



Master Thesis

Master Internacional de Programari Lliure

Estudis d'Informàtica Multimèdia i Telecomunicació
Universitat Oberta de Catalunya

Cognitive stimulation through task-oriented telerobotics

Estudiant: *Sergi Torrellas*
Tutor: *Juan Manuel Fernández Ramírez*
Consultor: *Gregorio Robledo*
Especialitat: *Desenvolupament d'aplicacions*
Data: *June 2013*

Copyright (C) 2013, SERGI TORRELLAS SOCASTRO.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.

A copy of the license is included in the section entitled "AnnexA: GNU Free Documentation License"

*"El fracaso no es una opción.
Es un privilegio reservado para los que lo intentan"*
Anónimo

"Soyez réalistes, demandez l'impossible"
Paris, May 1968

"Vindran de tot arreu per veure el que estem fent"
Antoni Gaudí

Abstract

Cognitive Stimulation Therapy is a brief treatment for people with mild to moderate cognitive disorder with the objective of stimulating and engaging people, whilst providing an optimal learning environment. Cognitive stimulation is to improve an individual's capacity to process information so as to enhance independence and dignity in everyday life. Currently, the treatment consists of hierarchically organized treatment tasks and exercises which require repetitive use of the cognitive system in a progressively more demanding sequence. Therefore, these repetitive tasks have an impact on motivation which becomes the biggest challenge in CST reducing the performance of the therapy¹.

The project presented, iCognos, consists of a flexible platform to assist end-users in performing a series of mental tasks with a sensitized mobile telerobotic platform aimed at mitigating the problems associated to cognitive disorders with an ecological cognition approach². The “ecological” cognition states that the tasks for stimulating the cognition need to follow the “holistic”, that is, taking into account the environment, rather than “analytic” approach. In other words, the project proposes a cognitive task aligned with the ecological cognition by means of a SmartObject. A SmartObject is defined as a physical artifact able to collect information of the environment to finally understand its surroundings to react accordingly. iCognos is expected to increase the motivation of the subject which, therefore, improves the cognitive skill of the user with this novel ecological cognition approach.

iCognos is in need of concentrating its efforts in successfully developing a platform for stimulating cognitively. To fulfill this objective, iCognos project has benefited of the Open-source software (OSS) and Open-source Hardware initiative (OSHW), intended to apply the successful principles of Open-source to the development of Hardware. iCognos is composed of two separated OSHW devices (1) the CSCS (Cognitive Stimulation Control System) implemented by a Raspberry Pi³ and interfaced with a WiiMote⁴; and (2) the CSTS (Cognitive Stimulation Telerobotics System) implemented by the ePuck robot. The software developed for providing such features relies on different OSS (e.g. Cwiid, Qt, Bluez) aligning the project with the outcomes of the community.

¹ M. Roussos, Issues in the Design and Evaluation of a Virtual Reality Learning Environment, M.Sc. Thesis, University of Illinois at Chicago (1997).

² Jonathan Bishop. Ecological Cognition: A new dynamic for Human-Computer Interaction
<http://www.scribd.com/doc/4680038/Ecological-Cognition-A-new-dynamic-for-humancomputer-interaction>

³ The Raspberry Pi Foundation. <http://www.raspberrypi.org/>

⁴ Wikipedia Article – WiiMote Controller. <http://es.wikipedia.org/wiki/Wiimote>

Agraïments

Aquesta tesi i el màster de programari lliure, han estat una realitat gràcies a la col·laboració de moltes persones. Tot i que és força difícil repassar a tothom perquè seria molt llarg voldria enrecordar-me d'algunes de les persones amb unes quantes línies.

En primer lloc, m'agradaria agrair a Felip Miralles pel seu guiatge a l'hora de decidir els estudis que més em convenien i empenyem a conèixer en més profunditat el món del programari. Tenies raó. No n'hagués fet res de uns estudis d'economia. També voldria aprofitar per agrair-li tota la gestió de projectes/persones equips que he après durant aquesta tesi. En la mateixa direcció, voldria agrair també a en Felip, i per extensió a la Fundació Privada Barcelona Digital, pel seu suport econòmic per la realització d'aquest màster.

Als meus companys del laboratori, Stefan, Juanma i Piero, pels seus consells, discussions i comentaris dels problemes tecnològics que ens hem anat trobant. El laboratori ha estat el vertader responsable de la creació d'aquest concepte innovador. He après molt al vostre costat. Gràcies a tots tres per l'entusiasme i per la feina ben feta.

Per part de la UOC, voldria agrair al meu tutor Gregorio Robledo per la seva ajuda entusiasme i guiatge en el disseny tecnològic i en tots els procediments necessaris.

Encara que ara ja són grans i no entenen el que faig, als meus pares. Perquè m'han empès sense tirar-me a terra, perquè m'han exigit sense ofegar i perquè han estat al meu costat sense tirar-s'hi a sobre.

Finalment, voldria fer una menció especial a la meva companya, la Núria, per suportar-me les angoixes i els "no arribo" durant tants i tants caps de setmana. Sempre empenyent i posant-hi practicitat i "peus a terra" quan cal. Sense tu, estic convençut de que no hagués estat possible. Ara els dos coneixem el món del programari lliure ;).

Table of contents

| | |
|--|-----------|
| TABLE OF CONTENTS | 5 |
| 1 INTRODUCTION | 9 |
| 1.1 MOTIVATION | 9 |
| 1.2 GOAL OF THE MASTER THESIS | 10 |
| 2 VIABILITY ANALYSIS | 12 |
| 2.1 PROJECT DESCRIPTION | 12 |
| 2.2 SCIENTIFIC AND TECHNOLOGICAL OBJECTIVES | 12 |
| 2.3 IDENTIFICATION OF RISKS AND CONTINGENCY PLANS | 13 |
| 2.4 WORKPLAN AND TIMETABLE | 14 |
| 3 DOMAIN CONTEXT..... | 15 |
| 3.1 ROBOT-AIDED REHABILITATION | 15 |
| 3.1.1 <i>Personal assistant robots</i> | 15 |
| 3.1.2 <i>Robot assisted motor recovery</i> | 17 |
| 3.2 THE OPEN-SOURCE HARDWARE MOVEMENT | 18 |
| 3.2.1 <i>Definition</i> | 19 |
| 3.2.2 <i>Popular projects</i> | 19 |
| 3.3 WIRELESS COMMUNICATION PROTOCOLS | 20 |
| 4 SYSTEM COMPONENTS | 24 |
| 4.1 THE RASPBERRY PI..... | 24 |
| 4.1.1 <i>Board components</i> | 24 |
| 4.1.2 <i>Built-in ARM processor</i> | 25 |
| 4.1.3 <i>GPU : Graphics Processing Unit</i> | 26 |
| 4.1.4 <i>Firmware for boot</i> | 27 |
| 4.1.5 <i>GNU/Linux Distributions</i> | 28 |
| 4.1.5.1 <i>Debian wheezy</i> | 28 |
| 4.1.5.2 <i>Raspbian</i> | 29 |
| 4.2 THE EPUCK | 30 |
| 4.2.1 <i>Origin of e-Puck</i> | 30 |
| 4.2.2 <i>Hardware</i> | 30 |
| 4.2.2.1 <i>Mechanical Design</i> | 30 |
| 4.2.2.2 <i>Sensors, actuators and heart of the robot</i> | 31 |
| 4.2.2.3 <i>Ground Sensor extension</i> | 32 |
| 4.2.3 <i>Software</i> | 33 |
| 4.2.3.1 <i>Libraries</i> | 33 |
| 4.3 WIIMOTE..... | 34 |
| 4.4 QT : GRAPHICAL USER INTERFACES | 35 |

| | | |
|---|---|-----------|
| 4.4.1 | <i>QML : Qt Meta language</i> | 36 |
| 5 | IMPLEMENTATION | 37 |
| 5.1 | SYSTEM DESCRIPTION | 37 |
| 5.1.1 | <i>iCognos featured open-source projects</i> | 38 |
| 5.2 | COMMUNICATION INTERFACES | 39 |
| 5.2.1 | <i>Bluetooth</i> | 39 |
| 5.2.2 | <i>Protocol for I1: ePuck – Raspberry Pi</i> | 40 |
| 5.2.3 | <i>Protocol for I2: Raspberry Pi – WiiMote</i> | 41 |
| 5.3 | COGNITIVE STIMULATION CONTROL SYSTEM..... | 43 |
| 5.3.1 | <i>Configuration of development environment</i> | 43 |
| 5.3.1.1 | Bluetooth configuration for R-PI | 46 |
| 5.3.2 | <i>Architecture of the application</i> | 49 |
| 5.3.3 | <i>User Interface development</i> | 51 |
| 5.3.3.1 | Hex-O-Select as GUI | 51 |
| 5.3.3.2 | Input control..... | 53 |
| 5.4 | CONGNITIVE STIMULATION TELEROBOTICS SYSTEM..... | 57 |
| 5.4.1 | <i>Configuration of development environment</i> | 57 |
| 5.4.2 | <i>Firmware development</i> | 59 |
| 5.4.2.1 | Command interpreter | 60 |
| 6 | ANALYSIS OF VALIDATION | 61 |
| 6.1 | TESTING ENVIRONMENT..... | 61 |
| 6.2 | DISCUSSION OF THE RESULTS..... | 62 |
| 7 | CONCLUSIONS | 64 |
| 7.1 | LAST WORD | 64 |
| 7.2 | FUTURE WORK | 65 |
| 8 | BIBLIOGRAPHY | 66 |
| ANNEX A : GNU FREE DOCUMENTATION LICENSE | | 68 |
| 0. | PREAMBLE..... | 68 |
| 1. | APPLICABILITY AND DEFINITIONS | 68 |
| 2. | VERBATIM COPYING..... | 70 |
| 3. | COPYING IN QUANTITY | 71 |
| 4. | MODIFICATIONS | 71 |
| 5. | COMBINING DOCUMENTS | 73 |
| 6. | COLLECTIONS OF DOCUMENTS | 73 |
| 7. | AGGREGATION WITH INDEPENDENT WORKS | 74 |
| 8. | TRANSLATION | 74 |
| 9. | TERMINATION | 75 |
| 10. | FUTURE REVISIONS OF THIS LICENSE | 75 |

List of figures

| | |
|---|----|
| Figure 1. Blooms taxonomy | 9 |
| Figure 2. The assistive robotics : REEM and Q.bo | 16 |
| Figure 3. The therapeutic robot, Paro..... | 17 |
| Figure 4. The arm and hand therapy for children, Armeo | 18 |
| Figure 5. Comparison of network wireless protocols [26] | 21 |
| Figure 6. The Raspberry PI | 24 |
| Figure 7. The ARM1176JZF-S architectural block..... | 26 |
| Figure 8. VideoCore IV GPU architecture | 27 |
| Figure 9. The Debian logo | 29 |
| Figure 10. The Raspbian logo | 29 |
| Figure 11: E-Pucks mechanical design | 31 |
| Figure 12: E-Pucks components | 31 |
| Figure 13: The outline of the electronic of the E-Puck | 32 |
| Figure 14: Ground sensors in detail | 33 |
| Figure 15: Connection between e-puck and the Ground Sensors extension board..... | 33 |
| Figure 16: Information about the E-Puck library | 34 |
| Figure 17: Nintendo's WiiMote..... | 34 |
| Figure 18: Qt framework logo | 35 |
| Figure 19: iCognos system description | 37 |
| Figure 20. iCognos system | 38 |
| Figure 21: WiiMote pairing procedure | 41 |
| Figure 22: Configuration of the development ecosystem | 43 |
| Figure 23: Qt Compilation with Acceleration HW (GPU) | 44 |
| Figure 24: SSH connection to the R-PI..... | 44 |
| Figure 25: Compilation of the Qt framework..... | 45 |
| Figure 26: Qt Creator | 46 |
| Figure 27: USB dongle used for Bluetooth communication (marked in Red)..... | 46 |
| Figure 28: Output of commands [<code>hciconfig dev</code>] and [<code>hcitool scan</code>]..... | 47 |
| Figure 29: Architecture of R-PI software..... | 49 |
| Figure 30: UML Diagram of the CSCS running on the R-PI..... | 50 |
| Figure 31: Signal&Slot mechanism | 51 |
| Figure 32: Hex-O-Select interface | 52 |
| Figure 33: QML Code developed for the Hex-O-Select interface | 53 |
| Figure 34: UML diagram for Hex-O-Select..... | 53 |
| Figure 35: Raspbian repositories | 54 |
| Figure 36: libcwiid1 installation via apt-get | 54 |
| Figure 37: Compilation of cwiid on R-PI..... | 55 |
| Figure 38:Using wmdemo..... | 56 |
| Figure 39:Combination of C and C++ language | 56 |
| Figure 40: C30 compiler tools data flow | 57 |
| Figure 41: E-pucks development environment, MPLAB IDE..... | 58 |
| Figure 42: ICD3 programmer/debugger connected to E-Puck..... | 58 |
| Figure 43: Tiny bootloader for PIC | 59 |
| Figure 44: ePuck UML state machine diagram..... | 59 |
| Figure 45: Conceptual map of the firmware..... | 60 |
| Figure 46: Code that turns the light on..... | 60 |
| Figure 47: Code to set the robots wheel speed..... | 60 |
| Figure 48: Final prototype of iCognos..... | 61 |

List of tables

| | |
|---|-----------|
| <i>Table 1. Risks identification and contingency plans</i> | <i>14</i> |
| <i>Table 2. Gantt Diagram for iCognos</i> | <i>14</i> |
| <i>Table 3. Comparison of wireless low-range technologies [27]</i> | <i>23</i> |
| <i>Table 4: Technical specifications of the Bluetooth protocol</i> | <i>40</i> |
| <i>Table 5: I1: CPCS-CSTS protocol description.....</i> | <i>41</i> |
| <i>Table 6: Cwiid API for communication with WiiMote</i> | <i>42</i> |
| <i>Table 7: Testing program of Bluetooth Connection.....</i> | <i>47</i> |
| <i>Table 8: Code for Bluetooth communication with the CSTS (ePuck).....</i> | <i>48</i> |

1 Introduction

1.1 Motivation

The human brain is not only the site of our personality, cognition, thoughts, feelings and other human characteristics. Cognition has to do with how a person understands and acts in the world constantly changing. Cognition relies on an ensemble of cognitive abilities, the brain-based skills we need to carry out any task from the simplest to the most complex. They are related to the cognitive mechanisms rather than with any actual knowledge. Any task can be broken down into the different cognitive functions needed to complete it successfully. For instance, answering the telephone involves at least: perception (hearing the ring tone), decision making (answering or not), motor skills (lifting the receiver), language skills (talking and understanding language), and social skills (interpreting tone of voice and interacting properly with another human being).

Cognition encompasses different abilities such as attention to tasks, memory, reasoning, problem solving, and executive functioning (e.g., goal setting, planning, initiating, self-awareness, self-monitoring and evaluation) but also other coming from psychology like awareness of oneself and surroundings. Different taxonomies have been developed so as to understand the different processes involved in cognition. The taxonomy of Bloom [1], developed in 1956, is one of the most popular due to the educational nature of the taxonomy.



Figure 1. Blooms taxonomy⁵

Bloom's taxonomy proposes the following skills as being involved in the cognitive process of humans:

- *Analyze*: skills to examine and break information into parts by identifying motives or causes. Make inferences and find evidence to support generalizations
- *Evaluate*: skills to present and defend opinions by making judgments about information, validity of ideas or quality of work based on a set of criteria

⁵ <http://en.wikipedia.org/wiki/File:BloomsCognitiveDomain.svg>

- *Create*: skills to compile information together in a different way by combining elements in a new pattern or proposing alternative solutions
- *Apply*: skills to make use of new knowledge. Solve problems to new situations by applying acquired knowledge, facts, techniques and rules in a different way
- *Understand*: skills to understand of facts and ideas by organizing, comparing, translating, interpreting, giving descriptions, and stating main ideas
- *Remember*: Skills to exhibit memory of previously learned materials by recalling facts, terms, basic concepts and answers

The skills defined in Bloom's taxonomy are eventually combined for the so-called **Adaptive behaviour** to accomplish the appropriate attitudes necessary for people to live independently and to function safely and appropriately in daily life activities. Adaptive behaviours include real life skills such as grooming, dressing, safety, safe food handling, school rules, ability to work, money management, cleaning, making friends, or even social skills to establish relationships with other individuals.

1.2 Goal of the Master Thesis

The European Brain Council performed a study of the impact of brain diseases (e.g. psychiatric, psychological and cognitive) in Europeans [2], which accounts for a 35% of the total burden of all diseases in Europe. Health disorders related to cognition have many different sources of origin ranging from learning disability, autism or traumatic brain injury to age-related dementia or stroke. In functional terms, cognitive disabled might be taken as anyone whose ability to "perceive, organize and integrate information has been affected by their condition [3].

There is a close link between cognitive impairment and predominantly age-related conditions. Stroke is one of the most common causes of cognitive impairment among elderly people. To take just two examples, in Sweden 25 000 to 30 000 people every year have a stroke while in Britain the annual figure is 150 000. Increasingly people are surviving strokes (there are almost 1 million people in Britain today who have had strokes) but many are left with residual physical and cognitive impairments.

