
Authors: Séverin MARCOMBES
Bastien PAUL

Nowadays, more and more standard protocols enable consumer electronic devices to communicate together, and even to control each other. This offers new possibilities to the users, as they are now able to control several devices from another. Such functionalities are interesting to quadriplegic people, to whom the electronic world could bring more independence and thus a better quality of life.

However, most interfaces between quadriplegic people and the electronic world suffer from limited possibilities and are cosmetically difficult to accept. To address these issues, TKS developed an intra-oral tongue controlled device providing a good hardware interface between the user and the electronic world. Nevertheless, TKS's system can only control a computer at the time of this writing.

This project is intended in evolving TKS's tongue controlled system, by making it compatible with standard protocols for connectivity to other devices, service discovery and service access. It will also provide efforts to endow the current system with a visual interface, and offer an analysis of the best way to present devices to the user on this interface.

During semester 9th, the first part of this project enabled us to design a proof of concept providing a highly interoperable tongue controlled interface for quadriplegic people. This proof of concept is able to take control of Bluetooth enabled computers and Zigbee enabled lights. It was granted an award in France from the association Hanploi in March 2010.

The second part of this project focuses on improving this proof of concept on three different axis: energy consumption, security and safety. Our results indicate that, by including a Power Management System in the HAMAC framework, we are able to make the HAMAC platform 2.5 times more energy efficient and the radio chips up to 8 times more energy efficient. The other changes that we applied to the framework enable it to encrypt the data sent over the air by our proof of concept. They also address the safety of specific plug-ins such as a module for controlling a motorized wheelchair.

2010

HAMAC

Home Automation & Mobile Access Controller



Séverin MARCOMBES
Bastien PAUL

Aalborg Universitet
10th semester 2010

Abstract**TITLE:**

Home Assistive and Mobile Access Controller
(HAMAC)

SUBJECT:

Electronic/Software engineering
Wireless communications
Embedded systems

GROUP:

S-628-A / D-620-A

GROUP MEMBERS:

Séverin MARCOMBES
Bastien PAUL

SUPERVISORS:

Alexandre DAVID
Yannick LE MOULLEC

SEMESTER:

SSE4

PROJECT PERIOD:

01/2-2009 to 11/6-2010

COPIES:

5

Number of pages :

109

Supplement:

CD-ROM: ☐ Yes ☐ No

Nowadays, more and more standard protocols enable consumer electronic devices to communicate together, and even to control each other. This offers new possibilities to the users, as they are now able to control several devices from another. Such functionalities are interesting to quadriplegic people, to whom the electronic world could bring more independence and thus a better quality of life.

However, most interfaces between quadriplegic people and the electronic world suffer from limited possibilities and are cosmetically difficult to accept. To address these issues, TKS developed an intra-oral tongue controlled device providing a good hardware interface between the user and the electronic world. Nevertheless, TKS's system can only control a computer at the time of this writing.

This project is intended in evolving TKS's tongue controlled system, by making it compatible with standard protocols for connectivity to other devices, service discovery and service access. It will also provide efforts to endow the current system with a visual interface, and offer an analysis of the best way to present devices to the user on this interface.

During semester 9th, the first part of this project enabled us to design a proof of concept providing a highly interoperable tongue controlled interface for quadriplegic people. This proof of concept is able to take control of Bluetooth enabled computers and Zigbee enabled lights. It was granted an award in France from the association Hanploi in March 2010.

The second part of this project focuses on improving this proof of concept on three different axis: energy consumption, security and safety. Our results indicate that, by including a Power Management System in the HAMAC framework, we are able to make the HAMAC platform 2.5 times more energy efficient and the radio chips up to 8 times more energy efficient. The other changes that we applied to the framework enable it to encrypt the data sent over the air by our proof of concept. They also address the safety of specific plug-ins such as a module for controlling a motorized wheelchair.

CONTENTS

| | |
|---|-----|
| Contents..... | 3 |
| Acronyms and abbreviations | 5 |
| List of tables..... | 6 |
| List of figures..... | 7 |
| Related work..... | 9 |
| 1 Introduction | 13 |
| 1.1 Context | 13 |
| 1.1.1 The rise of inter-devices communication | 13 |
| 1.1.2 The case of severely disabled people | 13 |
| 1.1.3 Researches on man to device interaction for disabled people | 13 |
| 1.1.4 In-mouth tongue controlled systems | 14 |
| 1.1.5 TKS's system | 15 |
| 1.1.6 Limitations of TKS's system | 15 |
| 1.1.7 The HAMAC proof of concept..... | 16 |
| 1.1.8 Features to address | 18 |
| 1.2 Problem statement..... | 18 |
| 1.3 Project output..... | 19 |
| 1.4 Project scope | 19 |
| 1.5 Project assumptions | 20 |
| 2 Preliminaries | 21 |
| 2.1 Problem Analysis | 21 |
| 2.1.1 Functional need of quadriplegics..... | 21 |
| 2.1.2 Analysis of standard protocol for device interoperability | 23 |
| 2.2 Defining HAMAC | 25 |
| 2.2.1 The HAMAC Framework | 25 |
| 2.2.2 Our proof of concept | 32 |
| 3 Primary axes of improvement | 36 |
| 3.1 Make HAMAC more energy efficient..... | 36 |
| 3.1.1 HAMAC Platform..... | 36 |
| 3.1.2 Reducing the consumption of the Radio Chips..... | 57 |
| 3.2 Making HAMAC more secure | 81 |
| 3.2.1 Defining the security needs of HAMAC | 81 |
| 3.2.2 Data Security..... | 82 |
| 3.2.3 Safety | 88 |
| 4 Additional Axis of Improvements | 92 |
| 4.1 New core features | 92 |
| 4.1.1 Event Manager..... | 92 |
| 4.1.2 File Manager | 92 |
| 4.2 Plugins..... | 93 |
| 4.2.1 Plugins in HAMAC | 94 |
| 4.2.2 Architecture of a plugin | 97 |
| 4.2.3 Plugin manager | 98 |
| 5 Summary & Conclusion..... | 99 |
| 5.1 Achievements | 99 |
| 5.1.1 Energy consumption | 99 |
| 5.1.2 Security & Safety..... | 100 |
| 5.1.3 Other work..... | 100 |
| 5.2 What should be improved | 100 |

| | | |
|-------|----------------------------|-----|
| 5.3 | Future works | 101 |
| 5.3.1 | Short term objectives..... | 101 |
| 5.3.2 | Long term objectives..... | 102 |
| 5.4 | Word of the end | 102 |
| 6 | Appendix | 103 |
| 6.1 | Project Methodology | 103 |
| 6.1.1 | A3 Model | 103 |
| 6.1.2 | UML..... | 105 |
| 6.1.3 | “2 Track Up” model..... | 106 |
| 6.1.4 | Realization concepts | 108 |

ACRONYMS AND ABBREVIATIONS

| | |
|---|-----|
| ACPI: Advanced Configuration and Power Interface | 44 |
| AES: Advanced Encryption Standard..... | 86 |
| APM: Advanced Power Management | 44 |
| CPU: Central Processing Unit | 41 |
| DFVS: Dynamic Frequency and Voltage Scaling | 41 |
| GUI: Graphical User Interface | 95 |
| HA: Home Automation | 19 |
| HAMAC: Home Assistive and Mobile Access Controller | 16 |
| HID: Human Interface Device..... | 19 |
| ISM: Industrial, Scientific and Medical | 60 |
| MIC: Message Integrity Code | 86 |
| OMG: Object Management Group..... | 105 |
| PMS: Power Management System..... | 36 |
| UML: Unified Modeling Language..... | 26 |

LIST OF TABLES

| | |
|---|----|
| Table 1: Possible network stack combinations for HAMAC | 24 |
| Table 2: features of AT91SAM9263 microcontroller | 33 |
| Table 3: features of AT91SAM9263-EK development board | 33 |
| Table 4: Power characteristics of main hardware parts [12; 13; 14; 15] | 39 |
| Table 5: Current consumption of a SDRAM according to different mode | 43 |
| Table 6: Four states in which the system can be are defined | 45 |
| Table 7: Screen modes | 46 |
| Table 8: Power consumption of main components present in HAMAC's board | 53 |
| Table 9: CPU and SDRAM power consumption in conservative and sleep mode | 55 |
| Table 10: Power consumption in different modes | 56 |
| Table 11: Power consumption of HAMAC according to the use case defined in section 3.1.1.5.2. | 56 |
| Table 12: Summary of the strategies for making the radio chips consume less energy | 68 |
| Table 13: Summary of the main parameters impacting our strategies for lowering the energy consumption of the radio chips | 70 |
| Table 14: List of factors impacting the probability that a user will not need to take back the control of an inactive connection soon | 72 |
| Table 15: List of factors impacting the probability of finding more devices | 72 |
| Table 16: Main constraints of the states, applicable to a communication link or to a chip | 75 |
| Table 17: Energy consumption data for the CSR BlueCore4 chip – Supply 1.8V | 77 |
| Table 18 : Estimated power consumption of a Bluetooth chip during normal operation | 78 |
| Table 19: Estimated power consumption of the Bluetooth chip while using some of our strategies... | 79 |
| Table 20: Estimated time spent by a Bluetooth chip in each state, during a day | 79 |

LIST OF FIGURES

| | |
|---|----|
| Figure 1: TKS's tongue-controlled system [5] | 15 |
| Figure 2: TKS's system extended by the HAMAC device | 17 |
| Figure 3: Element Class..... | 26 |
| Figure 4: Device class..... | 27 |
| Figure 5: Service class | 28 |
| Figure 6: ConcurrentList class | 29 |
| Figure 7: GUI class and classes representing devices and services | 30 |
| Figure 8: Protocol classes | 31 |
| Figure 9: ProtocolManager class contains the list of Protocols instance | 32 |
| Figure 10: Power Management System Design Process. | 37 |
| Figure 11: A ³ model meets Power Management System design process | 37 |
| Figure 12: State transition diagram of microcontroller and SDRAM..... | 46 |
| Figure 13: State transition diagram of screen display. | 47 |
| Figure 14: Timer class | 48 |
| Figure 15: activity diagram of two implementations of timer | 49 |
| Figure 16: Power Management System Class | 49 |
| Figure 17: Activity diagram representing tasks performed on time out and when an input is detected. | 50 |
| Figure 18: Service Class..... | 50 |
| Figure 19: Activity diagram of methods which set the <i>power</i> and <i>screen</i> flags. | 51 |
| Figure 20: Daily schedule of Mr. Nielsen. Numbers indicate the time of the day. | 54 |
| Figure 21: Items in Mr. Nielsen's daily schedule | 55 |
| Figure 22: Hamac architecture with a wheelchair module | 88 |
| Figure 23: Inputs from the tongue system are given both to HAMAC's main board and to the wheelchair module | 89 |

| | |
|---|-----|
| Figure 24: Procedure in the wheelchair control | 90 |
| Figure 25: EventManager class | 92 |
| Figure 26: FileManager Class | 93 |
| Figure 27: Component diagram of the plugin system..... | 94 |
| Figure 28: Device plugin components..... | 95 |
| Figure 29: Service plugin components | 95 |
| Figure 30: protocol plugin components..... | 96 |
| Figure 31: Protocol specific device plugin component | 96 |
| Figure 32: Protocol specific service plugin components..... | 97 |
| Figure 33: Plugin package. It contains all classes necessary to define a plugin..... | 98 |
| Figure 34: Plugin manager class. This class is the central point of the plugin system..... | 98 |
| Figure 35: A3 model, the A3 model divides the development and implementation of an algorithm into three distinct focus areas, namely Application, Algorithm and Architecture..... | 103 |
| Figure 36: UML Logo | 105 |
| Figure 37: Typical UML class | 106 |
| Figure 38: The 2 Track Up model separates requirements and architecture. Then it is based on progressive methods..... | 108 |

RELATED WORK

1. **Bluetooth SIG.** *Bluetooth Core Specification v2.1 + EDR.* 2007.
2. **ZigBee Alliance.** *ZIGBEE SPECIFICATION.* s.l. : ZigBee Alliance, 2008.
3. **Digital Living Network Alliance.** Home - DLNA. *www.dlna.org.* [Online] [Citation : 10 06 2010.] <http://www.dlna.org/home>.
4. **Andreasen Struijk, Lotte N. S., et al., et al.** *Fully Integrated Wireless Inductive Tongue Computer Interface for Diasabled People.* 2009.
5. **Enemark Lund, Morten, et al., et al.** *A Framework for Mouse and Keyboard Emulation in a Tongue Control System.* 2009.
6. **Séverin Marcombes, Bastien Paul.** *HAMAC: Home Automation & Mobile Access Controler.* Aalborg : s.n., 2010.
7. **ZHU Feng, MUTKA Matt, NI Lionel.** *Classification of Service Discovery in Pervasive Computing Environements.* 2005. 1536-1268.
8. IPv6 over Low power WPAN (6lowpan). *datatracker.ietf.org.* [Online] [Citation : 10 06 2010.] <http://datatracker.ietf.org/wg/6lowpan/charter/>.
9. Vector - C++ Reference. *C++ Reference.* [Online] [Citation : 16 December 2009.] <http://www.cplusplus.com/reference/stl/vector/>.
10. **Texas Instrument.** CC2530 Development Kit - CC2530DK - TI Tool Folder. *www.ti.com.* [Online] Texas Instrument. [Citation : 09 06 2010.] <http://focus.ti.com/docs/toolsw/folders/print/cc2530dk.html>.
11. *Power Reduction Techniques For Microprocessor Systems.* **VASANTH VENKATACHALAM, MICHAEL FRANZ.** 3, Irvine : ACM Computing Surveys, 2005, Vol. 37.
12. **Atmel.** AT91SAM9263 Preliminary. *atmel.com.* [Online] 2009. [Citation : 16 03 2010.] http://www.atmel.com/dyn/resources/prod_documents/doc6249.pdf.
13. SYNCHRONOUS DRAM. *www.micron.com/dramds.* [Online] [Citation : 16 03 2010.] <http://www.datasheetcatalog.org/datasheet/micron/MT48LC32M8A2.pdf>.
14. **Texas Instrument.** low Power RF ICs - 2.4GHz CC2530 - TI.com. *www.ti.com.* [Online] 04 2009. [Citation : 20 05 2010.] <http://focus.ti.com/lit/ds/symlink/cc2530.pdf>.
15. LCD Screen. *www.hitachi-displays-eu.com.* [Online] [Citation : 16 03 2010.] <http://www.hitachi-displays-eu.com/doc/TX09D71VM1CCA.pdf>.
16. ASIC-System On Chip (SoC)-VLSI Design. *asic-soc.* [Online] [Citation : 07 05 2010.] <http://asic-soc.blogspot.com/2008/04/clock-gating.html>.

17. **Dale, Mitch.** Utilizing Clock-Gating Efficiency to Reduce Power. *edadesignline*. [Online] [Citation : 07 05 2010.] <http://www.edadesignline.com/howto/205800151>.
18. *Scheduling for Reduced CPU Energy.* **Mark Weiser, Brent Welch, Alan Demers, Scott Shenker.** 1994.
19. *Comparing Algorithms for Dynamic Speed-Setting of a Low-Power CPU.* **Kinshuk Govil, Edwin Chan, Hal Wasserman.** 1995.
20. *Process Cruise Control: Event-Driven Clock Scaling for Dynamic Power Management.* **Andreas Weissel, Frank Bellosa.** s.l. : ACM Press, 2002, Vol. Proceedings of the International Conference on Compilers, Architecture, and Synthesis for Embedded Systems. 238–246.
21. *Improving Dynamic Voltage Scaling Algorithms with PACE.* **Smith, Jacob R. Lorch and Alan Jay.** Berkeley : s.n., 2001.
22. *Hard real-time scheduling for low-energy using stochastic data and DVS processors.* **Gruian, Flavius.** Lund, Sweden : s.n., 2001.
23. *Dynamic Processor Throttling for Power Efficient Computations.* **Masaaki KONDO, Hiroshi NAKAMURA.** Tokyo : s.n., 2004.
24. *Dynamic voltage and frequency scaling based on workload decomposition.* **Kihwan Choi, Ramakrishna Soma, Massoud Pedram.** Los Angeles : s.n., 2004.
25. **Brodowski, Dominik.** Linux Kernel Documentation :: cpu-freq : user-guide.txt. *Linux Kernel Documentation.* [Online] [Citation : 07 05 2010.] <http://www.mjmwired.net/kernel/Documentation/cpu-freq/user-guide.txt>.
26. *The Synergy between Power-aware Memory Systems and Processor Voltage Scaling.* **Xiaobo Fan, Carla S. Ellis, Alvin R. Lebeck.** Durham : s.n., 2003.
27. **Atmel.** Atmel Products - Tools and Software. *Amel.* [Online] [Citation : 15 03 2010.] http://www.atmel.com/dyn/resources/prod_documents/doc6341.pdf.
28. **Brodowski, Dominik.** Linux Kernel Documentation :: cpu-freq : user-guide.txt. *mjm wired.* [Online] [Citation : 16 05 2010.] <http://www.mjmwired.net/kernel/Documentation/cpu-freq/user-guide.txt>.
29. **Terrehon Bowden, Bodo Bauer, Jorge Nerin, Shen Feng, Stefani Seibold.** <http://kernel.org/doc/Documentation/filesystems/proc.txt>. *kernel.org.* [Online] [Citation : 16 05 2010.] <http://kernel.org/doc/Documentation/filesystems/proc.txt>.
30. *getrusage(2): resource usage - Linux man page.* *linux.die.net.* [Online] [Citation : 16 05 2010.] <http://linux.die.net/man/2/getrusage>.
31. **Intel, Microsoft.** download.microsoft.com. *download.microsoft.com.* [Online] 1996. [Citation : 17 05 2010.] <http://download.microsoft.com/download/1/6/1/161ba512-40e2-4cc9-843a-923143f3456c/APMV12.rtf>.

32. **Hewlett-Packard Corp., Intel Corp., Microsoft Corp., Phoenix Technologies Ltd., Toshiba Corp.** . ACPI. *acpi.info*. [Online] 05 04 2010. [Citation : 17 05 2010.] <http://www.acpi.info/DOWNLOADS/ACPIspec40a.pdf>.

33. *Power Management in Linux-Based Systems*. **Srivatsa Vaddagiri, Anand K. Santhanam, Vijay Sukthankar, Murali Iyer**. 119, 2004.

34. **Sandra Loosemore, Richard M. Stallman, Roland McGrath, Andrew Oram, and Ulrich Drepper**. *www.gnu.org*. [Online] 27 10 2007. [Citation : 17 05 2010.] <http://www.gnu.org/software/libc/manual/>.

35. **Jonathan Corbet, Greg Kroah-Hartman, Alessandro Rubini**. *Linux Device Drivers*. s.l. : O'Reilly, 2005. p. 636. 0-596-00590-3.

36. **ATMEL**. Atmel products - Application notes. *Atmel*. [Online] 23 11 2006. [Citation : 28 05 2010.] http://www.atmel.com/dyn/resources/prod_documents/doc6217.pdf.

37. **CSR**. Bluecore4 ROM CSP EDR. [Datasheet]. September 2005.

38. **Morrow, Robert**. *Bluetooth - Operation and Use*. s.l. : McGraw-Hill, 2002. ISBN-10: 007138779X.

39. *Allemagne : 100 euros d'amende en cas de connexion Wi-Fi non sécurisée*. **ZDNet**. 14 05 2010. <http://www.zdnet.fr/actualites/allemande-100-euros-d-amende-en-cas-de-connexion-wi-fi-non-securisee-39751669.htm>.

40. *Skyblog victime d'une tentative de piratage*. **LEMONDE.FR**. 24 05 2010. http://www.lemonde.fr/technologies/article/2010/05/24/skyblog-victime-d-une-tentative-de-piratage_1362127_651865.html.

41. **Alan Burns, Andy Wellings**. *Real-Time Systems and Programming Languages*. 4th. s.l. : Addison Wesley, 2009. p. 23.

42. **Daintree Networks**. Zigbee Security - Daintree Networks. *Daintree*. [Online] [Citation : 02 06 2010.] <http://www.daintree.net/resources/security.php>.

43. **Gascón, David**. Wireless Sensor Network Research Group. *sensor-networks*. [Online] 05 02 2009. [Citation : 02 06 2010.] <http://www.sensor-networks.org/index.php?page=0903503549>.

44. **The Ganssle Group**. Great Watchdogs. *ganssle*. [Online] 01 2004. [Citation : 01 06 2010.] <http://www.ganssle.com/watchdogs.pdf>.

45. Serialisation. *www.boost.org*. [Online] [Citation : 23 03 2010.] http://www.boost.org/doc/libs/1_42_0/libs/serialization/doc/index.html.

46. Sweet Persist - Overview . *www.sweetsoftware.co.nz*. [Online] [Citation : 23 03 2010.] http://www.sweetsoftware.co.nz/persist_overview.php.

47. s11n.net object serialization/persistence in C++. *http://s11n.net/*. [Online] [Citation : 23 03 2010.] <http://s11n.net/>.

48. JSON. *www.json.org*. [Online] [Citation : 23 03 2010.] <http://www.json.org/>.
49. Extensive Markup Language (XML) 1.0 (Fifth Edition). *www.w3.org*. [Online] [Citation : 23 03 2010.] <http://www.w3.org/TR/REC-xml/>.
50. **Kristensen, Jes Toft, Bjerrum, Bo et Kristiansen, Klaus Dahl.** *Noise Reduction for Hands-free Car-phone*. Aalborg : AAU, 2007.
51. **Corneliussen, Andreas, et al., et al.** *Evaluation of FPGA based Turbo Coding Implementations - on a soft-core processor with hardware acceleration*. Aalborg : AAU, 2009.
52. **Peter August Simonsen, Jes Toft Kristensen.** *DS-CDMA Procedures with the Cell Broadband Engine*. Aalborg : AAU, 2007.
53. UML. *foldoc*. [Online] <http://foldoc.org/UML>.
54. **Piechocki, Laurent.** *UML en français*. [Online] <http://uml.free.fr>.
55. **Rumbaugh, James.** *Object-oriented modeling and design*. s.l. : Prentice Hall, 1991.
56. **Booch, Grady.** *Object-oriented analysis and design with applications*. s.l. : Redwood City, 1994.
57. **Jacobson, Ivar.** *Object-oriented software engineering : a use case driven approach*. s.l. : ACM Press, 1992.
58. Object Management Group. [Online] <http://www.omg.org/>.
59. Conduite de Projet. [Online] 2004. dept-info.labri.fr/~counilh/systeme-d-information/SI_0202.pdf.
60. The RAI (Resource Acquisition Is Initialisation) Programming Idiom. *www.hackcraft.net*. [Online] [Citation : 13 04 2010.] <http://www.hackcraft.net/raii/>.

1 INTRODUCTION

1.1 CONTEXT

1.1.1 THE RISE OF INTER-DEVICES COMMUNICATION

Electronic devices are everywhere. Devices such as computers and mobile phones have become indispensable for living every day. They provide services enabling to work better, to communicate better, or to relax in new kind of ways. This brings a new dimension of comfort to their users, who can now get without moving far more possibilities than they had in former times. For practical purposes, one can for example replace meetings by videoconferences or prefer shopping online than going to the grocery store.

This art of minimizing the efforts of the user by bringing services directly to him has crossed a new step with the recent tendency of trying to make devices communicate with each other. Technologies such as Bluetooth [1], Zigbee [2], or the emerging DLNA [3] protocol have the common goal of making devices interact with each other. This enables devices to share the services that they offer, by controlling one another. From the user side, this pushes the comfort further by reducing the effort required to move from device to device to get these services. These technologies enable for example to control the cursor of a computer from a mobile phone. Thanks to a home automation gateway, this mobile phone can also turn the lights of the house on and off. Instead of being accessible through a set of scattered devices, the services are thus reachable directly from the hand of the user.

1.1.2 THE CASE OF SEVERELY DISABLED PEOPLE

This kind of new possibilities is often the target a lot of criticism from the promoters of a healthy life. These technologies tend in fact to push the lazy users to remain sat down all day long, by reducing their need to move to do what they want. They become really helpful, however, when the user has no other choice than to stay sat down. People assigned for a long time or for life time in a wheelchair are an example of users in that case, to which this kind of technologies could really benefit.

A subset of these users, suffering from heavy malfunctioning of the motor system, is even more dependent on the ease of access offered by such technologies. These people are suffering from high spinal-cord injury, brain injury, or other impairment precluding them from using a large part of their body, up to their shoulders. This disability affects heavily their everyday life, by making it difficult for them to live without external help. The basics of social life, such as going out or having a job, are difficult to satisfy in this situation. This impacts directly their quality of life, by making them heavily dependent on others.

1.1.3 RESEARCHES ON MAN TO DEVICE INTERACTION FOR DISABLED PEOPLE

To remove this dependency, a lot of researches have been made to help these people interact with electronic devices. Accesses to the control of motorized solutions and to the digital

world are in fact two key factors for improving the quality of life of these people. These would help increasing their self-supportiveness by enabling them to control a wheelchair or to work on computers, for example. A lot of techniques were thus imagined, to exploit the different ways that these people have to interact with the world so that they can be used to control a device. A whole set of controlling devices were developed to exploit physical movements from the user, such as head movements, eye movements and blinking, or variations in breath intensity. Two other ideas were also to make electronic devices react to orders from their user, through voice recognition or measurements of the neurological activity.

Of all these innovative technologies, however, most have failed in offering a solution both easy to use and to accept. They often require heavy installation or activation procedures, which reduce the independence of the user that they aim to bring. Most often, the user is thus tied to a single device, and requires assistance to begin or to finish interacting with it. A repeatedly neglected aspect of these control devices is also their appearance. Although providing efficient ways of controlling electronic devices is their main objective, they should avoid enlarging the gap between disabled people and the others by making them look even more different. Most of these devices take in fact the shape of helmets, hiding part of the head of their users. Others require surgical intervention, or the constant use of unnatural movements of the head or of the face.

1.1.4 IN-MOUTH TONGUE CONTROLLED SYSTEMS

To reduce these drawbacks, several researches on in-mouth tongue controlled systems have been made in the past few years. Such systems have in fact the advantages of being discrete while being able to use the complex movements of the tongue to transmit orders to other devices. They respond to the user's need to look less different than the other people, by hiding the assistive device entirely in the mouth.

One key challenge for this in-mouth integration to be successful is to develop a system that does not prevent the user from using his or her mouth as normally as possible while in use. The goal of the device being to provide better independence to the user, he or she should be able to speak while the device is in the mouth. The requirement for the device to be discrete also implies that the user should be able to use the device without making unusual movements of the jaw. Such movements would in fact betray the presence of the device to the external world.

Several in-mouth tongue controlled systems developed recently answer to these needs, by providing most often a very small sized wireless device equipped with some mechanism for sensing the tip of the tongue of the user. On a large number of these devices, however, the use of mechanical switches as sensors requires the user to put a certain effort in activating each sensor. This reduces the comfort of the user as he or she needs to use his/her tongue with an unusual strength. On other devices, a second solution for sensing the movements of the tongue has been implemented: a piece of metal is fixed as the tip of the tongue, and some inductors are used on the device to detect its movements. This second solution has the inconvenient of requiring the user to fix something at the tip of his or her tongue. Nevertheless, the detection mechanism is sensible enough for the user to be able to pilot the device without making unnatural efforts with his/her tongue. Provided that the piece of metal fixed to the tongue be small enough, this solution is thus far more

comfortable. However, the drawback of inductors in this case is their sensitivity to humidity and temperature variations. It is observed that most of the devices implementing this second solution are thus difficult to control when their temperature or their humidity varies. Another drawback often encountered in intra-oral devices is that the space offered by the mouth being small, the number of controls made available to the user is limited.

1.1.5 TKS'S SYSTEM

Recent research performed by TKS A/S [4] [5] lead to build a system taking into accounts all these observations. TKS' intra-oral device is a compact wireless device made to be placed right under the palate of its user, and clamped to its top teeth. The device is controlled by an activation unit fixed near the tip of the user's tongue, as a piercing. The user interacts with the device by moving this activation unit at different places of his palate. Through a set of inductors, the device can detect the localization of this activation unit on its surface, and transform it into commands.

The first particularity of this system is the high number of inductors used in the device, as they are in the number of eighteen. This provides a far more precise control to the user than the solutions developed in previous works. To get rid of the variations of inductance provoked by variations in humidity and temperature, TKS uses some software component to filter the user input, and makes use of fuzzy logic. The intra-oral device can thus be used while eating and drinking, and provides both a comfortable and efficient solution to the user.

This intra-oral device is part of a larger system, enabling the user to control a computer thanks to the movements of his or her tongue. It is thus wirelessly connected to an embedded controller, made to be placed on the wheelchair of the user. This controller receives the input orders from the user and transforms them into commands. In the current system, these commands are wirelessly sent to a USB device, and are used to control the computer to which it is plugged.

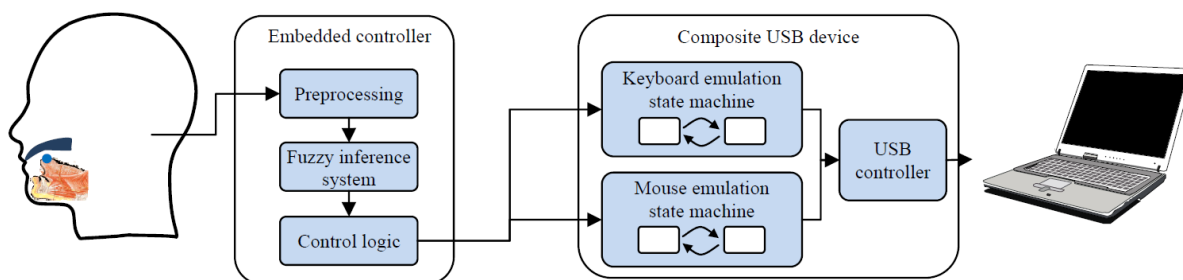


Figure 1: TKS's tongue-controlled system [5]

TKS's system is made of an intra-oral tongue controlled device, an embedded controller, and a USB dongle, enabling the system to act as a mouse and keyboard on any computer.

