/B41%5C\\_KLEINBERGER%20BECKER%20RAS%20HOLZINGER%20MUELLER%20\(2007\)%20ambient%20intelligence%20elderly%20people%20LNCS.pdf](http://user.medunigraz.at/andreas.holzinger/holzinger/papers%20en/B41%5C_KLEINBERGER%20BECKER%20RAS%20HOLZINGER%20MUELLER%20(2007)%20ambient%20intelligence%20elderly%20people%20LNCS.pdf).
- [4] F. L. Lewis, “Wireless sensor networks”, *Smart environments: technologies, protocols, and applications*, pp. 11–46, 2004.
- [5] F. G. Montoya, J. Gómez, A. Cama, A. Zapata-Sierra, F. Martínez, J. L. De La Cruz, and F. Manzano-Agugliaro, “A monitoring system for intensive agriculture based on mesh networks and the android system”, *Computers and Electronics in Agriculture*, vol. 99, pp. 14–20, Nov. 2013, ISSN: 0168-1699. DOI: <http://dx.doi.org/10.1016/j.compag.2013.08.028>. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0168169913002068>.
- [6] M. Portnoi and C.-C. Shen, “Location-aware sign-on and key exchange using attribute-based encryption and Bluetooth beacons”, ser. 2013 IEEE Conference on Communications and Network Security (CNS), Dept. of Comput. Inf. Sci., Univ. of Delaware, Newark,

- DE, United States BT - 2013 IEEE Conference on Communications and Network Security (CNS), 14-16 Oct. 2013: IEEE, 2013, pp. 405–406. DOI: [10.1109/CNS.2013.6682750](https://doi.org/10.1109/CNS.2013.6682750). [Online]. Available: <http://dx.doi.org/10.1109/CNS.2013.6682750>.
- [7] G. Banavar, T. Chandra, R. Strom, and D. Sturman, “A case for message oriented middleware”, in *Distributed Computing*, Springer, 1999, pp. 1–17.
- [8] NEST, *Life with Nest Thermostat*, 2015. [Online]. Available: <https://nest.com/thermostat/life-with-nest-thermostat/> (visited on 02/18/2015).
- [9] H. S. Ferreira, T. B. Sousa, and A. Martins, “Scalable integration of multiple health sensor data for observing medical patterns”, in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7467 LNCS, 2012, pp. 78–84.
- [10] K. Lee, “IEEE 1451: A standard in support of smart transducer networking”, in *Instrumentation and Measurement Technology Conference, 2000. IMTC 2000. Proceedings of the 17th IEEE*, IEEE, vol. 2, 2000, pp. 525–528.
- [11] E. Mackensen, M. Lai, and T. M. Wendt, “Bluetooth low energy (BLE) based wireless sensors”, ser. 2012 IEEE Sensors, Univ. of Appl. Sci. Offenburg, Offenburg, Germany BT - 2012 IEEE Sensors, 28-31 Oct. 2012: IEEE, 2012, 4 pp. DOI: [10.1109/ICSENS.2012.6411303](https://doi.org/10.1109/ICSENS.2012.6411303). [Online]. Available: <http://dx.doi.org/10.1109/ICSENS.2012.6411303>.
- [12] J. Moy, “OSPF Version 2”, IETF, RFC 2328, 1998, pp. 1–244. [Online]. Available: <http://tools.ietf.org/html/rfc2328>.
- [13] C. Harrison, J. Wiese, and A. K. Dey, *Achieving Ubiquity: The New Third Wave*, 2010. DOI: [10.1109/MMUL.2010.53](https://doi.org/10.1109/MMUL.2010.53).
- [14] Merriam-Webster, *Ubiquitous Definition*, 2014. [Online]. Available: <http://www.merriam-webster.com/dictionary/ubiquitous>.
- [15] OCLF, *ORNL Debuts Titan Supercomputer*, 2012.
- [16] A. Greenfield, *Everyware: The dawning age of ubiquitous computing*. New Riders, 2010.
- [17] M. Rouse, *Definition - machine-to-machine (M2M)*, 2010. [Online]. Available: <http://whatis.techtarget.com/definition/machine-to-machine-M2M> (visited on 01/21/2015).