Cognitive impairment is not, however, solely age-related. In some cases, impairment affects children. Cognitive impaired children are affected by **Learning Disability** which refers to a lifelong condition, usually present from birth or that develops before the age of 18. It is a **permanent condition, and people with learning disabilities have impaired skills in areas related to cognition, language, motor and social abilities**. Cognitive disability in children sets a barrier to their development and limits the equal opportunities for these communities. This situation has received the attention at different governmental levels both national and European. Indeed, the EU Disability Action Plan 2003-2010 [4] is aimed to improve the situation of people with disabilities in Europe working for the provision of equal opportunities for disabled people and take steps to eliminate barriers to their full participation in economic activity and in society more widely. From other perspective, the member states are also empowering the equal opportunities through Horizon 2020 [5], the new research instrument coordinated by the EC, in which one of the key priorities refers to

Societal Challenges – Inclusive, innovative and secure societies aimed at ensuring a opportunities for every citizen.

Cognitive Stimulation Therapy (CST) is defined as a brief treatment for people with mild to moderate cognitive disorder involving the stimulation and engage people, whilst providing an optimal learning environment. The cognitive stimulation is defined as a treatment to increase or improve an individual's capacity to process information so as to enhance independence and dignity in everyday life. Currently, the treatment consists of hierarchically organized treatment tasks and provides exercises which require repetitive use of the impaired cognitive system in a created, progressively more demanding sequence. **Motivation becomes the biggest challenge in CST which reduces the performance of the therapy** [6]. Motivation is about doing what it takes to accomplish an objective and begins with a particular goal but sustaining motivation depends on the mental acts of digging deep, keeping track of your progress, recalling past successes, and self-encouragement. The individual may very well wish to improve, but desire alone is not enough.

This Master Thesis, named the iCognos project , has developed a **cognitive stimulation therapy system by means of task-oriented telerobotics to improve accuracy of the programs and motivation of the subject** to continue with the therapies. Robotics for cognitive exercise has demonstrated to have a positive effect on the users [7]. The interaction with real objects, which follows the ecological cognition approach, improves the end-user mood towards the exercises, making them more active and communicative. The “ecological” cognition states that the tasks for stimulating the cognition need to follow the “holistic”, that is, taking into account the environment, rather than “analytic” approach. Furthermore, this system will suppose a **stepforward in the nowadays stimulation therapies proposing transversal cognitive games to the user in which different skills are required.**

Finally, the developed technology can be expanded to improve and extend the portion of population who will be now able to access to cognitive stimulation therapies due to the affordability of the platform and the flexibility it offers enlarging the number of people that can be benefited.

2 Viability Analysis

2.1 Project description

The technological objective of this proposal is to develop a flexible platform for performing task for cognitive stimulation. The system will consist of a low-cost small mobile robotic device, **the Cognitive Stimulation Telerobotics System (CSTS)**, which will be equipped with an actuator system to provide motion, luminosity, sound and vibration capabilities, together with a sensory system able to capture information from the user activity in the exercise (accelerometer, strain gauge, camera, microphone, proximity). The prototype of iCognos will propose the user to perform, in this first prototype, a fixed task combining the aforementioned actuators and sensors. The performance of the results of will be the recorded and controlled by **Cognitive Stimulation Control System (CSCS)**, running on an embedded Linux device. The CSCS provides the user interface which will be developed by following accessibility and usability rules and, additionally, this device is responsible for guiding the user through the task. Finally, the user interacts with the Nintendo's WiiMote controller that, talking with the CSCS, provides a direct manner of controlling the whole iCognos system by means of this flexible and capable platform.

Finally, it is remarkable that iCognos has the intention to be a complementary product for those who already have followed cognitive rehabilitation programs in the hospital. **The goal is to obtain a system with which the therapist is able to monitor the patient's progress while s/he is at home continuing with the therapy for maintenance for a larger period.** This is facilitated by the use of a novel paradigm, Open-source hardware, providing an affordable platform enabling every individual to be benefited by the features of the system to improve his/her cognitive skills.

2.2 Scientific and technological Objectives

Technological objectives

- To breakthrough the traditional means of cognitive rehabilitation by providing an ICT platform able to adapt automatically to the needs of the user, that is, the system proposes exercises to the user according to his/her needs
- To develop an affordable system so as to enlarge the target population
- To obtain a mobile device no bigger than 10cm in order to facilitate the transportation to user's home

Application objectives

- To develop both CSTS device driver and CSCS software modules with open and well-specified interfaces as a base for a flexible application
- To provide an enhanced user experience with transversal exercises following the principles of ecologic cognition

Social objectives

- To better include the cognitive disabled in society, increasing their opportunities and mitigating the problems associated to their impairment in accordance with the European social values.
- To decrease the necessary resources for healthcare in modern societies due to the deployment of a telemedicine platform

2.3 Identification of Risks and contingency plans

| |
|--|
| iCOGNOS-RISK-001 (P: 30%, I: HIGH)⁶ – Usefulness of cognitive stimulation |
| Description: iCognos proposes to create an easy-to-use system for improving the progress and cognitive skills of those who are impaired. For this, it becomes mandatory to reach a better performance in the progress of the recovery of cognitive skills. |
| Mitigation Plan: There will be a major effort put on the design of the system based on a wide research of literature in the related research areas, that is, cognitive rehabilitation and stimulation therapies. Additionally, the potential involvement of rehabilitation caregivers would be a valuable asset. |
| iCOGNOS -RISK-002 (P: 20%, I: HIGH) - Compatibility of combined technologies |
| Description: Compatibility of the combined technologies, regarding transition between standalone and combined implementation. Each of the concepts proposed to be combined were originally realized in different technologies and platforms. The risk is to lose important advantages provided by the proposed combination. |
| Mitigation Plan: To meet this mitigate this risk a systematic and step by step development has been planned, starting with careful assessment of possible designs, and the creation of subsequent prototypes as it is important to identify and to avoid currently unforeseen problems. |
| iCOGNOS -RISK-003 (P: 50%, I: LOW) – End user acceptance |
| Description: Although the iCognos principle and the user-centred design is put in first position during the design stages, the targeted end-users are not expected to have a technological expertise. In this setting, the new technology and the possibilities provided by ActivaKit will need familiarization for the end users. |
| Mitigation Plan: iCognos takes care from the very beginning of creating a platform with which the user feels comfortable. To meet this challenge, it will be essential to perform a deep study on the target population for their classification and categorisation of the targeted population. The outcome of this study will be taken into account when designing the system mainly those parts where the user interaction is required. |
| ACTIVAKIT-RISK-004 (P: 40%, I: HIGH, WP: 6,7) – Integration and Cost trade off |
| Description: Risk of difficulties with proper integration of the technology in a robust, safe |

⁶ P: Probability, I : Impact (H: High, M: Medium, L: Low)

and functional manner also suitable for mass production at a price that can be accepted by this very special market. This risk emerges from the high number of required sensor drivers and connections in the Intelligent Sensor Manager.

Mitigation Plan: This potential difficulty will be met utilising standardised mass produced components and utilising the expertise of the consortium with experience in similar products and, specially, a custom hardware platform with cheap digital systems.

Table 1. Risks identification and contingency plans

2.4 Workplan and timetable

| Week | March | | | | April | | | | May | | | | | June | | |
|--|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|------|------------|------------|
| | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 5 | 1 | | |
| WP1. Requirement definition and design of iCognos | [Task bar] | | | | | | | | | | | | | | | |
| 1.1 iCognos conception of functionalities | [Task bar] | [Task bar] | | | | | | | | | | | | | | |
| 1.2 Cognitive Stimulation Control System | | [Task bar] | [Task bar] | | | | | | | | | | | | | |
| 1.3 Cognitive Stimulation Telerobotics System | | | [Task bar] | [Task bar] | | | | | | | | | | | | |
| 1.4 Requirements for User Interaction | | | [Task bar] | [Task bar] | | | | | | | | | | | | |
| WP2. Configuration of development enviroment | [Task bar] | | | | | | | | | | | | | | | |
| 2.4 Networking communications for code development for RPI | | | [Task bar] | [Task bar] | [Task bar] | | | | | | | | | | | |
| 2.1 Cross-compilation of Qt libraries for RPI | | | | [Task bar] | [Task bar] | [Task bar] | | | | | | | | | | |
| 2.2 Configuration of Bluetooth interface for RPI | | | | | [Task bar] | [Task bar] | [Task bar] | | | | | | | | | |
| 2.3 Configuraion of firmware development environment for ePuck | | | | | [Task bar] | [Task bar] | [Task bar] | | | | | | | | | |
| WP3. Cognitive Stimulation Telerobotics System | [Task bar] | | | | | | | | | | | | | | | |
| 3.1 Bluetooth communication firmware | | | | | | [Task bar] | [Task bar] | [Task bar] | [Task bar] | [Task bar] | | | | | | |
| 3.2 Motion control firmware | | | | | | | [Task bar] | [Task bar] | [Task bar] | [Task bar] | [Task bar] | | | | | |
| WP4. Cognitive Stimulation Control System | [Task bar] | | | | | | | | | | | | | | | |
| 4.1 Development of user interface | | | | | | | | [Task bar] | [Task bar] | [Task bar] | [Task bar] | [Task bar] | | | | |
| 4.2 Communication with WitMote | | | | | | | | | [Task bar] | [Task bar] | [Task bar] | [Task bar] | | | | |
| 4.3 Development of communication with ePuck | | | | | | | | | | [Task bar] | [Task bar] | [Task bar] | [Task bar] | | | |
| WP5. Validation of iCognos | [Task bar] | | | | | | | | | | | | | | | |
| 5.1 Partial validation of components | | | | | | | | | | | | | | | [Task bar] | [Task bar] |
| 5.2 Field trials with end-users | | | | | | | | | | | | | | | | [Task bar] |

Table 2. Gantt Diagram for iCognos

3 Domain context

iCognos involves the combination of different disciplines in order to create the envisaged cognitive stimulation platform. Consequently, this section has been divided into three different areas which are aimed at providing the reader with an overview of the current solutions/alternatives studied and analysed during the development of iCognos.

The first subsection, 3.1 Robot-aided rehabilitation, illustrates the current solutions for creating SmartObjects for rehabilitation, that is, elements with which the user is allowed to interact as they include sensors to understand the context occurring and actuators to modify such environment. Indeed, SmartObject is defined as a physical artifact able to collect information of the environment to finally understand its surroundings to react accordingly. This term is clearly linked with the area of robotics. Indeed, a robot can be understood as an object able to understand the happening of its surroundings. This section continues with the Open-source hardware movement that applies the principles of open-source to the development of hardware multiplying the benefits both society and developers. It is remarkable that iCognos makes highly use of this concept by applying it to the general platform as explained in Section 5. Finally, section 3.3 Wireless communication protocols, reviews the most popular wireless communication technologies which are also a very important part of the iCognos platform. Indeed, all the components of iCognos will be wirelessly linked improving the usability and accessibility of the system.

3.1 Robot-aided rehabilitation

Even though traditional methods of helping people with a mental health problems have been widely adopted such as medication, counselling, cognitive behavioral therapies, exercise and a healthy diet; technological advances can also be used in conjunction with traditional methods to improve the effectiveness. Indications to date [8][9] suggest that robot-aided cognitive training does have a genuinely positive effect on reduction of impairment and the reorganization of the brain. The results in this field indicate that activity-dependent plasticity underlies neuro-recovery. In other words, **neurological changes that underlie recovery are facilitated by targeted sensorimotor activity**, and that this can be accomplished using robot technology. **The use of communicative and other new technologies as a supplement to mainstream therapies for cognitive disorders is an emerging mental health treatment field which is called to improve the accessibility, effectiveness and affordability of mental health care.**

3.1.1 Personal assistant robots

In the past, assistive robotics (AR) has largely referred to robots that assisted people with physical disabilities through physical interaction. This definition is no longer appropriate as it is lacking in scope: it does not cover assistive robots that assist to perform daily life activities, such as pill remainder or activity planning who are low-dependent citizens in a nursing home.

A clear example of assistive robotics are the so-called *Personal Assistant Robots (PAR)*. The term *PAR* is often used in the context of robotic assistants which either help an individual in doing some task or a robot able to assist the user in anything s/he requires. Indeed, PARs are defined as a versatile machine that interacts with both environment and people fully autonomously. Four capacities are required: companion: perception and cognition of environment; learning by observation; decision making; communication and interaction with humans.

For the sake of assisting dependent people, PARs play a very important role in providing the users with the assistance required in their everyday life activities as PARs are integrated in the environment. This kind of applications provides the synergy of Robotics and Ambient Intelligence technologies and their semantic integration to reach an assistive environment.

The robot REEM [10] illustrates the concept of PARs being a good example of this. REEM is a humanoid service robot created by PAL Robotics intended to help people in repetitive and tedious tasks. The designers highlight that REEM was conceived also to interact with people giving them information in public spaces like airports, hospitals. Other applications are more focused on human interaction including features like voice recognition, speech or object recognition. Another example is Q.bo [11], a fully programmable robot able to interact with humans for a general purpose use: in-home security, take care of seniors, play with children o disabled people taking advantage of a great variety of sensors (e..g cameras, microphones) and actuators (speakers, motors).



Figure 2. The assistive robotics : REEM and Q.bo

PAR currently available are designed so as to help people in their tasks, while few of them take care of maintaining cognitive skills of the user. The Paro project [12], though, is aimed at cognitive recovery by putting the accent in simulating animal therapy. Paro is an advanced interactive robot developed by AIST, a leading Japanese industrial automation pioneer. It relies in the well-studied benefits of animal therapy administered to patients with cognitive disorders. Paro has five kinds of sensors with which it can perceive people and its environment. By interacting with people, Paro responds as if it is alive, moving its head and

legs, making sounds, and showing your preferred behavior. Paro also imitates the voice of a real baby harp seal.



Figure 3. The therapeutic robot, Paro

Although it has been demonstrated its benefits for the user, manufacturers have not still put their attention on the cognitive stimulation with robots. Without the proper cognitive stimulation support, dementia and depression sufferers, especially the elderly, can deteriorate rapidly, demanding more from their carers.

This can be mitigated by the combination of mobile PARs (mobile facilitation) working collaboratively with a smart home environment (stationary facilitation) assisting in daily-life tasks and proposing new activities to the user. Such systems combine the use of robots in intelligent domestic environments, with the goal of creating a companion that assists people in their homes and helps them to be independent. The goal is to generate a simple and intuitive relationship between person and machine. These robots are assigned to assist in activities like managing the daily life of the dependent people, generating content for their cognitive stimulation, or analysing data on the state of their health.

3.1.2 Robot assisted motor recovery

There is a need to develop better ways to augment exercise training in a functional way. Using therapeutic adjuncts to facilitate clinical practice, such as robotics, is a new promising development. Robotics allows patients to train independently of a therapist and to improve upon their own functional level (i.e., robot-assisted therapy). In particular, there is strong evidence for robot-assisted therapy to increase treatment compliance introducing incentives to the patient, such as games. In addition, using computer assisted devices for regaining upper limb function, the robot can easily apply new constraints, in order to optimize the required movement pattern. Therefore, the complexity of a motor task to be learned can be controlled for more precisely with robotics than in conventional treatment approaches.

Different products are already in the market following this idea. For instance, the ArmeoSpring from Hocoma, a Swiss company, is devoted to help in the prescription of rehabilitation to regain movement of the arm and hand. ArmeoSpring exoskeleton is an ergonomic and adjustable arm support with integrated springs. It embraces the whole arm, from shoulder to hand, and counterbalances the weight of the patient's arm, enhancing any residual function and neuromuscular control, and assisting active movement across a large 3D workspace. Mounted on a trolley for quick and easy positioning, the ArmeoSpring offers

various self-initiated repetitive therapies to increase the patient's range of motion and selective control. The self-directed exercises motivate the patient to exert intense levels of both concentration and coordination. This product is designed for functional disabled people with low-level of dependency, though severely impaired people can be benefited, helped by the presence of a carer or relative.



Figure 4. The arm and hand therapy for children, Armeo

Many different studies [13][14] show a positive trend toward robot-assisted therapy for the limbs when compared to conventional treatment modalities with regard to motor recovery. However, distinguishing the benefits of the two interacting processes highlighted (motor and functional therapy) is critical for determining the future of robot-assisted therapy. If motor therapy is the dominant stimulus for movement recovery, then robotic actuators may turn out to be technological ornamentation. In this scenario, the question remains whether complex, and potentially expensive, devices are essential for maximizing the learning and recovery capabilities of the injured. However, Robotics for motor recovery does not cover rehabilitation in the scope of functionality of limb, that is, they are limited to a motion recovery. In spite of this, recovery for motor/cognitive disorders needs to be understood as a unique therapy so as to improve the performance. [15][16]

iCognos will collect the outcomes from the previous experiences of the combination of robotics and cognitive rehabilitation (e.g. Paro) combining them with the ability of fine motor rehabilitation (e.g. ArmeoSpring) to create a low-cost platform for performing cognitive stimulation exercises at home.

3.2 The Open-source Hardware movement

The open source software movement has had an enormous impact on today's technology. It has aided academic research, it has changed the way many tech companies do business, and it has changed our society. Open-source software projects have the advantage that the code is openly available for modification and is also often free of charge. It enables the customisation of the hardware easier than custom-building equipment, which often can be quite costly because fabrication requires the skills of machinists, glassblowers, technicians, or outside suppliers.