1.1.6 LIMITATIONS OF TKS'S SYSTEM

The system developed by TKS provides a really discrete, comfortable and efficient solution for transforming the movements of the tongue into commands. The intra-oral device designed, along

with the algorithms applied to manage its output, constitute the great interface that was missing between the user and the electronic world. This interface is cosmetically acceptable, and is sufficiently sophisticated to enable the control of complicated devices, such as computers.

However, the current system binds the user to a single computer. In fact, no interfaces were developed at the moment to enable the system to use other devices than a computer. To get control on a computer or switch between several computers, the user must also rely on the help of another person, as a USB key must be plugged onto the computer to control. These limitations reduce the independence of the user, as the user is forced to ask somebody whenever he or she wants to use a computer. Furthermore, the number of controllable devices being limited, the intra-oral device has little reason to be kept in the mouth once the user has finished using his or her computer. This makes the intra-oral device less comfortable and more difficult to accept, as frequent installation and removing of the device in the mouth by a tierce person may result uncomfortable.

Improving the comfort of the user while using the intra-oral device involves enabling this device to serve as an interface for far more other electronic devices than a single computer. In fact, giving a reason for the user to keep the device inside his or her mouth all day long will reduce his or her need to remove it or re-install it inside the mouth too often, which will improve his or her feeling of independence. Another key point for making this interface achieve its goal is to enable the user to switch from device to device without the need for external assistance. It is in fact important for an interface between a disabled user and the electronic world to remove as many obstacles as possible from the will of the user to the execution of actions by the electronic devices.

One of the main drawbacks of TKS's system is its use of a proprietary protocol between the embedded controller and the USB stick enabling the control of a computer. The use of such a proprietary protocol is in fact binding the system to the only devices adapted explicitly by TKS to be controllable by the system. One good improvement of the current system would be to enable it to interact with a larger pool of devices by endowing it with standard ways of controlling other devices. These standard ways include the support for standard protocols in the aim of connecting to other devices, but also the support for commonly used protocols enabling inter-device service discovery and access.

One other drawback of TKS's system is the lack of visual feedback that it provides to the user. Such a feedback is important for the user to know when he or she is in control of some electronic device, and which device he or she is controlling. Such an interface is essential to evolve the system for the use of multiple devices.

1.1.7 THE HAMAC PROOF OF CONCEPT

To address these drawbacks, we proposed in our previous semester project to evolve TKS's system to make it compatible with a large set of devices as well as more easy to use from the user's perspective. The result of this project is a device called HAMAC (Home Assistive and Mobile Access Controller).

The HAMAC device is made to interact directly with the embedded controller of TKS and can therefore be controlled by TKS's efficient intra-oral device. Highly modular, it is designed to support as many standard protocols as possible to control other devices wirelessly – replacing as such TKS's USB key system for controlling computers. Among other advantages, using standard protocols enables it to interact with unmodified commercial devices, designed for any user. It also enables the interface to adapt itself to the devices surrounding the user. The HAMAC device is engineered to limit the needs for external manipulations, thus taking care of its user's independence. It is also equipped with a small color screen offering a visual feedback to the user, to make it easier for him or her to interact with several devices. Figure 2 presents how HAMAC is integrated into TKS's system.

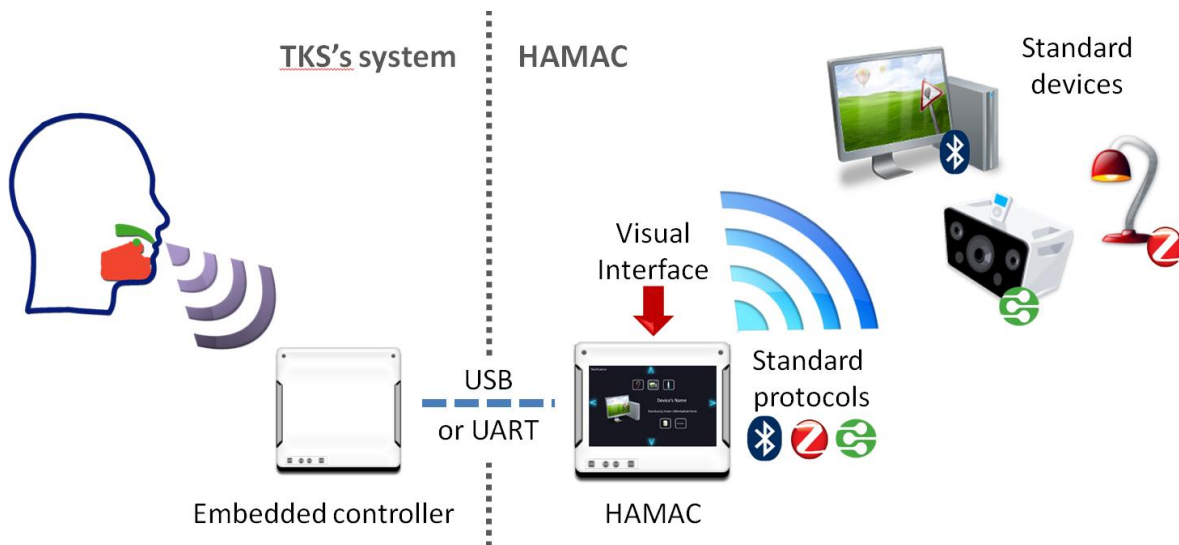


Figure 2: TKS's system extended by the HAMAC device

The HAMAC device improves TKS's system by completing its embedded controller with a solution for controlling a large set of devices in a standard way and for providing an intuitive and visual interface to the user.

During our previous semester project [6], HAMAC was developed as a proof of concept, on an Atmel AT91SAM9263-EK development board: a board equipped with an ARM processor and a small touch screen. The goal of this device was first to demonstrate how several standard technologies could be used to control unmodified consumer electronics devices. The proof of concept was also controlled by a modular framework able of presenting devices generically and intuitively to the user, while communicating with these through various protocols. The use of components specific to embedded devices to run the framework proved the feasibility of embedding the HAMAC device on a wheelchair.

At the end of the project, the HAMAC proof of concept was able of controlling unmodified Zigbee lights and Bluetooth equipped computers wirelessly. It could also detect the devices available around the user, to react to changes in the user's environment.

1.1.8 FEATURES TO ADDRESS

The HAMAC proof of concept is far from being a fully operational device. Even if it achieves to take control upon unmodified consumer electronics devices, it lacks several features that would be essential to make it a final product.

The next step to take for transforming HAMAC into a useable product is to replace its development hardware by a highly mobile and customized hardware platform. Siim SEPMAN chose to help us by undertaking this work during this semester, through another master project. Siim SEPMAN is working on building a battery powered device, embeddable on a wheelchair. However, the HAMAC framework designed during last semester is not able to control its power consumption, which makes it inappropriate to run on such a platform. On the software side, achieving mobility would thus require enabling the HAMAC framework to reduce its energy consumption.

While having the functionalities to make disabled people more independent, a fully mobile HAMAC device would still leave an important concern unaddressed: security. Even if taking control upon a large set of wireless devices is convenient, it often requires transmitting sensitive data over the air. Leaving this data unprotected could justify a strong refusal from the disabled people to use the HAMAC system. It would also contribute disadvantaging them compared to other people.

The current HAMAC framework can also be improved in a number of ways. As cited in our last report, future works should include integrating HAMAC with TKS's system to make it directly controllable with TKS's intra-oral device. They could also focus on enabling the framework to combine standard protocols together – i.e. to enable functionality sharing protocols to rely on several connectivity protocols – for better interoperability. Finally, it could be interesting to explore the needs of specific plugins that wouldn't use standard protocols, such as a plugin for controlling a motorized wheelchair.

1.2 PROBLEM STATEMENT

The HAMAC device is intended in evolving TKS's tongue controlled system, by making it compatible with standard protocols for connectivity to other devices, service discovery and service access. It also endows the system with a visual interface, and presents devices generically to the user on this interface.

The question that we began to answer in our previous semester project was formally written as the following problem statement:

How to evolve TKS's tongue controlled system to make it compatible with a large set of devices while making it more intuitive to use from the user's perspective?

By bringing improvements to the HAMAC proof of concept developed during our previous semester project, this project also contributes to answering the same question. However, our focus

will be put on the improvements themselves, and specifically on the methods that can be used to make the HAMAC framework more energy efficient and secure.

The formal problem statement that we intend to solve through this semester project and by the mean of this report is thus the following:

■ *How to improve the HAMAC framework to make it both more energy efficient and more secure?*

1.3 PROJECT OUTPUT

This project will be split in three parts.

In the first part, we will focus on the work done in our previous semester project, to summarize the analysis that we previously made and to present the HAMAC proof of concept as it was designed at the beginning of the current project. This part will also be focused on identifying which features of an ideal man-to-machine interface for disabled people miss in our proof of concept.

In the second part, effort will be put on bringing two major improvements to the HAMAC framework: a better energy efficiency and the ability to manage security. Special care will be put on taking into account the high modularity of HAMAC: most of the mechanisms to lower the energy consumption or increase security will be designed to be generic. A study case of the plugins available in our proof of concept will provide a good example of how these generic mechanisms can be used to improve HAMAC's plugins. All mechanisms presented in this part will be subject to an implementation on our proof of concept and to an evaluation.

The last part of this project is dedicated to additional improvements of the HAMAC device. It will present further works on the framework's architecture.

1.4 PROJECT SCOPE

This project will first provide an analysis of the different emerging and broadly used protocols for standard interoperability between several devices. It is to be noted that, far from being extensive, this analysis will be concentrated on a small number of these protocols, with the only aim to present the main ideas and concepts used today to achieve inter-device interoperability. This analysis must not be referred to as a summary of all the solutions existing today, as it will present only a selected set of protocols.

The proof of concept used as the basis of this project is only compatible with two protocols, Zigbee and Bluetooth, and supports only interaction with one profile of each of these protocols – namely the *Zigbee Home Automation (HA) Profile* and the *Bluetooth Human Interface Device (HID) Profile*. No other standard protocol or profile will be used in this project. In fact, the aim of the project is to show the ability of our algorithms and mechanisms to be efficient while using several standards technologies. It is not to add support for other technologies to the HAMAC device.

Although a hardware board for HAMAC will be built by Siim SEPMAN this semester, the software designed during this project will be tested only on the Atmel AT91SAM9263-EK development board. This decision will help us to focus on the problems introduced by our software improvements by guaranteeing us that the hardware on which HAMAC is run is bug free.

Finally, even if it might get our attention, the compatibility with TKS's device will neither be optimized nor tested, due to time limits. Therefore, interaction with TKS's embedded controller through a serial line is not bound to be implemented by the end of the project. However, we will continue making the HAMAC device controllable by a regular HID mouse, to enable interaction with TKS's intra-oral device when using the company's full system – which is capable of simulating a HID mouse.

1.5 PROJECT ASSUMPTIONS

Even if it might provide mechanisms for avoiding them, this project will assume that the interferences between several wireless transceivers and antennas put side to side in a hardware implementation and using overlapping frequencies to communicate can be neglected. More precisely, it will assume that the communication problems caused by these interferences will not prevent devices from establishing reliable communication links between one another. It is assumed that the protocols used for the wireless communication between devices provide the necessary algorithms and features to prevent these interferences from being an issue.

2 PRELIMINARIES

2.1 PROBLEM ANALYSIS

2.1.1 FUNCTIONAL NEED OF QUADRIPLLEGICS

To understand better which features have to be implemented in HAMAC, an analysis of the needs of quadriplegic people regarding man-to-machine interfaces is required. This analysis has been made in the first report [6] and is summarized in this section.

Quadriplegic people should be able to provide signals understandable by the digital world and thus control functionalities provided by the interface. Both hardware and software are important to answer these expectations. The HAMAC project relies on TKS's intra-oral tongue-controlled system as input system. Our focus will thus be put mainly on the expectations concerning the quality of the control functionalities provided to the users, which HAMAC aims to improve.

2.1.1.1 Hardware interface

The first concern of a man-to-machine interface is to enable the user to interact with the software interface in the most efficient way. This concern is generally addressed by providing input devices adapted to the abilities of the target user. These hardware interface should be well-thought enough to enable the user to forget about its disabilities, by being efficient, comfortable to use and cosmetically acceptable. The HAMAC project relies on the hardware interface provided by TKS, as we think that it responds particularly well to these three expectations.

2.1.1.2 Software interface

The software part of a man-to-machine interface has for goal to optimize the use of the input signals provided by the disabled user, so that to let him or her to interact with the electronic devices at least as well as people without disabilities.

This first implies trying to give back to the users as much as possible of the independence that they are deprived from because of their disabilities. It also includes making the interface powerful enough to be highly compatible with a lot of devices and at least as easy to use as the interfaces provided for the other users.

2.1.1.2.1 Functionalities enabling independence

2.1.1.2.1.1 *Full control of devices*

The functionalities enabling more independence include the full control of devices. It consists in bringing an alternative interface to regular devices which does not make the user feel the need to use the regular interface. Thanks to tools such as virtual keyboards, the user of a computer can for example benefit from the whole set of its functionalities by the only use of a pointing device.

2.1.1.2.1.2 Easy first-time installation process

To use a device as independently as possible, it is also essential that a user be able of accessing new devices without requiring heavy installation procedures. Although a lot of devices may require mechanical interaction to be controllable by a remote interface, this interaction should thus be as minimal as possible, to keep the user as independent as possible from the other people.

2.1.1.2.1.3 Easy second access to devices

If the ability of the user to use all the functionalities of devices is important, being able of getting access to the devices without any external help is also essential for his or her independence. The user is in fact expected to control several electronic devices per day. A well designed interface should thus memorize the parameters of every device that was already controlled once, to enable the user to control them again without performing any other installation process.

2.1.1.2.1.4 Location independent operation

The user should not be reduced in controlling devices only at home. The whole set of electronic devices available in public area should be controllable by disable people in order to give them more independence. Thus, the computer in a library, or an elevator in a city hall should be controllable by HAMAC.

2.1.1.2.1.5 Adaptability to the environment

Enabling the disabled users to get access to a large set of devices implies making the user choose which device he or she wants to control. A good interface should in fact enable a user that is near to a television, a light switch and a computer to specify which device he or she wants to use and then adapts itself to the surrounding environment. To this end, HAMAC should let the user know in some way the list of devices or functionalities that he or she can access to.

2.1.1.2.1.6 All day long availability

Quadriplegic people have very limited possibilities for interacting physically with their environment. Therefore, the man-to-machine interfaces dedicated to quadriplegic people are expected to require external help before and after usage. For example, TKS's intra-oral device needs to be put in the mouth before and removed from it after usage. This dependency from other people reduces a lot the independence of the users of such interfaces. For that reason, the number of operations per day to set up the interface and to stop using it should be as small as possible, and optimally limited to only two operations per day. This requirement involves that the device constituting the interface be able of running all day long, without running out of batteries. For this requirement to be met, the software interface must be particularly designed to be used under hard energy constraints. The use of hardware resources should be optimized to reduce as much as possible the energy consumption of the device.

2.1.1.2.2 Functionalities enabling control

2.1.1.2.2.1 High interoperability

The first feature that a disabled user should expect from a unique interface linking him/her with the electronic world is the ability to control a large set of devices. To this end, care should be taken to make HAMAC compatible with most of the standards elaborated to make electronic devices control each other. Finally, limiting the user to the use of specifically designed electronic devices should be avoided. The market of electronic devices is in fact moving too fast for HAMAC to be adapted to every device separately. Doing so would thus bind the user to a unique small set of devices, not always corresponding to its needs or wishes, which is contrary to the objectives of a good interface.

2.1.1.2.2.2 Consistency and ease of use

Consistency in controlling devices is brought in displaying identical functionalities of different devices that might be on different protocols, in a same way. It removes the need of the user to always adapt itself to a different interface, and a different way to use the same service. This brings an easy way to use HAMAC.

2.1.1.2.2.3 Multi-tasking capability

To enable disabled users to enjoy the same quality of service when controlling electronic devices than other users, an important feature to provide is a multitasking capability. In fact, in their day-to-day use of electronic devices, users switch frequently from one device to another to get services adapted to their needs. A user listening to a MP3 player while using a computer can for example switch from time to time from the computer to the MP3 player to change the song that he or she is listening to. Once the song has been changed, this user returns to the computer, and continues working on it. In the case of an interface providing access to these devices, it is important for the user experience that the time necessary to switch from device to device be minimized. It is also important that the MP3 player controlled by the user does not stop playing music if the user switches its control to the computer. These two objectives can be met by making HAMAC able to control several devices at a time, through multitasking.

2.1.1.2.2.4 Visual feedback

To finish, if an interface is made interoperable with a large set of devices and if it is designed to adapt itself continuously to the user's environment, it should be able to offer a visual feedback to the user. In fact, the user requires knowing which device he or she is controlling at any time. A changing list of controllable devices would make this information difficult to guess. A visual feedback, such as a TFT screen, would enable the user to see at any time what he or she is doing. It would also enable the interface to notify the user when new devices are available, or to ask the user for some parameters when connecting to other devices.

2.1.2 ANALYSIS OF STANDARD PROTOCOL FOR DEVICE INTEROPERABILITY

To meet the requirements seen in section 2.1.1, choosing the protocols that HAMAC will use is very important. This choice will depend on the needs of interoperability of the HAMAC device. As

presented in our first report [6], full interoperability requires two elements: interoperability of connectivity and interoperability of functionality sharing.

To achieve an interoperability of connectivity, two devices must be able to exchange information with one another. Due to the high number of technologies existing for transporting information and to the multiple ways information can be communicated, achieving this exchange between two devices built from different manufacturers is not easy without following well-defined standards. Standard network protocols were thus introduced. These protocols can often be combined together, if the parts of the communication that they are responsible for do not overlap. One protocol enabling the transmission of data bits over a wire can for example be used in addition with a protocol defining which bits to exchange to transmit certain information. In the case of HAMAC, we are interested to control wirelessly remote devices situated in the same room than our system. Moreover, the target devices should be controllable even when HAMAC is located at least one meter from them. The physical interface of the protocols to use is therefore well defined.

To achieve an interoperability of functionality sharing, devices must be able to do three things: to split their functionalities into independent sub-functionalities, to make other devices know that they can share these sub-functionalities and to provide a way for other devices to access these sub-functionalities. These three abilities are emphasized in the computer paradigm of Service Oriented Architecture (SOA) [7]. HAMAC needs to be able to take control upon wireless devices without requiring physical interaction with them. Therefore, it can only use protocols supporting a SOA at their application level. The list of standard protocols having this characteristic is reduced.

Consequently, the protocols responding the two requirements of HAMAC – in terms of physical interface and application architecture – are not numerous. We listed them in Table 1, as we did in our last report, to explain the choices made in this project.

| Solution | Application stack | Network stack | Physical stack |
|-----------------|--------------------------|----------------------|-----------------------|
| 1 | Bluetooth Profiles + SDP | Bluetooth | 802.15.1 (Bluetooth) |
| 2 | Zigbee Profiles + SDP | Zigbee | 802.15.4 |
| 3 | DLNA + UPnP | TCP/IPv4 | 802.11.X (Wi-Fi) |
| 4 | DLNA + UPnP | TCP/IPv6 | 802.11.X (Wi-Fi) |
| 5 | DLNA + UPnP | 6LoWPAN (TCP/IPv6) | 802.15.4 |
| 6 | DLNA + UPnP | TCP/IP + Bluetooth | 802.15.1 (Bluetooth) |

Table 1: Possible network stack combinations for HAMAC

This list presents only three physical stacks: integrating all these protocol combinations in one hardware device is thus feasible, which is promising in terms of interoperability. At the application level, the protocols compatible with HAMAC are first the Bluetooth and the Zigbee protocols. We also selected the DLNA over UPnP protocol, which provides a standard and broadly used way of sharing functionalities for devices communicating through IP-based protocols. Several

other IP-based protocols could be suitable for HAMAC in the future. However, none provides the same level of interoperability and standardization than the DLNA at the moment of writing this report. We also insist on the fact that, technology evolving rapidly, the HAMAC framework should be prepared to follow these changes as fast as possible. Further reading about the protocols listed in Table 1 can be found in [6; 1; 2; 3; 8].

2.2 DEFINING HAMAC

During the first part of this project, the needs of the disabled people in terms of interface, presented in section 2.1, were all taken into account to design HAMAC. Our works focused on two tasks: building a framework able to live up an interface device responding to these needs, and implementing a proof of concept to demonstrate this framework's capabilities experimentally. This section presents our achievements.

2.2.1 THE HAMAC FRAMEWORK

2.2.1.1 Overview

The HAMAC framework is the core of HAMAC. It consists in several software components enabling to remote control generically devices using several standard technologies. Using a visual interface, it presents the devices to control by functionality instead of technology to provide the user with the most intuitive interaction possible. The strength of this framework is due to the several levels of abstraction that it offers:

- The devices are represented with the same characteristics to the user, independently from the technology they use to communicate. They are presented in a single list, and are controlled, added, deleted or configured in the same way.
- The functionalities offered by the devices are split into several sub-functionalities, or "services", to allow the same functionality offered by two different devices using different technologies to always be controlled in the same way – and through the same interface.
- The communications technologies used by the framework are grouped into protocol packages, which are all controlled in the same way. In this way, the processes of discovering devices in the environment and of connecting to them can all be managed centrally by the framework.

These various levels of abstraction allow the framework to be highly modular, and thus to be adapted to new technologies with a minimum effort of development. To make the users benefit quickly from new possibilities, developers are free to include the framework the support for new services, new types of devices or new protocols, thanks to a plugin system.

2.2.1.2 Detailed design

This section presents the detailed design of the HAMAC framework, based on our first semester report [6]. Its aim is to provide a basic understanding of how the framework is designed, but also to emphasize changes that were made after the publication of our first report.

2.2.1.2.1 Methodology

This design task is situated in the algorithm domain of the A³ model. This model is presented in appendix 6.1.1. It is not a real algorithm but a response to the analysis made in the HAMAC project. The next step is the implementation of this design. This implementation will correspond to the architecture domain of the A³ model.

The design is described by using the Unified Modeling Language (UML). A description of UML is available in appendix 6.1.2. For a better understanding and modularity, we decided to design the HAMAC framework by using an Object Oriented (OO) language. This choice brings concepts that are not represented in procedural programming. These concepts are part of the UML language presented in appendix 6.1.2.

2.2.1.2.2 Presentation

The framework is divided into three software packages: the Core, the GUI, and the Communication packages. Each one has its particularities and a precise role. Each package contains several classes.

2.2.1.2.2.1 Core

The Core package is the skeleton of the framework. It contains all classes, abstraction mechanisms and algorithms enabling the framework to operate, to manage the list of devices to control and to coordinate the visual interface with protocols.

2.2.1.2.2.1.1 Element

The first class of the framework is `Element`. It represents an element discovered by HAMAC in the user's environment. An element is a device or functionality. This class stores basic information and provides methods to access them. Here is the UML representation of this class:

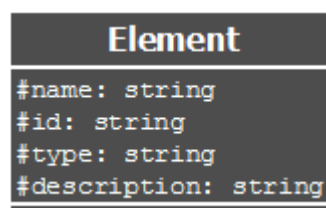


Figure 3: Element Class

This class stores the element's name, id, type and description. These data will be used by derived classes.

2.2.1.2.2.1.2 *Device*

A device is an object situated in user's environment. It can be a computer, a light, a wheelchair, etc. These objects are represented by the `Device` class. This class is derived from `Element`.

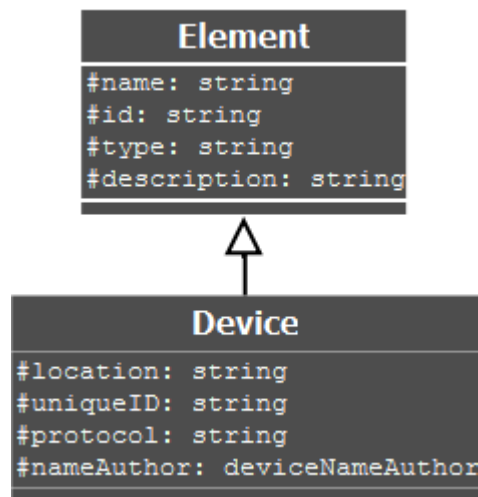


Figure 4: Device class

`Device` contains additional information. It stores the location of the device, its unique ID on the protocol network, the protocol used to communicate with it and some extra-information. For instance, a flag is stored inside the class to know who has given the name to the device. If it is a default name, it may ask the user to insert a new one more user friendly.

This is inherited by protocol dependent classes. It is the most specific class which will be instantiated. For instance, if a light is discovered on the Zigbee network, a `LightByZigbee` class will be created. But this is explained more deeply in the first report.

2.2.1.2.2.1.3 *Service*

A service is a functionality provided by a device. It is also an element. A service can be a mouse, a keyboard, an activation (to turn on or off a device), a dimmer (to dim a light for instance), etc. Functionality is represented by a `Service` class.

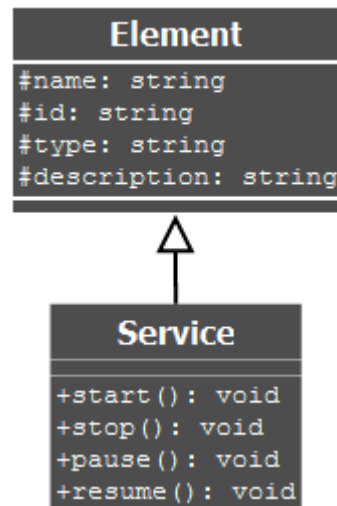


Figure 5: Service class

As seen in Figure 5, *Service* provides some methods to manage the device's functionality. It can be started, stopped, paused and resumed. This is useful when several services are to be used simultaneously. When the user switches the functionality, the previous one is paused, and the next is started or resumed.

This class is inherited by protocol specific classes. A same functionality can be used on different protocol networks, but the way they are used is different. The implementation is thus different. For instance, if a light situated on the Zigbee network provide the activation service, an *ActivationByZigbee* class will be instantiated. This is more deeply explained in the first report.

2.2.1.2.2.1.4 Concurrent List

All elements are stored in a list. A special container has been implemented to provide an additional functionality. All elements will be accessed by different thread, may be at the same time. To prevent unexpected behavior, a protection by mutex has been implemented. The UML design of this list is shown in Figure 6.

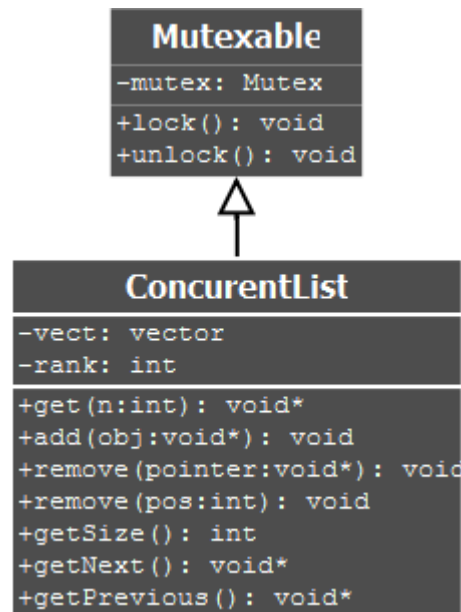


Figure 6: ConcurrentList class

As seen in Figure 6, this class contains a vector. A vector is a container defined in the STL library [9]. All write and read access to this vector are protected by a mutex. This class also provides some extra features methods to get the next or the previous element from the vector. These methods indicate by an exception if the next or the previous element is the same. This is the case when only one element is present. Indeed, `ConcurrentList` considers the vector as a circular list.

When a device or a service is discovered by HAMAC, a class will be created and stored inside a `ConcurrentList`. Then the GUI will read the list to display information on the screen. Only one `ConcurrentList` exists to store devices, but each device contains a `ConcurrentList` to store its own services.

2.2.1.2.2.2 Graphical User Interface (GUI)

The Graphical User Interface (GUI) package is responsible for displaying information about devices and services in an intuitive way. The Qt library has been chosen to perform this task, but another library can be used. In fact, the development has been thought to keep external libraries such as Qt independent from the rest of the framework. This choice allows us to change the library used to operate the GUI package if necessary. It keeps the framework extensible.

2.2.1.2.2.2.1 GUI class

The GUI class is the main class of the GUI package. It uses Qt to fetch user input and manage windows. This class stores the data of the windows being displayed. It is also responsible for managing the navigation between these windows and for displaying notifications. The Qt features used by the GUI class are presented in our first report [6]. These features are the event, signal and slots mechanisms. A special care is put in managing events, because GUI is the bridge between the

OS events and HAMAC. In fact, all interactions between the user and HAMAC are managed by the Linux drivers before being transmitted to Qt.

2.2.1.2.2.2.2 Other GUI classes

Within the GUI package, other classes were designed to display devices and services on the visual interface: `DeviceGUI` and `ServiceGUI`. These classes are derived from a class called `CommonGUI`, itself deriving from a Qt widget. The `CommonGUI` class enables its children to display an icon, a name and a description on a window. The instances of `DeviceGUI` and `ServiceGUI` are stored by the GUI class, as represented on Figure 7.

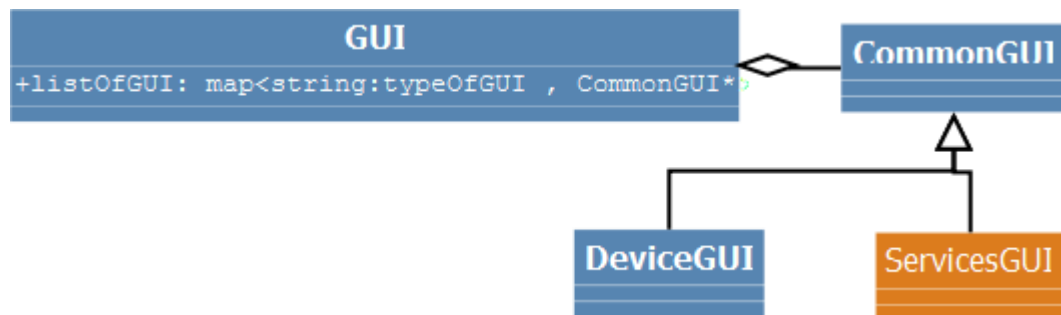


Figure 7: GUI class and classes representing devices and services

On Figure 7, the classes implemented in the GUI packages are drawn in blue. To keep the framework modular, the `ServiceGUI` classes are implemented in the service plugins.