- [18] S. Bushell and P. Budde, *M-commerce key to ubiquitous Internet - Computerworld*, 2000. [Online]. Available: [http://www.computerworld.com.au/article/84178/m-commerce%5C\\_key%5C\\_ubiquitous%5C\\_internet/](http://www.computerworld.com.au/article/84178/m-commerce%5C_key%5C_ubiquitous%5C_internet/) (visited on 01/23/2015).
- [19] CMU SCS Students, *MIT Coke Machine History*, 2005. [Online]. Available: [https://www.cs.cmu.edu/~coke/history%5C\\_long.txt](https://www.cs.cmu.edu/~coke/history%5C_long.txt) (visited on 01/23/2015).

- [20] Sri, “Appendix F: the internet of things (background)”, *Disruptive civil technologies conference report (CR ...*, pp. 1–19, 2008. [Online]. Available: <http://www.internet-of-things.eu/resources/documents/appendix-f.pdf>.
- [21] I. F. Akyildiz, “A survey on wireless mesh networks”, *IEEE Communications Magazine*, vol. 43, no. 9, S23–S30, 2005, ISSN: 0163-6804. DOI: 10.1109/MCOM.2005.1509968. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1509968>.
- [22] T. Clausen and P. Jacquet, *RFC 3626 - Optimized Link State Routing Protocol (OLSR)*, 2003.
- [23] C. Geyer and OASIS, *ISO and IEC Approve OASIS AMQP Advanced Message Queuing Protocol*, 2014. [Online]. Available: <https://www.oasis-open.org/news/pr/iso-and-iec-approve-oasis-amqp-advanced-message-queuing-protocol> (visited on 01/31/2015).
- [24] OASIS, “OASIS Advanced Message Queuing Protocol (AMQP) Version 1.0. 29”, *OASIS Standard*, no. October, 2012. [Online]. Available: <http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-complete-v1.0-os.pdf>.
- [25] STOMP Specification Group, *STOMP specification v1.2*, 2012. [Online]. Available: <https://stomp.github.io/stomp-specification-1.2.html> (visited on 01/31/2015).
- [26] A. Banks and R. Gupta, “MQTT Version 3.1.1”, *OASIS Standard*, no. October, 2014. [Online]. Available: <http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html>.
- [27] A. Stanford-Clark, H. L. Truong, and MQTT, “MQTT For Sensor Networks ( MQTT-SN ) Protocol Specification”, p. 27, 2013.
- [28] Pivotal Software, *RabbitMQ Documentation*, 2015. [Online]. Available: <https://www.rabbitmq.com/documentation.html> (visited on 02/01/2015).
- [29] Atchoo.org, *An Open Source MQTT v3.1 Broker*. [Online]. Available: <http://mosquitto.org/> (visited on 02/01/2015).
- [30] *.net - C# client library for subscribing/publishing MQTT (Really Small Message Broker) - Stack Overflow*, 2011. [Online]. Available: <https://stackoverflow.com/questions/6635440/> (visited on 02/01/2015).
- [31] *Apache ActiveMQ™ – How does ActiveMQ compare to AMQP*. [Online]. Available: <http://activemq.apache.org/how-does-activemq-compare-to-amqp.html> (visited on 02/01/2015).
- [32] K. Sachs, S. Kounev, S. Appel, and A. Buchmann, “A Performance Test Harness For Publish/Subscribe Middleware”, in *SIGMETRICS/Performance*, Citeseer, 2009.

- [33] S. Appel, K. Sachs, and A. Buchmann, “Towards benchmarking of AMQP”, in *Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems*, ACM, 2010, pp. 99–100.
- [34] M. Hadlow, *Message Queue Shootout!*, 2011. [Online]. Available: <http://mikehadlow.blogspot.pt/2011/04/message-queue-shootout.html> (visited on 02/02/2015).
- [35] M. Salvan, *A quick message queue benchmark: ActiveMQ, RabbitMQ, HornetQ, QPID, Apollo...* [Online]. Available: <http://blog.x-aeon.com/2013/04/10/a-quick-message-queue-benchmark-activemq-rabbitmq-hornetq-qp-id-apollo/> (visited on 12/07/2014).