Now this principle has arrived to hardware. In the book Democratizing Innovation [17], it is argued that a trend toward democratized innovation in physical products is occurring like the free and open-source software movement. The idea of opening the hardware has been getting a lot of attention in the latest years. Open-source hardware [18] (OSHW) consists of physical artifacts of technology designed and offered by the Open design movement. Both free and open-source software (FOSS) as well as open-source hardware is created by this open-source culture movement and applies a like concept to a variety of components. One of the main benefits is that geographically diverse communities sharing common hardware platform for different applications reduce the cost of equipment and helps sharing knowledge.

The term usually implies that information about the hardware is easily discerned. Hardware design (i.e. mechanical drawings, schematics, bill of materials, PCB layout data, HDL source code and integrated circuit layout data), in addition to the software that drives the hardware, are all released with the FOSS approach.

3.2.1 Definition

The open-source hardware statement of principles and definition were developed by members of the OSHW board and working group along with others. These documents were originally edited on the wiki at freedomdefined.org available for the general audience. The OSHW Definition 1.0 is based on the Open Source Definition for Open Source Hardware [19]. That definition was created by Bruce Perens and the Debian developers as the Debian Free Software Guidelines.

Open Source Hardware (OSHW) is a term for tangible artifacts — machines, devices, or other physical things — whose design has been released to the public in such a way that anyone can make, modify, distribute, and use those things. This definition is intended to help provide guidelines for the development and evaluation of licenses for Open Source Hardware. Hardware is different from software in that physical resources must always be committed for the creation of physical goods. Accordingly, persons or companies producing items (“products”) under an OSHW license have an obligation to make it clear that such products are not manufactured, sold, warranted, or otherwise sanctioned by the original designer and also not to make use of any trademarks owned by the original designer.

3.2.2 Popular projects

OSHW projects have been increased significantly for the last years. Indeed, it has been rising from less than 10 projects registered in 2005 to more than 200 in 2012 with widespread successful projects for the developer’s community such as Arduino with millions of revenues every year [20].

The concept has been applied to many different areas of engineering ranging from electronic projects for creating specific application devices (e.g. sensors and actuators) to more complex systems able to execute general purpose operating systems (e.g. Linux distributions). The following list provides a selection of the most popular open-source hardware projects related with the domain of iCognos:

- The **BeagleBoard** [21] is a low-power open-source hardware single-board computer produced by Texas Instruments in association with Digi-Key. The BeagleBoard was also designed with open source software development in mind, and as a way of demonstrating the Texas Instrument's OMAP3530 system-on-a-chip. The board was developed by a small

team of engineers as an educational board that could be used in colleges around the world to teach open source hardware and open source software capabilities. It is also sold to the public under the Creative Commons share-alike license.

- The **Raspberry Pi** [22] is a credit-card-sized single-board computer developed in the UK by the Raspberry Pi Foundation with the intention of stimulating the teaching of basic computer science in schools and/or universities. The Raspberry Pi has a Broadcom BCM2835 system on a chip (SoC),[3] which includes an ARM processor and originally shipped with 256 megabytes of RAM, later upgraded to 512MB. It does not include a built-in hard disk or solid-state drive, but uses an SD card for booting and long-term storage.[14] The Foundation's goal is to offer two versions, priced at US\$ 25 and US\$ 35.
- The **Arduino** [23] is a popular open-source single-board microcontroller, descendant of the open-source Wiring platform, designed to make the process of using electronics in multidisciplinary projects more accessible. The hardware consists of a simple open hardware design for the Arduino board with an Atmel AVR processor and on-board input/output support. The software consists of a standard programming language compiler and the boot loader that runs on the board. Arduino hardware is programmed using a Wiring-based language (syntax and libraries), similar to C++ with some slight simplifications and modifications, and a Processing-based integrated development environment
- The **e-puck** [24] robot was designed by Dr. Francesco Mondada and Michael Bonani in 2006 at EPFL, the Swiss Federal Institute of Technology in Lausanne. It was intended to be a tool for university education, but is actually also used for research. To help the creation of a community inside and outside EPFL, the project is based on an open hardware concept, where all documents are distributed and submitted to a license allowing everyone to use and develop for it. Similarly, the e-puck software is fully open source, providing low level access to every electronic device and offering unlimited extension possibilities. The e-puck robot has already been used in a wide range of applications, including mobile robotics engineering, real-time programming, embedded systems, signal processing, image processing, sound and image feature extraction, human-machine interaction, inter-robot communication, collective systems, evolutionary robotics, bio-inspired robotics, etc.

iCognos is clearly aligned with the objectives of the open-source hardware community since one of the implicit objectives is to make affordable an in-home cognitive stimulation therapy which is perfectly accomplished by the OSHW. Consequently, the project is willing to take advantage of the benefits of the open-source hardware projects and community. As part of this thesis, a preliminary study will be conducted in order to select the most suitable components for developing the envisaged platform.

3.3 Wireless communication protocols

Advances in wireless technology enable an unprecedented opportunity for ubiquitous real-time context monitoring without constraining the activities of the user for different applications such as healthcare or security. Wirelessly connected miniaturized sensors and actuators spread in the user's immediate environment are able to understand, monitor and react in front of different situations. This scenario coins the concept called PAN (Personal Area Networks), in which different appliances communicate each other to exchange

information of different nature like *environment* of the user such as temperature, humidity, light, motion, interaction with objects; or user information collected directly from the user itself such as physiological signals or lifestyle [25].

Wireless communication protocols are a fast growing technology for accessing networks and services without cables to provide the flexibility and mobility. Obviously, reducing the cable restriction is one of the benefits of wireless with respect to cabled devices. Other benefits are also included such as dynamic network formation, low cost, and easy deployment. Different actors, standards and technologies have come to play to provide the developers with a multitude of wireless protocols with different purposes as it is shown in Figure 5.



Figure 5. Comparison of network wireless protocols [26]

Among all these protocols, the following list provides an overview to the most popular wireless protocols that will be evaluated for the implementation of the iCognos project:



Wi-Fi is a trademark of the Wi-Fi Alliance that may be used with certified products that belong to a class of wireless local area network (WLAN) devices based on the IEEE 802.11 standards. Wi-Fi allows local area networks (LANs) to be deployed without wires for client devices, typically reducing the costs of network deployment and expansion. Spaces where cables cannot be run, such as outdoor areas and historical buildings, can host wireless LANs.

A typical wireless router using 802.11b or 802.11g with a stock antenna might have a range of 32 m indoors and 95 m outdoors. Due to reach requirements for wireless LAN applications, power consumption is fairly high compared to some other standards.



Bluetooth is a standard and a communications protocol primarily designed for low power consumption, with a short range (power-class-dependent: 100m, 10m and 1m, but ranges vary in practice) based on low-cost transceiver microchips in each device. Bluetooth makes it possible for these devices to communicate with each other when they are in range.

Bluetooth uses a radio technology called frequency-hopping spread spectrum, which chops up the data being sent and transmits chunks of it on up to 79 frequencies. Bluetooth provides a way to connect and exchange information between devices such as mobile phones, laptops, digital cameras, or video game consoles through a secure, globally unlicensed Industrial, Scientific and Medical (ISM) 2.4 GHz short-range radio frequency bandwidth.



ZigBee is a low-cost, low-power, wireless mesh networking proprietary standard. The low cost allows the technology to be widely deployed in wireless control and monitoring applications, the low power-usage allows longer life with smaller batteries, and the mesh networking provides high reliability and larger range.

ZigBee operates in the industrial, scientific and medical (ISM) radio bands; 868 MHz in Europe, 915 MHz in the USA and Australia, and 2.4 GHz in most jurisdictions worldwide. The technology is intended to be simpler and less expensive than other WPANs such as Bluetooth.

Because ZigBee can activate (go from sleep to active mode) in 15 msec or less, the latency can be very low and devices can be very responsive — particularly compared to Bluetooth wake-up delays, which are typically around three seconds. Because ZigBees can sleep most of the time, average power consumption can be very low, resulting in long battery life.

The following table provides technical comparison of the most relevant features of the protocols described above:

| | ZigBee | Wi-Fi | Bluetooth |
|---|---|---------------|-----------------------------|
| Range | 10-100 meters | 50-100 meters | 10 – 100 meters |
| Networking Topology | Ad-hoc, peer to peer, star, or mesh | Point to hub | Ad-hoc, very small networks |
| Operating Frequency | 868 MHz (Europe) 900-928 MHz (NA), 2.4 GHz (worldwide) | 2.4 and 5 GHz | 2.4 GHz |
| Complexity (Device and application impact) | Low | High | High |

| | | | |
|--|---|--|--|
| Power Consumption (Battery option and life) | Very low (low power is a design goal) | High | Medium |
| Security | 128 AES plus application layer security | | 64 and 128 bit encryption |
| Typical Applications | Industrial control and monitoring, sensor networks, building automation, home control and automation, toys, games | Wireless LAN connectivity, broadband Internet access | Wireless connectivity between devices such as phones, PDA, laptops, headsets |

Table 3. Comparison of wireless low-range technologies [27]

The selected technology/protocol for iCognos needs to be a well-tested standard accepted for the industry, which is accomplished by the three candidates evaluated. After a deep study of the characteristics of the three candidates, iCognos selects the Bluetooth protocol given the proper trade-off between consumption and range figures since those could be considered the main triggers for the environment in which the system will be deployed.

4 System components

4.1 The Raspberry Pi

The **Raspberry Pi** is a credit-card-sized single-board computer by the Raspberry Pi Foundation with the intention of stimulating the teaching of basic computer science in schools and provides a widespread and affordable open-source hardware platform for researchers worldwide.

4.1.1 Board components

A common mistake is to think of the Raspberry Pi as a microcontroller development board like Arduino, or as a laptop replacement. In fact it is more like the exposed innards of a mobile device, with lots of maker-friendly headers for the various ports and functions. Figure 6 shows the main the parts of the board, as described below

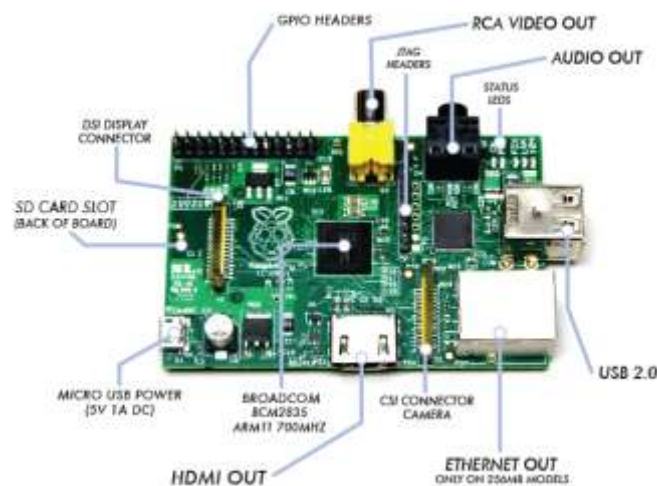


Figure 6. The Raspberry Pi

A. The Processor. The Raspberry Pi has a Broadcom BCM2835 system on a chip (SoC), which includes an ARM1176JZF-S 700 MHz processor, VideoCore IV GPU. At the heart of the Raspberry Pi is the same processor found in the iPhone 3G and the Kindle 2 showing the popularity of the ARM processors. The ARM processor is a 32 bit, 700 MHz System on a Chip, which is built on the ARM11 architecture. ARM chips come in a variety of architectures with different cores configured to provide different capabilities at different prices. The Model B has 512MB of RAM and the Model A has 256 MB. Section 4.1.2 provides further details on the ARM architecture whereas section 4.1.3 provides information on the GPU component.

B. The Secure Digital (SD) Card slot. The Raspberry Pi is not equipped with a hard drive on it; everything needs to be stored on an SD Card. It becomes then necessary a protective case

since the solder joints on the SD socket may fail if the SD card is accidentally bent making the Raspberry Pi unuseful.

C. The USB port. On the Model B there are two USB 2.0 ports, but only one on the Model A. Some of the early Raspberry Pi boards were limited in the amount of current that they could provide which might become problematic for some USB slave devices drawing up to 500mA. To avoid this issue, it could be necessary to use a powered external USB hub if any peripheral is in need of additional power supply.

D. Ethernet port. The model B has a standard RJ45 Ethernet port. The Model A does not, but can be connected to a wired network by a USB Ethernet adapter (the port on the Model B is actually an onboard USB to Ethernet adapter). WiFi connectivity via a USB dongle is another option.

E. HDMI connector. The HDMI port provides digital video and audio output. 14 different video resolutions are supported, and the HDMI signal can be converted to DVI (used by many monitors), composite (analog video signal usually carried over a yellow RCA connector), or SCART (a European standard for connecting audio-visual equipment) with external adapters.

F. Power input. Raspberry PI does not contain a power switch. The microUSB connector is used to supply power. In fact, this is not an additional USB port; it's only for powering the board. MicroUSB was selected because the connector is cheap USB power supplies are easy to find.

G. Status LEDs. The Pi has five indicator LEDs that provide visual feedback.

| | | |
|------------|--------|---|
| ACT | Green | Lights when the SD card is accessed |
| PWR | Red | Hooked up to 3.3V power |
| FDX | Green | Turned on when network adapted is full duplex |
| LNK | Green | Network activity light |
| 100 | Yellow | Turned on when network connection is 100Mbps |

4.1.2 Built-in ARM processor

Developed by Acorn Computer back in the late 1980s, the ARM architecture is relatively uncommon sight in the desktop world. Where it excels, however, is in mobile devices: the majority of the mobile phones have at least one ARM-based processing core hidden away inside. Its combination of a simple reduced instruction set (RISC) architecture and low power draw make it perfect choice over desktop chips with high power demands and complex instruction set (CISC) architectures [28].

The ARM-based BCM2835 is the secret of how the Raspberry Pi is able to operate on just 5V 1A power supply provided by the onboard micro-USB port. It is also the reason why you will not find any heat-sinks on the device: the chip's low power draw directly translates into very little waste heat, event during complicated processing tasks. It does, however, mean that the

Raspberry Pi is not compatible with traditional PC software for desktops and laptops is built with the x86 instruction set architecture in mind, as found in processors from AMD or Intel. It requires cross-compiling as it is described in section 5.3.1.

The BCM2835 uses a generation of ARM's processor design known as ARM11, which in turn is designed around a version of the instruction set architecture known as ARMv6. This is worth remembering since ARMv6 is lightweight and powerful architecture. The following figure shows the different modules included in the ARM processor of the R-PI, the ARM1176JZF-S.

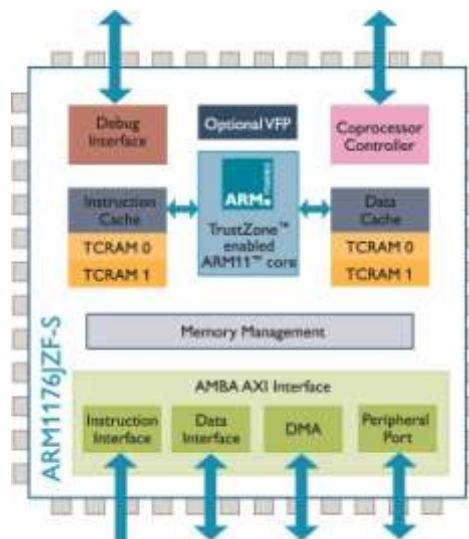


Figure 7. The ARM1176JZF-S architectural block

4.1.3 GPU : Graphics Processing Unit

The Raspberry Pi has a clear vocation towards providing enriched Graphical Interfaces and for this it becomes necessary the use of a GPU to accelerate the treatment of graphics. The Raspberry Pi contains a Broadcom VideoCore IV GPU providing API following the standards OpenGL ES 1.1, OpenGL ES 2.0, hardware-accelerated OpenVG 1.1, Open EGL. There are 24 GFLOPS of general purpose compute and a bunch of texture filtering and DMA infrastructure. Currently C header files and libraries for many of the Broadcom APIs are located in the path `/opt/vc/include` and `/opt/vc/lib` respectively, or available from GitHub within the same directory structure. Some documentation is contained within comments in the header files, however it is severely lacking and difficult to understand in a general sense for people wanting to experiment with the device due to it being proprietary [29].

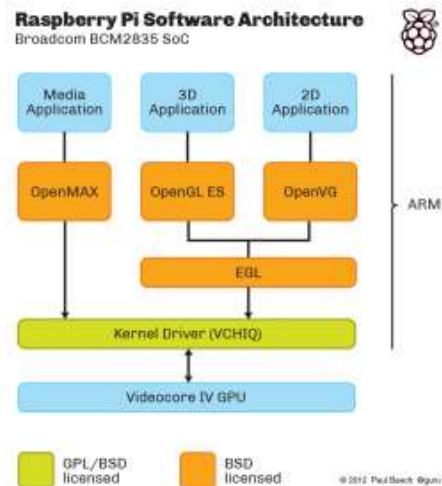


Figure 8. VideoCore IV GPU architecture

Initially, the drivers for accessing the VideoCore GPU were proprietary and not accessible to general public partially breaking the Open-source hardware paradigm supported by the R-Pi project. The Foundation has made strong efforts to solve this issue to create an open-source version of these drivers. Eventually, on October 2012, all of the VideoCore driver code which runs on the ARM is available under a FOSS license. It does actually mean that the BCM2835 used in the Raspberry Pi is the first ARM-based multimedia SoC with fully-functional, vendor-provided (as opposed to partial, reverse engineered) fully open-source drivers, and that Broadcom is the first vendor to open their mobile GPU drivers up in this way.