2.2.1.2.2.2.3 Notifications

The HAMAC framework is made to notify the user when a device or functionality is added or removed from the environment. When such events occur, a notification is created as an instance of the `Notification` class. The notification is then pushed inside a queue implemented inside the `NotificationList` class. When GUI reads the notification, the `NotificationList` launches a timer enabling it to hide the notification after a given timeout. When this timeout occurs, the next notification is read and displayed, if any.

2.2.1.2.2.2.4 Icon Manager

To manage all icons, an icon manager has been designed. By now its role is to store the path of each icon file used by the framework. In future works, this role will be extended to enable adding or removing icons dynamically during the execution and loading icons more efficiently. Because we created a specific class to handle the icons, we will be able to perform these improvements effortlessly. A decision should be made in the future to define the best way to load icons: the options include loading all icons at the beginning to allow a faster switch between windows, or loading icons only when they are displayed.

2.2.1.2.2.3 Communication

The communication package is responsible for managing interactions between HAMAC and the different communication technologies it can use. This package contains classes for managing the connections to the remote devices and others for interacting with the core package. It also contains classes representing the types of devices and the types of services attached to each protocol.

2.2.1.2.2.3.1 Protocol

The Protocol plugins represents specific protocols, along with the specific devices types and services attached to them. Each Protocol plugin contains a class deriving from the class `Protocol`. The `Protocol` class is a generic class enabling HAMAC to make abstraction of the specificities of each protocol. It provides functions enabling to discover devices in the user's environment and to manage the communication between the services accessible through the protocol and the `Service` class designed in HAMAC.

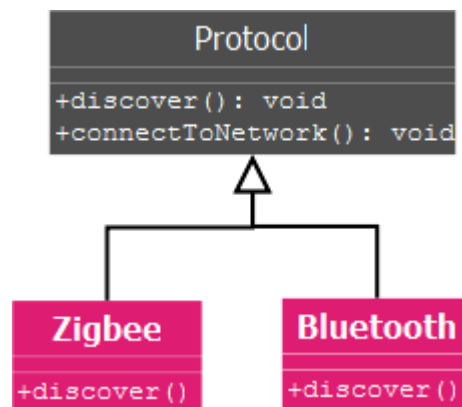


Figure 8: Protocol classes

As seen in Figure 8, the `Protocol` class is defined in the framework and its derived classes *Zigbee* and *Bluetooth* are put into Protocol plugins.

2.2.1.2.2.3.2 Protocol manager

The protocol manager creates a thread on which the instances of the `Protocol` classes will operate. This thread separates the execution of the `Protocol` classes from the rest of the framework. The instances of the `GUI` and `Protocol` classes are independent, which ensures that HAMAC always be reactive to the user's input.

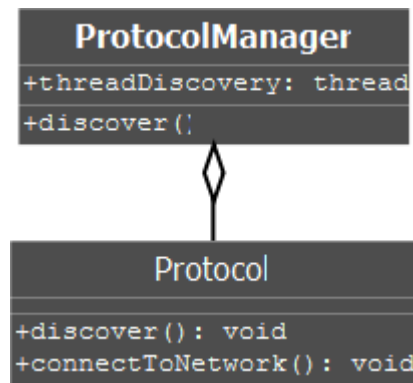


Figure 9: ProtocolManager class contains the list of Protocols instance

Figure 9 is the UML representation of `ProtocolManager`. We can see that the class contains the discovery thread presented in our first report [6].

2.2.2 OUR PROOF OF CONCEPT

To demonstrate that the HAMAC framework could be implemented, the first step of our project also included the design of a proof of concept. This section describes it.

2.2.2.1 Software design

To give a good example of the possibilities offered by the framework, we developed several plugins for it. Our goal was to demonstrate how several technologies can be used to control devices generically. Therefore, we developed the components necessary to control the mouse of Bluetooth enabled and the switching functionality of Zigbee enabled lights.

2.2.2.2 Hardware design

At the moment, the hardware of our proof of concept is composed of several development boards. In fact, we preferred to finish the software design of the framework before designing a hardware platform optimized for it. Nevertheless, we decided to run the framework only on components suitable to be embedded in a small mobile device. Our technology choices are described in our last report [6].

The execution of the framework is hosted on an Atmel AT91SAM9263-EK platform, a development board based on the Atmel AT91SAM9263 microprocessor. This microprocessor contains an ARM-9 core, embeddable but powerful enough to host a Linux operating system. Its main features are described in Table 2, while the features of the AT91SAM9263-EK board are described in Table 3.

| | | |
|----------------|---------------------|-------------------|
| LCD Controller | Ethernet MAC 10/100 | Clock Speed (MHz) |
| Enhanced USART | USB Device FS | ARM Core 926EJ-S |

Table 2: features of AT91SAM9263 microcontroller

| | |
|--|---|
| 64 Mbytes of SDRAM memory | Two USB Host port interfaces |
| One RS232 serial communication port | One Ethernet 100-base TX with three status LEDs |
| One 3.5" 1/4 VGA TFT LCD Module with TouchScreen and backlight | Two SD/SDIO/MMC card slots |

Table 3: features of AT91SAM9263-EK development board

To support the Bluetooth and Zigbee plugins, two USB dongles are plugged on the development board. The Zigbee dongle used is part of Texas Instruments' CC2530 Development Kit [10].

2.2.2.3 Functionalities

By the end of last semester, this proof of concept was therefore able to control standard Zigbee lights and Bluetooth computers. Controlled by a touchscreen and a USB mouse, it was also able to present the devices available in the environment around the user on a visual interface, and to notify the user each time a device was appearing or disappearing. Finally, it was able to split the functionalities of devices into services, to offer always the same interface for controlling a given functionality.

2.2.2.4 Points to improve

This proof of concept is only a demonstration of how the capabilities of the HAMAC framework could be implemented on an embedded device. It is thus far from being a stable and optimized implementation of the HAMAC device. It should be improved by following several axes:

Security: The current device does not implement any security measures to protect its communications with the other devices.

Compatibility: According to our tests performed last semester, our proof of concept is not fully compatible with the Bluetooth HID Profile. It cannot, in fact, take control upon computers using the Microsoft Windows Bluetooth stack. Moreover, the control of the other computers through the Bluetooth HID Profile presents some problems of latency. Our device should thus be improved both for a better compatibility with the profile and to achieve better latencies.

Furthermore, our device couldn't be integrated with TKS's intra-oral tongue-controlled device, due to the limited number of devices available for the tests.

Optimization: The optimization of our proof was out of the scope of this first part of the project. Our device can thus be optimized in several ways.

First, the usage of the radio communications should be optimized to avoid interferences to occur between the different radio links established. The link between TKS's intra-oral tongue-controlled device and TKS's embedded controller should particularly be protected from interferences.

The energy consumption of both the HAMAC device and the devices of its environment should also be made more efficient.

The hardware of the proof of concept is demonstration hardware. The hardware of the final HAMAC device should thus be redesigned so that to optimize the device's power consumption. A lot of components on the demonstration hardware are unused in our proof of concept.

The software of our proof of concept should also be improved. This can be made by using an optimized Linux kernel for the HAMAC device, by optimizing QT for an optimal usage, and by optimizing the file system of the HAMAC device. The algorithms used in our framework and the resources that they used should also be measured and optimized.

Although sufficient to offer the functionalities of the target HAMAC device, our framework could also be improved according to the following axes:

Modularity: Our framework is modular enough to enable new protocol stacks to be supported easily. However, it misses an axis of modularity enabling protocols to be combined. This modularity is required if we want to optimize the combination of several protocol stacks on the HAMAC device. It would for example enable the protocols Zigbee and 6LoWPAN to share the same 802.15.4 connectivity, or the DLNA + UPnP protocol to search for services both on an IP connection provided by the Bluetooth LAN Profile and another IP connection provided by a 802.11.X (Wi-Fi) stack. This other axis of modularity should be analyzed in further works.

2.2.2.5 Improvement brought by this report

To perfect our proof of concept, this report will bring several of these improvements:

- A power management system will be designed and implemented as described in section 3.1, to lower the energy consumption of the HAMAC device.
- This power management system will include mechanisms to perform radio collision avoidance, as required by the points to improve mentioned in the previous section.

- Section 3.2.2 will present design patterns improving the security of HAMAC, and section 3.2.3 other patterns improving the safety of some of its plugins.

3 PRIMARY AXES OF IMPROVEMENT

3.1 MAKE HAMAC MORE ENERGY EFFICIENT

3.1.1 HAMAC PLATFORM

3.1.1.1 About Power Management Systems

HAMAC (by its name) is a mobile and autonomous embedded system and is designed to be powered by a battery. Thus, it has a limited powered on time due to the battery life-time and this time should be as long as possible. The goal of the proposed Power Management System (PMS) is to adapt the energy consumption of HAMAC depending on the usage made by the user. It helps the system by saving energy when it is unused and by using the best possible amount of energy when users want to interact with it.

3.1.1.1.1 Definition

Power and Energy terms are often confused when talking about power and energy savings; the basic idea is the same but practical effects are different. Vasanth Venkatachalam and Micheal Franz [11] define energy as the “*total amount of work a system performs over a period of time*” and power as the “*rate at which the system perform that work*”. $P = \frac{dW}{dT}$ (1) And $E = P \times T$ (2) define power and energy, respectively.

$$P = \frac{dW}{dT} \quad (1)$$

$$E = P \times T \quad (2)$$

Where P: Power (Watt), E: Energy (Joules), T: specific time interval, W: total work performed in that interval.

In the context of embedded system, power is defined by the rate of electrical energy used and energy is defined by the electrical energy used over time. Reducing power does not mean necessarily reducing energy because by definition they are different, e.g. reduced power often means reduced computational power, which means longer computational times and thus possibly equal (and sometimes increased) energy. The target of the PMS is to reduce the energy consumption in order to increase the battery life.

3.1.1.1.2 Design Method

To design the power management system of HAMAC, two tasks are performed first. On one hand, specifications are defined according to the context use of HAMAC in order to identify the requirements needed to select the battery. On the other hand, the energy consuming parts of HAMAC are identified and a policy for reducing the electrical consumption during operating time is defined. By using these data, the power management system is then designed and implemented. This process is represented in Figure 10

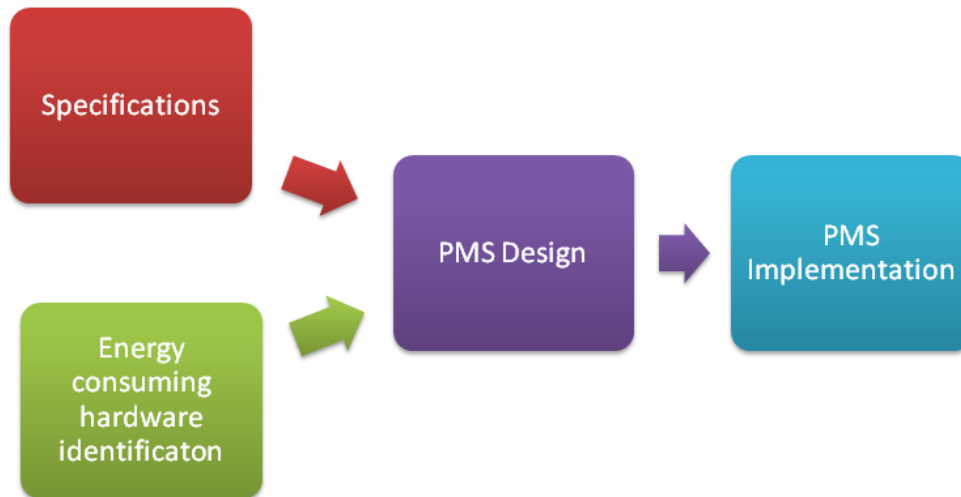


Figure 10: Power Management System Design Process.

This process meets the A^3 requirements introduced in Appendix 6.1.1. The two first blocks define the application. The design block (in purple in Figure 10) specifies algorithms and the implementation block (in blue in Figure 10) represents the implementation of algorithms in architecture circle of A^3 model. This is represented in Figure 11.

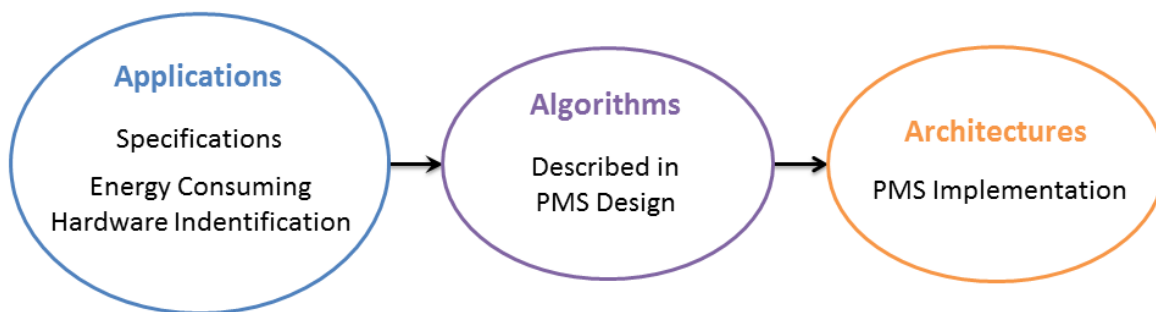


Figure 11: A^3 model meets Power Management System design process

3.1.1.2 Managing Power in HAMAC

3.1.1.2.1 Strategy

3.1.1.2.1.1 Specifications

The power management strategy is straightforward. We should use as little energy as possible. The question is in which range or for which goal. To build HAMAC as a product, specifications regarding the energy and power consumption are needed. Nevertheless, it is not the purpose of this report to present a business plan. Some ideas and key concepts are still presented.

To have an idea of the characteristics of HAMAC, the running time is defined and then the energy consumption of HAMAC is calculated in different contexts of use. Both the best and the worst case of energy consumption are thus calculated. Hardware and software optimizations are applied to HAMAC to have better performances in term of power use. At the end, we have all data needed to

perform a final design and to choose a battery for a final product. The goal is to have a battery as little as possible. All methods are applied to reduce the power and energy consumption of HAMAC.

In order to choose the right duration time, some research as customer quizzes, marketing work, business plan, etc. are generally performed. But for this report, basic understanding of what disable people need is used to have an idea of a duration time. According to the analysis presented in the first [6], HAMAC should be used all day long, without recharging the battery. A disabled people should use HAMAC at its worst case utilization during at least one day. It means that HAMAC will be able to run 16 hours at full use.

3.1.1.2.1.2 Idea

When using a system like HAMAC, all hardware components are not used at the same time. Even not all an entire component is used at a same time. For instance:

- HAMAC can be performing a device discovery when the user is doing a random task (talking to another person, or something else). Nothing has to be displayed since the user is not watching the screen. The screen is unused.
- When the user is controlling a computer, he/she is looking at the computer screen and not at HAMAC, its screen is unused.
- The microcontroller is not using a CAN or I²C hardware in the application. These parts of the chip can be turned off.

We can understand that the behavior of the system can be adapted regarding its use. This is the role of the power management system. To design it, the idea is to identify what it is possible to act on, in order to reduce the energy consumption; thus, power consuming components are listed with the facilities provided by them to save power. Basic ideas to design the power management system are:

- To design different modes corresponding to use cases.
- To calculate switching cost in order to avoid over-power-consuming switching.
- To design negotiation between HAMAC and its plugin to set the most suitable mode according to plugin needs.

These ideas are part of the strategy used to reduce the system's energy consumption. They are developed in the next sections.

3.1.1.2.2 Ways of saving Energy

We can act on several levels to save power and energy consumption.

- HAMAC Platform:

In this level, several actions are possible to reduce power and energy consumption. These actions concern the electronic board of HAMAC. The first way is to use as less electrical component as possible. Then the component choice to build a board is very important. When putting several components in comparison, the power and energy consumption is an important issue to make a choice.

- HAMAC software:

Software techniques can be used to improve the power and energy consumption. In this level we can implement a way to adapt HAMAC to its use. We can act on compiler optimization, code profiling, or implement a specific behavior at application level.

- Radio links:

In this level, actions are made on radio chips. Protocols such as Zigbee or Bluetooth are made to be embedded and thus have mechanisms to save power and energy. These mechanisms will be described and used to use as less electricity as possible.

3.1.1.3 Reducing the consumption of the HAMAC Platform

3.1.1.3.1 List of features enabling components to save energy

A list of components with their need of power supply has been made. Not all hardware parts are presented but only the most power consuming. The power and energy consumption of the other components are estimated by practical measurements. Table 4 lists the power characteristics of the main hardware parts.

| | Static Current | (Leak) Current | Voltage | Power |
|-------------------------------|-------------------|----------------|---------|---------|
| Microcontroller (core) | 700μA | 70mA | 1.2V | 84mW |
| Screen LCD | | 200mA | 3.3V | 660mW |
| Screen Backlight | | 20mA | 3.75V | 75mW |
| Touch Panel | | 25mA | 5V | 125mW |
| Memory (SDRAM) | 5μA | 135mA | 3.3V | 445,5mW |
| Zigbee chip | | 33.5mA (max) | 3V | 100.5mW |
| Bluetooth chip | | 36.3mA (max) | 1.8V | 65,34mW |

Table 4: Power characteristics of main hardware parts [12; 13; 14; 15]

Table 4 shows that the most power consuming parts of the board are the LCD screen and the memory chip. An action has to be made on these parts to save power and energy as much as possible. In a future hardware design, more energy efficient components will be taken in consideration. Actions are also applied to the backlight, microcontroller and touch screen. It should be noticed that the value for the Bluetooth and Zigbee chip are maximum ones when the radio is at full power.

The microcontroller contains several power pins. The main ones are the one who power the core, the ones who power the external bus interface connected to the SDRAM memory and the ones who power the peripherals, USB included. Only the core power pin is independent of the design. It is the reason why, only the power consumption of this one is presented. Nevertheless, the memory chip is powered by the external bus interface (EBI) power pin.

3.1.1.3.1.1 Microcontroller

3.1.1.3.1.1.1 Generalities

The microcontroller is one of the most power consuming hardware after the touch screen and the SDRAM memory. Several approaches exist to reduce power and energy consumption. We should understand first the causes of this consumption. Then an approach can be found to reduce this consumption.

According to [11], there are two forms of power consumption: dynamic power consumption and static power consumption. Dynamic power consumption concerns circuit activities such as register writes. It has two sources:

- Switches capacitance, which arises from the charging and discharging of capacitors at the outputs of circuits.
- Short-circuit current, which occur when a NMOS and PMOS transistors are simultaneously on.

Static power is also known as idle power or leakage power. It is a circuit aspect and we cannot work on it by software. It is not the purpose of this project to work on the leakage power.

As indicated in [11], dynamic power is dependent on four factors as expressed in (3).

$$P_{dynamic} \approx aCV^2f \quad (3)$$

Where a is the activity factor, C the physical capacitance (F), V the supply voltage (V), and f the clock frequency (Hz).

3.1.1.3.1.1.2 Physical Capacitance

Power consumption can be reduced in four ways. The first way is to decrease the physical capacitance of the circuit. The physical capacitance depends on transistors size and wire design. Capacitance can be reduced with the size reduction of transistors and wires.

3.1.1.3.1.1.3 Switching activity

The second way is to reduce the transistor switching activity. A hardware technique exists; it is named clock gating [16; 17]. It consists in disabling portion of circuits to avoid flip-flop changing state. It is not the purpose of this project to work on this aspect.

Another solution is to minimize the processor utilization. But it is paradoxical to choose a powerful processor to not use it. However, some execution aspects are power consuming such as cache misses, or context switches. We can act on them to reduce the number required instructions for performing a certain task. A simple way is to use the compiler features to optimize the software. A compromise has to be found between the size of the executable file and the speed optimization.

3.1.1.3.1.1.4 Clock frequency

The third way is to reduce the clock frequency. This case is tricky because we cannot know exactly the effect of the frequency scaling on performance. A naive way is to assume that dividing the frequency by two will increase the execution time for performing a certain task by two. The power is reduced but not the energy consumption. This technique is useful in case of temperature reduction since the peak and average power dissipation will be reduced. But in real life, the time performing a task is not necessarily inversely proportional to frequency reduction. Indeed, at full frequency, the processor is not used at full computational ability because it has to wait for instructions and data from memory which is generally slower. So a decrease of frequency can save energy. The difficult task is to know by which factor(s) the consumption is decreased. This depends on the software being executed and a practical evaluation is needed to know this information.

3.1.1.3.1.1.5 Voltage supply

The forth way is to reduce the voltage supply. In this case the clock frequency has to be reduced too. This is related to the third way of reducing energy consumption explained above. This technique is called dynamic frequency and voltage scaling (DFVS). According to Equation $P_{dynamic} \approx \alpha CV^2 f$ (3), if the voltage supply is decreased, power is reduced in a quadratic way. This technique is difficult to carry out for different reasons.

The first reason is the difficulty of knowing the workload of a task. A task involves many different ways of using the microcontroller and is basically unpredictable in real systems. The fact that it can be preempted at any time by interruptions in addition to pipelining, hyper threading (when available), or cache misses, renders the program workload analysis very difficult. Without knowing in advance the processor needs, dynamics measurements are needed to apply DFVS. Furthermore, because it is also difficult to predict the future workload, DFVS can result in a lack of performance. Some algorithms have been developed to apply DFVS in a most efficient way. These are *Interval based approach* [18; 19], *Intertask approach* [20], *Intratask approach* [21; 22] and *memory bounded code approach* [23; 24].

The second reason involves mechanisms internal to the CPU. The frequency hopping stops the execution of instructions during the hop and has an effect on clock-dependent tasks such as the serial communication for instance. The DFVS utility implemented in the kernel should be able to manage the other kernel modules to ensure that the hop will occur without any execution problems. A utility exists [25] for Linux systems but it is not implemented in all architectures.

Furthermore, some misconceptions are usually made. The first one is that total microprocessor system power is quadratic in supply voltage. It is true for a CMOS transistor, but not necessarily for an entire system. For instance, the AT91SAM9263 processor has height different types of power supply pins [12], at different voltages. If one of the supply voltages varies, this will not reduce the power consumption of the entire system quadratically.

The second misconception is that it is not necessarily more power efficient to execute a task at the minimum frequency needed to meet its deadline. Indeed, if the task is executed slower, then

the peripherals, memory and others are activated longer, consuming more power. The energy versus slowdown curve is then “U” shaped [26].

The third one is that the execution time is inversely proportional to the clock frequency. The execution time depends on more factors than the frequency like cache misses, memory latency or memory throughput.

To conclude, it is necessary to implement DFVS capability to improve the energy efficiency of HAMAC. Nevertheless, it should not result in a lack of performance or in an energy waste due to the reasons presented above. A simple approach is used in HAMAC, implemented at application level. It is described in section 3.1.1.3.3.3. It basically consists in applying DFVS when the system is not used by the user. Instead of having a workload monitoring, HAMAC monitors the user activity and reacts accordingly.

3.1.1.3.1.1.6 Hibernation

Finally, to reduce drastically the dynamic power consumption, the processor can simply be set in sleep mode. This is possible when the system is not used at all. The processor does not need to be powered on all the time, above all when the user is not using it. The difference between setting the CPU in sleep and turning off the system is that wireless communication can be kept alive. HAMAC stays connected to its environment network and allows a fast recovering from sleep. CPU is also not the only component where hibernation technique is useful. Memory and screen can also provide some power saving facilities.

3.1.1.3.1.1.7 Peripheral shutdown

The AT91SAM9263 allows the disabling of peripheral clock [12]. This results in a peripheral shutdown and thus it does not consume any power. Some peripherals provided by this microcontroller are not used in this project. For instance, the CAN or I²C peripheral are useless in this scope. They should be shutdown in order to not waste any energy.

3.1.1.3.1.2 Memory

The memory is also a cause of energy consumption. We focus on the SDRAM chip used on Atmel development board AT91SAM9263-EK [27]. According to [13] and as seen in Table 4, the memory can use 135mA at 3.3V. But the chip can be set in some state to save power when it is unused. These states are presented in Table 5.

| Mode | Current |
|--------------------------------|---------|
| Active | 135mA |
| Standby: Power down | 2mA |
| Standby: Active | 40mA |
| Burst | 135mA |
| Auto-refresh: during a refresh | 270mA |
| Auto-refresh: idle | 3.5mA |
| Self-refresh Standard | 2.5mA |
| Self-refresh Low Power | 1.5mA |

Table 5: Current consumption of a SDRAM according to different mode

The particularity of a SDRAM chip is its need to be powered on in order to keep data. Each memory cell needs a refresh to maintain data consistency. It occurs each 64ms. According to [13], the 256Mb SDRAM needs 8,192 refresh in 64ms. Thus it needs one refresh each 7.81 μ s. In self-refresh mode, this is done internally but data is inaccessible.

The idea is to set the memory chip in standby or in self-refresh mode when the RAM is not used. Typically, this will occur when the microcontroller is set in sleep mode to save some energy because this mode does not allow any read or write access. Otherwise, the other modes are the normal operating modes managed by the kernel.

3.1.1.3.1.3 Screen

The screen is the most power consuming hardware on the board. There is no magic action to do in order to save some energy. The basic method is to turn off the backlight and the display when the user is not using it. It is the role of our power management system to decide when to turn off the screen, and when turn it on. And second way is to adapt the backlight by dimming it.

The touch screen is one of our input systems; it is thus difficult to turn it off. But in case of the use of the tongue system, the user will be able to turn it off manually or to configure to turn it off and on automatically in case of a plug of the tongue input.

3.1.1.3.2 Abstraction mechanisms offered by the system

3.1.1.3.2.1 How to reduce the processor frequency

The possibility to change the processor frequency on Linux is given by "CPUFreq" [28]. This kernel module is composed of three parts:

- The core provides standardized interface for the CPUFreq architecture drivers as well as to "notifiers". The "notifiers" are used to communicate to other part of the kernel that the frequency is changing.
- The CPU driver is used to change the frequency
- The governor is used to follow a policy of frequency changes.

A governor is running as a kernel module. It switches the frequency following a specific policy. Several kinds of governors are available:

- Performance: keep the frequency at the highest frequency
- Powersave: keep the frequency at the lowest frequency
- Userspace: put information and control available in userspace in order to be controlled by another program.
- Ondemand: scale the CPU frequency depending on the needs. It increases the frequency in putting the maximum value and decreases it step by step.
- Conservative: It works like Ondemand but by increasing the frequency step by step.

CPUFreq maintains information about the CPU in the virtual file system. Programs running in user space are thus allowed to give some commands in order to implement their own policy.

3.1.1.3.2.2 How to monitor resources usage

All information about processes in Linux is stored within the processes file system [29]. HAMAC can access to the “/proc” directory, which is a virtual file system in order to get information about kernel. This file system is also used by “procp” utility which makes available some commands to control processes in user space. Information about HAMAC software is situated in the “/proc/HAMAC_PID” directory where HAMAC_PID is the actual pid of HAMAC running on Linux. CPU usage but also used RAM memory and/or opened files can be seen inside this directory.

The “getrusage” function available in <sys/resource.h> can be used to get information about the running process (e.g. HAMAC) [30]. This function gives us a structure with the time running in user space, the time running in system mode, and other process information.

3.1.1.3.2.3 Power Management in Linux

3.1.1.3.2.3.1 APM

APM (Advanced Power Management) is a set of specification developed by Intel and Microsoft for managing power in computers. It defines an independent software interface between hardware specific power management software and an operating system power management policy driver [31]. APM uses a layer approach. Application using APM communicates with APM driver of the operating system which communicates with the BIOS which communicate with APM-aware hardware. It is also possible to bypass the BIOS and communicate directly with the device. This system has been mainly developed for X86 computer architecture.

3.1.1.3.2.3.2 ACPI

ACPI (Advanced Configuration and Power Interface) is the replacement for APM. It does not work only on power management but also in device configuration [32]. In a same way, it provides a unified interface for operating systems to manage power. It is also mainly developed for X86 computer architecture.

3.1.1.3.2.4 Drivers

The problem is that APM and ACPI implementations are not available in the HAMAC board architecture. To perform power management, both Linux kernel and HAMAC software have to communicate with device drivers. DFVS is performed by means of the CPUFreq driver, memory management through the memory driver and screen management through the corresponding drivers. The open-source community provides drivers to manage devices presented in section 3.1.1.3.1.

Communication with drivers is performed through the “/sys” interface. It is a virtual file system maintained by the kernel and allowing user-space application to communicate with devices. For instance, on a standard computer running Linux, changing the value in “/sys/class/backlight/.../brightness” file will alter the intensity of the backlight.

3.1.1.3.3 Defining global energy modes

Section 3.1.1.3.1 showed that devices allow several power states. The goal of the power management system is to set these devices in the lower power state as long as possible so that HAMAC consumes as less power as possible. Section 3.1.1.3.2 lists the features available in Linux to manage power. With these data, the power management system design can be started.

The design consists in defining several states in which the system will run. States correspond and are adapted to different uses cases. Furthermore, the establishing of a power state transition diagram is recommended by [33] of Linux Journal. But before presenting a diagram, both states and transitions between them are defined. States are defined for the microcontroller and memory on one hand, and for the screen on the other hand. The screen states are independent from the microcontroller's ones.