- [36] M. Klas, “Porovnání protokolů pro M2M komunikaci.”, Masarykova Univerzita, 2014, p. 42.
- [37] Gilles Thonet, “ZigBee FAQ”, vol. 1, no. 7, pp. 1–7, 2006.
- [38] IEEE Computer Society, *Part 15.1: Wireless medium access control (MAC) and physical layer (PHY) specifications for wireless personal area networks (WPANs)*, June. 2005, vol. 2005, ISBN: 0738147079. DOI: [10.1109/IEEESTD.2003.94389](https://doi.org/10.1109/IEEESTD.2003.94389).
- [39] G. Fleishman and Ars Technica, *UWB group shutter, sends tech to Bluetooth, USB groups*, 2009. [Online]. Available: <http://arstechnica.com/gadgets/2009/03/ultrawideband-groups-disbands-doesnt-despair/> (visited on 02/03/2015).
- [40] B. news, *Bluetooth rival unveiled by Nokia*, 2006. [Online]. Available: <http://news.bbc.co.uk/2/hi/technology/5403564.stm>.
- [41] Bluetooth SIG, *Bluetooth Smart Beacons in Retail*, 2015. [Online]. Available: <http://www.bluetooth.com/Pages/beacons-retail-location.aspx> (visited on 02/03/2015).
- [42] —, *Bluetooth technology creates huge opportunities in medical*, 2015. [Online]. Available: <http://www.bluetooth.com/Pages/Health-Wellness-Market.aspx> (visited on 02/03/2015).
- [43] —, *Bluetooth Technology Makes Wireless Home Automation Possible*, 2015. [Online]. Available: <http://www.bluetooth.com/Pages/Smart-Home-Market.aspx> (visited on 02/03/2015).
- [44] ZigBee Alliance, *ZigBee Specification FAQ*. [Online]. Available: <http://old.zigbee.org/Specifications/ZigBee/FAQ.aspx>.
- [45] Nordic Semiconductor, “nRF24L01+ Product specification v1.0”, vol. 21, no. September, pp. 21–22, 2008. DOI: [10.1080/09613219308727250](https://doi.org/10.1080/09613219308727250). [Online]. Available: <https://www.nordicsemi.com/eng/Products/2.4GHz-RF/nRF24L01P>.
- [46] WPAN IEEE 802.15 3c Task Group, *IEEE 802.15 WPAN Task Group 3c (TG3c) Millimeter Wave Alternative PHY*, 2009. [Online]. Available: <http://www.ieee802.org/15/pub/TG3c.html> (visited on 02/03/2015).



- [47] “IEEE Standard for Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Specific Requirements Part 15.3: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications F”, *IEEE Std 802.15.3-2003*, 0\_1–315, 2003. DOI: [10.1109/IEEESTD.2003.94395](https://doi.org/10.1109/IEEESTD.2003.94395).
- [48] IEEE Computer Society, *60 GHz WPAN, PHY and MAC*. 2009, ISBN: 9780738160504.
- [49] L. Yuansheng and H. Xi, *Analysis of the Maximal Transmission Rate Based on NRF24L01 Chip System*, 2010. DOI: [10.1109/ICIECS.2010.5678223](https://doi.org/10.1109/ICIECS.2010.5678223).
- [50] C.-H. Hallard, *NRF24L01 real life range test*, 2013. [Online]. Available: <http://hallard.me/nrf24l01-real-life-range-test/> (visited on 02/05/2015).
- [51] A. Opel, “Bluetooth - Authentication - Authorisation - Encryption”, p. 1, 2003.
- [52] Atmel, “Atmel AT02845 : Coexistence between ZigBee and Other 2.4GHz Products”, *Application Note*, 2013.
- [53] R. Balani, “Energy consumption analysis for bluetooth, wifi and cellular networks”, *Networked & Embedded Systems Laboratory, NESL Technical Report TR-UCLA-NESL-200712-01*, 2007.
- [54] ATMEL, “AT03663 : Power Consumption of ZigBee End Device”, 2014.
- [55] D. P. Consumption, D. Halperin, B. Greenstein, A. Sheth, and D. Wetherall, “Demystifying 802.11n Power Consumption”, *Proceedings of the 2010 Workshop on Power Aware Computing and Systems (HotPower’10)*, pp. 2–6, 2010. DOI: [10.1.1.173.7044](https://doi.org/10.1.1.173.7044).