4.1.4 Firmware for boot

When the R-Pi is powered up, the first thing that starts working is the GPU core. The GPU core is the main part responsible for the first booting steps of the Pi. The GPU core uses some internal firmware to start accessing the SD card slot. The SD Card should be formatted in fat32 (at least for the first partition) and should contain a bunch of files that are used from the GPU core to boot the system [30] which are described below:

(1) bootcode.bin is the first file loaded from the SD card by the GPU core. This is a GPU binary that is loaded into the L2 cache and this file has the duty to enable the SDRAM memory of the system (disabled until this time) and load the next bootloader stage.

(2) loader.bin: This file contains code able to load elf binaries. After this stage, the system is able to load the **start.elf** file, the last and most interesting stage of the boot process.

(3) start.elf: This file is the last bootloader stage (still a GPU binary). It loads a file called **kernel.img** in memory at the SDRAM address 0x8000. The **start.elf** loader supports some configuration options. If a file called *config.txt* is present in the SD root then the *start.elf* file loads it configuring the system according to the specified options there. In short the options configurable vary from the video resolution to the frequency of the ARM core. After all of this has been done the **start.elf** loader resets the ARM core and the ARM processor start execution from the loaded kernel starting address.

(4) Kernel.img file is an ARM executable. This is the part we are interested in building. In the default package provided from the Pi Foundation this is an ARM linux kernel, but if you build your own program you can substitute the kernel with your own file and boot it up.

All these files stored in the boot partition (kernel.img excluded) represent the **R-Pi firmware**, which is downloadable from [<https://github.com/raspberrypi/firmware>]. Finally, there are also different versions of the start.elf file (named armXXX_start.elf), the difference between these corresponds to the amount of SDRAM memory allocated for the GPU and CPU respectively.

4.1.5 GNU/Linux Distributions

The operating system is a software program which is executed at bootstart consisting of, in simple words, a software layer to access to hardware functions and common service for computer programs (e.g. process scheduling, memory management). For instance, an application that accesses the Internet makes use of the operating system's functions to do so. The operating system abstracts developers enabling them to not be in need of acquiring the knowledge for every single component involved in an application. In the other example, the Internet access related hardware.

The majority of desktop and laptop computers available today run either Microsoft Windows or Apple OS X. Both platforms are closed source, created in a secretive environment using proprietary techniques. The users are able to obtain the finished software, but never see how it is made, even when purchasing the full version. The Raspberry Pi, by contrast, uses an operating system which is based on GNU/Linux. GNU/Linux is the perfect choice for R-PI since it is free software and open source keeping the prices low and customizable for the end-users. The Raspberry Pi Foundation provides several Linux ARM distributions for download from which the principal distributions used by the board are Raspbian and an adapted Debian wheezy described lately in this section. However, there are still other distributions which are taking their community of users behind such as a flavor of Arch Linux for R-PI or the OpenELEC distribution. There are even a few non-Linux OS options available out there.

4.1.5.1 *Debian wheezy*

Debian is an operating system composed of software packages released as free and open source software primarily under the GNU General Public License along with other free software licenses. Debian GNU/Linux, which includes the GNU OS tools and Linux kernel, is a popular and influential Linux distribution. Debian is distributed with access to repositories containing thousands of software packages ready for installation and use. Latest version of Debian OS released is 7.0 also called "wheezy" published on June 2012. Debian was ranked second only to Ubuntu (which is derived from Debian) for Most Used Linux Distribution for both personal and organizational use in a 2007 survey by SurveyMonkey.com [31].

Most popular Debian binaries are prepared for the x86 architecture mentioned in the previous section which is not compatible with the R-PI architecture, based on ARM. Nevertheless, the source code for this distribution is open source and users are allowed to customised the software according to their needs. As such, the R-PI Foundation prepared a Debian binary package ready-to-use with the R-PI.



Figure 9. The Debian logo

4.1.5.2 Raspbian

The official Debian Squeeze image issued by the Raspberry Pi foundation uses "soft float" settings. The foundation found it necessary to use the existing Debian port for less capable ARM devices due to time and resource constraints during development of the Raspberry Pi. Therefore, it does not make use of the Pi's processor's floating point hardware - reducing the Pi's performance during floating point intensive applications - or the advanced instructions of the ARMv6 CPU. Raspbian is an unofficial port of Debian Wheezy armhf with compilation settings adjusted to produce optimized "hard float abi" code that will run on the Raspberry Pi. This provides significantly faster performance for applications that make heavy use of floating point arithmetic operations. All other applications will also gain some performance through the use of advanced instructions of the ARMv6 CPU in Raspberry Pi.

This difference in performance comes from the hard float feature of the ARM processors. The soft-float applications passes floating point parameters in integer registers while the hard-float using floating point unit built-in. The two methods are not compatible because they use different registers. It is possible to use hardware floating point with the soft-float but doing so means that whenever a floating point value is passed to or returned from a function it must be transferred to an integer register incurring a performance penalty.



Figure 10. The Raspbian logo

Raspbian is a free operating system based on Debian (in turn, based on GNU/Linux) optimized for the Raspberry Pi hardware with the hard-float feature. An operating system is the set of basic programs and utilities that make your Raspberry Pi run. However, Raspbian provides more than a pure OS: it comes with over 35,000 packages, pre-compiled software bundled in a nice format for easy installation on your Raspberry Pi. The initial build was completed in June of 2012. However, Raspbian is still under active development with an emphasis on improving the stability and performance of as many Debian packages as possible.

4.2 The ePuck

In this chapter, the robot and its design is introduced separating hardware and software, going through for what need it was conceived, it's mechanical design, it's components, taking in account sensors as well as actuators and finally a study is made on how to program it and how the firmware runs on the microcontroller.

4.2.1 Origin of e-Puck

Motivation for a new robot arose from an absence of a very small educational robot, which is sufficiently efficient, and can be used for education in many research areas. E-Puck was developed in summer of 2004 at the École Polytechnique Fédérale de Lausanne (EPFL) as an open tool. E-Pucks designers used open software and hardware development model. E-Puck can be applied in automatic control, signal processing or distributed intelligent system research. Its structure is robust and simple but still, the designers opted for the use of cheap components and mass production manufacturing techniques in order to keep the price low.

Since 2006, the first generation of e-Pucks has been replaced by the second stable generation and so far, more than 2000 robots have been manufactured. There are several distributors for Switzerland and America, and one for Japan:

- <http://www.roadnarrows.com/robotics/>
- <http://www.aai.jp/>
- <http://www.k-team.com/>
- <http://www.cyberbotics.com/products/robots/e-puck.html>
- <http://www.gctronic.com/>
- <http://www.aai.ca/robots/e-puck.html>

The price of e-Puck is between 450 and 550 Euros.

4.2.2 Hardware

4.2.2.1 Mechanical Design

The mechanical design of E-Puck is shown in Figure 11. It consists of a rounded transparent body, a battery, two stepper motors, wheels, a printed circuit board, plastic ring of LEDs, a camera and a default extension board. There are extensions like floor sensors, a rotating scanner or a turret with linear cameras, which can scale up E-Puck's sensors. If we are talking about E-Puck in this thesis, it is meant E-Puck with the default extension plus the ground sensors.

The battery is placed in the bottom and can be easily extracted and recharged separately. Two stepper motors with wheels are screwed to a plastic body and are located on axis of e-Puck in order to allow robot turn around on place. Wheels have the diameter of 41 mm and the perch is 53 mm. The printed circuit board is fixed to the top of the body and a ring of LEDs' is mounted around the board. A camera is placed on the front side of robot lying on the axis between the wheels. The extension board covers the main printed circuit board.

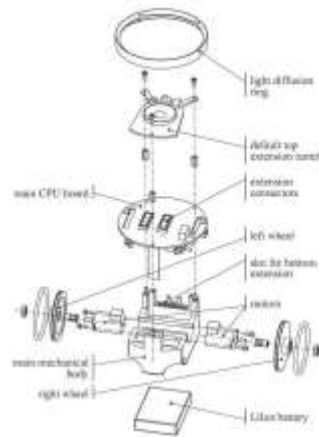


Figure 11: E-Pucks mechanical design

4.2.2.2 Sensors, actuators and heart of the robot

Sensors and actuators determine the possible robot usage. Luckily, E-Puck has many sensors of different kinds and is equipped with typical actuators. The following paragraphs shortly describe e-Puck's sensors and actuators.

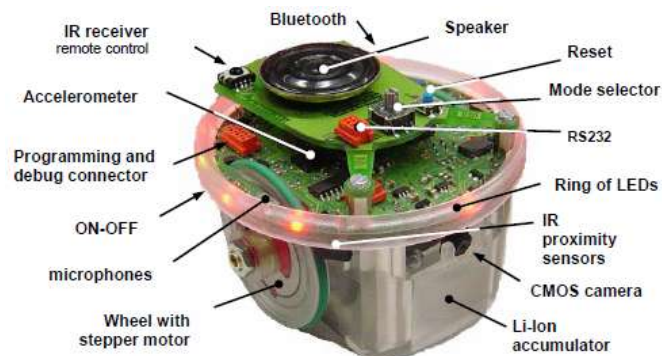


Figure 12: E-Pucks components

The actuators consist of motors, 8 red LEDs on perimeter, four green body LEDs, which are turned on/off together, a front LED and a speaker. Stepper motors are a great advantage of E-Puck, because the motion of wheels can be split into small steps. One wheel revolution corresponds to 1000 steps of motor. The diameter of the wheel is 41 mm. If the wheel makes one revolution, the wheel goes 128.8 mm. In conclusion, a thousand of steps matches 128.8 mm of linear movement and one step is 0.1288 mm. Motors are equipped with encoders and the motors are very accurate, which in combination with simple odometry really helps in localization tasks. A nice feature of encoders is that their value can be set at any time. The maximum speed of the motors is one revolution per second in both directions.

All e-Puck LEDs can be used for debugging or for making robot visible to other devices. Furthermore, the front LED is usually used to illuminate the terrain in front of e-Puck. Each LED can be turned off, turned on or set to an inverse state.

Three axis accelerometers are placed inside the robots body. In the rest position, the accelerometers measure the slant of e-Puck. They can also measure acceleration of e-Puck and for example detect a collision or a falling state of e-Puck.

The other available elements present on the E-Puck are the eight infrared sensors located around its body, a speaker, three microphones distributed in a triangle and a 640x480 camera. Special features that this extra components provide, would be obstacle avoidance with the IR sensors, and audio source detection via triangulation of the three microphones. About the camera mention that since the E-Puck only disposes of 8 kB of RAM, the actual size of the picture has to be reduced in order to save the picture in memory.

The processor (a dsPIC 30F6014) is the heart of e-Puck and runs at 60 MHz, which correspond to 15 MIPS. It has C oriented instructions and supports compiling from GNU compilers. Apart from standard 16 bits core unit, Digital Signal Processor brings very high performance for computation FFT or other signal processing. Programs can be downloaded to flash memory with 144 kB and are loaded to RAM memory according to the selector position.

Communication with other devices is provided by IR port, Bluetooth and RS232 serial interface. Both, Bluetooth and RS232, can be used to download programs to E-Puck's flash memory. In addition, Bluetooth can be used to communicate with other E-Pucks or with a computer using BtCom library available at www.e-puck.org. The counter part of BtCom on computer is connected to the serial port.

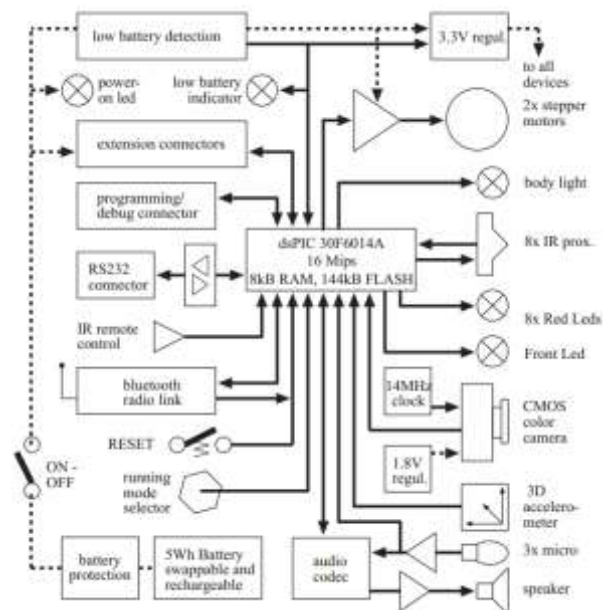


Figure 13: The outline of the electronic of the E-Puck

4.2.2.3 Ground Sensor extension

The e-puck Ground Sensors extension was designed and developed at the Laboratory of Intelligent Systems (LIS) at École Polytechnique Fédérale de Lausanne (EPFL) in Lausanne, Switzerland by Adam Klaptocz and Jean-Christophe Zufferey under supervision of Professor Dario Floreano.

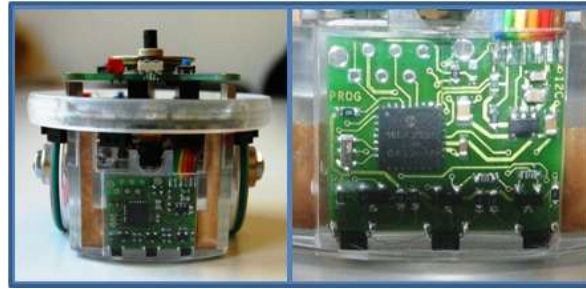


Figure 14: Ground sensors in detail

The sensor is comprised of three active infrared (IR) proximity sensors placed in the front of the E-Puck pointing directly at the ground. These sensor elements are mounted on a small printed circuit board (PCB) which is placed in the vertical slot at the front of the e-puck's plastic base. The PCB also includes a microcontroller that continually samples the IR sensor elements. The values obtained by the sensor elements and formatted by the microcontroller can then be read by the e-puck through the I2C serial interface.

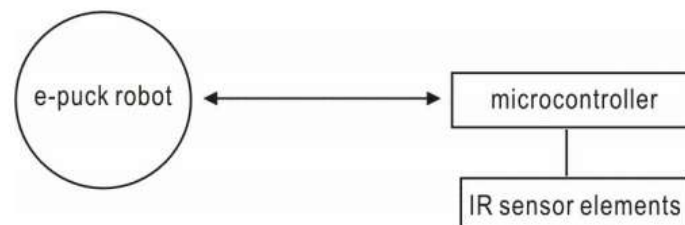


Figure 15: Connection between e-puck and the Ground Sensors extension board

One IR sensor element consists of an IR-emitting diode (LED) and a phototransistor. The infrared LED is used to emit a constant amount of infrared beam, whereas the phototransistor simply detects the amount of signal reflected by a surface. A white surface reflects much more infrared signal than a black surface, as will a surface closer to the sensor element, as opposed to one that is farther away.

4.2.3 Software

The software, also called firmware, running in the E-Puck is devoted to control the operational procedures that provide the cognitive stimulation capabilities. In order to obtain this, several tools are needed to generate this firmware, that is, the software executed by the E-Puck. These tools include the environment in which the little robots firmware is developed and the libraries given by the developers of the E-Puck to facilitate the interaction with its hardware.

4.2.3.1 Libraries

The E-Pucks manufacturer provides the developers with the basic libraries to access the sensors and actuators as well as the Bluetooth to be able to communicate with the PC. On the website, the download looks like:

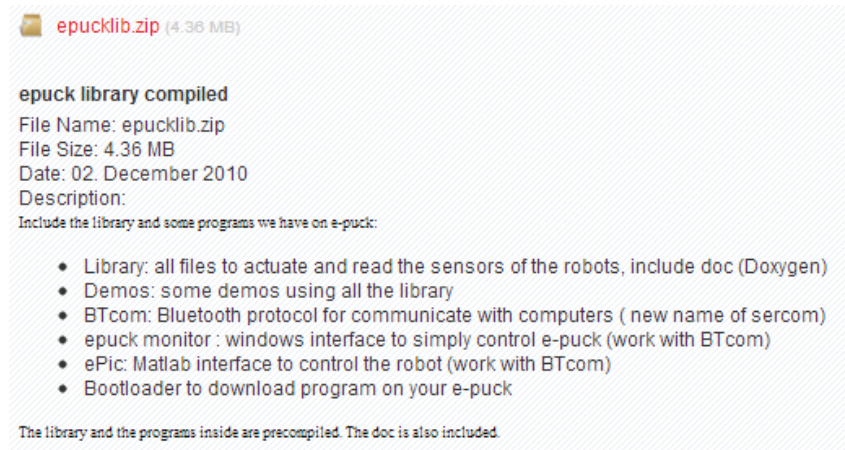


Figure 16: Information about the E-Puck library

As it shows on the above figure, the library contains the files necessary to abstract the developer of the low level programming of the hardware. It contains code written in C and in assembler, which translate the complex and difficult programming of registers and ports into easy callable C functions. This topic will be further discussed in the E-Puck implementation section.