3.1.1.3.3.1 States definition

Four states can be defined for the microcontroller. They are listed in Table 6.

| | |
|---------------------|--|
| Run | It is the normal state, when everything is powered on |
| Shutdown | It is the state when everything is powered off |
| Conservative | This state represents a mode where the system is degraded. It means that the microcontroller works at a lower frequency. |
| Sleep | In this state, the microcontroller is set in sleep mode. The RAM memory is put in self-refresh mode. The wireless chip will be notified that the system is sleeping so that they will be able to manage their power policy in a way to consume the less energy as possible. All methods are used to allow a fast transition in run mode. |

Table 6: Four states in which the system can be defined.

The advantage of the *run* mode is that the system is at his fastest mode. The response time is little because it has not to change state. It is already at its fastest point. But it is also the mode where it consumes the most. The benefit of *sleep* mode is the very little amount of energy used by the system. But in this mode, the user cannot manipulate HAMAC. *Conservative* is a mix of *sleep* and *run*.

It consumes less energy than *run*, but more than *sleep*. It is slower than *run* but the system stay usable. Concerning the screen, three modes are defined. They are listed in Table 7.

| | |
|---------------|--|
| ON | In this mode, the screen is ON with the maximum backlight intensity. |
| Dimmed | In this mode, the screen is ON but with reduced backlight intensity. |
| Off | In this mode, the screen and backlight are OFF. |

Table 7: Screen modes

Now, with states defined, the transitions between them are specified.

3.1.1.3.2 State Transition

In case of inactivity, the system can pass from *run* to *conservative* mode. In this mode, both the voltage and frequency of the processor are reduced. If the inactivity keeps going, the system goes from *conservative* to *sleep* mode. The processor is set in sleep mode; the memory is set in self-refresh mode (see section 3.1.1.3.1.2) and the wireless chip in stand-alone mode in a way that they can save the most energy without ending network connections. In case of activity, the system is set to *run* mode. This is represented in Figure 12.

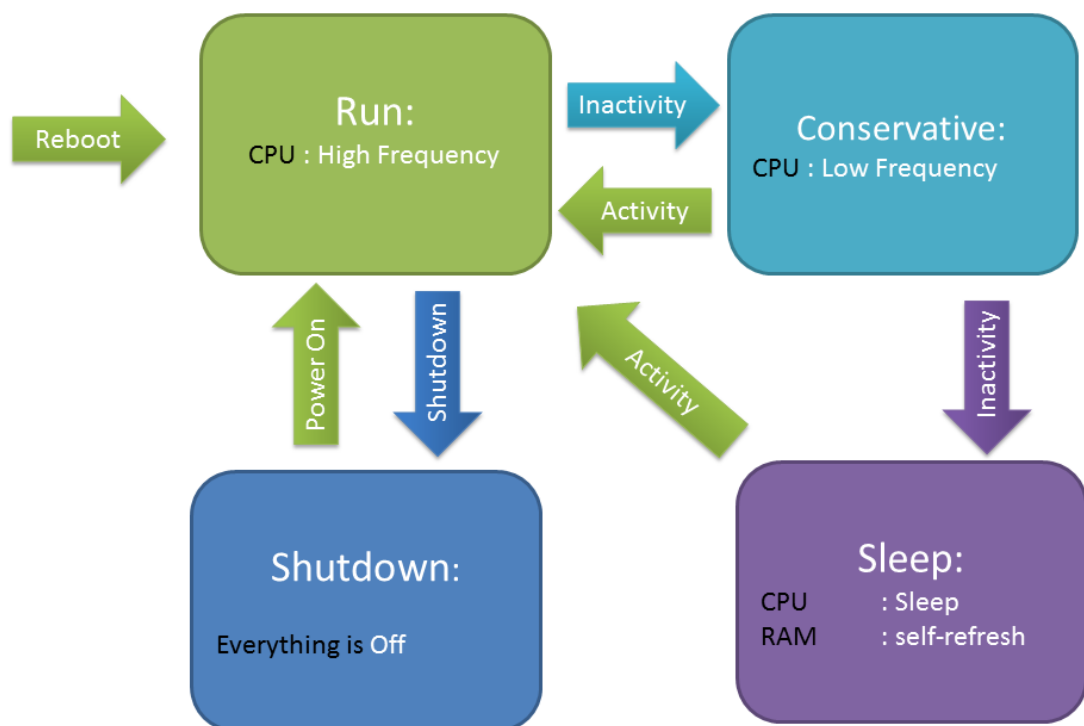


Figure 12: State transition diagram of microcontroller and SDRAM

In the same way, the transitions between screen states occur in case of inactivity. If the user is not using the screen, it starts to dim, then it turns off. If the user is using the screen, it turns on. The transition follows the representation shown in Figure 13.

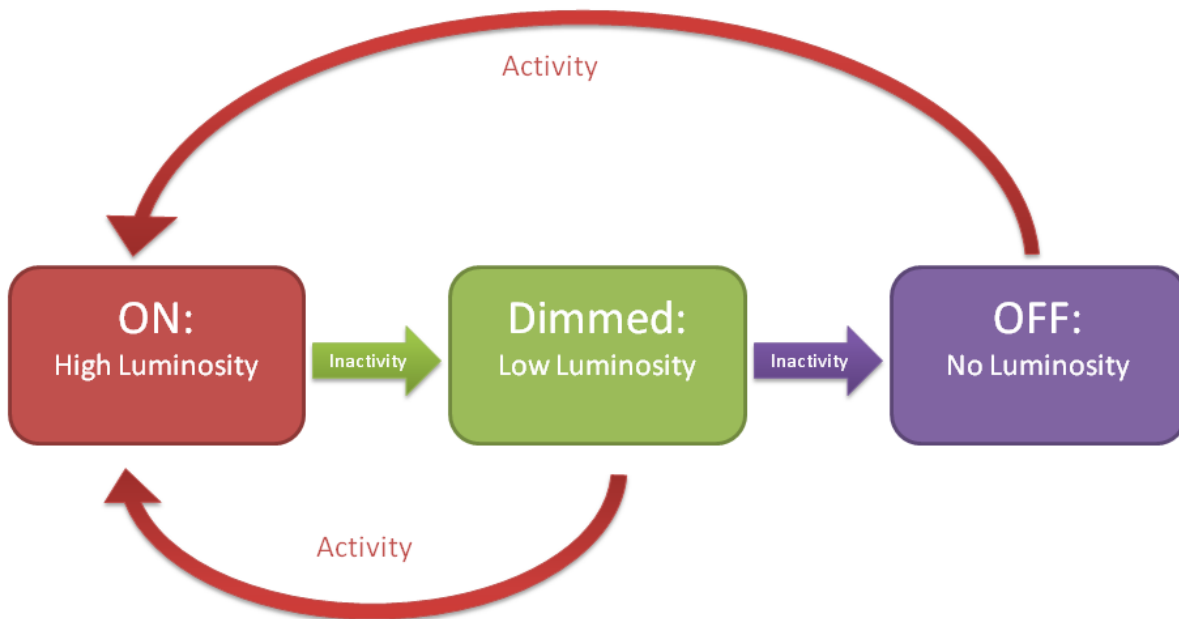


Figure 13: State transition diagram of screen display.

We define an activity as an action made by the user to control HAMAC. The way of monitoring user's activities is to watch on user input. These inputs are:

- Touch screen
- Tongue device

A threshold time is defined to assume that the user is not using HAMAC anymore. This can be easily implemented by a timer reset each time that the system receives an input event. On time out, the system state changes. For instance, if the threshold is one minute, the system waits for an input during one minute. If there is one, the timer is reset and the system is waiting again for one minute. Otherwise, after one minute, HAMAC enters in *conservative* mode and energy is saved. The same mechanism can be implemented for the transition between *conservative* and *sleep* modes.

Another timer is used to adapt the backlight. Its operation is very similar to the one used for the processor. In case of inactivity, the backlight is reduced. If the inactivity keeps going, the backlight is reduced even more, until being turned off together with the screen.

The values of the timers are not fixed yet and adjustments are required. The threshold time is important. Indeed, changing state can be energy consuming. For instance, in case of DFVS, the processor needs to perform some tasks before changing its working frequency. These tasks represent a cost. Indeed, all system clocks have to be recalibrated. For instance, timers used for refreshing the SDRAM and for communicating through a serial line depends on the master clock of the chip which depends on the working frequency [12]. These timers have to be recalibrated to have the same time out rate. If HAMAC is changing its state too often, the cost is added and power is wasted. On the other hand, if HAMAC is taking too long time before changing its state, power can be wasted in case of the system is unused. A balance has to be found. This value corresponds to the threshold time.

3.1.1.3.3.3 Flags

The mechanism presented above is quite simple. More information should be monitored. Indeed, if the user is using a service (mouse via Bluetooth for instance), he/she does not need the screen to be active but the processor should be in active mode to use the service. Two possibilities exist to prevent the system to enter in *sleep* if the user is using a service.

The first possibility is to monitor the CPU activity by using the *getrusage* system call described in section 3.1.1.3.2. If this activity is too high, the system stays in *conservative* or *run* mode. But this can be difficult to implement according to section 3.1.1.3.1.1. Indeed, an analysis has to be performed to predict the future and this analysis can drive a non-negligible quantity of computational and electricity power.

The second solution is to use a flag system. Each service knows in which scope the system can change its state. When a service is used, it can set a flag in the power management system to indicate HAMAC how the transition can be performed and what is not allowed to perform.

A flag is needed when a service decides to use the full ability of the processor computational power. This flag prevents the system to enter in *conservative* mode and thus prevents the use of DFVS. Without the ability to enter in *conservative*, the system cannot obviously be set in *sleep* mode. This flag is named “power”. A second flag is needed when a service decides that a feedback is needed on the screen. This flag prevents the system to enter in *dimmed* and *off* modes. It is named “display”

These flags extend the power management system by bringing more flexibility. More different modes are thus possible and allow the system to adapt itself to its use by the user.

3.1.1.3.4 Design impact on the HAMAC Framework

A timer class is represented by Figure 14. It contains four main methods which are “set” to set a timer, “start” to start the timer, “stop” to stop it and “reset” to reset it. To complete the design, more methods can be added like “pause” to pause the timer.

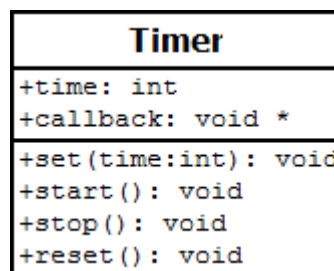


Figure 14: Timer class

The basic use of a timer is to create it, give a callback method, and start it. On time out, the timer will execute the callback and wait to be activated again. The implementation can take also into consideration that the timer should be started again during the execution of the callback. Then on time out, it sends a signal and can be started directly after. A handler intercepts the signal and executes the callback. Both of these situations are illustrated in Figure 15.

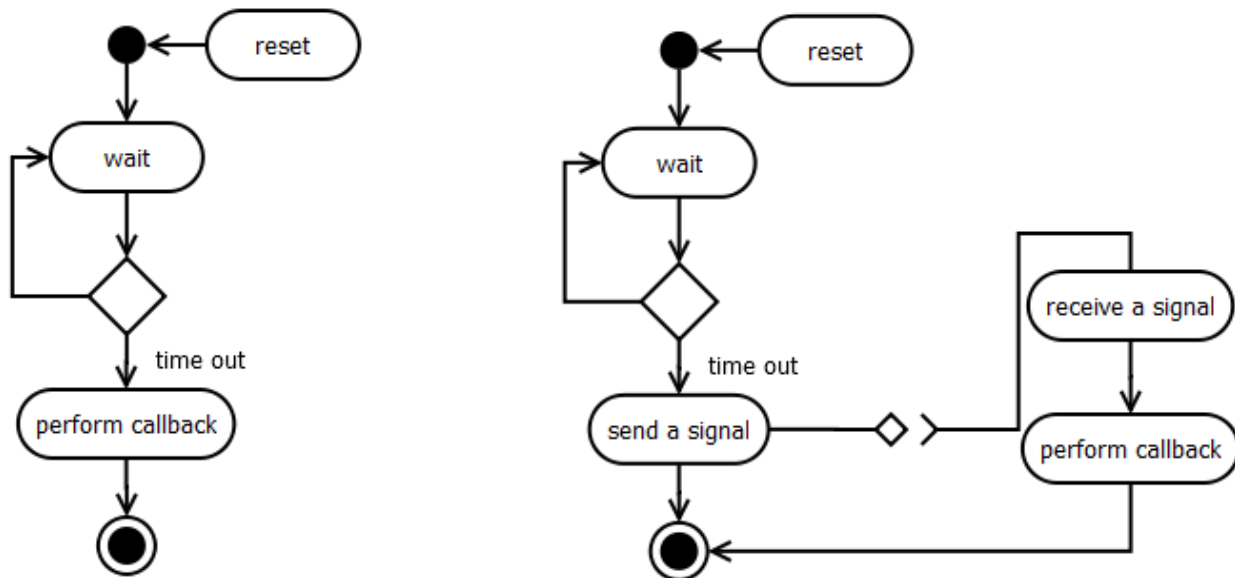


Figure 15: activity diagram of two implementations of timer

The power management system is designed by a class. This class contains all data and methods needed by PMS. More precisely it consists of the flags presented in section 3.1.1.3.3.3, timers, internal attributes, methods access flags, methods to manage timers, and methods to change modes. This class is shown in Figure 16.

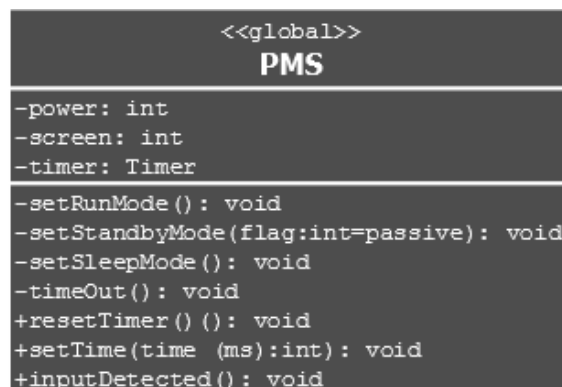


Figure 16: Power Management System Class

This class is instantiated at the end of HAMAC initialization. The principle is that at its creation, the class launches a timer. On time out, it watches on which mode the system runs, watches which flags are set and takes a decision to change the mode or to remain as before. This is represented by the activity diagram seen in Figure 17.

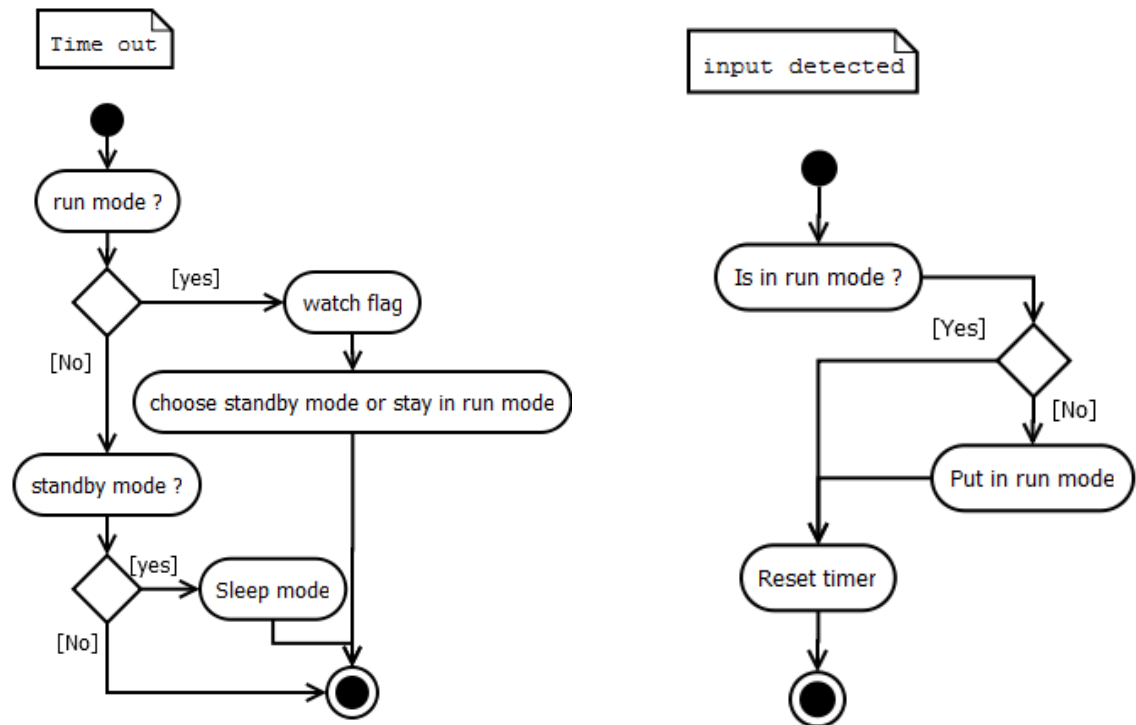


Figure 17: Activity diagram representing tasks performed on time out and when an input is detected.

To prevent the system changing its state in case of input event, the timer is reset on each detected event. If the system was in *conservative* or *sleep* mode, it is set in *run* mode in detecting an input event. This is shown in Figure 17. Since the GUI is used to fetch input event, it is its responsibility to reset timers. In some situations, the user input is directly used by the service, without being transmitted to the GUI. In this case, the GUI cannot reset the timer and HAMAC can be put in *conservative* and *sleep* mode. It is then the responsibility of the service to reset the timer. It is the role of a service to update PMS' flags. The Service class is described in section 2.2.1.2.2.1.3, but the class definition is reminded in Figure 18.

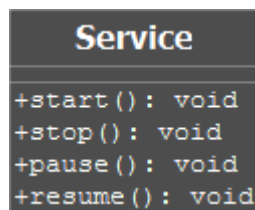


Figure 18: Service Class

We can see in Figure 18 that four methods are defined: Start, Stop, Pause and Resume. These methods are called when a service is used. Inside the implementation of these, the service sets the PMS flag according to its activity. For instance, if the service does not need the screen but a lot of computational power, it sets the *power* flag defined in the PMS class.

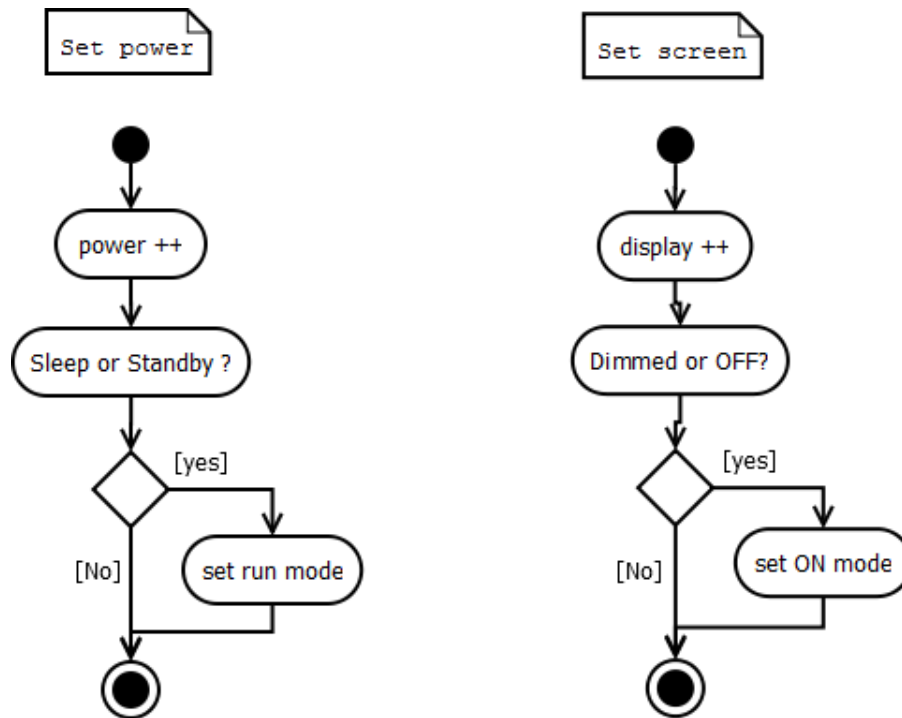


Figure 19: Activity diagram of methods which set the *power* and *screen* flags.

Figure 19 describes the tasks performed by methods which set the *power* and *screen* flags of the PMS class. We can see that if both of these flags are set, the system is set in *run* mode. Indeed, this mode corresponds to both needs for lots of computational power and screen feedback.

3.1.1.4 Implementation Details

In this section, details regarding the implementation are described. There is a discussion about how to implement a timer, and one about the way to communicate to drivers.

3.1.1.4.1 Timer implementation

The easiest way to implement timers in our case would be to use the *Timer* class provided by Qt library. But in this case, the independency of GUI library is not achieved. To respect the choices made to have the most independent package, another implementation choice has to be made.

Two main methods have been thought. The reader should be aware that in programming, there are many different ways to perform the same task and it is not the goal here to enumerate all of them.

3.1.1.4.1.1 Timers using threads

One method used to implement timers is the use of thread. Basically, the thread is launched, wait a time and then execute a command. The reset is made in cancelling the thread and starting again. One thread is created for each timer. The thread can also send a signal instead of executing the command. Then it is the role of the timer to catch the signal and execute the right callback. Nevertheless, this implementation can be a little bit heavy by the use of thread mechanisms.

3.1.1.4.1.2 Timers using “alarm” mechanism

The second method use the “alarm” method provided by C library [34]. The basic use is to configure the alarm time and then start it. In case of time out, the signal “SIGALRM” is sent to the process. This signal has to have a handler since the default action of the process receiving it is to terminate. The advantage is that this system is lighter than the use of threads but only one alarm can be used by process. To bypass this limitation, the duration of several timers is stored. Then the alarm is set for least value. On timeout, the alarm is set again with the difference between the previous duration value and the next one. This is the method used in our system.

3.1.1.4.2 Driver communication

Drivers in Linux are often developed as kernel modules. There are three classes of modules in Linux systems [35] which are:

- Character devices: it can be accesses by a character file or stream.
- Block devices: it supports a file system and can only perform I/O operations.
- Network interfaces: support a network connection.

Methods to communicate to drivers are different according to the class they belong to. The focus is put in the first class: character devices. A driver of this type has an interface on the virtual file system maintained by the kernel. Communication is performed by writing and reading bytes through a file descriptor.

3.1.1.4.2.1 Backlight driver

Files to manage backlight are situated in the “/sys” file system. Its location can change according to the hardware. The path looks like “/sys/class/backlight/atmel/” In this directory several files can be found. Interesting ones are:

- “brightness”: this file allows us to change the intensity of the backlight
- “max_brightness”: this file contains the maximum value of the brightness allowed by the system.

3.1.1.4.2.2 CPU driver

Files to manage cpu are situated in “/sys” file system. The path is “/sys/devices/system/cpu/cpu0/cpufreq/” In this directory several files can be found. Interesting ones are:

- “*cpufreq_min_freq*”: This file contains the minimum value of frequency acceptable by the processor in kHz
- “*cpufreq_max_freq*”: This file contains the maximum value of frequency acceptable by the processor in kHz
- “*cpufreq_cur_freq*”: This file contains the current value of frequency the processor is running at. It is in read-only mode.
- “*scaling_min_freq*”: This file allows putting manually a minimum value of frequency. The value has to be between *cpufreq_min_freq* and *cpufreq_max_freq*.
- “*scaling_max_freq*”: This file allows putting manually a maximum value of frequency. The value has to be between *cpufreq_min_freq* and *cpufreq_max_freq*.

- “*scaling_setspeed*”: This file allows putting a new value of frequency acceptable by the processor. This value has to be between *scaling_min_freq* and *scaling_max_freq*.

3.1.1.5 Evaluation

In order to evaluate the usefulness of the power management system, a comparison needs to be made between HAMAC with and without this system. The goal is to find out the amount of energy saved with the PMS. We will perform this evaluation theoretically.

In this evaluation, the power consumption of HAMAC will be calculated with the maximum values of the chips. This data represents the system energy consumption without any tools to regulate it. In a second time, a use case is defined. A duration value for the different microcontroller modes, presented in section 3.1.1.3.3, is defined for this use case. Similarly, a duration value for screen modes, presented in section 3.1.1.3.3, is defined. In a third time, power is estimated for the use case and compared to the maximum power value.

It is very important to notice that there can be thousands of different use cases. The goal is not to perfectly reproduce the reality but to show that theoretically, the power management system allows a significant reduction of the power and energy consumption for a typical user.

3.1.1.5.1 Worst case of power and energy consumption

The worst case appears when all components are turned on and are using the maximum value of electrical power. In order to calculate the worst case power consumption of HAMAC, the values presented in Table 4 are used. They come from datasheets of products present in AT91SAM9263EK board. As specified in section 3.1.1.5, the values of radio chips are skipped in this evaluation. The maximum values measures at 25°C are listed in Table 8.

| | Power |
|------------------------|----------|
| Microcontroller (core) | 84mW |
| Screen LCD | 660mW |
| Screen Backlight | 75mW |
| Touch Panel | 125mW |
| Memory (SDRAM) | 445,5mW |
| Total | 1389,5mW |

Table 8: Power consumption of main components present in HAMAC's board

The worst case total power consumption of elements present in Table 8 is 1389.5mW.

3.1.1.5.2 Use case

The goal of a use case is to represent the typical actions made by the user in using HAMAC. It is not an easy task because the user can manipulate the system in thousands of different ways. Nevertheless, a use case can be defined to evaluate the energy consumption of HAMAC in a typical day. For an easier reading, the user is named Mr. Nielsen. In this use case, Mr. Nielsen is using exclusively the tongue system. The touch panel is thus deactivated.

The method consists in establishing a daily schedule of Mr. Nielsen. During this schedule, the duration of use of HAMAC is defined in addition to its running mode. At the end, are defined a percentage of duration time during a day for each mode. From this, the energy consumption during a day is evaluated with the value of power for each mode.

Several activities are defined: Sleep, Miscellaneous, Eat, Work and Watch TV. These activities are spread over 24 hours. For each activity, the time spent in each mode is represented. At the end, the total of time in each mode is calculated and then the energy consumption of HAMAC during this day can be found out.

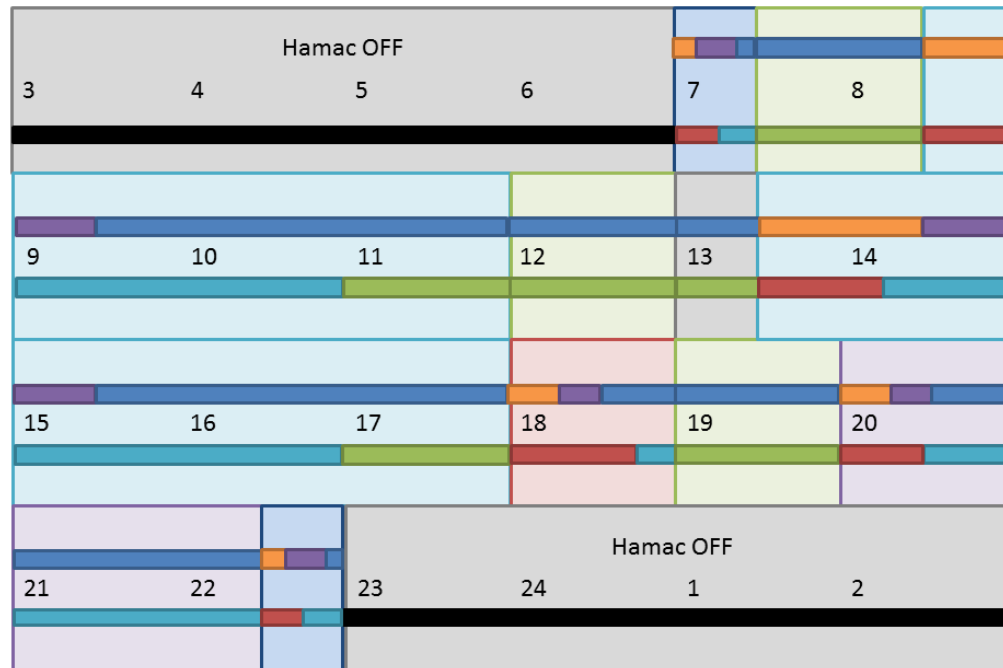


Figure 20: Daily schedule of Mr. Nielsen. Numbers indicate the time of the day.

Mr. Nielsen's daily schedule is presented by Figure 20. The items from Figure 1 are detailed in Figure 21.

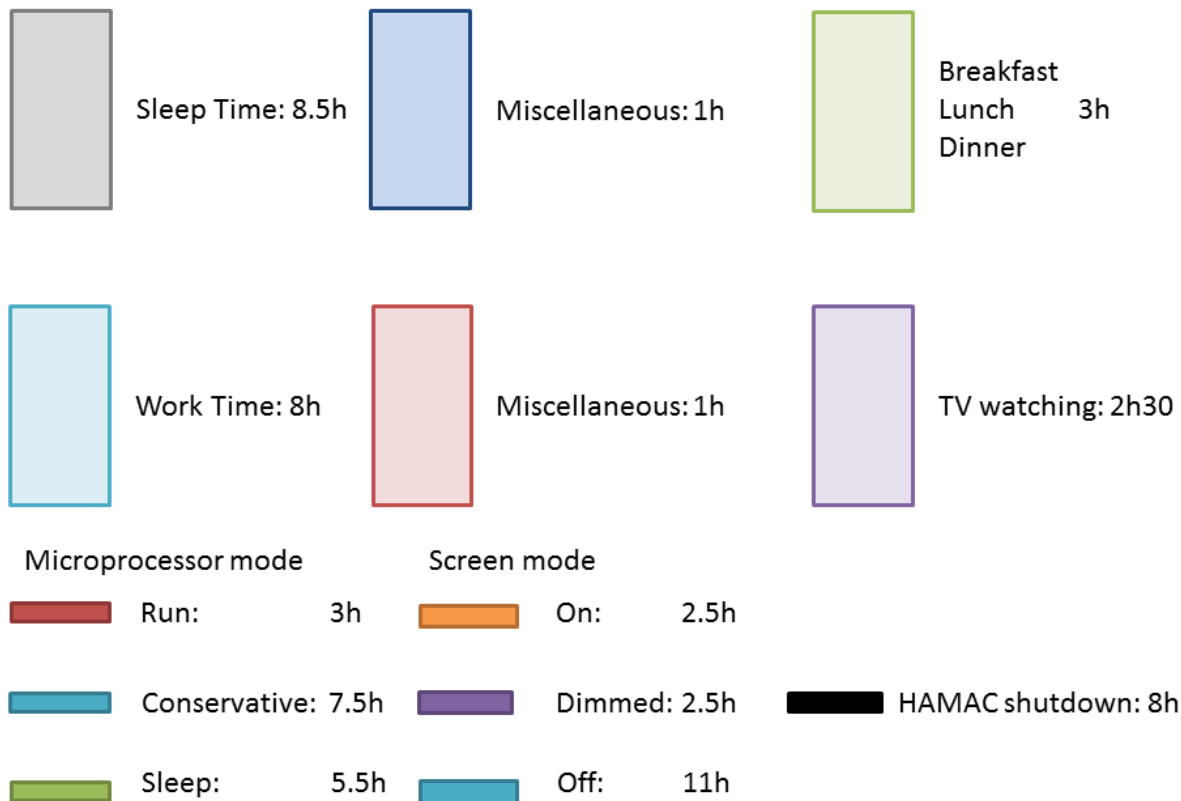


Figure 21: Items in Mr. Nielsen's daily schedule

During sleep time, HAMAC is not used, it is shutdown. The blue miscellaneous time corresponds to the time after and before sleep. Mr. Nielsen is going to move around in a wheelchair and to turn lights off and on. The red miscellaneous time corresponds to the time after work. In this time, the wheelchair use is the most important. During the work time, the computer use is more important. During meals, Mr. Nielsen is not going to use HAMAC so it stays in *sleep* mode. When watching TV, HAMAC is mainly used to control the lightning and the TV.