- [56] Android Developers, *What is Android*, 2011. [Online]. Available: <http://developer.android.com/guide/basics/what-is-android.html>.
- [57] AOSP, *ART and Dalvik | Android Developers*. [Online]. Available: <http://source.android.com/devices/tech/dalvik/> (visited on 02/05/2015).
- [58] Microsoft, *Windows 10 for Raspberry Pi 2*, 2015. [Online]. Available: <https://dev.windows.com/en-us/featured/raspberrypi2support> (visited on 02/05/2015).
- [59] Raspberry Pi Foundation, *What is a Raspberry Pi?* [Online]. Available: <http://www.raspberrypi.org/help/what-is-a-raspberry-pi/> (visited on 02/05/2015).
- [60] —, “Raspberry Pi Model B+ datasheet”, p. 1, 2014.
- [61] —, *Raspberry Pi Compute Module: new product!*, 2014. [Online]. Available: <http://www.raspberrypi.org/raspberry-pi-compute-module-new-product/> (visited on 02/05/2015).
- [62] Marvell Technology Group, “Sheeva Plug”, 2012. [Online]. Available: <https://www.globalscaletechnologies.com/p-46-sheevaplug-dev-kit.aspx>.
- [63] Marvel Semiconductor, “Marvel MV78200 SoC with Sheeva Technology”, pp. 1–2,

- [64] PlugPBX Project, *About*. [Online]. Available: <http://www.plugpbx.org/> (visited on 02/06/2015).
- [65] Memsic, “TelosB datasheet”, 2013. [Online]. Available: [http://www.memsic.com/userfiles/files/DataSheets/WSN/telosb%5C\\_datasheet.pdf](http://www.memsic.com/userfiles/files/DataSheets/WSN/telosb%5C_datasheet.pdf).
- [66] Texas Instruments, “2.4-GHz Bluetooth ® low energy System-on-Chip”, no. June, 2013.
- [67] Nordic Semiconductor, *nRF51822 Product Specification v3.1*, 2014.
- [68] J. Figueiras, H. Schwefel, and I. Kovacs, “Accuracy and timing aspects of location information based on signal-strength measurements in Bluetooth”, in *Personal, Indoor and Mobile Radio Communications, 2005. PIMRC 2005. IEEE 16th International Symposium on*, vol. 4, 2005, 2685–2690 Vol. 4. DOI: [10.1109/PIMRC.2005.1651931](https://doi.org/10.1109/PIMRC.2005.1651931).
- [69] M. Barahim, “Low-cost bluetooth mobile positioning for location-based application”, *Internet, 2007. ICI 2007. ...*, 2007. [Online]. Available: [http://ieeexplore.ieee.org/xpls/abs%5C\\_all.jsp?arnumber=4401707](http://ieeexplore.ieee.org/xpls/abs%5C_all.jsp?arnumber=4401707).
- [70] S. Malik and H. Maidasani, *Cat Gear*, College Park, 2014. [Online]. Available: <http://cmssc838f-s14.wikispaces.com/Cat+Gear>.
- [71] C. H. O. Hyunggi, K. Myungseok, K. I. M. Jonghoon, and K. I. M. Hagbae, “Zigbee based location estimation in home networking environments”, *IEICE transactions on information and systems*, vol. 90, no. 10, pp. 1706–1708, 2007.
- [72] W.-H. Kuo, Y.-S. Chen, G.-T. Jen, and T.-W. Lu, “An intelligent positioning approach: RSSI-based indoor and outdoor localization scheme in Zigbee networks”, in *Machine Learning and Cybernetics (ICMLC), 2010 International Conference on*, vol. 6, 2010, pp. 2754–2759. DOI: [10.1109/ICMLC.2010.5580783](https://doi.org/10.1109/ICMLC.2010.5580783).
- [73] J. Rekimoto, T. Miyaki, and T. Ishizawa, “LifeTag: WiFi-based continuous location logging for life pattern analysis”, in *LoCA*, vol. 2007, 2007, pp. 35–49.