4.3 WiiMote

The Wii Remote is the Wii's main input device for the Nintendo's Wii videoconsole. It is a wireless device, using standard Bluetooth technology to communicate with the Wii. It is built around a Broadcom BCM2042 bluetooth System-on-a-chip, and contains multiple peripherals that provide data to it, as well as an expansion port for external add-ons. The Wii Remote uses (and, at times, abuses) the standard Bluetooth HID protocol to communicate with the host, which is directly based upon the USB HID standard. As such, it will appear as a standard input device to any Bluetooth host. However, the Wii Remote does not make use of the standard data types and HID descriptor, and only describes its report format length, leaving the actual contents undefined, which make it useless with standard HID drivers (but some Wiimote Drivers exist). The Wii Remote actually uses a fairly complex set of operations, transmitted through HID Output reports, and returns a number of different data packets through its Input reports, which contain the data from its peripherals.



Figure 17: Nintendo's WiiMote

The WiiMote incorporates the following features :

- Bluetooth Communication
- Core Buttons
- Accelerometers
- IR Camera
- Power Button
- Speaker
- Player LEDs

4.4 Qt : Graphical User Interfaces

Qt [32] is a cross-platform application framework that is widely used for developing application software with a graphical user interface (GUI) (in which cases Qt is classified as a widget toolkit), and also used for developing non-GUI programs such as command-line tools and consoles for servers. Some of the well known applications developed with Qt are KDE, Opera, Google Earth, Skype, VLC, Maya or Mathematica. Qt was first publicly released on May 1995. It is dual licensed. It can be used for creating open source applications, with an open-source license, as well as commercial ones by using a specific license. Indeed, Qt is available under a commercial license, GPL v3 and LGPL v2. All editions support many compilers, including the GCC C++ compiler and the Visual Studio suite.

Qt toolkit is a very powerful toolkit. It is well established in the open source community. Thousands of open source developers use Qt all over the world. Qt uses standard C++ but makes extensive use of a special code generator (called the Meta Object Compiler, or moc) together with several macros to enrich the language. Qt can also be used in several other programming languages via language bindings. It runs on the major desktop platforms and some of the mobile platforms. It has extensive internationalization support. Non-GUI features include SQL database access, XML parsing, thread management, network support, and a unified cross-platform application programming interface (API) for file handling.



Figure 18: Qt framework logo

The following list provides an overview of the most important modules available in the Qt framework:

QtCore – contains core non-GUI classes, including the event loop and Qt's signal and slot mechanism, platform independent abstractions for Unicode, threads, mapped files, shared memory, regular expressions, and user and application settings

QtGui – contains most GUI classes; including many table, tree and list classes based on model–view–controller design pattern; also provides sophisticated 2D canvas widget able to store thousands of items including ordinary widgets

QtNetwork – contains classes for writing UDP and TCP clients and servers; implementing FTP and HTTP clients, supporting DNS lookups; network events are integrated with the event loop making it very easy to develop networked applications

QtOpenGL/QtOpenVG – contains classes that enable the use of OpenGL/OpenVG in rendering 3D graphics

QtScript/QtScriptTools – an ECMAScript-based scripting engine

QtSql – contains classes that integrate with open-source and proprietary SQL databases. It includes editable data models for database tables that can be used with GUI classes. It also includes an implementation of SQLite

QtXml – implements SAX and DOM interfaces to Qt's XML parser

QtXmlPatterns – provides support for XPath, XQuery, XSLT and XML Schema validation

Qt Declarative module is a declarative framework for building fluid user interfaces in QML

4.4.1 QML : Qt Meta language

Qt Quick is a collection of technologies that are designed to help developers create the kind of intuitive, modern, and fluid user interfaces that are increasingly used on mobile phones, media players, set-top boxes, and other portable devices. Qt Quick consists of a rich set of user interface elements, a declarative language for describing user interfaces, and a language runtime. A collection of C++ APIs is used to integrate these high level features with classic Qt applications.

Qt Quick applications are defined with the QML (Qt Meta Language or Qt Modeling Language), a JavaScript-based, declarative language for designing user interface-centric applications. QML elements are a sophisticated set of building blocks, graphical (e.g., rectangle, image) and behavioral (e.g., state, transition, animation). These elements can be combined to build components ranging in complexity from simple buttons and sliders, to complete internet-enabled programs.

QML element capabilities might be augmented by standard JavaScript both inline and via included “.js” files. Elements can also be seamlessly integrated and extended by C++ components using the Qt framework.

5 Implementation

This section is aimed at exposing the developments specifically carried out to create the iCognos project. It is remarkable to say that iCognos combines of different elements from both disciplines hardware and software working in concert and, therefore, knowledge in both areas was required for developing the system.

5.1 System description

iCognos provides a novel concept in cognitive stimulation therapies by using SmartObjects to improve the motivation. The system is composed of a telerobotics platform, called CSTS (Cognitive Stimulation Telerobotics System) controlled by the CSCS (Cognitive Stimulation Control System). The following figure depicts all the elements and their relationships taking part in the iCognos project together with their interactions.

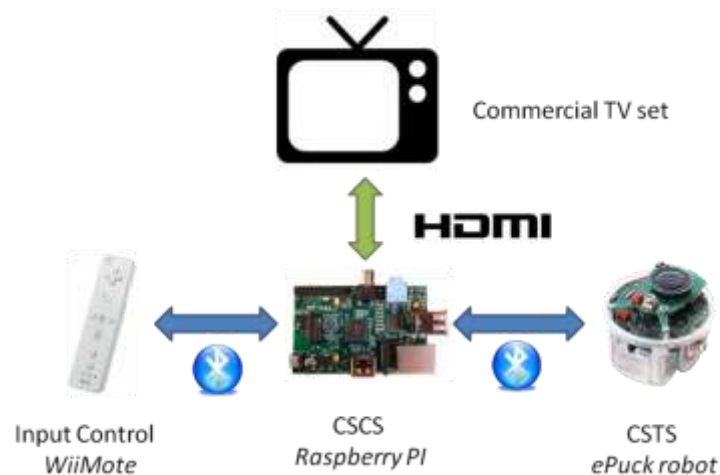


Figure 19: iCognos system description

CSTS (Cognitive Stimulation Telerobotics System) is the telerobotics subsystem, which can be understood as a SmartObject to provide the physical interaction to the user enhancing his/her motivation. See Section 3 for further details on SmartObjects.

CSCS (Cognitive Stimulation Control System) is the subsystem used to propose the task to the user guiding him/her through the different stages of the task and finally giving a result of the performed exercise. Therefore, the control of the CSTS is one of the main responsibilities of the CSCS. As part of the CSCS, **Input Control** is the main interaction point of the user with the system. Given the targeted users of iCognos, this device takes a dramatic importance to ease the use of the system.

iCognos needs to be understood as a platform for validating the usage of SmartObjects in cognitive stimulation therapies. In order to do so, a cognitive game was proposed to the user which called "FollowMyLine". The next picture depicts the "FollowMyLine" task proposed to the user which development is as follows: the user is requested to make the CSTS to follow a

line by means of controlling it with the Input Control (WiiMote) while selecting the specific commands through a dedicated interface, the Hex-O-Select (HoS). This interface consist of six 6 hexagons distributed in a circle with a central arrow spinning. When the arrow selects, the wished command the user has to press the A button in the WiiMote. See section 5.3.3.1 for further details on the HoS interface.



Figure 20. iCognos system

5.1.1 iCognos featured open-source projects

iCognos relied on open-source (OSS) and open-source hardware (OSHW) projects to successfully fulfill the objectives of the project. Both elements of iCognos, the CSCS and the CSTS, make use of projects following these principles.

Hardware development

CSTS takes advantage of the *ePuck* robot to provide an affordable system with which the user interacts, while the CSCS makes use of the *Raspberry Pi* project to provide an ICT platform for the guidance and support during the development of the cognitive stimulation therapy.

Software development

The CSCS contains a software application developed making the good of different open source software projects. Such projects are listed below with a short summary of their role in whole ecosystem.

- **Raspbian** is an open source project which created a GNU/Linux based distribution for the R-PI taking advantage of the different hardware feature offered to improve the global performance of the system (e.g. hard-floating point unit). See section 4.1.5.2 for further details.
- **Qt Project** is a software framework developed in C++ mainly used to facilitate the development of UI. The iCognos interface, and in particular the HoS, has been developed using this framework.
- **Cwiid** is an open-source software package which enables the connection with the Nintendo's WiiMote by means of a Bluetooth Link. The public API provided facilitates the connection with functions to open, close, send and receive messages from the WiiMote. The connection to the Input Interface (i.e. WiiMote) used this OSS library.

- **Bluez** is the Bluetooth stack for Linux. Its goal is to make an implementation of the Bluetooth wireless standards specifications for Linux. As of 2006, the BlueZ stack supports all core Bluetooth protocols and layers. It was initially developed by Qualcomm, and is available for Linux kernel versions 2.4.6 and up. The connection to the CSTS (i.e. the ePuck) used this library which is also implicitly used by the Cwiid project.

The CSTS firmware also is based on software libraries developed by the ePuck project. They are published under open-source licenses, abstracting the designer of the specific operation of the hardware and letting him to concentrate in the added value (i.e. cognitive stimulation). It is remarkable to say that these libraries are to be used within the firmware development for the microcontroller of the ePuck. In particular, the ePuck libraries provide the following modules:

Analog&Digital Module includes routines to interact with the A/D outputs

Bluetooth Module holds the communication with the Bluetooth transceiver via UART for robot-to-robot communication

Camera Module incorporates the functions to interact with light sensor (i.e. camera)

Codec Module is able to control the audio codec of the ePuck which is connected to the speaker and microphone

FFT Module includes the routines to perform on-board Fast Fourier Transform for processing the images coming from the image sensor

I2C Module library is to facilitate the communication with other elements of the device (e.g. camera)

Motor_led Module controls the capacities to interact with the motors, accessed directly without any additional driver, and the on-board LEDs

UART Module establishes the serial communication with on-board devices as for example the Bluetooth transceiver.

5.2 Communication interfaces

This section describes the different interfaces used in iCognos project: the communication between the ePuck and the R-PI, named I1, and the R-PI and the WiiMote, named I2. Both links are based on the wireless communication protocol Bluetooth by means of Bluez. Bluez is the Bluetooth stack for Linux, that is, the software used for communicating using this technology, making available an implementation of the Bluetooth stack for GNU/Linux systems. The bluez package also incorporates other software for diagnosis and monitoring such as the bluez-utils and bluez-firmware such as dfutool to interrogate the Bluetooth adapter chipset and determine whether its firmware can be upgraded.

5.2.1 Bluetooth

Bluetooth is a wireless technology for exchanging data over short distances wirelessly, using short-wavelength radio transmissions in the ISM band from 2400–2480 MHz, with high levels of security. Created by Ericsson in 1994, it was originally conceived as a wireless alternative to RS-232 data cables.

In past years, Bluetooth made a substantial breakthrough due to the scaled implementation within small devices as well as laptops. Indeed, the Bluetooth technology has been developed for low cost appliances, which increase the number of potential applications. However, many other wireless competitors have appeared in the last years such as Zigbee or Z-Wave competing mainly in consumption. For this, Bluetooth is now developing a new version, version 4.0, which reduces dramatically the overall consumption and autonomy of the appliances. This latest version of Bluetooth will become a towering competitor, as its specifications seem to become better than main competitors.

Table 4, recites the technical specifications of the Bluetooth protocol.

| Standard | Bluetooth |
|-----------------------|------------|
| IEEE spec. | 802.15.1 |
| Frequency band | 2.4 GHz |
| Max signal rate | 1 Mb/s |
| Nominal range | 10 m |
| Nominal TX power | 0 - 10 dBm |
| Number of RF channels | 79 |

Table 4: Technical specifications of the Bluetooth protocol

The following sub-sections provide a brief description of the protocol implemented on top of the Bluetooth link, for the interfaces I1 and I2.

5.2.2 Protocol for I1: ePuck – Raspberry Pi

All the messages going through the wireless communication have the same format, an initial header that determines the action that needs to be carried out, the necessary parameters to complete the action or the information that the message carries, and are finalised with a '\r\n' to indicate the end of the message. To separate the command header and the different parameters, the comma sign was employed.

On the following tables, the specific protocol is show for each command including header and the necessary parameters for the communication between the ePuck and R-PI.

| | | | |
|--------------------|---|-------------------|-----------------|
| Id | 1 | Action | Reset the robot |
| Header | R | Parameters | No parameters |
| Explanation | This command is used to reset the robot, and leave him in a known state | | |
| Example | R\r\n | | |

| | | | |
|--------------------|--|-------------------|---------------------------|
| Id | 2 | Action | Send robot to destination |
| Header | T | Parameters | Movements to do |
| Explanation | It sends a series of commands that the robot has to do to | | |
| Example | T,100,190,150,w90,50\r\n <ul style="list-style-type: none"> • <i>move forward 100mm</i> • <i>turn left 90°</i> • <i>move forward 150mm</i> • <i>turn right 90°</i> | | |

| | |
|--|--|
| | <ul style="list-style-type: none"> • <i>move forward 50mm</i> |
|--|--|

| | | | |
|--------------------|--|-------------------|----------------------------|
| Id | 3 | Action | Move the robot |
| Header | Z | Parameters | Right and left wheel speed |
| Explanation | This command sends the exact speed (positive and negative) of each wheel. It is mainly used for turning left/right the robot | | |
| Example | Z,200,-200\r\n <i>Spin the robot at a velocity of 200</i> | | |

| | | | |
|--------------------|---|-------------------|--|
| Id | 4 | Action | Controls the 8 LEDs |
| Header | W | Parameters | Led1, led2, led3, led4, led5, led6, led7, led8 |
| Explanation | This command sends for each led around the E-Puck it if has to be turned on or off. | | |
| Example | W,0,0,1,1,1,0,1\r\n <i>Turn on the leds 3, 4, 5, 7 and 8; and turn off the leds 1, 2 and 6</i> | | |

Table 5: I1: CPCS-CSTS protocol description

5.2.3 Protocol for I2: Raspberry Pi – WiiMote

The WiiMote communicates with the host via standard Bluetooth protocol. Holding down the 1 and 2 button continuously will force the WiiMote to stay in discoverable mode without turning off. When in discoverable mode, a number of the LEDs will blink. During device inquiry performed by the host, this one will find all discoverable nearby WiiMote. Finished this stage, the host can establish a Bluetooth baseband connection to the WiiMote and the HID channels can be opened and used for reading and writing reports from/to the WiiMote.



Figure 21: WiiMote pairing procedure

So as to facilitate the interface with the WiiMote, iCognos makes use of the open-source library Cwiid, a collection of Linux tools written in C for interfacing Wiimote relying on the bluez library. The Cwiid package contains four different modules: (1) a communication library (libcwiid), (2) an event mouse joystick driver with plugin architecture (wminput), (3) a GUI control panel and test application (wiigui) and (4) a test/demo application (wiidemo). The Cwiid can be seen as an abstraction layer of the specific exchange information performed between the R-PI and the WiiMote. Cwiid and specifically libcwiid, exposes a

public API to interact with the device, which main functions are described in the following list.

| |
|--|
| <code>cwiid wiimote_t *cwiid_open(bdaddr_t *bdaddr, int flags);</code> |
| Establish a connection with a wiimote. |
| <code>int cwiid_close(cwiid_wiimote_t *wiimote);</code> |
| Disconnect from wiimote. |
| <code>int cwiid_get_id(cwiid_wiimote_t *wiimote);</code> |
| Accessor function for the wiimote id, which is guaranteed to be unique among all wiimotes in a single process. |
| <code>int cwiid_enable(cwiid_wiimote_t *wiimote, int flags);</code> |
| Option flags and their meanings are as follows: CWIID_FLAG_MSG_IFC: Enable the message based interfaces (message callback and <code>cwiid_get_mesg</code>). CWIID_FLAG_CONTINUOUS: Enable continuous wiimote reports CWIID_FLAG_REPEAT_BTN: Deliver a button message for each button value received, even if it hasn't changed. CWIID_FLAG_NONBLOCK: Causes <code>cwiid_get_mesg</code> to fail instead of block if no messages are ready. |
| <code>int cwiid_disable(cwiid_wiimote_t *wiimote, int flags);</code> |
| Disables option flags enabled by <code>cwiid_connect</code> or <code>cwiid_enable</code> . |
| <code>int cwiid_set_mesg_callback(cwiid_wiimote_t *wiimote, cwiid_mesg_callback_t *callback);</code> |
| Sets the message callback function, which is called when a set of messages are received from the wiimote (each message in a set was generated from the same packet and therefore arrived simultaneously). Note that option flag <code>CWIID_FLAG_MSG_IFC</code> must be enabled for the callback to be called. Passing NULL for the callback cancels the current callback, if there is one. |
| <code>int cwiid_get_mesg(cwiid_wiimote_t *wiimote, int *mesg_count, union cwiid_mesg *mesg[]);</code> |
| Retrieve the oldest undelivered message set. Messages are delivered either through a callback, or <code>cwiid_get_mesg</code> . |
| <code>int cwiid_get_state(cwiid_wiimote_t *wiimote, struct cwiid_state *state);</code> |
| Retrieve the current state of the wiimote. |
| <code>int cwiid_command(cwiid_wiimote_t *wiimote, enum cwiid_command command, int flags);</code> |
| Issue command to wiimote. Available commands and their associated flags are as follows: CWIID_CMD_STATUS Request a status message (delivered to the message callback) (flags ignored) CWIID_CMD_LED Set the LED state. The following flags may be bitwise ORed: <ul style="list-style-type: none"> • CWIID_LED1_ON • CWIID_LED2_ON • CWIID_LED3_ON • CWIID_LED4_ON CWIID_CMD_RUMBLE Set the Rumble state. Set flags to 0 for off, anything else for on. CWIID_CMD_RPT_MODE Set the reporting mode of the wiimote, which determines what wiimote peripherals are enabled, and what data is received by the host. The following flags may be bitwise ORed (Note that it may not be assumed that each flag is a single bit - specifically, <code>CWIID_RPT_EXT = CWIID_RPT_NUNCHUK CWIID_RPT_CLASSIC</code>): <ul style="list-style-type: none"> • CWIID_RPT_STATUS • CWIID_RPT_BTN • CWIID_RPT_ACC • CWIID_RPT_IR • CWIID_RPT_NUNCHUK • CWIID_RPT_CLASSIC • CWIID_RPT_EXT |

Table 6: Cwiid API for communication with WiiMote

5.3 Cognitive Stimulation Control System

This section covers the description of the Cognitive Stimulation Control System (CSCS) used in iCognos responsible for guiding the user through the requested task and providing visual feedback through the UI and maintaining the communication with all the other hardware platforms. The CSCS is implemented with the OSHW project Raspberry Pi. Further details on this platform, can be found in section 4.14.2 .