3.1.1.5.3 Power used in each mode

Finding the power used by components in modes where HAMAC uses less power is quite difficult. Nevertheless, Atmel provides an application note [36] with power consumption values of the AT91SAM9261 microcontroller. We have taken these values and extrapolated them to the AT91SAM9263 microcontroller by a cross multiplication since there is no document available for this microcontroller. Table 9 presents the power consumption values of the microcontroller (μC) and the SDRAM in *conservative* and *sleep* modes. The values in *run* mode are the maximum ones presented in Table 4.

| | Conservative Mode | | | Sleep Mode | | |
|--------------------------|-------------------|------------|-------------|------------|-----------|-------------|
| μC (Core) | 1.2V | 0.56mA | 672 μ W | 1.2V | 3 μ A | 3.6 μ W |
| μC (Static current) | 1.2V | 0.7mA | 840 μ W | | | |
| SDRAM | 3.3V | 125mA [36] | 412mW | 3.3V | 2mA | 6.6mW |

Table 9: CPU and SDRAM power consumption in conservative and sleep mode

The power consumption values of the screen are presented in Table 4. The calculation is simple since both screen and touch panel have only two states: on or off. The calculation of the power consumption of the backlight is trickier because the intensity of brightness is varying. To have simple consideration, a value of 50% of brightness is taken for the dimmed mode. The current is 10mA with a voltage at 3.75V [15]. The power is thus 37,5mW.

3.1.1.5.4 Use case's power and energy consumption

The time spent by HAMAC in each different modes defined in section 3.1.1.5.2 is associated to the power consumption of HAMAC in these modes calculated in section 3.1.1.5.3. The values of Table 4 and Table 9 are used. The result is presented in Table 10 below:

| Modes | Formula | Power | Duration (H) |
|--------------|--|-----------|--------------|
| Run | $\max(\mu C) + \max(\text{SDRAM})$ | 529,5mW | 3 |
| Conservative | $\text{Conservative}(\mu C) + \text{conservative}(\text{SDRAM})$ | 413,512mW | 7.5 |
| Sleep | $\text{Sleep}(\mu C) + \text{sleep}(\text{SDRAM})$ | 6.6mW | 5.5 |
| On | $\text{On}(\text{screen}) + \max(\text{Backlight})$ | 735mW | 2.5 |
| Dimmed | $\text{On}(\text{screen}) + 50\%(\text{Backlight})$ | 697.5mW | 2.5 |
| Off | $\text{Off}(\text{screen}) + \text{Off}(\text{Backlight})$ | 0mW | 11 |

Table 10: Power consumption in different modes

To calculate the average power consumption of HAMAC in its time of use (16 hours) defined in section 3.1.1.5.2, first, the average of the consumption of the microcontroller and SDRAM is determined, then the average of the display system (screen and backlight) is reckoned and finally, the average of both values is computed. These calculations are presented in Table 11 below:

| Elements | Formula | Result |
|------------------------|---|--|
| $\mu C + \text{SDRAM}$ | $\frac{\sum(\text{mode's power} \times \text{duration})}{\text{total duration}} = A1$ | $\frac{529,5 \times 3 + 413,512 \times 7.5 + 6.6 \times 5.5}{16} = 295,38mW$ |
| Display system | $\frac{\sum(\text{mode's power} \times \text{duration})}{\text{total duration}} = A2$ | $\frac{735 \times 2.5 + 697.5 \times 2.5 + 0 \times 11}{16} = 223.83mW$ |
| HAMAC (core) | $A1 + A2$ | 519,21mW |

Table 11: Power consumption of HAMAC according to the use case defined in section 3.1.1.5.2.

According to data from Table 10 the average power consumption is: 519,21mW. Taking in consideration the maximum value of power presented in section 3.1.1.5.1 which is 1389.5mW, the PMS allows minimizing the power by a factor greater than 2.5.

3.1.2 REDUCING THE CONSUMPTION OF THE RADIO CHIPS

Now that we have covered the mechanisms for lowering the power consumption of the general HAMAC platform, we'll focus on optimizing the power consumption of the radio chips present on this platform.

3.1.2.1 Specificities of the radio chips

The radio chips require special interest because of the modularity of HAMAC: there can be as many radio chips on the HAMAC board as protocol plugins loaded in the HAMAC Framework. Controlling the energy consumption of the radio chips efficiently can thus be a key factor for reducing the overall energy consumption of a very interoperable HAMAC board.

Before describing how such a control can be implemented, let's describe the main characteristics of the radio chips: they will frame the mechanisms for saving energy.

3.1.2.1.1 A common goal: communicate

To begin, the radio chips are designed for communicating. This communication involves two main activities: transmitting and receiving radio waves. The specifications of the communication protocols define how these transmissions and receptions are organized over time for data to be exchanged between two devices. Understanding this organization is essential for optimizing the number and length of the transmissions and receptions performed by a device. The radio chips consuming a lot of power through these activities, it is also a good step for optimizing the energy consumption of these chips.

All wireless communication protocols divide the life cycle of a communication in at least four steps: device discovery, connection, communication and disconnection. Let's look at them more closely.

3.1.2.1.1.1 *Device discovery*

Device discovery is a necessary step to interconnect wireless devices that do not know anything about each other. It enables devices to detect which other devices are around and to get enough information about them for initiating connections. This step is important in wireless communications, as the user cannot link manually two devices together with a wire: the connection have to be initiated by the devices themselves.

As any kind of communication, device discovery requires devices to exchange data packets between each other. To be detected, a device needs to listen for incoming discovery requests when they come and to answer to them. However, listening continuously for incoming data packets is power consuming. Therefore, most protocols define complicated mechanisms to limit the power consumption introduced by device discovery.

The counterpart of such mechanisms is that they make device discovery more time consuming. The first version of the Bluetooth protocol required for example 10.24 seconds for one device to detect all other devices in its environment [1]. Nevertheless, some latency for discovering

new devices is often acceptable by the user. Continuously searching for other devices has also very little sense, as the environment of the wireless device is not expected to change very often. Detecting other devices around should be though as a balance between the latency of this process and the power consumption of the radio chips. One solution used by wireless devices is to perform this device discovery every few minutes, to keep an active list of the devices around without impacting power consumption too much.

3.1.2.1.1.2 Connection

After having collected some information about each other, wireless devices can begin the process of connection. This step is equivalent to the step of connecting two devices with an invisible wire. During it, the two devices decide if they accept to be connected to each other. They also exchange additional information to configure their communication link, and eventually define the way it should be encrypted.

Performing all these tasks take some time and energy. To minimize those, the protocols often differentiate the processes of first connection and of reconnection. The reconnection is made easier, as devices are allowed to remember the parameters of the first communication link that they established.

Both processes, however, have a latency that can be undesirable for some applications. This reason pushes most devices to maintain their communication links open as long as possible, to avoid going through time consuming disconnections and reconnections.

3.1.2.1.1.3 Communication

Once the two devices connected together, a communication link is finally established between them. This communication link can be used for exchanging data, but it can also be kept open without being used. In fact, applications do not often use the full bandwidth of the protocols that link them to other devices. Therefore, the communication can be made without the radio chips transmitting or receiving data continuously. A communication link can also stay open without any data being transferred on it. The advantage in this case is that the two devices can begin re-exchanging data when needed, with very little latency.

The protocols' specifications define both the way data should be transferred between the two devices and the conditions following which the communication link can stay open when idle. These conditions include most of the time a timeout which specifies how much time a device is considered connected after its last transmission. Because keeping the communication link open is convenient for a lot of applications, several protocols also enable devices to poll each other periodically to avoid closing it. Thanks to this mechanism, two devices can stay connected with the transmitters and receivers of their radio chips shut down – which is really good to lower their power consumption – without reducing the applications' latency.

To further reduce the power consumption of the radio chips in such situations, some protocols define several “sleep modes” enabling to increase the polling period. These “sleep modes” are mechanisms thanks to which two devices can decide to trade the low latency of their

communication link against lower power consumption. Some protocols offer several “sleep modes” to make these tradeoffs correspond more precisely to the application’s needs. Radio chips can in this way shut their transmitters and receivers for a very long time and still be able to come back to an active state faster than if the communication link was closed.

3.1.2.1.1.4 Disconnection

Disconnection of the communication link can be made either voluntarily or involuntarily. Voluntary disconnection is provoked when one of the two devices sends a disconnection request to the other. Involuntary disconnection happens when one of the two devices did not transmit to the other during more than a maximum determined time. It is useful in the cases when devices get out of range of each other or out of energy without having the time to send a disconnection packet.

3.1.2.1.2 A common medium: the radio

Another specificity of the radio chips is that their communication activities are all using the same medium to transmit information: the Hertzian waves. Knowing how they can access this same medium together is important when designing a device such as the HAMAC device, which is made to use several radio chips.

The Hertzian waves can be differentiated by their frequency, which is an important factor for defining how they propagate in the air. Nowadays, electronic components are also very good at filtering signals by frequency. Because of this, radio chips can focus on the messages transmitted within the range of frequencies defined by their related protocol. Therefore, two radio chips using non-overlapping ranges of frequencies to communicate can transmit and receive messages from the same place without impacting each other.

However, a problem can occur if two messages are sent at the same moment and at the same or at neighbor frequencies by two nearby transmitters. In fact, receivers in the range of the two transmitters will receive the sum of the waves generated by both signals. Because the signals are emitted at the same frequency, their filters will not enable them to differentiate the two messages. This special case is called a ‘collision’: it often causes one or the two messages to be indecipherable by the receivers.

When happening during a radio communication, this phenomenon does not impact much the reliability of the data transferred, because of special mechanisms implemented in radio protocols to guarantee data integrity. Invalid data is most of the time detected by the receiver, and a process of acknowledgement of the data packets between the devices makes the transmitter of the information repeat it later. However, this process requires several additional transmissions and receptions to be made during the communication. As a result, going through it causes additional power consumption.

The risk when using several radio chips together in the same device is to cause this collision effect to impact very often the communications. This risk is increased because of the small number of frequencies used nowadays by the wireless protocols interesting for HAMAC. In fact, the regulations imposed by governments concerning the radio waves reserve several ranges of frequencies – or bands – to applications in the fields of defense, mobile communications or civil

radio. This leaves the protocols with only a small numbers of bands to communicate in. The most known one is the 2.4 GHz Industrial, Scientific and Medical (ISM) band, which stays unlicensed all over the world. It is used for example by the Bluetooth, Zigbee or Wi-Fi protocols.

Wireless communication protocols divide the band they operate in into smaller ranges of frequencies, or channels. Some protocols enable radio chips to select the channels on which the communication links are established, while others require changing dynamically the channels used for the data transfers during communication. These characteristics give opportunities to avoid some radio collisions, as several protocols enable the user to select which channel or set of channels should be used for the communication. Making the radio chips work together to avoid radio collisions is therefore possible and should reduce the chips' power consumption.

3.1.2.1.3 Various power consumptions

With all these common specificities, radio chips still have a major difference: their power consumption. This difference is mainly due to the protocols operated through the chips, which specifications make more or less power consuming. It is also related to the field of application of these protocols. The Wi-Fi protocol is for example made for high data rate transfers, while the Zigbee protocol is made for low data rate interactions for sensors and remote controlling: the Zigbee chips are therefore far less power consuming than the Wi-Fi chips.

Reducing the power consumption of the radio chips can therefore be more beneficial concerning some chips than others. However, applying generic mechanisms for lowering the power consumption of the radio chips will enable us to reduce further the overall power consumption of the system, by optimizing the consumption of even the less power consuming chips.

3.1.2.2 Mechanisms for lowering the energy consumption

The mechanisms for lowering the energy consumption of the radio chips make an exhaustive use of the specificities described in the previous section. This section describes techniques for saving some power when communicating wirelessly. It underlines the advantages and constraints of each technique, and analyses when they could be interesting to use in the case of HAMAC.

3.1.2.2.1 Shutdown mode

3.1.2.2.1.1 Principle

The first way of saving power is simply to shut down the radio chips. In fact, HAMAC being highly interoperable, there is a high probability that the users will not need all the technologies with which it will be compatible. Therefore, a very low probability to discover a device using a given technology in the user's environment can justify powering the corresponding radio chip down.

3.1.2.2.1.2 Constraints

The main constraint of this technique is that it stops all activities of the target radio chip. The shutdown mode is thus inappropriate to use when some communication links are active on the chip. The radio chip shut down will also have to be woken up every time that the HAMAC framework needs to perform a device discovery using its technology. This operation costs both energy and time,

as the radio chip has to start-up again its internal logic and its communication link with the HAMAC platform before being able to perform a discovery. Therefore, these energy and time needs should be well known for the efficiency of this technique to be evaluated.

3.1.2.2.1.3 Conditions making this technique efficient

As a conclusion, shutting down radio chips is efficient when:

- The radio chip is not used in any active connection.
- During the estimated time in which the chip will be non-active, the energy consumed by the chip in idle mode would exceed the energy required for powering the chip down and then up.

3.1.2.2.2 Automatic disconnection

3.1.2.2.2.1 Principle

In many cases, shutting down the radio chips is a less efficient solution than keeping the chips awake but in idle mode. Once in idle mode, the radio chips need less energy and time to come back to normal operation. Another technique for lowering power consumption is therefore to put the chips in idle mode whenever no activity is needed and that shutting down the chip is less efficient than doing so.

This technique can also be extended by automatic disconnection. In some cases, in fact, disconnecting HAMAC from other devices will not impact the user's comfort of operation. This is the case when the interaction with a given remote device is only based on sending well-spaced and short control signals – the control of a Zigbee light is a good example of such an interaction. Another case where this technique can be applied is when a user took the control upon a given device but didn't use it for a long time. Disconnecting HAMAC from the remote device in such a situation can sometimes be an efficient way to save power.

3.1.2.2.2.2 Constraints

Of course, the point to clarify before using such a mechanism is when disconnecting HAMAC from a remote device impacts the user's comfort of operation. To this end, two main constraints are to take into account.

First, in some interaction patterns, cutting the communication link can provoke the remote device to come back to a default state, or more generally to prevent the user to come back to the state it was in before the disconnection. In such situations, disconnecting the communication link is strongly not recommended. It will prevent the user from controlling efficiently some devices, but can also put the user in danger by provoking unexpected behavior from the remote device.

The second constraint is the time required by a reconnection to the remote device. In fact, this technique of automatic disconnection should not prevent the user from taking back the control of the remote device within an acceptable delay. Therefore, the reconnection delays should always

be weighted to evaluate whether the automatic disconnection is acceptable or not regarding the user's comfort.

A last constraint concerns the energy consumed by the operations of disconnection and reconnection. If this energy is greater than the estimated energy consumed without disconnecting HAMAC from the remote device, the technique of automatic disconnection is obviously inefficient. As it is not possible to know when the user will need to interact with remote device again, the calculation determining whether if a disconnection can save power or not should be based on a probabilistic approach.

3.1.2.2.3 Conditions making this technique efficient

As a conclusion, disconnecting automatically is efficient when:

- The radio chip is involved in an active connection unused by the user.
- Disconnecting and reconnecting this connection will not impact the state of the remote device.
- The user can accept to wait the time needed for reconnecting HAMAC to the remote device before taking control upon the remote device again.
- During the estimated time in which the user will not use the connection, keeping the connection active would consume more energy than disconnecting the communication link and reconnecting it again.

3.1.2.2.3 Reduction of the power of transmission

3.1.2.2.3.1 Principle

Some protocols enable to reduce the power consumption of active connection by controlling the transmission power and reception sensitivity of the radio chips. This functionality is useful for reducing the power consumption of a radio chip when all the devices it is communicating with are at a shorter distance than the one covered by its maximum range. It can also be interesting to reduce the range of some radio chips so that the user can only control the devices within the same room as him or her.

3.1.2.2.3.2 Constraints

The main constraint when lowering the range of a radio chip is that this chip will not be able neither to detect nor to communicate with the devices outside its new range. This can be problematic if a new device appears in the environment outside this shortest range, or if the user decides to move while being connected to some of the devices.

To avoid this technique from impacting the functionalities of the HAMAC device, care should therefore be taken in verifying periodically that the adopted range is broad enough to enable the radio chip to communicate with all the devices in the environment properly. This involves taking periodically two actions: launching device discoveries with a range large enough to cover a room, and measuring the quality of the active communication links to adapt the range of the chip to the movements of the user or of the remote devices.

These actions can require more energy from the radio chip. The measure of the quality of the communication links can for example require additional interaction between the chip and the remote device. To make sure that this technique is efficient, the energy consumed in performing them should therefore be compared to the additional energy required when keeping the range of the chip to a constant value.

3.1.2.2.3.3 Conditions making this technique efficient

As a conclusion, reducing the power of transmission of a radio chip is efficient when:

- The devices to which HAMAC may connect itself can be accessed to when using a lower range.
- The HAMAC device is kept able of detecting periodically all devices within the room.
- The energy consumed to adapt the range of the radio chip and to measure periodically the quality of the active radio links is less important than the energy consumed when keeping a constant range.

3.1.2.2.4 Adaptive discovery schemes

3.1.2.2.4.1 Varying periods of discovery

3.1.2.2.4.1.1 Principle

When no device using their technology is available in range, the only requirement of the radio chip is to perform periodical device discoveries. However, as stated in section 3.1.2.2.1, the high interoperability of HAMAC makes it highly probable that the user will not use certain technologies very often. On one hand, performing device discoveries every now and then when the probability of discovering a device is extremely low is thus an unnecessary power consuming task. On the other hand, it is not possible to affirm that the user will never encounter a device using a certain technology.

One technique for lowering the power consumption of radio chips is to adapt the period of device discoveries according to the probability that the user has to encounter a new device in the environment. This probability can depend on several factors. Here are some examples that could decrease it if proven to be true or increase it on the contrary:

- The user has never used the target technology before.
- All devices used at least once by the user – using the target technology – were already discovered.
- No networks using this technology are detected.
- No devices using this technology are detected.
- No devices were added or removed from the environment recently.
- The quality of the currently open communication links – using any technology – is not varying.

The period of discovery could also be modified if the user does not make any input on the HAMAC device after some time. This can further reduce the power consumption of the HAMAC device when the user does not use it.

3.1.2.2.4.1.2 Constraints

The constraint of lowering the periods of discovery of the HAMAC device is to lower the ability of the device to react quickly to a change in the environment. This can impact the comfort of the user, as waiting several minutes before being able of controlling a device nearby can be very frustrating. Therefore, it is recommended that the discovery periods stay within a reasonable limit. Adapting them as precisely as possible to each situation is also the best way to ensure a good probability that the user will remain satisfied.

Because a balance is sometimes hard to reach, other solution may include letting the user chose by which factor the discovery periods should be extended when the discovery of a new device is not probable. Other types of configuration options could also let the user to disable the use of a certain technology, or force a global device discovery.

It is also important to ensure that getting the information required to calculate the periods of discovery for each technology do not require more energy than using a constant device discovery period.

3.1.2.2.4.1.3 Conditions making this technique efficient

As a conclusion, using varying periods of discovery is efficient when:

- The environment surrounding the HAMAC device does not change often.
- The user is offered options avoiding its usage of HAMAC to be too much impacted.
- Calculating the ideal periods of discovery does not require using too much additional energy.

3.1.2.2.4.2 Varying types of discovery

3.1.2.2.4.2.1 Principle

To reduce even further the power consumption of the discovery process, the various ways offered by protocols to detect other devices should be analyzed. Some protocols enable to detect only devices offering a certain type of service, for example. This feature can shorten the discovery process if only a few service plugins are available in the HAMAC Framework for a certain technology.

Another feature offered by some protocols enables to shorten the discovery process by accepting a varying probability of detecting all devices available around. Such shortened discoveries could be interesting to use when the period of discovery is short, as several of them could enable to detect all devices in the environment anyway.

Finally, when the probability of encountering new devices using a certain technology is low, trying to connect blindly to devices previously used by the user can sometimes be less power consuming than performing a device discovery.

3.1.2.2.4.2.2 Constraints and Efficiency

The possibility to use various types of discovery is protocol specific. This technique should therefore be controlled by the protocol plugins, in relation with the probability of discovering new devices calculated by the HAMAC core. The constraint of using several mechanisms of discovery is that a wrong discovery strategy can impact both the overall power consumption of the radio chip and the user's comfort of operation. To always make the better choice, protocol plugins should define precisely the advantages and constraints of each discovery scheme, and evaluate which combination brings the better balance between power consumption and usability.

3.1.2.2.5 Avoid making HAMAC discoverable or connectable when not needed

3.1.2.2.5.1 Principle

Some services of some protocols require the HAMAC device to be discoverable or connectable by other devices of its environment. This is for example the case of the Bluetooth HID service, which requires computer to be able to detect and connect to the HAMAC device upon the first connection. However, this requirement can be very power consuming. Moreover, it does not apply to every service and is only required temporarily by the services concerned. Another technique for saving energy is therefore to fulfill this requirement only when explicitly needed.

3.1.2.2.5.2 Constraints & Efficiency

The only constraint of this technique is to make sure that it does not impact the functionality of the given services. It is efficient as soon as the visibility of HAMAC is reduced.

3.1.2.2.6 Sleep modes

3.1.2.2.6.1 Principle

As introduced in section 3.1.2.1.1.3, several protocols define "sleep modes", which enable to lower the power consumption of the radio chips involved in a communication. These sleep modes can lower the number of interactions performed between two chips when they are connected. They can also help the chips to keep an idle connection active during a long time without exchanging any control packet. Therefore, they are very useful when an application does not use the full bandwidth of the protocol on which it relies. They can also be used when two devices do not need to exchange any more data but need to be able to take back communication quickly when needed.

3.1.2.2.6.2 Constraints

The first constraint of the sleep modes is that, by lowering the number of required interactions between devices, they also lower the communication links' reactivity. For that reason, using sleep modes inappropriately can create discomfort for the user. Nevertheless, sleep modes can make a good compromise between the options of keeping the connection alive or not between two devices. Often varied and configurable, they enable to adjust precisely the balance between power consumption and reactivity.

The second constraint of using sleep modes is that switching between modes of operation requires devices to exchange some data packets to agree on the next mode to use. As these

additional exchanges require energy, switching between sleep modes or from the normal mode to a sleep mode can sometimes be less efficient than letting the communication link in its current state. Consequently, using sleep modes requires evaluating carefully the power consumption needed both to enter and to leave the sleep mode.

3.1.2.2.6.3 Conditions making this technique efficient

The implementation of sleep modes can vary a lot between protocols. To be efficient, their management should thus be made within the protocol plugins. However, their use being closely related to the applications' needs in terms of reactivity, it should be based on an indicator given by the HAMAC core to adjust as precisely as possible the balance between power consumption and user comfort. All the protocol plugins should implement a function enabling to define which sleep mode to use depending on the user needs and the specificities of each mode.

3.1.2.2.7 Radio collision avoidance

3.1.2.2.7.1 Principle

In section 3.1.2.1.2 we introduced the problems which may occur when several nearby radio chips are using the same range of frequencies to communicate. To prevent the radio chips from having to retransmit too often their data packets, care should be taken in avoiding as much as possible radio collisions. This would both increase the reliability of the communication and lower the overall power consumption of the radio chips.

Fortunately, several protocols using the same bands of frequencies implement mechanisms for letting the user choose which radio frequencies to use, or more often which radio frequencies to avoid. Although these mechanisms are sometimes limited, a technique for lowering the power consumption of radio chips should try as much as possible to take advantage of them. One interesting approach is to make the protocol plugins indicate the band of frequencies that they will use to the HAMAC framework. In case of conflict, they can then negotiate which protocol should use the incriminated frequencies, depending both on the requirements of each protocol and on their ability to change their operating frequencies. This way, the number of frequencies in conflict should be decreased, thus decreasing the power consumption.

3.1.2.2.7.2 Constraints

The first constraint of this technique is that changing the frequencies used by an open communication link involves additional negotiation between the two devices in collaboration. For this reason, the estimated energy lost by retransmitting some data packets should be compared to the energy lost by performing such negotiations before taking any decision. One way of avoiding as much as possible this kind of negotiations is to initiate communication links on the least conflicted bands of frequencies.

However, the second constraint of this technique can make it difficult. As a matter of fact, by forcing protocols to use fewer channels of frequencies than they could, the chances to overload the selected channels are higher. Consequently, the reliability or speed of the communications can be affected. This can cause both user discomfort and the inoperability of the HAMAC device. Therefore,

this technique should be handled with care. More than only taking into account the frequencies in conflict and the ability of protocols to avoid them, the algorithms for allocating the frequencies should evaluate precisely the needs of each active communication link before taking decisions.

3.1.2.2.7.3 Conditions making this technique efficient

As a conclusion, radio collision avoidance is efficient when:

- Several active radio chips use the same bands of frequencies to communicate.
- The protocols related to these chips implement mechanisms to avoid using certain frequencies while in communication.
- The bands of frequencies in conflict are small enough to enable each protocol to communicate in a reliable way while avoiding them.
- The interferences risk creating a larger energy loss than the one induced by the negotiations necessary to change the set of frequencies used by the active communication links.

3.1.2.3 Energy saving strategies

The mechanisms presented in the previous section provide a large numbers of ways to save some energy. The optimization of the energy consumption of the radio chips should thus be conducted on several axes. This section presents these axes. It focuses on how these mechanisms can be combined to save as much energy as possible during the operation of HAMAC. Each of these mechanisms having its constraints, this part of the report defines how they should be used to improve the energy consumption of HAMAC without bringing their drawbacks to the device.

3.1.2.3.1 Presenting the strategies

Table 12 summarizes the various strategies that we will undertake concerning the radio chips. It shows several differences between these strategies, which we will explain in this section.

| Ref | Description | Obtrusiveness | Base | Mechanism(s) used |
|-----|---|---------------|-------------|---------------------------------------|
| 1 | Limit HAMAC's discoverability & connectivity | Unobtrusive | Negotiation | 3.1.2.2.5 |
| 2 | Limit the range of HAMAC to the size of a room | Unobtrusive | Static | 3.1.2.2.3 |
| 3 | Perform radio collision avoidance | Unobtrusive | Negotiation | 3.1.2.2.7 |
| 4 | Use low power modes for active connections when possible | Unobtrusive | Negotiation | 3.1.2.2.2; 3.1.2.2.6 |
| 5 | Use low power modes between discoveries when no connection is active | Unobtrusive | Calculation | 3.1.2.2.1 |
| 6 | Putting connections to standby modes according to their probability of use | Obtrusive | Calculation | 3.1.2.2.1; 3.1.2.2.2; 3.1.2.2.6 |
| 7 | Varying the period and type of discovery according to the probability of finding more devices | Obtrusive | Calculation | 3.1.2.2.4.1; 3.1.2.2.4.2 |

| | | | | |
|---|---|-----------|-------------|-----------|
| 8 | Limit the range of HAMAC further when communicating with nearby devices | Obtrusive | Negotiation | 3.1.2.2.3 |
|---|---|-----------|-------------|-----------|

Table 12: Summary of the strategies for making the radio chips consume less energy

3.1.2.3.1.1 Obtrusive and Unobtrusive strategies

Primarily, Table 12 distinguishes obtrusive and unobtrusive strategies. This point is important, as saving energy should not imply lowering the functionality of the target device. The best strategies for saving energy are unobtrusive strategies: their use does not impact at all the comfort of the user. However, the energy that can be saved through these strategies is limited; that's why we also define obtrusive strategies.

In obtrusive strategies, the main point is to balance usability versus low power consumption. To save more energy, we accept to lower the usability of the device in certain situations. Because usability is the first priority, calculations are made to impact the functionalities of the device only when the user has a very low probability of using them. For example, one can accept devices to be detected less frequently if the environment around him or her has not changed for a long time. This is a way of adding a slight discomfort to the user to release him or her from the huge discomfort that would be to see HAMAC device running out of energy.

Of course, obtrusive strategies are best used with configuration options settable by the user, which enable him or her to choose which functionalities to trade against energy. These options can make the user accept a greater loss of functionality than obtrusive algorithms can allow themselves to produce. Consider for example the situation when a user never uses Zigbee devices. In this case, even the most efficient obtrusive algorithms will consume energy to keep the support of the unused Zigbee protocol in HAMAC. This energy can be saved by making the user specify explicitly to the HAMAC framework that he or she never uses Zigbee.