- [74] R. Paradiso, G. Loriga, and N. Taccini, “A wearable health care system based on knitted integrated sensors”, *Information Technology in Biomedicine, IEEE Transactions on*, vol. 9, no. 3, pp. 337–344, 2005, ISSN: 1089-7771. DOI: [10.1109/TITB.2005.854512](https://doi.org/10.1109/TITB.2005.854512).
- [75] Fundació Privada CETEMMSA, Telefónica Investigación y Desarrollo, INESC Porto – Instituto de Engenharia de Sistemas e Computadores do Porto, University of Plymouth Enterprise Ltd, University of Limerick, Corscience GmbH & Co KG, Fundació Hospital Comarcal Sant Antoni Abat, Fraunhofer Portugal, L. TeleMedic Systems, Zentrum für Kardiovaskuläre Telemedizin GmbH, and G. National University of Ireland, “eCAALYX”, 2009. [Online]. Available: <http://www.aal-europe.eu/projects/ecaalyx/>.
- [76] Fraunhofer Portugal AICOS, BCB Informática y Control SL, Università degli Studi di Ferrara, KOHS PIMEX, Portugal Telecom Comunicações, Ab.Acus Srl, Grado Zero Espace, and K. RK Tech, “Clockwork”, 2014. [Online]. Available: <http://www.aal-europe.eu/projects/clockwork/>.

- [77] Scuola Universitaria Professionale della Svizzera Italiana (SUPSI), D. d. E. e. I. Politecnico di Milano, Info Solution SpA, VCA Technology Ltd., Istituti Sociali di Chiasso, Clinica Hildebrand, and University of Wurzburg, “ALMA”, 2013. [Online]. Available: <http://www.aal-europe.eu/projects/alma/>.
- [78] C. Styles and ARM Ltd, *mbed FAQs*, 2010. [Online]. Available: [https://developer.mbed.org/media/uploads/chris/mbedqa%5C\\_v1.0.pdf](https://developer.mbed.org/media/uploads/chris/mbedqa%5C_v1.0.pdf).
- [79] Nordic Semiconductor, “nRF51822 Evaluation Kit User Guide v1.2”, 2013.
- [80] —, *Introduction to the S110 SoftDevice*, 2011. [Online]. Available: [https://devzone.nordicsemi.com/documentation/nrf51/4.2.0/html/group%5C\\_%5C\\_nrf518%5C\\_%5C\\_lib%5C\\_%5C\\_ble%5C\\_%5C\\_s110%5C\\_%5C\\_intro.html](https://devzone.nordicsemi.com/documentation/nrf51/4.2.0/html/group%5C_%5C_nrf518%5C_%5C_lib%5C_%5C_ble%5C_%5C_s110%5C_%5C_intro.html).
- [81] *Region RAM overflowed with stack - Nordic Developer Zone*. [Online]. Available: <https://devzone.nordicsemi.com/question/38781/region-ram-overflowed-with-stack/> (visited on 06/15/2015).
- [82] —, *nRF51 IoT SDK Documentation*. [Online]. Available: [https://developer.nordicsemi.com/nRF51%5C\\_IoT%5C\\_SDK/doc/iot/html/index.html](https://developer.nordicsemi.com/nRF51%5C_IoT%5C_SDK/doc/iot/html/index.html) (visited on 06/16/2015).
- [83] J. Hui and P. Thubert, “RFC 6282”, 2011.
- [84] Nordic Semiconductor, *nRF51 SDK for Internet of Things applications using Bluetooth Smart*. [Online]. Available: <https://www.nordicsemi.com/eng/Products/Bluetooth-Smart-Bluetooth-low-energy/nRF51-IoT-SDK> (visited on 06/15/2015).
- [85] *lwIP - A Lightweight TCP/IP stack - Summary [Savannah]*. [Online]. Available: <https://savannah.nongnu.org/projects/lwip/> (visited on 06/16/2015).
- [86] E. Stock, *Memory*, 2015. [Online]. Available: <https://github.com/eliotstock/memory>.
- [87] Fraunhofer Portugal, *Smart Companion UI Design & SC-Lib*, 2014.
- [88] *What’s the maximum size for an advertisement package? - Nordic Developer Zone*. [Online]. Available: <https://devzone.nordicsemi.com/question/75/whats-the-maximum-size-for-an-advertisement-package/> (visited on 06/19/2015).