5.3.1 Configuration of development environment

The code development might be performed within the R-PI itself but the limited resources in terms of computational power and memory will result in longer time development. The alternative to this is to develop the code in development station which output is ultimately executed by the R-PI. The following picture depicts the ecosystem used for the development of the application with this approach: a *Development Station* to compile the code resident in a network shared folder (e.g. CIFS [33] Network Folder) accessible for the *Raspberry Pi* where the code is run controlling the execution with a SSH console.

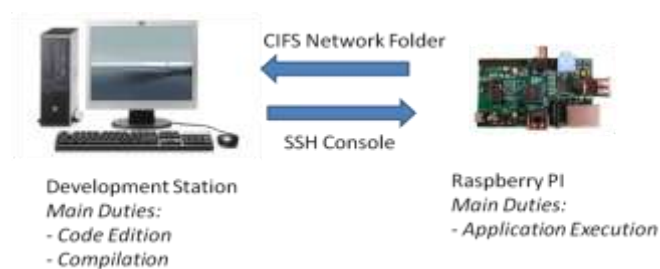


Figure 22: Configuration of the development ecosystem

R-PI includes a CPU with an ARM architecture while the personal computers usually are based on a x86 architecture. The consequence of this difference is that the binary code created after compilation on a x86 machine is not executable on the R-PI neither vice-versa. It becomes, therefore necessary to develop the code under a cross-compiling scheme. A cross-compiler is a software program (i.e. a compiler) capable of creating binary code for a platform other than itself. Cross compiler tools are used to generate executables for embedded system or multiple platforms like, for instance, the R-PI. It is used to compile for a platform upon which it is not feasible to do the compiling, like microcontrollers that don't support an operating system. It has become more common to use this tool for paravirtualization where a system may have one or more platforms in use.

QT Cross-compilation

iCognos has used the Qt framework for UI development which gives visual feedback to the user on how s/he is performing the task. The Qt framework is configured to make use of the

R- GPU via an OpenGL API enabling to use the GPU as painting engine. To do so, we have prepared cross-compiled version the Qt framework to obtain a version of Qt libraries to be used in an ARM architecture taking the GPU as painting engine while the Qt code being developed in the *Development station*.

The following figure summarises the different software layers taking part in the Qt framework using GPU as painting engine, that is, during the execution of the binary code. In the figure, the application corresponds to iCognos which uses the Qt framework for generating the GUI. Qt libraries rely on the the OpenGL-compatible API to access the Acceleration Hardware, in our case the GPU. This approach improves the graphical performance of the whole system enlarging the possibilities of the system and improving ultimately the user-experience.

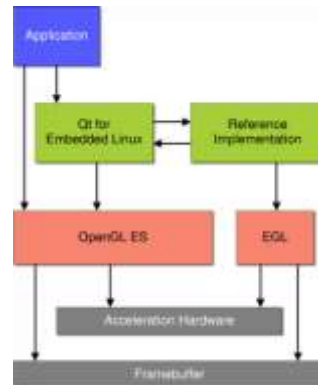


Figure 23: Qt Compilation with Acceleration HW (GPU)

To cross-compile the Qt framework with the scheme above mentioned, the following steps were made:

1. Establish an ssh session from the development station to the R-PI [ssh root@192.168.1.160]

```
sergi@sergi-laptop: ~
File Edit View Terminal Help
sergi@sergi-laptop:~$ ssh root@192.168.1.160
The authenticity of host '192.168.1.160 (192.168.1.160)' can't be established.
RSA key fingerprint is 88:85:97:fd:83:18:92:98:3d:cd:7a:a7:9d:03:71:a9.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.160' (RSA) to the list of known hosts.
root@192.168.1.160's password:
Permission denied, please try again.
root@192.168.1.160's password:

sergi@sergi-laptop:~$ ssh root@192.168.1.160
root@192.168.1.160's password:
Linux raspberrypi 3.1.9+ #108 PREEMPT Sat Jul 14 18:56:31 BST 2012 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

Type 'startx' to launch a graphical session

Last login: Sun Jul 15 19:42:50 2012
root@raspberrypi:~#
```

Figure 24: SSH connection to the R-PI

2. Download the code from the github site with the command [git clone git://gitorious.org/qt/qt5.git qt5]

3. Initiate the repository with the init-repository script from Qt5 [perl ./init-repository]
4. Qt compilation will need the OpenGL libraries from the Raspbian distribution, necessary to interact with the GPU. Therefore, we need to mount the Raspbian distribution, downloaded from the site of the foundation, in the folder /mnt/rasp-pi-rootfs with the command where the OpenGL libraries are. [sudo mount -o rw,loop,offset=62914560 /home/sergi/Downloads/raspbian/2012-10-28-wheezy-raspbian.img /mnt/rasp-pi-rootfs/]
5. Configure the Qt framework with the command [./configure -opengl es2 -device linux-rasp-pi-g++ -device-option CROSS_COMPILE=/opt/gcc-4.7-linaro-rpi-gnueabi-hf/bin/arm-linux-gnueabi-hf- -sysroot /mnt/rasp-pi-rootfs -opensource -confirm-license -optimized-qmake -reduce-relocations -reduce-exports -release -make libs -prefix /usr/local/Trolltech/Qt-embedded-RPI-5.0.0 -no-pch -v && make]. This step could take a few moments.

```

sergi@sergi-laptop: ~/Downloads/Qt5_2/qt5/qtbase
File Edit View Terminal Help
NIS auto-detection... ()
^C
sergi@sergi-laptop:~/Downloads/Qt5_2/qt5/qtbase$ ./configure -opengl es2 -device
linux-rasp-pi-g++ -device-option CROSS_COMPILE=/opt/gcc-4.7-linaro-rpi-gnueabi-h
f/bin/arm-linux-gnueabi-hf- -sysroot /mnt/rasp-pi-rootfs -opensource -confirm-lic
ense -optimized-qmake -reduce-relocations -reduce-exports -release -make libs -p
refix /usr/local/Trolltech/Qt-embedded-RPI-5.0.0 -no-pch

This is the Qt Open Source Edition.

You are licensed to use this software under the terms of
the Lesser GNU General Public License (LGPL) versions 2.1.

You have already accepted the terms of the license.

<srcbase> = /home/sergi/Downloads/Qt5_2/qt5/qtbase
<outbase> = /home/sergi/Downloads/Qt5_2/qt5/qtbase
Creating qmake. Please wait...
make: Nothing to be done for 'first'.
Note: PKG_CONFIG_LIBDIR automatically set to /mnt/rasp-pi-rootfs/usr/lib/pkgconf
ig:/mnt/rasp-pi-rootfs/usr/share/pkgconfig:/mnt/rasp-pi-rootfs/usr/lib/arm-linux
-gnueabi-hf/pkgconfig
Note: PKG_CONFIG_SYSROOT_DIR automatically set to /mnt/rasp-pi-rootfs

```

Figure 25: Compilation of the Qt framework

6. Installation of the installed libraries with [make install]

QTCreator

To develop the code of the CSCS, we have made use of an IDE called Qt Creator. Qt Creator is a cross-platform C++ integrated development environment which is part of the Qt SDK. It includes a visual debugger and an integrated GUI layout and forms designer. The editor's features include syntax highlighting and autocompletion as the majority of IDE for code development. On Linux, Qt Creator uses the C++ compiler from the GNU Compiler Collection whereas, on Windows, it can use MinGW with the default install.

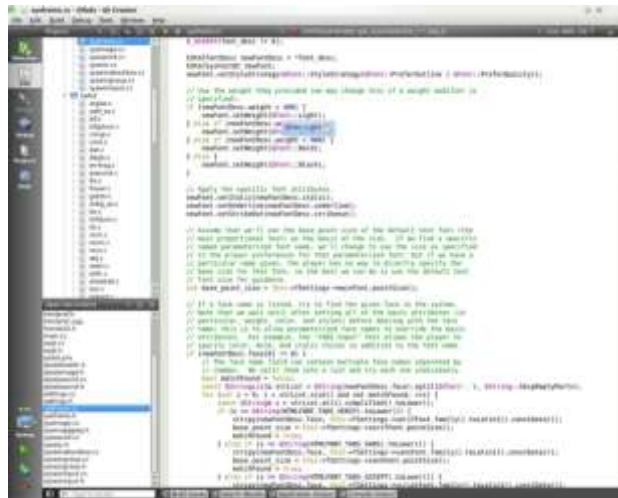


Figure 26: Qt Creator

5.3.1.1 Bluetooth configuration for R-PI

The R-PI does not include the Bluetooth hardware (i.e. antenna, transceiver). To perform this communication, a USB Bluetooth dongle will be used taking advantage of one of the USB ports of the R-PI.



Figure 27: USB dongle used for Bluetooth communication (marked in Red)

From the software perspective, we need to configure our Raspbian distribution for communication via Bluetooth since the two components of the system, the ePuck and the WiiMote, are linked with this protocol. The following step list summarise the actions to be performed to configure the Bluetooth on the R-PI and establish a connection to the ePuck. Similar steps were performed for the WiiMote.

1. Establish an ssh session from the development station to the R-PI [ssh root@192.168.1.160]

2. Install the necessary packages to enable the use of the Bluetooth stack. [sudo apt-get install Bluetooth python-gobject python-dbus libbluetooth-dev python-bluetooth bluez]
3. The installed packages provide the system with the capability to connect to Bluetooth devices together with a set of tool to monitor the link. As an example, the command [hciconfig dev] to list all the Bluetooth devices present in the system or the command [hcitool scan] listing all Bluetooth devices in the surroundings.

```
sergi@sergi-laptop: ~
File Edit View Terminal Help
78:F3:95:CB:13:4A FDD-BRAINABLE
00:1A:00:FF:00:00 FDD-NRUIZ
root@raspberrypi:~# /etc/init.d/rc
rc rcS rcnologin rsync
rc.local reboot rpcbind rsyslog
root@raspberrypi:~# /etc/init.d/rc
rc rcS rcnologin rsync
rc.local reboot rpcbind rsyslog
root@raspberrypi:~# service bluetooth restart
[ OK ] Stopping bluetooth: /usr/sbin/bluetoothd.
[ OK ] Starting bluetooth: bluetoothd +fcms.
root@raspberrypi:~# hcitool scan
Scanning ...
00:1F:C5:02:6A:4E Nintendo RVL-CNT-01
10:00:E8:AD:79:A8 e-puck_2464
00:1A:00:FF:00:00 FDD-nrui2
78:F3:95:CB:13:4A FDD-BRAINABLE
root@raspberrypi:~# hciconfig dev
hci0: Type: BR/EDR Bus: USB
BD Address: 00:09:00:50:00:0A ACL MTU: 310:10 SCO MTU: 64:0
UP RUNNING PSCAN
RX bytes:17701 acl:0 sco:0 events:249 errors:0
TX bytes:1381 acl:0 sco:0 commands:89 errors:0
root@raspberrypi:~#
```

Figure 28: Output of commands [hciconfig dev] and [hcitool scan]

The installed packages include the necessary Bluetooth Python modules to verify the Bluetooth communication with the devices. The following listing is a Python code establishing a Bluetooth connection to the ePuck platform and sending the command “C\r\n”.

```
PyBluetoothClient.py
from bluetooth import *
import sys

sock=BluetoothSocket( RFCOMM )
sock.connect(("10:00:E8:AD:79:A8",1))
sock.send("C\r\n")
sock.send("C\r\n")
sock.close()
```

Table 7: Testing program of Bluetooth Connection

Following the notation from section 5.2, the I1 interface was fully developed within the project whereas the I2 interface is abstracted by the open-source library cwiid. The I1 is implemented from the scratch by using the operations provided by the bluez library. As an example of the implementation the following code sends a command to the ePuck with address “10:00:E8:AD:79:A8” with the developed protocol.

```
rfcomm-client.c
#include <stdlib.h>
```



```

#include <stdio.h>
#include <unistd.h>
#include <sys/socket.h>
#include <bluetooth/bluetooth.h>
#include <bluetooth/rfcomm.h>

/*****
Function:      main
Overview:    main method
*****/
int main(int argc, char **argv)
{
    struct sockaddr_rc addr = { 0 };
    int s, bytesOp;
    char dest[18] = "10:00:E8:AD:79:A8";
    char buf[3] = { 0 };
    int status = 0;

    // allocate a socket
    s = socket(AF_BLUETOOTH, SOCK_STREAM, BTPROTO_RFCOMM);
    if( s < 0 )
        printf("It didnt work");

    // set the connection parameters (who to connect to)
    addr.rc_family = AF_BLUETOOTH;
    addr.rc_channel = 1;
    str2ba( dest, &addr.rc_bdaddr );

    // connect to server
    status = connect(s, (struct sockaddr *)&addr, sizeof(addr));
    usleep(200);

    int iterations=0;
    while(iterations < 5){

        // send a message
        bytesOp = send(s, "R\r\n", 3, 0);

        printf("\nWritten bytes : %d %d\n\n", bytesOp, iterations);

        // Receiving from the client
        printf("\nReceiving ... \n");
        bytesOp = recv(s, buf, sizeof(buf), 0);
        if( bytesOp > 0 ) {
            printf("received [%s]\n", buf);
        }

        if( strcmp(buf, "r\r\n") == 0) break;
        iterations++;
        // wait for some time
        usleep(500);
    }

    // Iterations
    if(iterations == 5){
        printf("Not able to connect ePuck. Exiting code\n");
        close(s);
        exit(-1);
    }

    printf("INFO: ePuck device found!\n");

    close(s);
    /**/
    return 0;
}

```

Table 8: Code for Bluetooth communication with the CSTS (ePuck)

5.3.2 Architecture of the application

The main functionality of the CSCS is to provide the UI to guide the user and maintain the proper communication between all the modules/devices, that is, the Input Device and the CSTS (ePuck robot). The architecture of the CSCS is conceptually illustrated with the following figure.

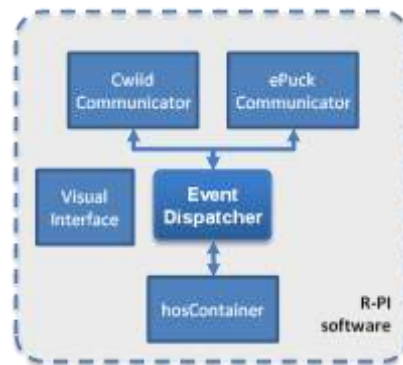


Figure 29: Architecture of R-PI software

- CwiidCommunicator holds all the functionalities related to the communication with the Input Controlled, the WiiMote
- ePuckCommunicator maintains the connection with the ePuck implementing the necessary functions to send and receive commands
- EventDispatcher connects the events from the different modules
- hosContainer handles the events from the Hex-O-Spell (see section 5.3.3.1 for further details) to understand the current selection of the user
- Visual Interface contains all the static elements that shape the interface such as groupboxes, layouts or labels.

The following picture provides a more detailed description of the architecture by means of a UML diagram.