3.1.2.3.1.2 Static, Negotiation or Calculation based strategies

Table 12 also defines a "base" for each strategy. This base indicates how each strategy compiles its input data to take the best decision at any time. In fact, we can abstract the strategies for energy saving as mathematical functions, which behavior depends on several parameters. The way these parameters are obtained is described by the "base" of the strategy.

"Static" strategies are based on static parameters. These strategies are based on a configuration statically written into the framework. For example, the only static strategy listed in Table 12, which aims to limit the range of HAMAC to the size of a room, needs the protocol plugins to respect a maximum value for the power of transmission of their chips. This maximum value should be calculated through researches and real life experiments. It will not change during the operation of the framework.

On the contrary, "Negotiation" based strategies will see their parameters change during the operation of HAMAC. Parameters based on negotiation change according to the state of the framework, and more specifically according to the needs of each service or protocol being used at a certain time. For example, the strategy presented in Table 12 to limit the discoverability and the

connectivity of the HAMAC device requires each service to specify whether it needs the HAMAC device to be discoverable or connectable to operate properly. This strategy will base its action on a summary of the needs of each service. In the same way, all the negotiation based strategies require their input parameters to be determined thanks to a negotiation between services or protocols, using mechanisms such as the flag system presented in section 3.1.1.3.3.3.

“Calculation” based strategies bring another dimension of complexity, as their input parameters have to be calculated thanks to external functions. These strategies are based on variable parameters that may change through time following a multitude of events. These parameters are calculated to reflect the effects of several changing probabilities and to respond to the various constraints imposed by the protocols. For example, the strategy presented in Table 12 to put the connections in standby mode according to their probability of use bases its action both on the probability that the user will use a given connection in the future and on the characteristics of the standby modes made available by each protocol. One key step in the process of elaborating calculation based strategies is to define methods for calculating these parameters balancing properly low energy consumption and usability.

3.1.2.3.2 Parameters impacting the strategies

All our strategies for saving energy are based on several parameters. We decided to summarize the most important of these parameters in Table 13. We indicate for each parameter which strategy of Table 12 it impacts, and also at which level it should be taken into account in our framework. Each parameter can be either static, either based on calculations depending on several factors. To address this second kind of parameters, we will present the factors impacting them and propose a way to calculate them further down in this section.

| Ref | Description | Level | Type | Strategies impacted |
|-----|---|----------|-------------------|---------------------|
| 1 | Need for HAMAC to be discoverable | Service | Static | 1 |
| 2 | Need for HAMAC to listen for incoming connections | Service | Static | 1 |
| 3 | Minimum power of transmission required to cover an entire room. | Protocol | Static | 2 |
| 4 | Radio frequencies used by each protocol | Protocol | Variable | 3 |
| 5 | Ability of a protocol to adjust the frequencies it uses | Protocol | Static | 3 |
| 6 | Time and energy consumption constraints of changing the set of frequencies used by a protocol | Protocol | Static | 3 |
| 7 | Minimum rate of transmission of the data packets during communication | Service | Static | 4 |
| 8 | Possibility to disconnect from the remote device without impacting its state | Service | Static | 4 |
| 9 | Time and energy consumption constraints of the low power modes of the radio chips | Protocol | Static | 4; 5; 6; 7 |
| 10 | Time estimated before the user needs to take back the control of an inactive connection | PMS | Probability based | 6 |
| 11 | Maximum time acceptable between an action from | PMS | Variable | 6 |

| | | | | | |
|----|---|----------|-------------------|------|--|
| | the user and the reactivation of a service | | | | |
| 12 | Time & energy consumption constraints of each discovery mode of each protocol | Protocol | Static | 7 | |
| 13 | Probability of finding more devices | PMS | Probability based | 7; 8 | |
| 14 | Maximum acceptable duration of a complete discovery process | PMS | Variable | 7; 8 | |
| 15 | Maximum acceptable period of discovery | PMS | Variable | 7; 8 | |

Table 13: Summary of the main parameters impacting our strategies for lowering the energy consumption of the radio chips

3.1.2.3.2.1 Defining variable parameters

Several of the parameters listed in Table 13 are described as “Variable”. They are parameters which can change through time, but are not based on any probability. This section describes them further.

3.1.2.3.2.1.1 Radio frequencies used by each protocol

The radio frequencies used by each protocol are variable because of the mechanisms implemented in several protocols to modify them during their operation. To apply strategy #3 of Table 12, protocols will negotiate the frequencies that they should use as described in section 3.1.2.3.3.2. This negotiation requires each protocol to store the frequencies that it is being using. We decided to store this information as a list of ranges, each range being defined by a start and an end frequency. The ranges of frequencies attached to each protocol will change during HAMAC’s operation to avoid as much as possible radio interferences.

3.1.2.3.2.1.2 Maximum time acceptable between an action of the user and the reactivation of a service

When using obtrusive strategies for saving energy, care should be taken to avoid impacting too much the quality of service offered to the user. Putting radio communications links to a standby mode has the undesirable effect of impacting the comfort of the user when he or she wants to use these communication links again. The radio chip will in fact need to re-activate the connection before being able to take back operations, which will introduce a latency of the HAMAC device. The variable “Maximum time acceptable between an action of the user and the reactivation of a service” puts an upper limit to this latency.

This upper limit should be low if the user is using HAMAC intensively, and may be increased if HAMAC was not used for a certain time. Therefore, we decided to make it vary according to the energy modes used by the HAMAC platform, described in section 3.1.1.3.3. This variable will have one arbitrary value per mode, the value being higher for the modes attributing to HAMAC the lower levels of availability. These values should be defined after real life experiments, and should not exceed a few seconds.

3.1.2.3.2.1.3 Maximum acceptable duration of a complete discovery process

The variable defining the maximum acceptable duration of a complete discovery process puts an upper limit to the time necessary for a radio chip to detect all devices in its environment when using several discovery schemes. This value should depend on the probability of finding a new device in the environment, but also on the user activity. In fact, our analysis in section 3.1.1.3.3 provides energy modes shutting down the screen of HAMAC in case of a low user interaction. In such energy modes, the device will not be able to show to the user that a new device was found. We can thus assume that the user expectations in terms of latency of discovery are lower.

Consequently, we decided to make this second upper limit vary according to two factors: the energy mode used by the HAMAC platform, and the probability of encountering a new device in the environment. A good calculation scheme could be to make this variable proportional to the probability of finding a new device in the environment – up to a given limit – when HAMAC is active, and to put a minimum to its value when the HAMAC platform switches to inactive modes.

3.1.2.3.2.1.4 Maximum acceptable period of discovery

The variable defining the maximum acceptable period of discovery should be calculated in the same way and in relation to the variable “Maximum acceptable duration of a complete discovery process”, as the two parameters are closely related.

3.1.2.3.2.2 Calculating the probabilities

Several of our obtrusive strategies are based on the estimation of future events. Therefore, Table 13 lists several parameters based on probabilities. This section explains the solution that we offer to calculate these probabilities. All the choices that it presents are arbitrary. However, our approach follows a structured methodology which could be reused to make more accurate calculation after real life experiments.

3.1.2.3.2.2.1 Methodology

Our method of calculation of probabilities is rather simple. We define each probability as a value between 0 and 100. We then define a list of conditions which, if true, will higher the probability of a given value. Both the list of conditions and the values associated to these conditions are selected arbitrarily, but can be refined with further studies. During the operation of HAMAC, the value of the probability is at any time the sum of the values associated to all the asserted conditions, up to the maximum value of 100.

3.1.2.3.2.2.2 Factors impacting the probabilities

Here is, for the two probability-based variables presented in Table 13, examples of conditions impacting them, with their associated value.

3.1.2.3.2.2.2.1 Probability that the user will not need to take back the control of an inactive connection soon

| Condition | Value |
|--|-------|
| The user is actively using another service. | 40 |
| The connection is inactive for more than 30 seconds. | 5 |
| The connection is inactive for more than 1 minute. | 10 |
| The connection is inactive for more than 5 minutes. | 20 |
| The connection is inactive for more than 10 minutes. | 30 |
| The connection is inactive for more than 20 minutes. | 50 |
| The HAMAC device is in sleep mode. | 70 |
| The rate of interaction with the service transported by the connection exceeds the average rate of interaction for this service. | 30 |

Table 14: List of factors impacting the probability that a user will not need to take back the control of an inactive connection soon

3.1.2.3.2.2.2.2 Probability of finding more devices

| Condition | Value |
|--|-------|
| The user has already used the target technology before. | 50 |
| The devices used at least once by the user – using the target technology – were not all discovered. | 20 |
| The user used a device using the target technology several times at the same period of the day. This device has not yet been detected. | 10 |
| At least one network using this technology is detected. | 10 |
| At least one device using this technology is detected. | 5 |
| A device was added or removed from the environment recently. | 30 |
| The quality of the currently open communication links – using any technology – is varying. | 30 |

Table 15: List of factors impacting the probability of finding more devices

3.1.2.3.3 Negotiation mechanisms

Among the strategies listed in Table 12, several are based on negotiation between services or protocols. Section 3.1.2.2 describes which components should negotiate between each other, and on what their negotiation should be based. This section explains the algorithms that will enable such negotiations to take place.

3.1.2.3.3.1 Flag based mechanisms

A first method to make several components negotiate between each other is to use a system similar to the flag system described in section 3.1.1.3.3.3. This kind of mechanism is useful when

several components need to justify the use of a given functionality according to their needs. It enables a given functionality to be enabled only if at least one component needs it.

The strategy #1 from Table 12 needs to use this negotiation algorithm to define when the HAMAC device should be discoverable, and when it should listen to incoming connections. Strategy #6 of this same table can also use this algorithm to define whether the operation of the active services can be endangered in case of disconnection of their communication link. Finally, strategy #8 needs this system to adapt the power of transmission of the radio chips according to the quality of each communication link.

3.1.2.3.3.2 Radio usage negotiation

The radio collision avoidance mentioned in strategy #3 of Table 12 cannot be performed by using the flag system. There are in fact too many frequencies to allow protocol plugins to define a flag for each of them. However, the idea is the same.

As explained in section 3.1.2.3.2.1.1, we decided to represent the frequencies used by each protocol as a list of ranges, defined by a lower frequency and a higher frequency. Instead of increasing the value of a flag integer, each protocol plugin will have to register the ranges of frequencies it uses regularly in a `RadioCollisionAvoidanceManager`. This manager will then split the range to register into several channels of variable width in a way enabling it to state for each channel how many protocols are using it. In the same way as with the flag system, the manager will thus be able of detecting quickly the overlapping ranges of frequencies.

Once possible to detect the overlapping ranges, several mechanisms will enable the protocol plugins to negotiate their use of the Hertzian medium.

First, each protocol will, when registering a band of frequencies, indicate whether it can avoid using it or not, and at which – energetic – cost. This will enable the `RadioCollisionAvoidanceManager` to know which protocols can release which channel, and to optimize the cost of the operation.

Secondly, a protocol able of using several bands of frequencies will have to ask to the manager which band of frequency from its list is free before registering them. This will enable each protocol to use the free channels in priority.

In case of conflict, if a protocol absolutely has to register at least one band of frequency already used by another protocol, the following will happen:

- The protocol plugin will give to the manager the bands of frequencies which it can use, and the minimum number of bands that it needs to register.
- If there is not enough free channels to satisfy the need of the protocol plugin, the manager will begin searching for channels which can be freed by other protocols.

- If it finds a protocol able to switch to an unused channel, it will ask it to do so, and register the freed channel to the requesting protocol. If several protocols are able to perform this switch, the manager will first ask to the protocol for which the operation is the least energetically expensive.
- If no protocol able to perform a switch is found, the manager will register the less occupied channels to the requesting protocol.

Finally, protocols which do not need a band of frequency anymore will be able to unregister it from the manager. The manager will in turn be able of minimizing the number of conflicted channels by reassigning protocols to the newly freed channel.

An additional feature can be added to the `RadioCollisionAvoidanceManager` to make it avoid as much as possible the registration of a given band of frequency. This is useful to ensure a better compatibility of the HAMAC device with external uncontrolled wireless devices, such as the intra-oral interface from TKS.

3.1.2.3.3 Communication requirements negotiation

Another mechanism of negotiation is required for the operation of strategy #4 of Table 12. This strategy requires in fact that services using the same protocol and controlling the same device specify their needs in terms of connectivity to the `ConnectionManager` of their protocol plugin. Knowing these needs will enable the `ConnectionManager` to optimize the energy consumption of communication links by applying sleep modes to them without obstructing their activity. This coordination process is thus another case of inter-service negotiation.

Applying a sleep mode on an active communication enables the radio chip to avoid transmitting and receiving data when the service does not need it. It is a way to keep the communication link active when the service does not use it often enough to do so during normal operation. Services have three different types of requirements concerning the radio links:

- Some services require data to be sent periodically;
- Others require data to be sent at any time with a limited latency;
- Some services require a specific amount of data to be transmitted or received each time they use the radio link.

There are thus three needs that the services need to communicate to the `ConnectionManager` to support strategy #4: the amount of data to send each time that the radio link is used, the maximum latency authorized by the service to send some information, and the period following which the information will have to be transmitted – a period of 0 defining a non-periodic operation.

To use the optimal sleep mode for the communication link, the `ConnectionManager` will have to compile this data together for a negotiation to take place. The sleep mode chosen has basically to answer to the following requirements:

- It must not make any service miss their latency requirement: the latency introduced by the sleep mode, L_S , should not be higher than the minimum latency limit defined by the services. The latency introduced by the transmission of data from other services should also be taken into account: if two services have latencies requirements L_1 and L_2 such that $L_1 < L_2$, the service having the latency requirement L_1 should transmit its data first. Also, assuming that this service takes T_1 time to transmit its data, L_S should comply to: $L_2 < L_S + T_1$.
- It must let the services meet their periodical deadline.
- It must enable the services to achieve their normal throughput even when transmitting one after the other.

3.1.2.3.4 Calculation mechanisms

Some strategies presented in Table 12 are based neither on single static parameters nor on negotiation. They are based on a multitude of parameters, among which some might result of probability calculations (described in section 3.1.2.3.2.2). The main algorithms that we use to operate these calculation based strategies are described in this section.

3.1.2.3.4.1 Choosing the best state for a chip or a connection

The most common type of calculation used in our strategies has for goal to select the best state for a chip or a communication link to operate in. This state has to be selected among a set of several one, each state having its constraints and its advantages. Two steps have to be taken to perform this kind of calculation: first the set of possible states available to the chip or to the communication link must be defined; then the most energy efficient state should be chosen.

To perform the first of these steps, we first need to understand and list the constraints of each state. These constraints are generic and are common to the states of a connection and to the states of a chip. Table 16 lists them, along with their description.

| Constraint | Description |
|-------------------|---|
| GoInTime | Time necessary for the chip or the connection to enter the state. |
| GoOutTime | Maximum latency required for the chip or the connection to leave the state. |
| GoInConsumption | Energy consumed in the process of entering the state. |
| GoOutConsumption | Energy consumed in the process of leaving the state. |
| Consumption(time) | Energy consumed while within the state. Depends on the time the state stays active. |

Table 16: Main constraints of the states, applicable to a communication link or to a chip

To select a state within the set of possibilities offered to the chip or to the communication link (the target); the algorithms should check that its constraints are compatible with the constraints applied to this chip or communication link.

The main constraint is the maximum allowed latency L_{max} before the chip or communication link can go back to its active state. To be efficient, the algorithm must select states verifying:

$$\text{GoOutTime} < L_{max} \cdot (4)$$

The algorithm must also check that the target chip or communication link will have the time at least to enter and leave the state before it needs to get active again. If we define T_{sleep} the estimated time before the target needs to get active again, we must thus have:

$$\text{GoInTime} + \text{GoOutTime} < T_{sleep} \cdot (5)$$

Following these simple constraints will enable the algorithms to select the possible states available to the target. The second step of the process is to select the best state, which is the one consuming the less energy. The energy E consumed by switching the target to a given state can be calculated as:

$$E = \text{GoInConsumption} + \text{GoOutConsumption} + \text{Consumption}(T_{active}) \cdot (6)$$

with:

$$T_{active} = T_{sleep} - \text{GoInTime} - \text{GoOutTime} \cdot (7)$$

For this simple algorithm to work, each possible state and its constraints should be statically specified into the protocol plugins.

3.1.2.3.4.2 Selecting the best discovery scheme

These calculations to define the best state for a chip have to be combined to another algorithm to enable the optimization of the discovery scheme mentioned in strategy #7 of Table 12.

Defining a discovery scheme means changing two important things: the period between discoveries, as described in section 3.1.2.3.2.1.4, and the type of discovery used – depending on the possibilities offered by the protocols. The only constraint to respect is the maximum time required to perform a complete discovery, described in section 3.1.2.3.2.1.3.

To find an optimal period between discoveries, the calculation algorithm must find the most energy efficient way to discover devices while respecting this constraint. This means changing the period of discoveries, which modifies the T_{sleep} value described in the previous section. This algorithm thus has to find the best combination between the period between discoveries, the discovery type to use, and the state of the chip whenever no discovery has to be performed.

This combination can be found by specifying, for each discovery type, the maximum period enabling the discovery process to meet its deadline. By lowering this period step by step, the algorithm will then be able to find several possible combinations including several chip states between discoveries. Once all chip states are contained in the list of possibilities, we can assume that this list contains all the best options. In fact, the energy consumed to enter and leave each state makes the sleep modes more efficient when they are used on a larger period of time.

The best option in this list can therefore be assumed to be the best one available. If several options offering the same energy consumption are available, the one enabling a faster discovery should be preferred.

The different types of discovery made available by the protocols can be differentiated following several variables: the number of iterations required to perform a complete discovery, the length of one iteration, and the efficiency of one iteration.

As a future work, we could also include another requirement in the choice of the discovery scheme to use: the estimated time required to find at least 70% – or another percentage – of the devices in the environment. This requirement would add another quality limit to the discovery scheme to select.

3.1.2.4 Theoretical efficiency evaluation

To evaluate the efficiency of the strategies presented in this section, we will compare a use case using some of them with another use case which does not, in a theoretical evaluation.

3.1.2.4.1 Evaluation context

This evaluation will analyse the energy consumption of a Bluetooth chip during a typical day. In the first case, no energy saving strategies will be applied to the Bluetooth chip. In the second case, the strategies #1, #4 and #7 of Table 12 will be used to lower the energy consumption of the chip. The day schedule will be taken from our previous use case, presented in section 3.1.1.5.2.

3.1.2.4.2 Calculating the consumption of the chip in several states

To calculate the energy consumption of a Bluetooth chip over one day, we need to estimate its power consumption when in several states. Our analysis is based on the data contained in [37].

3.1.2.4.2.1 Energy consumption data

The relevant current consumption data presented in the datasheet of the BlueCore4 chip [37] is presented in Table 17 :

| Operation Mode | Connection Type | Average |
|--------------------------------------|-----------------|------------|
| Inquiry & page scan | | 0.77 mA |
| ACL data transfer with file transfer | Slave | 17 mA |
| SCO connection HV1 | Master | 34 mA |
| ACL data transfer 40ms sniff | Slave | 1.5 mA |
| Reset (RESETB low) | | 40 μ A |

Table 17: Energy consumption data for the CSR BlueCore4 chip – Supply 1.8V

3.1.2.4.2.2 Normal operation

We can directly extract the data presented in Table 17 to deduce the power consumption of the Bluetooth chip when in several states, during normal operation.

During normal operation, the control of a computer requires to use an ACL link continuously, as a Slave device – the target computer being the master. When controlling a computer, the chip is thus consuming the same power as when performing an “ACL data transfer with file transfer”.

During the day, the fact that HAMAC be constantly discoverable and constantly listen to incoming connections makes the Bluetooth chip consume energy all day long for the operations of “Inquiry & page scan”. This also prevents the chip from being put in idle mode.

A data not included in [37] is the energy consumption of the Bluetooth device during a device discovery process. However, the consumption of the radio chip when communicating constantly is indicated: it is equal to its consumption when performing a “SCO connection HV1”. We also know through [38] that a Bluetooth device performing a discovery communicates about 31.2% of the time, during 10.24 seconds. If we fix the period of discovery in normal mode to 1 minute, we can thus roughly estimate the average current consumption of a periodic discovery scan as being:

$$A_{inq_1} = 31.2\% \times 34 \text{ mA} \times \left(10,24 \text{ s} / 60 \text{ s}\right) \approx 1.81 \text{ mA.} \quad (8)$$

Table 18 summarizes the results of our estimations during normal operation.

| State | Power consumption |
|--|-------------------|
| Controlling a computer | 30.60 mW |
| Periodic discovery | 3.26 mW |
| Listening for incoming connections & Discoverability | 1.39 mW |
| Idle state | 0.00 mW |

Table 18 : Estimated power consumption of a Bluetooth chip during normal operation

3.1.2.4.2.3 Using the strategies

When using strategy #4 of Table 12, a Bluetooth chip controlling a computer can use the sleep mode “sniff” to reduce the number of packets it must exchange with the remote computer. We can allow HAMAC to communicate with the computer only every 40ms. In fact, this would already provide it with 25 values par second to actualize the position of the computer mouse: it is enough to ensure a good user experience – 25 images per second being a rate known to be comfortable in video. The consumption of the chip while controlling a computer corresponds therefore, when using strategy #4, to the consumption when performing an “ACL data transfer 40ms sniff”.

Moreover, strategy #1 helps saving the energy consumed in normal mode for making the HAMAC device constantly discoverable and listening to incoming connections. This energy is never consumed, as these functionalities are not needed: the user is supposed to have a very low probability of using a computer for the first time within a regular day. As a consequence of this, the HAMAC device will be able to enter idle mode when neither discovering other devices nor communicating. The consumption in this mode is equal to the consumption when the chip is “Reset”. To simplify the calculation, we will assume that the chip consumes the “Idle mode” power even when performing discoveries.

To finish, strategy #7 enables HAMAC to lower its rate of discoveries when the environment around the user does not change. We can honestly assume that the environment of the user is static during at least 80% of his or her day. We can also define that the period of discovery is increased to 10 minutes when the environment does not change. From this data, we can roughly estimate the current consumption of the Bluetooth chip caused by the periodic discovery scans as being:

$$A_{in_{q_2}} = 31.2\% \times 34 \text{ mA} \times \left[20\% \times \left(\frac{10,24 \text{ s}}{60 \text{ s}} \right) + 80\% \times \left(\frac{10,24 \text{ s}}{600 \text{ s}} \right) \right] \approx 0.51 \text{ mA} \quad (9)$$

Table 19 summarizes the results of our estimations when strategies #1, #4 and #7 of Table 12 are used.

| State | Power consumption |
|--|-------------------|
| Controlling a computer | 2.70 mW |
| Periodic discovery | 0.92 mW |
| Listening for incoming connections & Discoverability | 0.00 mW |
| Idle state | 0.07 mW |

Table 19: Estimated power consumption of the Bluetooth chip while using some of our strategies

3.1.2.4.3 Defining the duration of each state

Now that we have the power consumption of each state, we will calculate the duration the device will have to stay in each of these states during the typical day already presented in section 3.1.1.5.2. Figure 20 and Figure 21 split the day of an example user, Mr. Nielsen, into several periods of time. Let's associate a state for the Bluetooth chip during each period:

- During the "Sleep time", we will assume that the Bluetooth chip will be shutdown.
- During the "TV watching" period, the periods dedicated to meals and the first "Miscellaneous" period of the day, we can assume that the user will not be using any Bluetooth enabled device: the chip will stay idle.
- During the "Work time period", we the user will probably use a computer during about 7 hours. The Bluetooth chip will remain idle in the hour left.
- During the second "Miscellaneous" period of the day, the chances that the user uses a computer at home are high. We can therefore count that the chip controls a computer during this time.

Table 20 takes these considerations into account to summarize the time the Bluetooth chip will spend in each state, during the day.

| State | Time spent in the state |
|--|-------------------------|
| Controlling a computer | 8h |
| Periodic discovery | 15h30 |
| Listening for incoming connections & Discoverability | 15h30 |
| Idle state | 7h30 |

Table 20: Estimated time spent by a Bluetooth chip in each state, during a day

3.1.2.4.4 Comparaison

During the time it is not shutdown, the average power consumption of the Bluetooth chip in normal mode and when using the strategies can be defined respectively as P_{normal} and $P_{strategies}$ such that:

$$P_{normal} = \frac{(30.6 \text{ mW} \times 8h) + (3.26 \text{ mW} \times 15.5h) + (1.39 \text{ mW} \times 15.5h)}{15.5 h} \approx 20.44 \text{ mW} \quad (10)$$

$$P_{strategies} = \frac{(2.70 \text{ mW} \times 8h) + (0.92 \text{ mW} \times 15.5h) + (0.07 \text{ mW} \times 7.5h)}{15.5 h} \approx 2.35 \text{ mW} \quad (11)$$

By using our strategies, the energy consumption of a Bluetooth chip can therefore be lowered to about only $2.35\text{mW}/20.44\text{mW} \approx 12\%$ of the energy consumption when the chip is in normal mode. Moreover, this improvement of a factor of nearly 10 is made when using only 3 of our 8 strategies for saving energy. Using our other strategies would certainly increase this performance.

3.2 MAKING HAMAC MORE SECURE

3.2.1 DEFINING THE SECURITY NEEDS OF HAMAC

Security is an important word that has several meanings. If a web engineer is asked about security, he will answer about data integrity, encrypted communication, password mechanisms, etc. But in the opposite, if a health engineer is asked about the same word: security, he will answer about mechanisms to avoid putting the user in danger. In this case, the word safety is used. Both meanings are taken in account in HAMAC.

The first meaning is important to keep privacy and data integrity. It is also in the heart of actuality with a user condemned in Germany for not having protected his WiFi network [39], the thievery of Skyblog's passwords [40] and all privacy debates around social networks. The meaning is also important to keep users away from danger which may be induced by the use of HAMAC.

3.2.1.1 Reason to be implemented

3.2.1.1.1 Protect the user physically

A use case is more suitable than an introduction. Mr. Nielsen is in his wheelchair and is going to cross the street. He is controlling the chair by the tongue unit via HAMAC. The pedestrian light is red and a car is coming. But suddenly and without any explication, HAMAC is freezing and the wheelchair is blocked in a forward mode. The user is thus crossing the street and an accident occurs. This case can happen if no security scheme is designed and implemented.

Other use cases and examples can be related. Mr. Nielsen lives in an automation environment. He has a device placed on a tap to open it by Zigbee. He can also control his cooker. A communication problem or a HAMAC shutdown could forbid Mr. Nielsen to turn off his tap of his cooker on which there is of course a kettle full of boiling water. Another protection is needed in domestic environment. The conclusion is that it is necessary to protect the user physically in all environments that he is likely to evolve.

3.2.1.1.2 Protect the user virtually

The virtual protection concerns data communication. Nowadays, if a radio transceiver is put close to an open WiFi network; all data packet can be fetched. Then all emails can be read, and all the network life of people can be watched. It is a big intrusion in the privacy of people. Wireless connections need to be encrypted in order to ensure privacy. It is true for Zigbee and Bluetooth. There is no reason to forget the privacy of the disable people.

In a same way, a special care is made in inter-devices connection. A particularity of radio waves is that they are going in every direction. Without any protection, a neighbor can connect a Zigbee remote control to the user network and start using Zigbee devices as lights, heater, or managing intruder alarm system. Those connections have to be forbidden.

3.2.1.2 What needs to be secured

The security problem has been divided in different kind of protection. It has been seen that security has several meanings. A response exists for each meaning and level of danger.

3.2.1.2.1 Data

As presented in section 3.2.1.1.2, data transfers need to be secured. Data security concerns the way that data is encrypted before being transmitted. It is ensured in HAMAC that all its communication are secured and encrypted to forbid an external user to monitor all communications and spy the user activity. For this, the features brought by Bluetooth and Zigbee are presented.

3.2.1.2.2 Connection

As depicted in section 3.2.1.1.2, the device connection should be controlled carefully. It is not acceptable that an external person can use the user's network in being able to connect any devices. It is thus very dangerous for the user. Solutions taken in account in HAMAC are presented.

3.2.1.2.3 Time critical control

Is called time critical control all devices control that needs a timely response. It is typically the wheelchair control which is a hard real-time system. A real-time system is defined as *"any information processing activity or system which has to respond to externally generated input stimuli within a finite and specified delay"* [41]. A hard real-time system is one which is absolutely imperative that the response arrive in the specified delay. A wheelchair is responding to user's input stimuli in order to move, and if the user wants to stop the wheelchair, this one must respond in the specified delay.

In taking the example of section 3.2.1.1.1, it is absolutely imperative that the user can stop the wheelchair in the millisecond. It is also imperative that the control be failsafe. In addition, it is imperative that an exception procedure be implemented in case of system crash. Section 3.2.3 develops these features. The focus is put on wheelchair for the time critical control.

3.2.1.2.4 Standard device control

Standard device control consists in securing the control of device that can be dangerous but in a lower level than the case seen in section 3.2.1.2.3. This is illustrated by the second example of section 3.2.1.1.1. These systems are defined as soft real-time system. A soft real-time system is a system which still functions correctly if a deadline is occasionally missed [41]. It can also be defined as a real-time "like" system in the way that a deadline miss is not critical but still important. A security scheme is also needed in case of failure to keep the user integrity. This is described in section 3.2.3.

3.2.2 DATA SECURITY

3.2.2.1 Bluetooth

This section presents the security mechanisms made available by the Bluetooth specification to protect the communications using Bluetooth. It also identifies the main security threats for the

user of HAMAC and summarizes what changes should be applied to the HAMAC framework to make it compatible with Bluetooth security. Its content relies on the description of Bluetooth security found in [38] and may be better understood after reading the short description of the Bluetooth protocol found in our previous semester report [6].

3.2.2.1.1 Presentation

To ensure its security, Bluetooth relies on two main mechanisms: authentication and encryption.