- [89] Radius Networks, *AltBeacon Protocol Specification v1.0*, 2015. [Online]. Available: <https://github.com/AltBeacon/spec>.
- [90] E.-E.-L. Lau, B.-G. Lee, S.-C. Lee, and W.-Y. Chung, “Enhanced RSSI-based high accuracy real-time user location tracking system for indoor and outdoor environments”, *International Journal on Smart Sensing and Intelligent systems*, vol. 1, no. 2, pp. 534–548, 2008.
- [91] M. Slade, *Boyer Moore Horspool Algorithm - YouTube*, 2014. [Online]. Available: <https://www.youtube.com/watch?v=PHXAOKQk2dw> (visited on 06/22/2015).

- [92] R. S. Boyer and J. S. Moore, “A fast string searching algorithm”, *Communications of the ACM*, vol. 20, no. 10, pp. 762–772, 1977.
- [93] A. Hume and D. Sunday, “Fast string searching”, *Software: Practice and Experience*, vol. 21, no. 11, pp. 1221–1248, 1991.
- [94] L. Olimex, *Weekend Programming Challenge Issue 16 – Infix to Postfix converter*. [Online]. Available: <https://olimex.wordpress.com/2013/07/05/weekend-programming-challenge-issue-16-infix-to-postfix-converter/> (visited on 06/23/2015).
- [95] E. W. Dijkstra, *ALGOL-60 translation*. Mathematisch Centrum, 1961.
- [96] Stack Overflow, *c++ - What is the running time of the translation of infix to postfix using queue and stack?* [Online]. Available: <https://stackoverflow.com/questions/5305215/what-is-the-running-time-of-the-translation-of-infix-to-postfix-using-queue-and> (visited on 06/28/2015).
- [97] Rosetta Code, *Parsing/Shunting-yard algorithm*. [Online]. Available: [http://rosettacode.org/wiki/Parsing/Shunting-yard%5C\\_algorithm](http://rosettacode.org/wiki/Parsing/Shunting-yard%5C_algorithm) (visited on 06/23/2015).
- [98] S. R. Schmitt, *RPN Calculator*, 2004. [Online]. Available: [http://www.abecedarical.com/javascript/script%5C\\_reverse%5C\\_polish.html](http://www.abecedarical.com/javascript/script%5C_reverse%5C_polish.html) (visited on 06/23/2015).
- [99] Rosetta Code, *Parsing/RPN calculator algorithm*. [Online]. Available: [http://rosettacode.org/wiki/Parsing/RPN%5C\\_calculator%5C\\_algorithm%5C#C](http://rosettacode.org/wiki/Parsing/RPN%5C_calculator%5C_algorithm%5C#C) (visited on 06/23/2015).
- [100] A. Dobie and Android Central, *Bluetooth Low Energy support coming to future Android version | Android Central*. [Online]. Available: <http://www.androidcentral.com/bluetooth-low-energy-support-coming-future-android-version> (visited on 06/23/2015).
- [101] Stack Overflow, *bluetooth lowenergy - Does BluetoothLeAdvertiser work on a Nexus 5 with Android 5.0? - Stack Overflow*. [Online]. Available: <https://stackoverflow.com/questions/26441785/does-bluetoothleadvertiser-work-on-a-nexus-5-with-android-5-0/26441948%5C#26441948> (visited on 06/28/2015).
- [102] Nordic Semiconductor, *nRF51 IoT SDK Documentation*, 2014.
- [103] OASIS, *MQTT Version 3.1.1*. [Online]. Available: <http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html> (visited on 06/28/2015).
- [104] Nordic Semiconductor, *IoT SDK v0.8.0 Changelog*, 2015. [Online]. Available: <https://www.nordicsemi.com/eng/nordic/Products/nRF51-IoT-SDK/nRF51-IoT-SDK-zip/41601> (visited on 06/26/2015).
- [105] A. Dunkels, *uIP*, 2013. [Online]. Available: <https://github.com/adamdunkels/uip> (visited on 06/29/2015).

- [106] O. Schneider, *Pimatic*, 2014. [Online]. Available: <http://pimatic.org/> (visited on 06/26/2015).