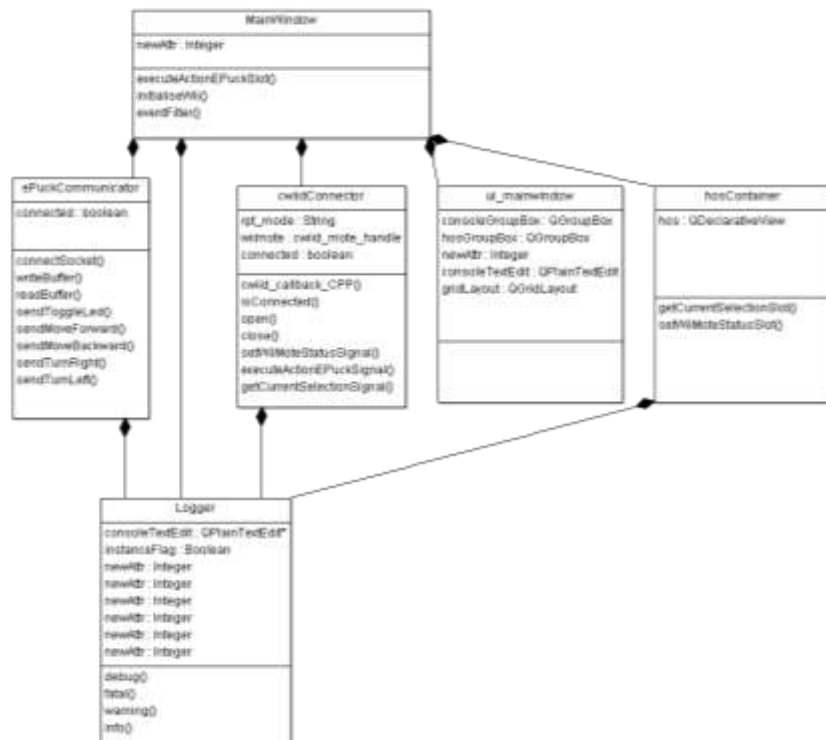


Figure 30: UML Diagram of the CSCS running on the R-PI

Signals & Slots

The callback functions are very common in GUI development but they introduce a lot of complexity to the developments. CSCS takes advantage of the Qt framework to avoid their use replacing them by the signal&slot mechanism. Indeed, signals and slots are used for communication between objects. The signals and slots mechanism is a central feature of Qt and probably the part that differs most from the features provided by other frameworks. In GUI programming, when a change occurs in one component, it is often necessary to notify another component. More generally, it is required that objects of any kind to be able to communicate with one another. Qt solves this common issue with the signal&slot mechanism.

A signal is emitted when a particular event occurs. Qt's widgets have many predefined signals, but we can always subclass widgets to add our own signals to them. A slot is a function that is called in response to a particular signal. Qt's widgets have many pre-defined slots, but it is common practice to subclass widgets and add your own slots so that you can handle the signals that you are interested in.

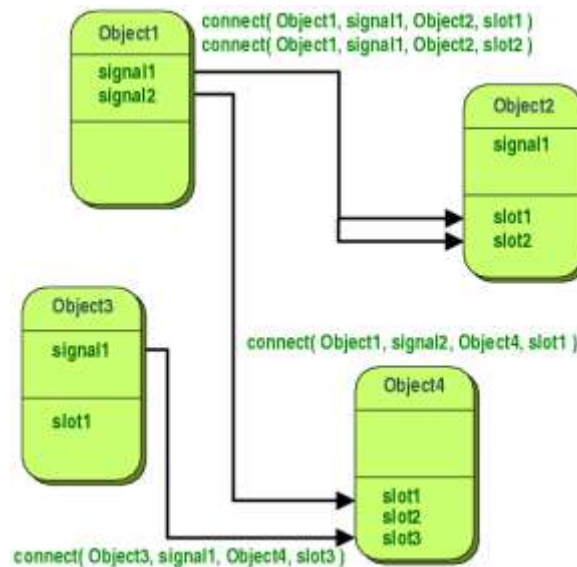


Figure 31: Signal&Slot mechanism

5.3.3 User Interface development

The user interface is the space where interaction between humans and machines occurs. The goal is effective operation and control of the machine, and feedback from the machine which aids the operator in making operational decisions

5.3.3.1 Hex-O-Select as GUI

Graphical user interface (GUI) is a type of user interface that allows users to interact with electronic devices using images rather than text commands. GUIs can be used in computers, portable media players or gaming devices, household appliances, office and industry equipment. A GUI represents the information and actions available to a user through graphical icons and visual indicators such as secondary notation, as opposed to text-based interfaces, typed command labels or text navigation. The actions are usually performed through direct manipulation of the graphical elements which have an effect on the system.

The usefulness of a system is tightly linked to usability and accessibility of the user interfaces. It becomes, then, clear that the user interface is a key element in the success or failure of iCognos given the target population envisaged. End-users of iCognos are cognitive impaired people and, consequently, the graphical user interface design needs to be as a main issue to be treated carefully. After the study of different alternatives on the look and feel (e.g. web design), iCognos has implemented a graphical user interface based on the novel paradigm Hex-O-Select. Hex-O-Select consists of six hexagons distributed in a circle with a central arrow turning clockwise. The user is able to select the option which is highlighted and pointed by the arrow.

Specifically the system allows the following five commands:

- **Up** to move the CSTS onwards
- **Left** to turn the CSTS 90 degrees clockwise
- **Connect** which makes the CSCS to establish a connection to the CSTS

- **Down** to move the CSTS backwards
- **Right** to turn the CSTS 90 degrees anticlockwise

The following figure is a screenshot of the Hex-O-Select interface developed for iCognos.

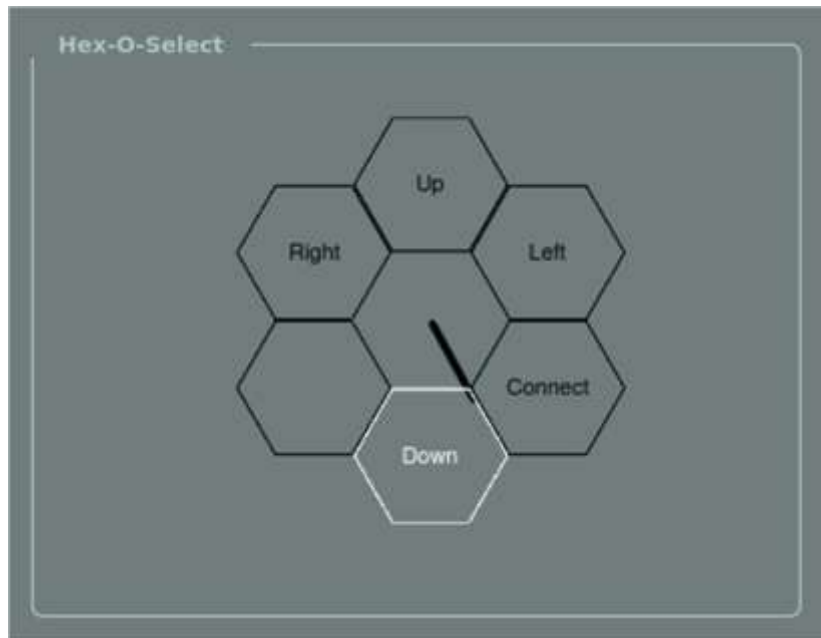


Figure 32: Hex-O-Select interface

As first approach, the Hex-O-Select interface was developed by using the standard elements of Qt (e.g. buttons, labels) and the 2D capabilities of the framework. However, the lack of flexibility and the complexity of the code necessary forced the development of the final version using QML. QML is a powerful extension of Qt focused on treating the graphical and animated side of the system (see section 4.4.1 for further details). It is remarkable to say that iCognos relies on the `QDeclarativeComponent`⁷ class of the Qt Framework. This class enables to mix both Qt and QML code leveraging the powerfulness of the Qt-based applications.

⁷ <http://doc.qt.digia.com/4.7-snapshot/qdeclarativecomponent.html>

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

Figure 33: QML Code developed for the Hex-O-Select interface

The following figure illustrates the software architecture designed for the development of the Hex-O-Select within iCognos:

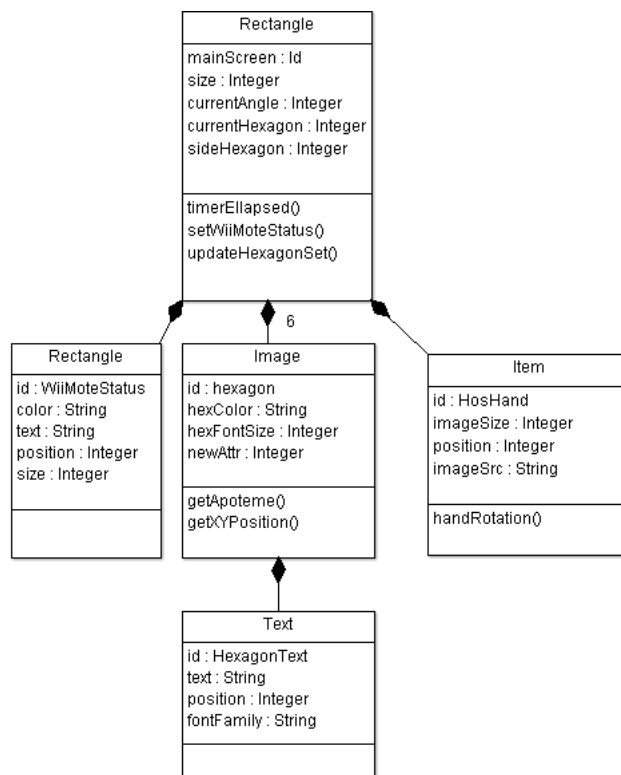


Figure 34: UML diagram for Hex-O-Select

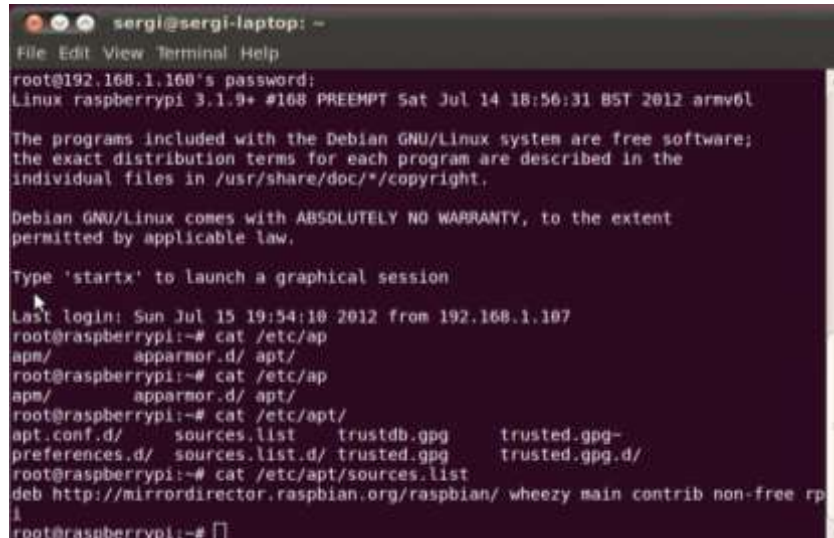
5.3.3.2 Input control

Cwiid [34] is a software package that actuates on top of the bluez package, responsible for the Bluetooth communication. Specifically, Cwiid provides a public API to interact with the WiiMote by means of Bluetooth commands, the libcwiid. The project also provides a series of different software applications that take advantage of the libcwiid to demonstrate the powerfulness of the project such as wmgui, wmdemo or wminput.

Compilation of Cwiid

To obtain Cwiid running in the R-PI, two different alternatives were followed:

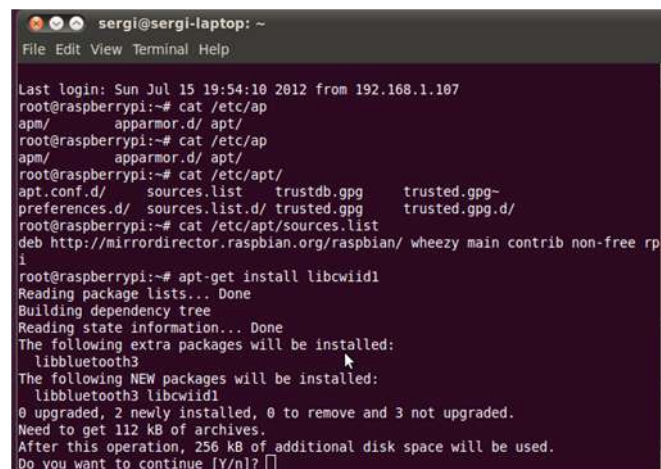
Alternative 1 - Install a the libcwiid1 package from the raspbian repositories. These repositories are available under the URL [<http://archive.raspbian.org/raspbian>] as it is shown in the figure. This alternative makes use of the pre-compiled binaries of the official raspbian repositories with a functional compiled cwiid package.



```
sergi@sergi-laptop: ~  
File Edit View Terminal Help  
root@192.168.1.160's password:  
Linux raspberrypi 3.1.9+ #168 PREEMPT Sat Jul 14 18:56:31 BST 2012 armv6l  
  
The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
  
Type 'startx' to launch a graphical session  
Last login: Sun Jul 15 19:54:10 2012 from 192.168.1.107  
root@raspberrypi:~# cat /etc/ap  
apm/      apparmor.d/ apt/  
root@raspberrypi:~# cat /etc/ap  
apm/      apparmor.d/ apt/  
root@raspberrypi:~# cat /etc/apt/  
apt.conf.d/  sources.list  trustedb.gpg  trusted.gpg-  
preferences.d/  sources.list.d/  trusted.gpg  trusted.gpg.d/  
root@raspberrypi:~# cat /etc/apt/sources.list  
deb http://mirrordirector.raspbian.org/raspbian/ wheezy main contrib non-free rp  
i  
root@raspberrypi:~#
```

Figure 35: Raspbian repositories

To get this package, we make use of the command `apt-get install libcwiid1`. This command, shown in the next figure, queries the official raspbian repositories, downloads and installs the package into the system.



```
sergi@sergi-laptop: ~  
File Edit View Terminal Help  
Last login: Sun Jul 15 19:54:10 2012 from 192.168.1.107  
root@raspberrypi:~# cat /etc/ap  
apm/      apparmor.d/ apt/  
root@raspberrypi:~# cat /etc/ap  
apm/      apparmor.d/ apt/  
root@raspberrypi:~# cat /etc/apt/  
apt.conf.d/  sources.list  trustedb.gpg  trusted.gpg-  
preferences.d/  sources.list.d/  trusted.gpg  trusted.gpg.d/  
root@raspberrypi:~# cat /etc/apt/sources.list  
deb http://mirrordirector.raspbian.org/raspbian/ wheezy main contrib non-free rp  
i  
root@raspberrypi:~# apt-get install libcwiid1  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following extra packages will be installed:  
  libbluetooth3  
The following NEW packages will be installed:  
  libbluetooth3 libcwiid1  
0 upgraded, 2 newly installed, 0 to remove and 3 not upgraded.  
Need to get 112 kB of archives.  
After this operation, 256 kB of additional disk space will be used.  
Do you want to continue [Y/n]?
```

Figure 36: libcwiid1 installation via apt-get

Alternative2 - Compile the cwiid project from source. The download from the source code can be done from the cwiid website. The compilation has to be done in the Raspberry PI itself since the architecture of the R-PI (ARM) is not the same as the architecture of the development station (x86). The necessary steps to be done are the following:

1. Establish an ssh session to the R-PI [`ssh root@192.168.1.160`]

2. Download the code from the website with the `wget` command
`wget http://abstrakraft.org/cwiid/downloads/cwiid-0.6.00.tgz`
3. Create a folder [`mkdir cwiidDownloaded`] and untar the downloaded package [`tar xafv cwiid-0.6.00.tgz`]
4. Install the necessary packages [`sudo apt-get install bison flex libgtk2.0-dev python-dev`] to perform the compilation
5. Compile the code with the command [`./configure && make`]. This command will compile all the cwiid package, that is, the library itself together with all the demonstration programs before mentioned

```
sergi@sergi-laptop: ~
File Edit View Terminal Help
root@raspberrypi:~/Downloads#
root@raspberrypi:~/Downloads#
root@raspberrypi:~/Downloads# dir
root@raspberrypi:~/Downloads# cd cwiid/
root@raspberrypi:~/Downloads/cwiid# dir
cwiid-0.6.00  cwiid-0.6.00.tgz
root@raspberrypi:~/Downloads/cwiid# cd cwiid-0.6.00/
root@raspberrypi:~/Downloads/cwiid/cwiid-0.6.00# ./configure
checking for gcc... gcc
checking for C compiler default output file name... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking for gawk... no
checking for awk... awk
checking for flex... flex
checking lex output file root... lex.yy
checking lex library... [
```

Figure 37: Compilation of cwiid on R-Pi

6. Install the library with [`make install`]

Cwiid connection test

The cwiid project has prepared a series of application to test the proper installation of cwiid in the system. In particular, we have made use of the application `wmdemo`, a command-line application that contains all the functionalities implemented for communication with the WiiMote.

```
sergi@sergi-laptop: ~
File Edit View Terminal Help
common  configure.ac  doc  Makefile.in  README
root@raspberrypi:~/Downloads/cwiid/cwiid-0.6.00# LD_LIBRARY_PATH=./libcwiid/ wmdemo/wmdemo
Put Wiimote in discoverable mode now (press 1+2)...
Note: To demonstrate the new API interfaces, wmdemo no longer enables messages by default.
Output can be gathered through the new state-based interface (s), or by enabling the messages interface (c).
1: toggle LED 1
2: toggle LED 2
3: toggle LED 3
4: toggle LED 4
5: toggle rumble
a: toggle accelerometer reporting
b: toggle button reporting
e: toggle extension reporting
i: toggle ir reporting
m: toggle messages
p: print this menu
r: request status message ((t) enables callback output)
s: print current state
t: toggle status reporting
x: exit
[
```


Figure 38:Using wmdemo

Finally, it is remarkable to say that this short application was used as a reference to getting started in including the Cwiid API into iCognos.

Although there is not a public specification from Nintendo's, different open-source projects have published the commands used to interact with the device such as the WiiBrew⁸ project, which gives detailed information of the operational functioning of the WiiMote. Given the fact that Cwiid abstracts the connection, it is not necessary to fully understand the communication with the WiiMote, though this information could be very useful for debugging.