The mechanism of authentication requires that, for sharing sensitive tasks particularly, two devices trust each other. This trust is acquired by asking the users of the two devices to enter the same password on the devices during this process. This password serves as a shared secret. Its use enables the users to check that the device to which they are connecting is really the device to which they want to connect. From this password, the two devices performing the authentication process will exchange some keys and save information about each other. Once the two devices “trust” each other, they are able to re-authenticate each other automatically and without intervention of the user each time that they reconnect.

This authentication process serves also as a base for an encryption mechanism. Based on the secret keys that they share, the two authenticated devices are able to start encrypting their communications upon need, without exchanging more information about the encryption keys. Thanks to these mechanisms, Bluetooth communication is made very difficult to eavesdrop, as some components for building the encryption key necessary to decrypt the communication are exchanged only once during the authentication process. Another component, the shared password entered by the users, is not even transmitted over the air.

The Bluetooth devices have several levels of security. Therefore, some devices require their communication links to be constantly encrypted, while others do not.

3.2.2.1.2 User interactions required

The disadvantage of the Bluetooth authentication system is that it requires the user to interact with the devices at the two sides of the communication link, to enter a password. The users of HAMAC will thus require an external person to help them to establish this authentication when using a remote device for the first time. They will also need either to be able to enter a password on the HAMAC device either to be shown this password on the screen of HAMAC to complete the authentication process – to simplify the process, we chose the second solution.

Another kind of interaction can be required from the user in the case when a remote device asks to begin an authentication process with HAMAC. This situation can happen if the user wants to use services which require the remote device to initiate the connection, such as the Bluetooth HID service. In such a case, the user will certainly ask an external person to initiate the connection to HAMAC from a remote device. On the side of HAMAC, the framework will then have to decide or not to accept the incoming connection, and eventually to ask to the user its permission. This can be problematic as the intuitivism of the device will be endangered if the user is asked such a question

while doing something else. It can also represent a risk for comfort or security, as it may interrupt the user from his or her activity. To keep the HAMAC device intuitive, we decided that incoming authentications will only be accepted if the user explicitly asked to use a service from their source devices. To begin using a service requiring an incoming connection, the user will thus have to click on a button making HAMAC accept automatically any authentication request from the target device. The framework will then show instructions guiding the user during the connection process.

3.2.2.1.3 Threats in the context of HAMAC

In the context of HAMAC, the main threats for the user when using Bluetooth are the communication of sensible information to a wrong device and the eavesdropping of this information by a bad-intentioned person.

The information sent over Bluetooth to control a TV or a computer mouse is not very sensible. To maximize the comfort of the user, it should therefore not be protected if this protection requires passing through a new authentication process – except if the user specifically asks for it.

However, other Bluetooth transported information can be sensitive. This is the case of the information for remote controlling a keyboard which may contain passwords or other secret data. In the same way, the IP traffic over Bluetooth can also be considered as sensitive.

To secure the use of HAMAC, we thus recommend that services requiring the transport of sensitive information be identified. We also recommend that this identification be used to force the encryption of the communication links when the remote devices do not require systematic encryption. By opposition, we encourage that non-sensitive information be not encrypted when remote devices can be controlled without authentication: this will preserve the user's comfort of use.

3.2.2.1.4 Integration to the framework design

3.2.2.1.4.1 Bluetooth devices

The modifications of the framework to integrate Bluetooth security are many. First, the framework should be able to know whether a Bluetooth device has already been authenticated, and whether or not it requires or support encryption for its communication. These variables can be easily added to the class `BluetoothDevice` of our framework. According to their values, the framework will know whether or not to encrypt communication links, but also when to perform authentication processes.

The security strategy concerning Bluetooth is simple:

- If a device does not require authentication to be controlled and if it can be controlled without transmitting sensitive data, then authentication should be avoided.
- If a device requires authentication to be controlled, then authentication should be performed on the first time it is used.
- If a device has already been authenticated and supports encryption, then all communications with it should be encrypted.

- If a device does not support encryption but requires the transmission of sensitive information to be controlled, then the user should be warned that its communication is not encrypted.

3.2.2.1.4.2 Bluetooth services

On the service side, this strategy requires the HAMAC framework to know which service can possibly transmit sensitive information. A constant variable should thus be added in the `BluetoothService` class to indicate for each service if it requires the communication links to be encrypted.

3.2.2.1.4.3 Configuration options

Data security requires balancing data protection and user comfort. However, because the default balance chosen in HAMAC might not be adapted to every case, the user should be able to ask for more security and less comfort. Configuration options concerning the Bluetooth security should thus offer several levels of security:

- **Low:** Avoid authentication processes as much as possible, even when using services requiring transmitting sensitive data.
- **Standard** (default): Avoid authentication processes as long as it does not require sensitive data to be transmitted without encryption.
- **High:** Encrypt all data, even non-sensitive information, and hence require all devices to be authenticated.

3.2.2.1.4.4 Interface

To enable the user to perform authentication processes, the framework should also be able to display several types of wizards and or dialog boxes:

- An outgoing authentication wizard explaining to the user how to authenticate his or her device and giving him a random password to enter on the remote device.
- An incoming connection/authentication wizard explaining to the user that he needs to ask for external help to initiate a connection to HAMAC from a remote device.
- A dialog box asking the user if he or she wants to pursue the establishment of an unsecure connection when sensitive data may be transmitted on it.

3.2.2.2 Zigbee

3.2.2.2.1 Presentation

The presentation of security in Zigbee networks come from [42; 43; 2].

Zigbee security inherits from IEEE 802.15.4 specifications. It adds the key management and the device control to the frame protection.

3.2.2.2.1.1 Frame protection

The frame protection is a feature brought by IEEE 802.15.4. It consists in encrypting the frame with the AES (Advanced Encryption Standard) algorithm with a 128bits key. This encryption is also using to validate data integrity thanks to a Message Integrity Code (MIC). This code is generated with the same key using for encryption. It allows a node to verify the origin of the frame and discard the message if it comes from an untrusted node. IEEE 802.15.4 defines several encryption policies. Data can be send encrypted or clearly with or without integrity verification.

3.2.2.2.1.2 Key Management

Zigbee protocol brings security to the network and application layer. It also defines how keys used for data encryption are shared in the network. There are three kinds of keys:

- The master key is used to share the link key.
- The Link key is used to encrypt data between two nodes. It is common to two nodes.
- The network key is shared in by the nodes of a same network and is used to encrypt data.

3.2.2.2.1.3 Trust center

The trust center is a special node who rules the security policy. It is typically the network coordinator which takes this role but it can also be another node in the network. The trust center authenticates devices who want to join the network. It can whether allow or forbid the connection. It maintains and distributes the keys. And finally, configure the security policy of the network.

3.2.2.2.1.4 Security modes

Zigbee protocol has two security modes: standard and high. High security mode is only provided by the PRO version of the Zigbee stack whereas the standard mode is compatible with the 2006 version. The High security mode enforces the encryption of the key when they are sharing. Furthermore, it brings the ability of use a unique key (Link key) between two devices and the maintenance by each devices of authentication and permissions tables. This permits of each devices to know if its neighbor is allowed to communicate with itself or not.

3.2.2.2.2 Integration to the framework design

The integration of security features brought by the Zigbee protocol should have the less interaction with the user as possible. This does not mean that the framework will not be configurable. But HAMAC is designed for people that already use a lot of movements to do simple things. It is not the goal of this project to force the user to use even more movements in order to

control objects. A special care is made in the framework integration to make HAMAC the less annoying as possible.

3.2.2.2.1 Network join

The connection of HAMAC to an existing network is needed to be set up once. The connection procedure depends on the facilities provided by the network manufacturer. HAMAC makes the procedure as easy as possible but it cannot induce on a different implementation. Once the connection is established, the trust center of the network will allow the future HAMAC connection. Then HAMAC can negotiate to take control upon the network. If HAMAC does not have any default network then it will attempt to connect to any networks that it can see. The number of attempt decreases after a moment. It is always possible to attempt a manual connection through the configuration panel.

3.2.2.2.2 Network protection

The network protection does not depend on HAMAC. The protection is provided by the network manufacturer. If a device wants to join HAMAC network, then it will allow it and display it in the list of devices. But it will not give it the network management. Thus, if a user wants to connect himself to HAMAC, he will succeed, but he will never be allowed to control the network. When HAMAC is shutdown, the Zigbee network needs another coordinator, to be the trust center controlling network accesses. This is the role of manufacturers to think about these.

3.2.2.2.3 Device connection control

Device connection consists in binding HAMAC to other devices. HAMAC binding policy consists in attempt a connection to all networks. Once accepted, HAMAC fetches the list of devices and displays it. During the discovery of devices, HAMAC uses the file manager described in section 4.1.2 to store information about them. It thus knows if it is the first time or not that it is connecting to the device.

3.2.2.2.4 Communication policy

All communications by Zigbee are encrypted. HAMAC uses the features presented in section 3.2.2.2.1.1 and present in Z-stack. Z-stack is the Zigbee stack provided by Texas Instrument and used by HAMAC. More information is presented in the report of the first semester [6]. The communication policy assumes that it is not acceptable to have a clear communication. Nevertheless, if a device does not support encrypted communication, it is assumed that it is more important to give the opportunity to the user to control this device. In this case, a notification indicating that the communication is not secured is displayed.

3.2.3 SAFETY

It has been said in section 3.2.1.2.3 and 3.2.1.2.4 that safety is brought by a real-time system. The problem is that HAMAC is not one of those. Then, the question is: “how to achieve real-time in a system that is not real-time?”

Safety has to be achieved in two levels. The first one concerns the use of the wheelchair or other devices which have the same level of danger. The second level concerns other devices like those evoked in section 3.2.1.2.4. First the design of these systems is presented then an implementation scheme is developed.

3.2.3.1 How safety is integrated in HAMAC design

3.2.3.1.1 Wheelchair module

3.2.3.1.1.1 Presentation

The aim of HAMAC is to be compatible with the most devices, as explained in section 2.1. Constructors of wheelchair do not have a standard way to control it. A module is thus needed. The idea is to have a type of module for each kind of wheelchair, and be compatible with all these modules. A module is composed of a hardware board, and a software plugin. The role of the plugin is first to bring the abilities to communicate with the wheelchair, and secondly to bring the security as both the real-time and fail-safe capability. It is more specific than HAMAC’s typical plugins.

The hardware board has two interfaces: one specific to the wheelchair, and one standard to communicate with HAMAC. Figure 22 is a schema of this module.

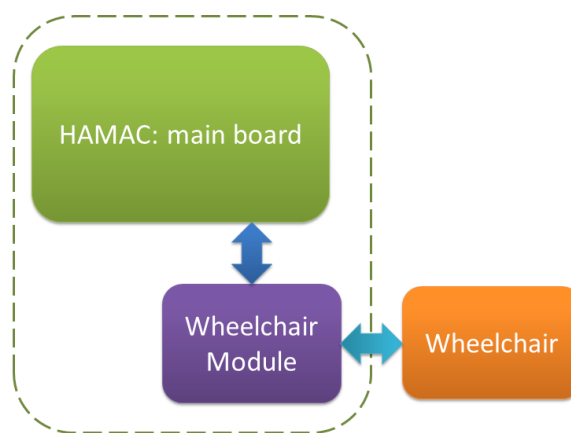


Figure 22: Hamac architecture with a wheelchair module

As shown in Figure 22, the wheelchair module is separated from the main HAMAC system. It is commutable with another module in the case that the user changes his wheelchair. The plugin supporting this hardware module is part of the HAMAC software running on the main board. A communication standard is defined to communicate with the hardware module which implements the way to communicate to the wheelchair.

When the wheelchair hardware module is present, the plugin associated is activated. It adds a service in HAMAC's GUI. The control of the wheelchair will be activated in the case that the user will choose to use the service. When activated the service starts an initialization procedure to activate mechanisms ensuring security. Several methods follow to show how to integrate security in HAMAC design in the case of wheelchair module

3.2.3.1.1.2 Tongue system input

In this section are defined procedures specific to the tongue system input.

3.2.3.1.1.2.1 Input redirection

To be controlled, HAMAC fetches user's inputs and responds to it. The problem in using the wheelchair is that if HAMAC crashes or takes too much time to perform a task, the tongue inputs are not given to the wheelchair module on time. The idea is to bypass HAMAC to give to the wheelchair module the user's inputs. This is shown in Figure 23.

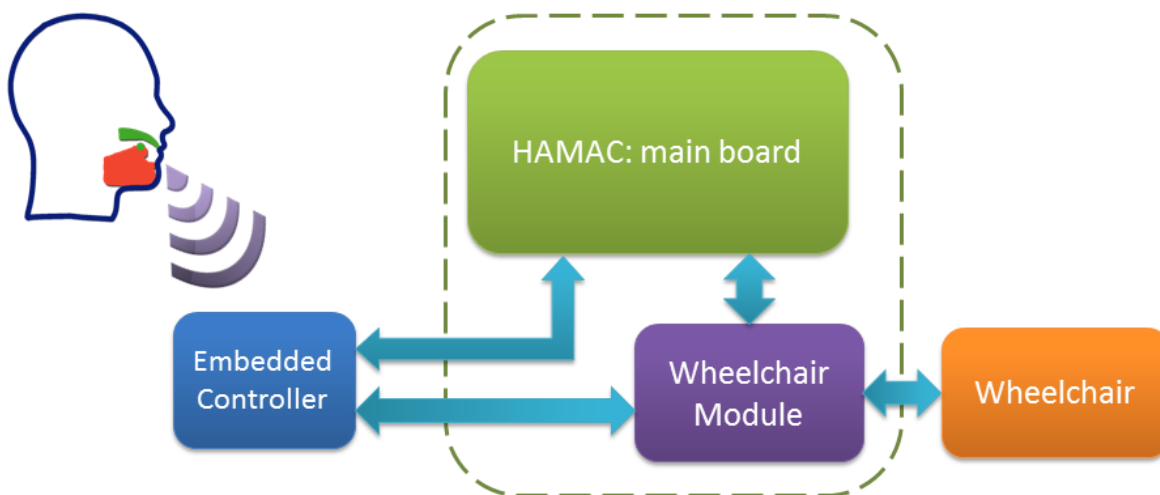


Figure 23: Inputs from the tongue system are given both to HAMAC's main board and to the wheelchair module

As seen in Figure 23, the embedded controller sends information both to the HAMAC's main board and to the wheelchair module. The role of HAMAC in this disposition is to tell who take inputs from whom.

The procedure in the wheelchair control is presented in Figure 24:

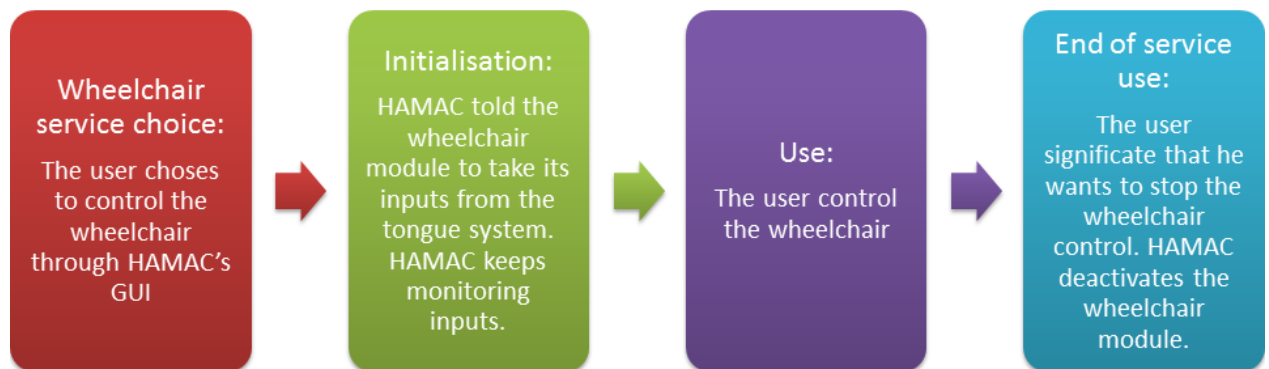


Figure 24: Procedure in the wheelchair control

As presented in Figure 24, HAMAC is still monitoring inputs, even if the wheelchair module is using them directly. In this case, the user can notify HAMAC to its wish to stop using the wheelchair, and thus HAMAC can disable the wheelchair module. The fact that HAMAC is not responding anymore does not affect the control by the tongue unit.

3.2.3.1.1.2.2 Initialization procedure

As explained in section 3.2.3.1.1.2.1, during the initialization, HAMAC's wheelchair plugin system activates the input redirection. It is not the only task performed. This plugin, also inform the PMS that the screen deactivation is allowed. Furthermore, HAMAC will not enter in *conservative* or *sleep* mode because the inputs sent to it induced the timer reset. This is explained in section 3.1.1.3.3. The fact that the processor stays in *run* mode ensures that HAMAC has a low latency in its response. Indeed, HAMAC may not use the full computational power of the processor, but it cannot wait for it to change its mode (and because it is slower) to perform a critical task.

3.2.3.1.1.3 Screen input

The case of the screen input is trickier because user's commands go through HAMAC before going to the wheelchair module. It has to be ensured that the information can go through HAMAC on time and with failure handled. The wheelchair service adds real-time like abilities.

First, it ensures that the system will stay in *run* mode to have low latency. Then it avoids the execution of long algorithms by taking the maximum of resources. It does it in disabling the service discovery. This is the only part of HAMAC which can take resources. The GUI is entirely dedicated to transmit the input from the screen to the wheelchair service.

To ensure that HAMAC is responding, the hardware module is constantly asking if it is running. It is a part of the description of the communication system between the both. It is simply a "ping - pong" algorithm like in TCP/IP protocol. If the hardware module sees that HAMAC is not answering, then it goes in failure mode and stops the wheelchair. It can also start failure modes provided by the wheelchair manufacturer.

3.2.3.1.1.4 Real-time design

The design of the hardware module comprises the real-time capability. It is not the scope of this report to develop the hardware design and implementation, but independently from the content of this module, if it is considered as a black box, it is considered as a hard real-time system. Indeed, the commands received are delivered to the wheelchair on time.

3.2.3.1.2 Standard devices

The design of security of standard devices refers to section 3.2.1.2.4. In the case evoked, the danger is provoked by the fact that because HAMAC is down, it cannot deactivate devices like taps or cookers. The work that can be done in HAMAC is a fast recovery. Then it can take control back and manage these devices. Otherwise it is the role of the manufacturer to manage problems that can happen.

It would consist of the implementation of a watchdog [44]. A watchdog is a timer which, if it is not reset, resets the microcontroller. In our case, if HAMAC is not resetting the watchdog of the device that it is using, then the device goes in fail-safe and stops running.

4 ADDITIONAL AXIS OF IMPROVEMENTS

4.1 NEW CORE FEATURES

4.1.1 EVENT MANAGER

To separate the GUI from the rest of the framework, a class bridge has been designed for QT's events. It is the *EventManager* class. *GUI* has to know when a device and a service are added to the framework in order to display a notification. *ConcurrentList* will call a method from the *EventManager* and this method will create and send an event to *GUI*. This class is thus GUI specific.



Figure 25: EventManager class

According to Figure 25, it is possible to send several kinds of events and also to lock *GUI*'s mutex. Indeed *GUI* is derived from *Mutexable*. This is made to prevent *GUI* modifications when the content of a *ConcurrentList* instance changes.

4.1.2 FILE MANAGER

The File Manager is responsible for managing information in files. It is related to the configuration system but is independent from it. The File Manager stores information from Devices, Services and Framework. There are several ways of storing data in files and HAMAC's needs are defined to choose the right system.

Serialization is the process of converting an object into a stream of bytes. Object's attributes are stored but also references of other classes, the name of the class and assembly containing the class. In this way, the class can be transferred in the network or put in a storage medium. The goal is to have a perfect clone of the class. The boost library [45], Sweet Persist [46] or s11n [47] are libraries which are implementing this way of serialization.

Another way of serialize a class consists in putting object's attribute in a file thanks to a markup language. The object is not converted in a stream of byte, but in a stream of characters and data is put between beacons. The file is thus human readable. JSON [48] and XML [49] are two markup languages which allow the storage of data.

The type of data stored in HAMAC system is mainly characters chains or strings. When storing, the object integrity is kept. Furthermore, Data is saved only if the user ask for it (it can be implicit). The *Device* and *Service* classes will not be restored from file but when a *Device* is created, it will look in the file to see if the user has stored some information it should fetch.

The cloning of data is not relevant in HAMAC since only some attributes are stored and fetched. Furthermore, references are not kept. Moreover, HAMAC is an embedded system and the library used should be as tiny as possible. For these reasons, the boost library, Sweet Persist and s11n are not retained for the solution.

For the same reason, XML and JSON have been retained. The choice between these two libraries depends on the size and complexity of them. Only basic features are used in HAMAC, and because it is on an embedded environment, the library with the smallest size is chosen.

We have chosen to use the JSON [48] type because it is more compact than XML [49] and we do not need all features provided by the XML language. JSON defines object and array structures. It also defines several type of value like a string or a number. In HAMAC, the object structure is used to describe a class and the array structure is used to store a collection of object.

The File Manager class is defined as shown in Figure 26.

```
<<global>>
FileManager
-files: File
+save(e1:Element): void
+load(e1:Element): void
```

Figure 26: FileManager Class

As presented in Figure 26, the class contains a reference to files where data is. This class also provides two methods that can be overloaded. In these methods you can give a reference to a *Device*, *Service* or *Framework* class. The methods “*save*” look for an existing data and replace information with the one provided. Otherwise it adds a new object in the file. The method “*load*” looks for an existing object. If one is found, it fetches information and writes them in the class. Otherwise it lets the default information.

4.2 PLUGINS

Nowadays, most technologies evolve very rapidly. This is a difficulty when developing a system: if it does not evolve with the tools it uses, it can be rapidly obsolete. We have therefore devised a modular system, so that changes can be made easily and rapidly. Modifications can also be made without knowing all the framework implementation.

The plugin system takes four types of features in account. On the one hand, it allows the developer to support new technologies easily by bringing protocol dependent and profile plugins. On the other hand, it allows the developer to support new functionalities easily by bringing type of

device and service plugins. The plugin system of HAMAC is part of its modularity. It has been developed in this way in order to add new features or updates in a simplest manner.

The plugin system has been one of the focuses of last semester. Nevertheless, it has not been described precisely. In this section is made a description of this system.

4.2.1 PLUGINS IN HAMAC

4.2.1.1 Organization

The organization of the plugin system is shown in Figure 27.

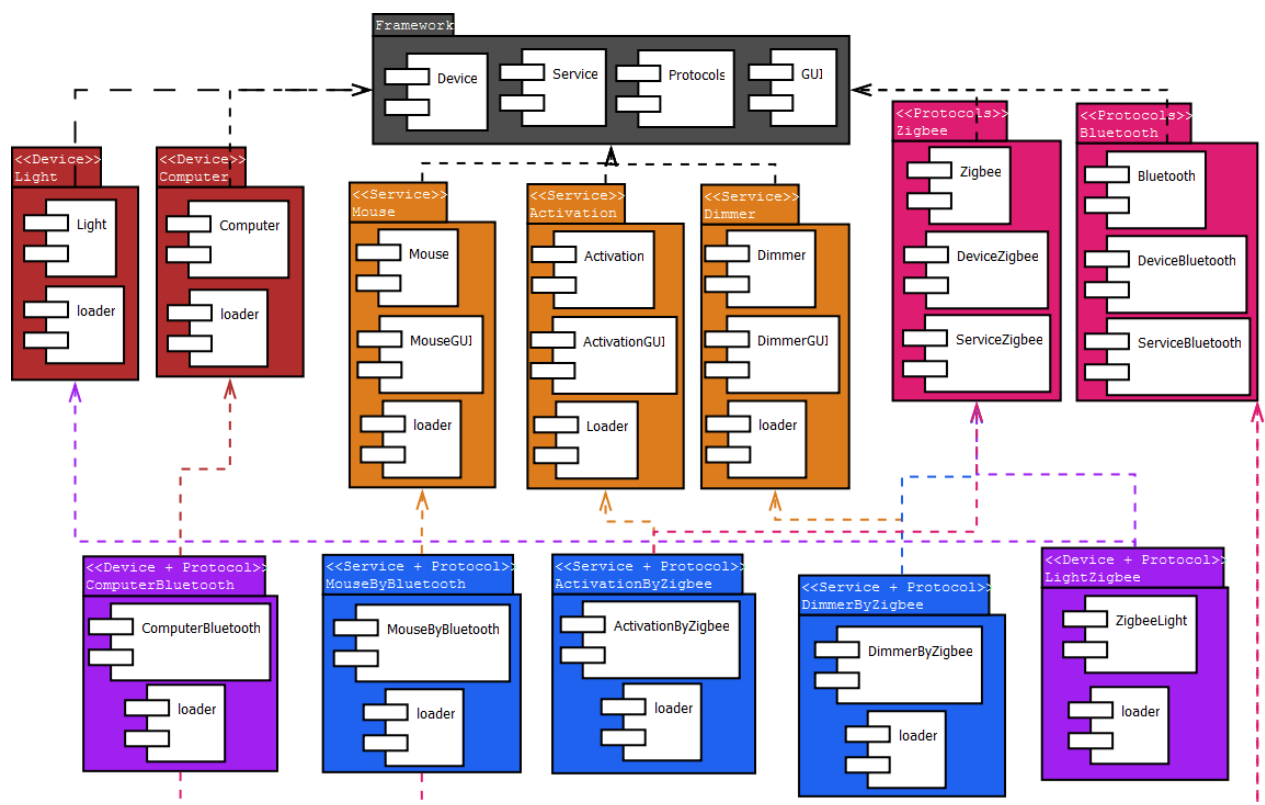


Figure 27: Component diagram of the plugin system

As shown in Figure 27, in grey is the HAMAC framework. In red are *Device* plugins, in orange *Service* plugins, in pink *Protocol* plugins, in purple *Device* definition according to one protocol plugins, and in blue *Service* definition according to one protocol plugins. Dependencies between plugins are also shown in Figure 27. Thereby purple plugins are dependant of one red and one pink because it brings more specification according to one device and one protocol being to mix of both. In a same way, blue plugins are dependant of orange and pink ones. Indeed, they bring the specification of general services to a particular protocol.

4.2.1.2 Description

4.2.1.2.1 General device plugins

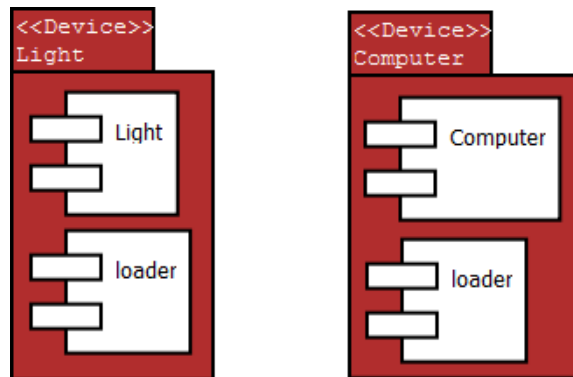


Figure 28: Device plugin components

Plugins shown in Figure 28 define a general device. A general device is for instance a computer, a light, a wheelchair, a heater, etc.

4.2.1.2.2 General service plugins

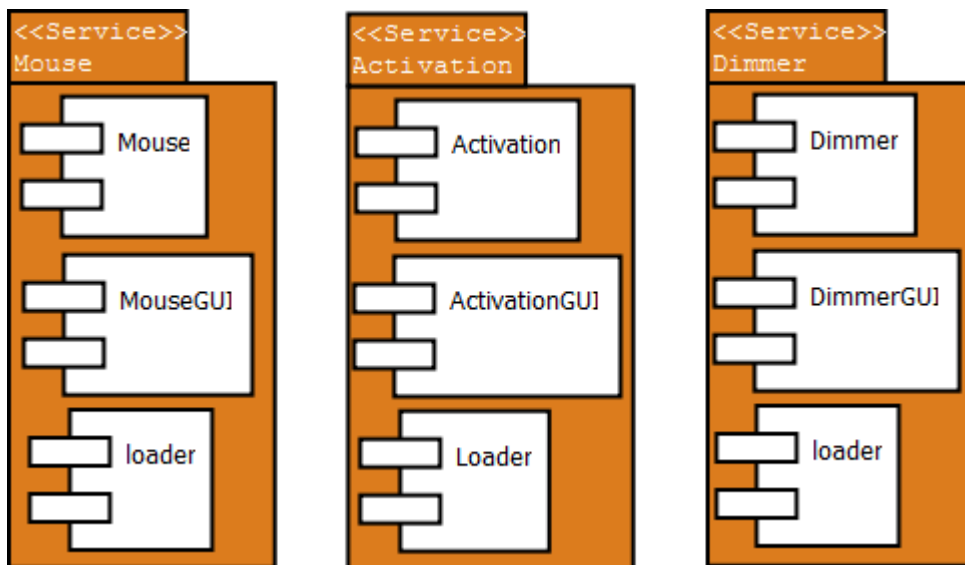


Figure 29: Service plugin components

Plugins shown in Figure 29 define a service and an associate graphical user interface (GUI). The GUI will be presented each time the user will use a service. The same interface is presented for different sub specification of a service. For instance, if the user uses the activation service of a light, switch, or of another device, the same GUI will be presented.

4.2.1.2.3 Protocol plugins

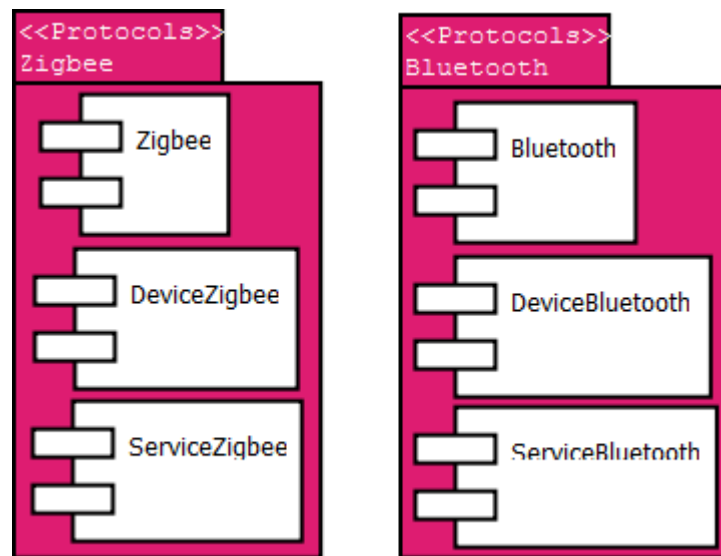


Figure 30: protocol plugin components

Plugins shown in Figure 30 define protocols. These plugins can be connectivity plugin and bring only a connectivity support. Then another protocol will use the connectivity provided to bring more features like service discovery support. Or both the connectivity and the discovery are grouped in a same plugin. It is the case for Bluetooth and Zigbee. These plugins manage all necessary processes to use a protocol network.

4.2.1.2.4 Protocol specific device plugins

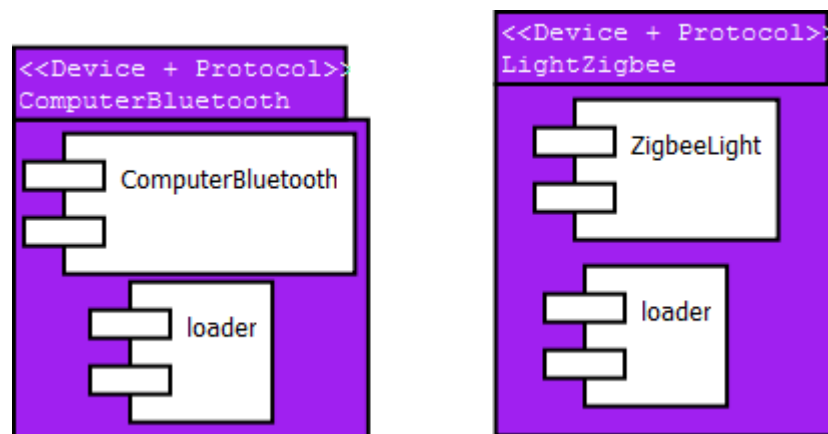


Figure 31: Protocol specific device plugin component

Plugins shown in Figure 31 are the specification of device plugin shown in Figure 28 according to one protocol plugin shown in Figure 30. They define protocol specificities to each kind of device.

4.2.1.2.5 Protocol specific service plugin

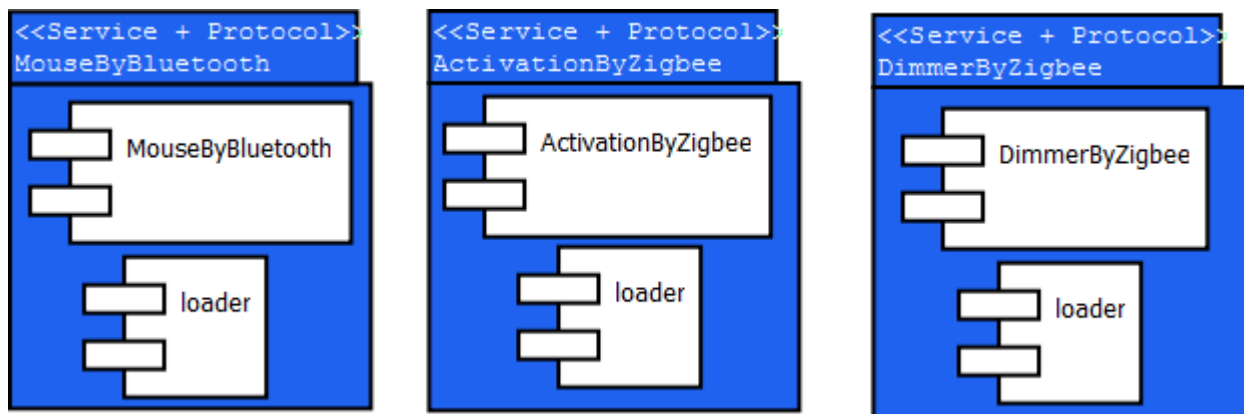


Figure 32: Protocol specific service plugin components

Plugins shown in Figure 32 are the specification of service plugins shown in Figure 29 and protocol plugins shown in Figure 30. They define the service specification according to one protocol and to the general interface. For instance, the way to activate an object is different depending on whether it is on a Zigbee network or a Bluetooth network.

4.2.2 ARCHITECTURE OF A PLUGIN

A plugin is composed of three classes:

- The implementation class: depend of the plugin type. For instance, if it is a device plugin, then it is the implementation of the device.
- The plugin class. This class will create an instantiation of the implementation class. It also contains the dependency information between plugins.
- The loader class. This class will register the plugin class inside the HAMAC framework.

These three kinds of class are contained in a plugin package presented in Figure 33.

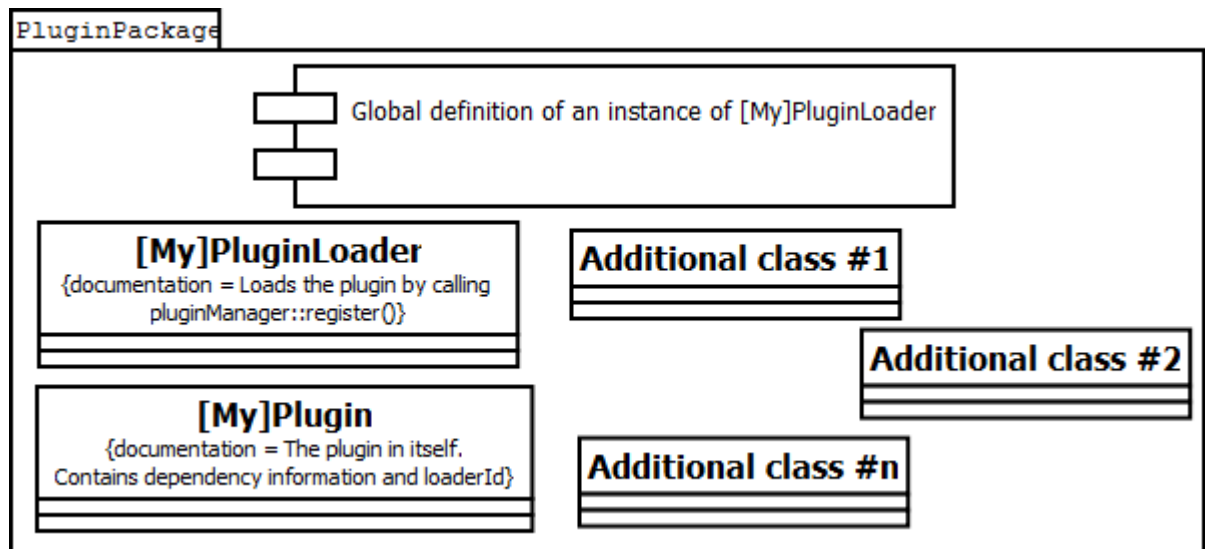


Figure 33: Plugin package. It contains all classes necessary to define a plugin.

4.2.3 PLUGIN MANAGER

The role of the plugin manager is to load all plugins. When a plugin is loaded, it registers automatically itself inside the plugin manager. Then it can proceed to the dependency check. It is also this class who can unload plugins. Then all plugins are available for the entire system. Thus, a protocol can ask for a device plugin and start to use it. The plugin manager class is defined in Figure 34.

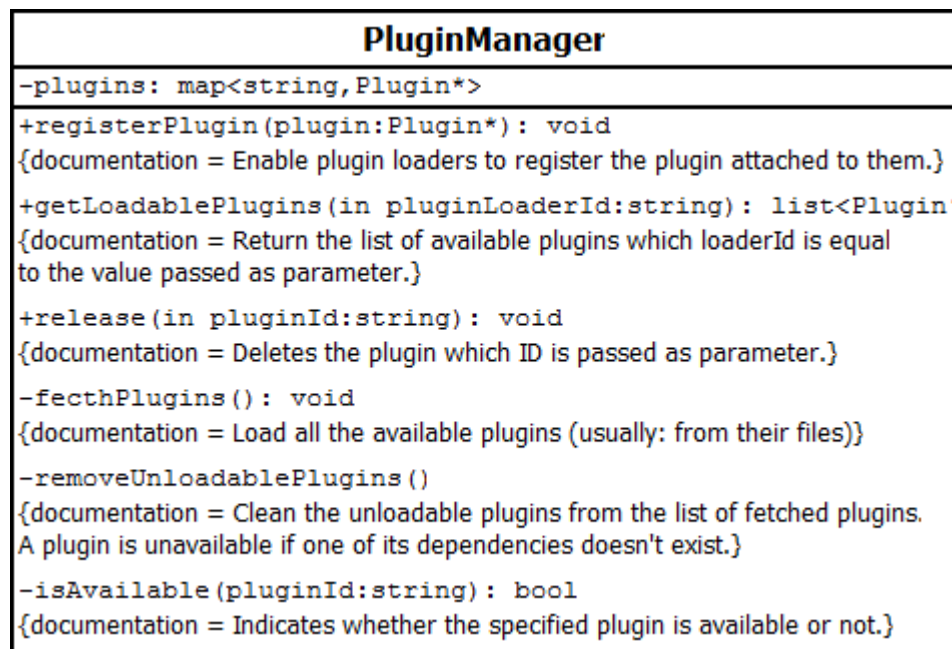


Figure 34: Plugin manager class. This class is the central point of the plugin system.

5 SUMMARY & CONCLUSION

This project was aimed at offering solutions to improve the man-to-machine interfaces dedicated to quadriplegic people. It was elaborated in a context where TKS's intra-oral tongue-controlled device provided an excellent hardware interface between quadriplegic people and the electronic world, and where interoperability between devices for functionality sharing was becoming possible due to the more and more broaden usage of standard protocols enabling it in consumer electronics devices. The focus of our studies was put on creating HAMAC: an improvement of TKS's system able to control more devices than a single computer and providing an intuitive interface offering visual feedback to its user.

In the first part of this project, we focused on building the HAMAC framework, the core component of HAMAC. This framework was made to combine the services of several standard technologies into a highly interoperable system for controlling electronic devices. This system was provided with a visual interface enabling this control to be made generically. Highly modular, the HAMAC framework was also made to support new technologies and features easily. Our analysis was backed up by the implementation of a proof of concept, using this framework to control Bluetooth enabled computers and Zigbee lights.

This report describes the second part of this project, dedicated to improve our former work. Its main focus is put on improving our system in terms of power management and security. It also presents small improvements that were required in order to strengthen our framework.

This section summarizes our achievements, describes the current limitations of our work, and summarizes the future works necessary to make HAMAC fully achieve its objectives.

5.1 ACHIEVEMENTS

5.1.1 ENERGY CONSUMPTION

The first step of our study was dedicated to make the HAMAC framework less energy consuming. Our focus was first put on analysing the sources of energy consumption of the HAMAC hardware, and on understanding how energy could be saved when operating the HAMAC device. We emphasized the possibilities offered by many hardware components to adapt their energy consumption to their usage. This led us to define mechanisms for adapting the hardware resources used by HAMAC to the needs of the application. Our approach to evaluate the resources required by the HAMAC software at any time was based on an analysis of the user needs. We first looked at shutting down the hardware components that were never needed for the operation of HAMAC. We then underlined the dependency between the hardware resources and the services being used by the user, which we addressed by making each service specify its resource needs through a flag based system. Finally, we decided to take advantage of the periods of time in which the user does not need the HAMAC device to make it switch between several low energy modes. The application of these mechanisms to lower the energy consumption of the HAMAC platform enabled us to make it 2.5 times more energy efficient, according to our theoretical evaluation.

In addition to these efforts, we decided to analyse the energy consumption of the radio chips with a special care. To optimize the use of these chips, we strengthened our energy saving strategy with probability based mechanisms, to further reduce their energy bill without impacting too much HAMAC's functionalities. Based on these additional mechanisms, our approach enabled us to make these chips nearly 10 times more energy efficient, also according to our theoretical evaluation.

5.1.2 SECURITY & SAFETY

The second step of our study focused on improving HAMAC's security. We distinguished the need to secure the data manipulated by the user from the need for the operation of HAMAC to be safe for him or her.

To address the first of these needs, we looked closely at the security mechanisms enabling to secure a communication link on various wireless protocols. This analysis helped us to emphasize the modifications needed for our framework to be ready for data security. These modifications mainly concerned the persistency of the configuration of HAMAC and the visual interactions required to help the user initiate secure communications.

HAMAC also needs to ensure the safety of its users. We underlined the fact that, although the goal of HAMAC is to control mostly non-harmful consumer electronic devices, some devices – such as wheelchairs – can put the user in danger if they are not controlled properly. To prevent HAMAC from impacting the safety of the user, such devices should still be controllable even if the HAMAC framework fails to execute properly. To handle the case of such specific plugins, we defined a way to enable some hardware components to take control upon the HAMAC device without relying on the HAMAC framework. This feature is based on a hardware notification mechanism and on a management of control priorities on the framework's side. Our work allows for example wheelchair plugins to react to the inputs of the user even if the HAMAC framework is crashed. It also enables the framework to come back to a convenient state once restarted.

5.1.3 OTHER WORK

To finish, the last part of our study includes work on new or improved features of the framework, which were required to support the implementation of the works of this semester. We worked on re-designing the management of events in the framework and on enabling it to save configuration options persistently. We also included into this part of the report a better description of the plugin management system that we built last semester.

5.2 WHAT SHOULD BE IMPROVED

The power management system designed this semester is efficient and provided us with satisfying results. However, it can still be perfected in several ways.

First, the evaluation of our work is only theoretical. It is based both on an arbitrarily defined use case and on a model representing the effects of our power saving strategy. To improve our work

and maybe optimize our strategy, measurements should be performed on the final hardware that will be used for the HAMAC device.

Secondly, several of our mechanisms for saving energy are based on arbitrarily defined parameters, such as the time to wait before considering that a user whom does not produce any inputs does not need the HAMAC device. Instead of being fixed, the best value for each of these parameters could be calculated based on real life experiments. The HAMAC framework could also calculate the best value for some of these parameters through an observation of the user behaviour.

Another improvement of our work could be to adapt the energy saving mechanisms of the HAMAC framework to the final hardware of the HAMAC device. In fact, most of the mechanisms presented in this report can be generically used on any hardware platform. However, the final hardware of the HAMAC device may have other functionalities enabling to add new mechanisms to our energy saving strategy.

To finish, our study to lower the energy consumption of HAMAC focused only on adapting its hardware resources to the needs of the framework and of the user. It did not address the optimization of the operating system running on the HAMAC device, and barely mentioned the optimization of the HAMAC software in terms of energy. These optimizations can be the next important step to take to reduce even further the energy consumption of HAMAC.

5.3 FUTURE WORKS

With this project, we were able to produce a stable, secure and energy efficient proof of concept of HAMAC, able to control both Zigbee lights and Bluetooth enabled computers. However, our work could still be improved by performing several tasks. We can split these tasks in two parts, according to whether these tasks should be performed in the short term or in the long term. Performing the short term tasks should be sufficient to get a usable – but not necessarily complete – product.

5.3.1 SHORT TERM OBJECTIVES

In the short term, our objectives are to improve our software design and to integrate the HAMAC framework on a hardware platform specifically adapted to its needs.

The works to undertake thus include performing real-life measurements to perfect our power management system and continuing our study to better define the arbitrary parameters on which our energy saving strategy is based.

We also need to ensure the full compatibility of the HAMAC framework with the Bluetooth and Zigbee profiles used in our proof of concept, and to integrate our system with TKS's one so that it can be used with TKS's intra-oral device for a minimum additional cost or energy consumption.

Finally, adapting our power management system to the final hardware of HAMAC is an important point. The operating system used on the HAMAC device should also be configured and optimized to run on the final hardware platform with a minimum of resources.

5.3.2 LONG TERM OBJECTIVES

Long term objectives for the HAMAC project should include the improvement of its interoperability capabilities. Although our proof of concept is in fact limited to the use of one profile per protocol and of only two protocols, its modularity enables it to support easily other protocols. Our analysis shows in fact that high interoperability can be achieved on the HAMAC device by adding only the support for the R2FCE Zigbee Profile, the AVRCP Bluetooth Profile and the DLNA + UPnP + 802.11.X (Wi-Fi) protocol stack to it.

Moreover, further works may include, in the long term, a better study of the operating system and graphical interface to use on the HAMAC device, with the objective to achieve more performance and/or even less energy consumption. The implementations of our framework and of the QT interface using it were intentionally made independent from each other during the first part of this project to ease the process of adapting a new interface to the framework.

5.4 WORD OF THE END

To close this one year project, we can only underline the more than satisfying results that we obtained when building HAMAC, a next generation interface between the disabled people and the digital world. These results have been put on the front scene by being granted the *Handi's Ability* award from the French association Hanploi in March 2010. They are also honored by the little amount of work that we are left with to transform HAMAC from a student project to a reliable product.

By focusing on constraints such as high modularity, interoperability, intuitivism, mobility and ease of use without external help, we defined our own way to look at man-to-machine interfaces for disabled people. As students, we can only hope that our vision will be followed in the future to provide a better quality of life to all the quadriplegic people.

6 APPENDIX

6.1 PROJECT METHODOLOGY

To support ourselves in the development of this project, we decided to make use of several methodology models. Such models are in fact useful both for describing the output of a project and for organizing the work to be done. They help to think about every aspect of the project and to forecast the incoming difficulties. There are a lot of different methodology models available, each model having strengths and weaknesses adapted to different kinds of projects. Some are software oriented, while others are hardware oriented. Some are made for big projects in many different domains, and others are well-suited for a little team or for a more specific subject.

We used three of these models during this project. The A^3 and the UML (Unified Modeling Language) models helped us to get a better overview of the system to build, and to describe it in a precise way. To organize the different tasks that we had to perform, we also used the “2 Tracks Up” model. All these models are described below.

6.1.1 A^3 MODEL

As presented in papers [50], [51] and [52], the A^3 model divides the development and implementation of an algorithm into three distinct focus areas, namely Application, Algorithm and Architecture (AAA, and thus A^3). These areas can be represented by three circles connected together by a set of links.

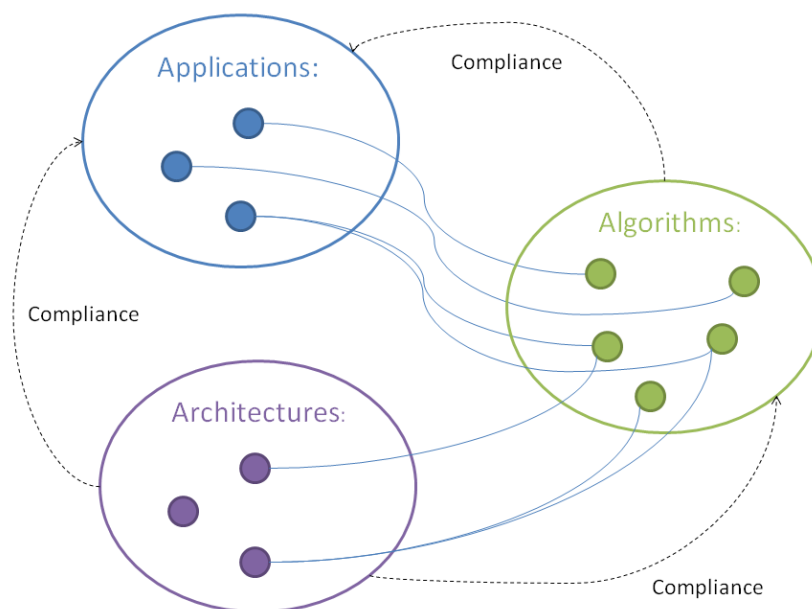


Figure 35: A^3 model, the A^3 model divides the development and implementation of an algorithm into three distinct focus areas, namely Application, Algorithm and Architecture.

The first thing described in this model is the set of applications enabled by the system being built. Each of these applications implies the development of one or more algorithms. Each algorithm can be implemented in several architectures. This model is flexible, so that we can implement our system in many ways.

The framework is iterative. On each step of the process, compliance between modules should be verified. Tests should be performed on each implementation and feedback should be added, for instance, to improve the specification of the applications. Physical or functional constraints can appear: a good example is timing constraints.

6.1.1.1 Applications

The “Applications” area constitutes the highest level of this model. It is used to support a first analysis of the problem. The objectives of the final product should be listed in this area, for a quick evaluation of the requirements of the project. This process ensures a better understanding of the project, by forcing the team to define its aims and scope. Each application listed in this area of the model should be part of the answer to the simple question: what are the objectives of the project? It should also be described in sufficient details to prepare the next step of the project.

6.1.1.2 Algorithms

The next step is the definition of algorithms which will help us to build the applications. One application can be implemented by using several algorithms. This step involves listing the different algorithms available or required, and evaluating them. For instance, Matlab could be used to simulate some signals treatment. Another good evaluation of how algorithms fit the requirements of the project consists in fast developing and evaluating a piece of software on a classical computer, to see the performances required by the use of these algorithms.

6.1.1.3 Architecture

Each algorithm chosen puts some constraints on the architecture of the final device to design. The different architecture candidates are listed in the Architecture bubble. Algorithms are then mapped to architectures that can support them, helping the developers to see the trade of between using an algorithm instead of another or one architecture instead of another.

6.1.1.4 Compliance

On each step of the process, tests are performed to check that the requirements of the upper level are met. Changes from previous levels are tested for feasibility in the present level. When a solution is found, the next level should be analyzed, and functional tests adapted to the new solution should be performed. If the improvements of a certain level of abstraction require too many changes, the requirements of its upper level of abstraction can be changed. However, this operation involves performing all the tests again on the system, which is an expensive process.

6.1.2 UML



Figure 36: UML Logo

“UML is a non-proprietary, third generation modeling language. The Unified Modeling Language is an open method used to specify, visualize, construct and document the artifacts of an object-oriented software-intensive system under development. The UML represents a compilation of “best engineering practices” which have proven successful in modeling large, complex systems.” [53]

6.1.2.1 Overview

These information comes from <http://uml.free.fr> [54]

UML is born in the 90th by a fusion of three methods: OMT [55], Booch [56] and OOSE [57]. UML has been developed to unify all the object oriented modeling languages. A lot of companies and experts work together to improve and extend the functionalities of UML. This model is now a reference in the design of object oriented projects.

Designing object oriented architectures is a difficult task for the human brain. Strict rules should be used in this process, and a very specific language should be employed to communicate within the project team. Many mistakes come from wrong terms used during the modeling process. The use of a unified language is thus needed. That’s why we decided to use UML in our project.

UML provides a language to:

- Represent abstract concepts
- Avoid misunderstandings
- Better analyze the project architecture

UML provides a model:

- Helping to think about the project in an object oriented way since the beginning of the design
- Enabling to describe all aspects of a system





UML is part of the Object Management Group (OMG) [58]. OMG™ is an international, open membership, non-profit computer industry consortium created in 1989. Its goal is to promote standards to be used in object oriented projects.

UML provides several tools to design a system, enabling to describe both static and dynamic views of it through set of fourteen diagram models. We will not use all UML tools, but some of them are well-suited for our problem. The UML diagrams are split in two categories: structure and behavior.

We present below diagrams used in the project design. This is a sum up of the first report [6].

6.1.2.2 Class diagram

The class diagram describes the structure of a software system by showing its classes, their attributes and methods and the relationship between them. The class diagram brings some important object oriented concepts. Basic concepts are:

- Visibility of members
- Different possible relationships between classes :
 - Link (basic relationship)
 - Association 
 - Aggregation 
 - Composition 
 - Generalization 

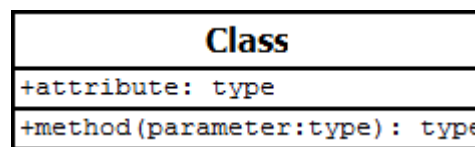


Figure 37: Typical UML class

The class diagram is used to design the structure of our application. The framework has been designed thanks to this model. It is software oriented and is not used to hardware design.

6.1.2.3 Component diagram

The component diagram enables to represent a system as a set of components. It describes it physically and statically. For instance, in a software project, the component diagram depicts all source files, libraries, binaries, etc.

Thanks to this diagram, we can have a quick look on and understand better the dependencies between the several components composing the system. Component diagrams can be used to illustrate the structure of arbitrarily complex systems. Each component can for example be described internally by its own component diagram.

6.1.3 “2 TRACK UP” MODEL

This section is based on this document: “Conduite de Projet “ [59].

Management models are useful to help conducting the project and finishing it on time. They also help to organize the design activities. There are different styles of project management methodologies. Some are sequential, others are iterative.

For the sequential methodologies, tasks are in cascade. The whole project is divided in several steps, which are finished one after the other. The project is thus managed step by step. The advantage of this approach is to reduce development risk by decreasing the amount of uncertainty and decomposing the project in a set of understandable sub-tasks. This is the “divide and conquer” idea. The drawback of this approach is that it does not take into account the user of the final product being designed. The tasks described with this methodology are too technical, while the quality control is performed only in the end of the project. For a better management of the quality of the work being made, this quality control should be taken into account continuously. The risk of not doing so is the user refusal of the final product.

For the iterative methodologies, there are two ways of performing tasks:

The progressive tasks are based on both versatility and multidisciplinary. If the requirements change, these methods decrease the impact on these changes on the project. Another advantage is that users, forgotten in the previous methods, are included in the project. Quality can also be verified since the beginning. The drawback is the product is the result of an amount of update. It becomes more complex and support is difficult.

The object method is based on the separation between requirements and architecture. Tasks are paralyzed. For the rest, all is performed like progressive methods. Advantages are we can take in consideration architecture problems since the project start. We use also the object approach in managing the project.

The “2 Track Up” model is part of the last family.

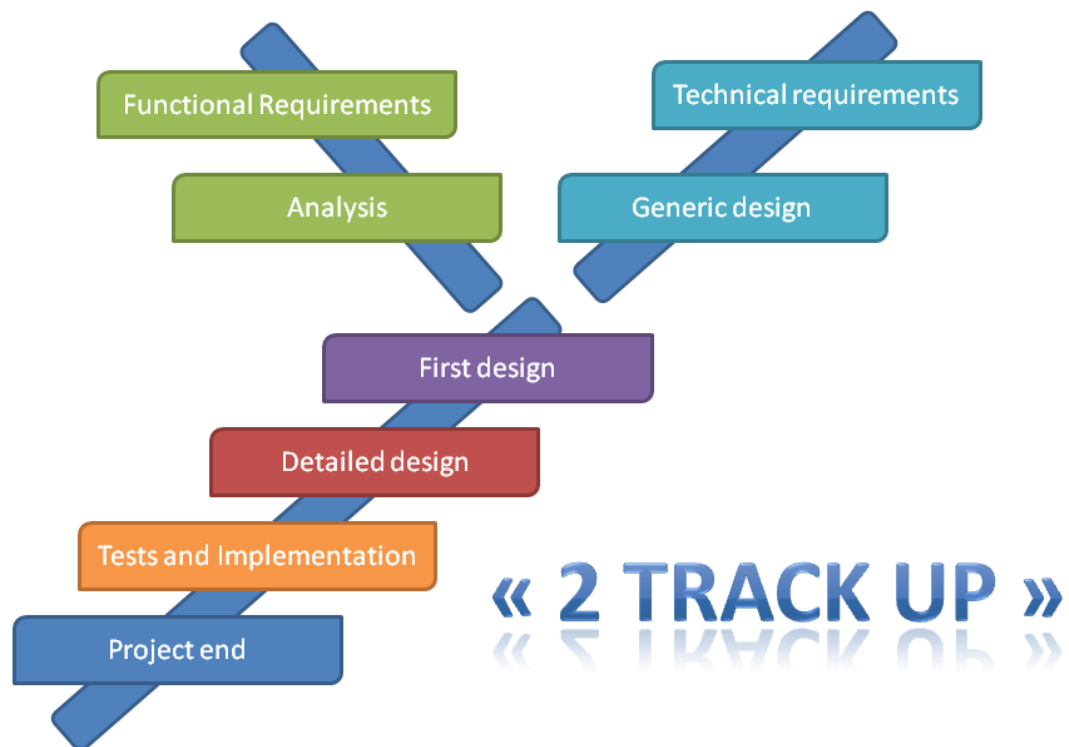


Figure 38: The 2 Track Up model separates requirements and architecture. Then it is based on progressive methods.

The project begins in two distinct parts. The first of these parts is dedicated to the research of the functional needs of the project, and to the analysis of how they can be answered. The second part focuses on the research of technical requirements of the product to build. A generic conception of the system can be designed in this part, to provide a first prototype to evaluate the hardware.

A first design of the system is only started once these two parts are processed. This design is then completed during a phase of “detailed design”, before being implemented and tested. The project ends when a successful implementation of the detailed design was developed.

6.1.4 REALIZATION CONCEPTS

In this section, Resource Acquisition Is Initialization (RAII) concept is presented. This concept is not the fruit of work performed in this project but a result of research on the way to implement the design in the most efficient and safe way. The goal of this section is to present this concept in order to be reused in a future work.

6.1.4.1 Resource Acquisition Is Initialization

This paragraph is based on [60].

In some point inside the program, when a resource is acquired, it has to be released. A resource can be:

- A file
- A mutex
- A network connection
- A dynamic memory allocation

The problem is that if an exception occurs, or something wrong happens after the resource acquisition and before the release, the resource is still acquired and memory can be leaked. For instance if the program enters in a critical section in locking mutex, and then an exception occurs inside the critical section, the mutex is never unlocked. In this case, the execution can be stopped forever.

To prevent this problem, a programming idiom exists. It is the Resource Acquisition Is Initialization (RAII) idiom. In C++ language, it consists in acquiring the resource inside the constructor of a class and releasing it inside the destructor. Then when we create an instance of this class on the stack, and an exception occurs, the object's destructor is called and the resource is released.

This is a programming idiom and some well-known libraries (STL [60]) are following it. This is important to mention in order to not forget to have a good behavior in implementing HAMAC's design.