Inclusion of Cwiid in iCognos

The CSCS application is cross-compiled in the development station but executed in an embedded Linux, the R-PI. Therefore, it becomes compulsory to include the Cwiid library for the ARM architecture (libcwiid.a) in the project folder to perform the compilation. The inclusion of these libraries was done by means of the static version of the library avoiding problems associated to search path.

The code developed to incorporate the library uses the clause `[#include <libcwiid/cwiid.h>]`. This header file holds all the function prototypes to make use of the cwiid library. The cwiid library is developed using the C language which needs to be incorporated to the CSCS ecosystem developed in C++. This can be accomplished with the clause `[extern "C" { ... }]` that enables the combination of C and C++ code in a single application.

```
13
14
15  Extern C Functions
16  .....
```

```
17 extern "C" {
18
19     CwiidConnector *cwiidConnector_C;
20     void cwiid_callback(cwiid_wiimote_t *wiimote, int msg_count,
21                       union cwiid_msg msg[], struct timespec *timestamp)
22     {
23         cwiidConnector_C->cwiid_callback_CPP(wiimote, msg_count, msg, timestamp);
24     }
25
26     void err(cwiid_wiimote_t *wiimote, const char *s, va_list ap)
27     {
28         if (wiimote) printf("%d:", cwiid_get_id(wiimote));
29         else printf("-1:");
30         vprintf(s, ap);
31         printf("\n");
32     }
33
34 }
35
36
```

Figure 39:Combination of C and C++ language

⁸ <http://wiibrew.org/wiki/Wiimote>

5.4 Cognitive Stimulation Telerobotics System

The CSTS is implemented by means of the low-cost robot ePuck which development is described in this section together with an explanation of the development environment. Further details on the e-puck can be found in section 4.2.

5.4.1 Configuration of development environment

The ePuck robot contains at its heart a dsPIC microcontroller which is the piece of hardware able to be programmed with a dedicated firmware. The code for the dsPIC is developed by using C language and compiled with the C30 compiler. This compiler translates the C code to the binary code lately interpreted by the microcontroller. The compiler-generated files are assembled and linked with other object files and libraries to produce the final application program in executable COFF or ELF file format. The instruction set to which the C code is translated is defined by the manufacturer⁹. The following figure provides a conceptual map of the tools data flow when using the C30 compiler.

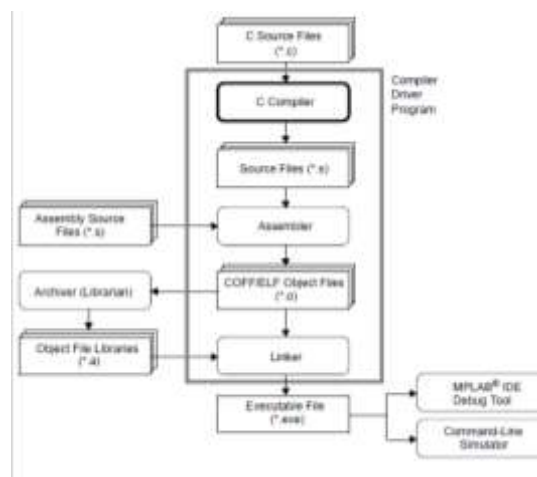


Figure 40: C30 compiler tools data flow

To develop the code, Microchip provides a software platform for development of firmware called MPLAB. MPLAB Integrated Development Environment (IDE) is a free, integrated toolset for the development of embedded applications employing Microchip's PIC and dsPIC microcontrollers. MPLAB IDE runs as a 32-bit application on Windows, is easy to use and includes a host of free software components for application development and debugging. MPLAB IDE also serves as a single, unified graphical user interface for additional Microchip and third party software and hardware development tools.

⁹ For further details, see <http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en010334>

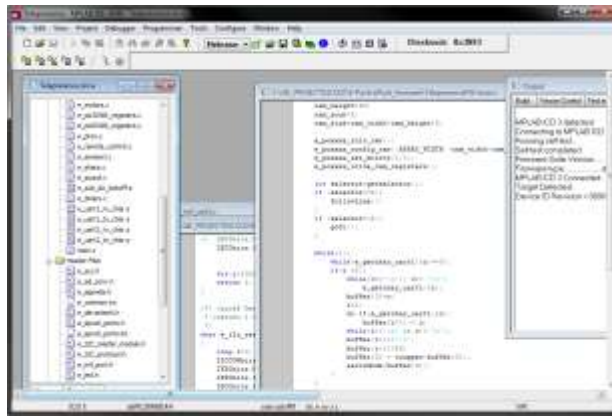


Figure 41: E-pucks development environment, MPLAB IDE

To burn the created firmware into the microcontroller a programmer is needed and, for iCognos, the ICD 3 from the Microchip was used. The figure below shows how the programmer/debugger is connected to the ePuck program port.

The programmer is recognized by the MPLAB IDE with the corresponding drivers and the MPLAB IDE itself is able to recognize the target device and program/debug it. There is a big advantage in this approach of developing embedded systems. All embedded systems have a hardware element and a software element, which are separate but tightly interdependent: running the code in final hardware is the proper validation. For this, the ICD3 allows the software element to be run and tested on the actual hardware on which is to run, allowing programmer to help isolate faulty code, such as single-stepping.



Figure 42: ICD3 programmer/debugger connected to E-Puck

An alternative to the use of the ICD3 device is to make use of a bootloader, a program that stays in the microcontroller and communicates with the PC. Once the communication is established, the bootloader receives a user program from the PC and writes it in the flash memory, then launches this program in execution. Bootloaders can only be used with those microcontrollers that can write their flash memory through software. The bootloader itself must be written into the flash memory with an external programmer. In order for the bootloader to be launched after each reset, a "goto bootloader" instruction should exist somewhere in the first 4 instructions of the microcontroller.

The bootloader selected for the ePuck is the Tiny Bootloader, see figure below. This bootloader is connected through a serial interface, which in the case of the ePuck is an emulated serial port relaying the data to the Bluetooth interface. In this manner, it is possible to program the ePuck by means of an OTAP (Over-The-Air Programming) feature.

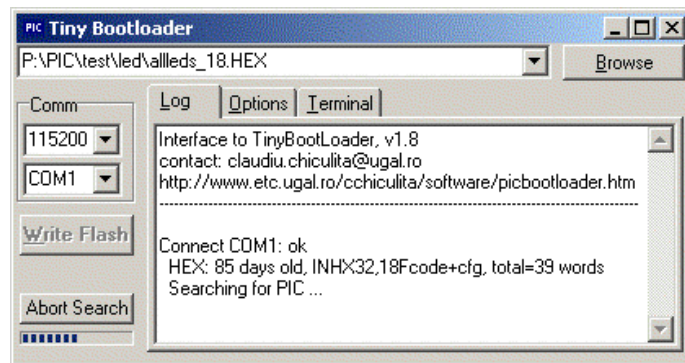


Figure 43: Tiny bootloader for PIC

5.4.2 Firmware development

The CSTS operation needs to take care of the different inputs of the on-board sensors together with the control of the different actuators. For this, the use of multithreading seems to be the optimal choice for development of the firmware. The ePuck relies on a PIC microcontroller, a lot less powerful than a microprocessor and no native multithreading option from the manufacturer is available. Multithreading is very powerful tool in distributed systems where different applications need to talk each other. As such, some tests with RTOS (real time operating system) were done, to get a multithreading environment on the PIC microcontroller. FreeRTOS [35], an OSS project, was managed to be run on ePuck's microcontroller, though the amount of memory necessary were higher and was eventually discarded.

As such, a single-threaded application (e.g. polling-based) was devised to be embedded into the microcontroller. The polling-based firmware application resides in an infinite loop and once an event is received (e.g. received a command) treats it with the Interpreter module. The following figure illustrates the ePuck firmware for treating a message received by providing a conceptual overview of the code developed. It is remarkable to say that the commands accepted by the firmware are listed in the section 5.2.2.

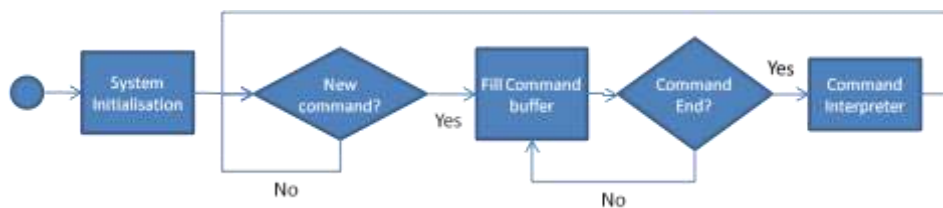


Figure 44: ePuck UML state machine diagram

Finally, the complete architecture of the firmware running in the ePuck is based in the following modules:

- *Bluetooth Communicator* responsible for maintaining the communication with the on-board Bluetooth chip via the UART interface
- *Command interpreter* includes all the algorithms to understand the command received

- *I/O Manager* incorporates all the functions devoted to control the on-board peripherals



Figure 45: Conceptual map of the firmware

5.4.2.1 Command interpreter

The implementation of the command interpreter is simple: a switch to treat it separately suffices. Below some clear and short examples:

```
//to turn on the leds
case 'W': //
    sscanf(buffer, "W,%d,%d,%d,%d,%d,%d,%d,%d,%d\r", &led[0], &led[1], &led[2],
        |&led[3], &led[4], &led[5], &led[6], &led[7]);
    for (i=0; i<8; i++){
        e_set_led(i,led[i]);
    }
    break;
```

Figure 46: Code that turns the light on

On this example, the LEDs around the epuck are turned on and off accordingly to the parameters received. The function first fills the *led* array with the eight parameters (0 or 1 depending if the LED is on or off) and then for each led the *e_set_led(int, int)* function is called, which is directly from the ePuck software library.

```
//setting individual speed of each wheel
case 'Z': //
    sscanf(buffer, "Z,%d,%d\r", &left, &right);
    e_set_speed_wheels(0,0);
    e_set_speed_wheels(left,right);
    sendGSensorValue();
    break;
```

Figure 47: Code to set the robots wheel speed

6 Analysis of validation

The current implementation of iCognos has to be understood as a proof-of-concept and, therefore, the validation presented here becomes of predominant importance; the information concluded is very valuable to adjust the current development. In this chapter, the final platform developed is tested and validated by three different subjects with different profiles to collect impressions with different perspectives. The feedback collected from the different sessions is reported below.

6.1 Testing environment

On the next figure, the final set up for the validation of iCognos is shown. It consists of a R-PI and a WiiMote which are part of the CSCS together with the ePuck robot corresponding to the CSTS. Additionally, the prototype makes use of a wooden white box with black lines painted on it; the testing subject is requested to make the robot to follow such lines with the help of the CSCS. Indeed, the user of iCognos will be able to send the robot autonomously to any destination on the wooden white box and move it independently.

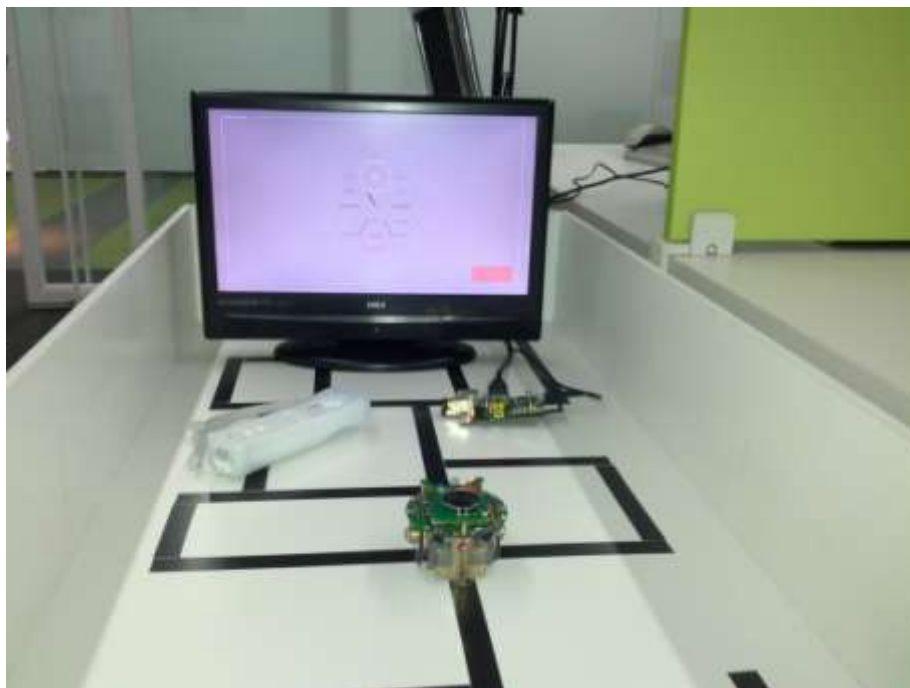


Figure 48: Final prototype of iCognos

To see the prototype in action, a video demonstration is available at the following URL:

<http://www.youtube.com/watch?v=N6j51JhuT0A&noredirect=1>

6.2 Discussion of the results

To test iCognos, we asked three persons to try the platform. The feedback collected was analysed and classified into three different aspects : (1) Concept, (2) Usability and Accessibility, and (3) Improvements.. The outcome of this analysis is included in the following tables:

TEST USER-001 Age : 32
Profession : Supervisor of iCognos

Concept: The system as it is developed is just a proof-of-concept of the potential. As such, it has to be understood like this and a further refinement of different aspects needs to be tackled so as to obtain a commercial solution such as the casing of the robot or the aspect of the UI.

The system developed here needs to help the end-users to improve their performance while doing cognitive stimulation tasks. It will be, then, necessary to including algorithms to understand the current interaction of the user and propose tasks adapted to their profile. For me, the major concern comes from the R-PI since this device has a limited amount of resources, mainly in terms of CPU, that potentially could be a problem for deploying such algorithms.

Usability and Accessibility: iCognos proposes to the user a single interface to interact with which could become a problem for covering all the cognitive disorders. The system needs to take into account this and have the possibility to present interfaces in different forms to the end-user to ensure his/her comfortability. Additionally, this could be a manner of improving several cognitive skills since the user needs to interact with the system differently.

Improvements: As future work, it would be nice to exploit as much as possible to full features offered by the robot. In particular, the ePuck has more sensors, and more actuators that could serve to improve the current games or develop new ones. Examples of this are the use the 3D accelerometer, the speaker, or even the microphones to determine the source of an audio signal.

***TEST USER-002 Age : 35**
Profession : Research in Rehabilitation

Concept The project is aligned with the idea of ecological cognition as we proposed, following the recommendations from the clinicians, for the development. For me, it seems clear that a system like iCognos is closer to the activities of daily life than a serious game. This has a clear impact on the performance and the progress of the users suffering from cognitive disabilities.

Usability and Accessibility: The interface developed is reused from an FP7 project called BrainAble which is very useful for a low –latency interfaces such as the BNCI technology. Although it seems to be challenging for the end-users if we understand the interface as part

of the stimulation, I am not completely sure that is the scope of iCognos.

Improvements: On the first impression using the system, I had problems in understanding the operation of the Hex-O-Select interface. I would strongly recommend preparing an instructions or any other preliminary aid which facilitates the user to understand what is required from him/her.

Additionally, I would like to remark that there is a big amount of blank space in the interface which could be used to improve the interaction of the user (e.g. feedback from the progress of the task).

TEST USER-003 Age : 27

Profession : Journalist

Concept A great project, It has been very nice and funny to interact with the little robot (I actually “shouted it lives!” when first seeing it move). Then, I must agree that concept of motivating people through the use of a little robot seems to be nice.

In my view, everyone in the office found it very attractive which leads me to think that also the cognitive impaired will. The equipment is not very bulky which helps to fulfill the aimed telemedicine spirit of iCognos.

Usability and Accessibility: The use of a WiiMote is nice but the project has a very limited use of it. I think the system should make use of the WiiMote as Nintendo’s games do as for example, with the movements of the asset. For instance, you could try to move the robot as the WiiMote is moving. This will give more possibilities of rehabilitation combining both motor and cognitive rehabilitation.

Improvements: The game what I was asked to play it is not adaptable in levels of difficulty. It would be a good idea to integrate the concept of levels in order to adapt the task to every impairment and individual reaching a personalized cognitive stimulation. I think it could be a good improvement to have cognitive stimulation adapted to the person currently performing the task.

7 Conclusions

The conclusion of this master thesis is divided in two sections, the conclusion and the future work. In the first part, I present what I gained on its development, but I also show where I found problems and had special troubles; on the future work section, I explained how the project could be improved and brought to another level.

7.1 Last word

The goal of this master thesis to develop a **prototype to show the potential of cognitive stimulation through task-oriented telerobotics** has been accomplished. iCognos presents a stable environment to demonstrate of the capabilities and benefits of including robots in the cognitive rehabilitation programs. The results of the validation lead to think that the concept of implementing cognitive stimulation tasks by means of SmartObjects is very likely. Indeed, the reporting presented in section 6, demonstrates the increase of motivation of the user when interacting with a real/physical object, the basis of iCognos.

From the technical point of view, iCognos proposed different challenges that I would like to mention. These challenges have changed the manner in which I understand the software life-cycle and the development of hardware

During the different courses I took, the main focus was posed over the consolidated concept of open-source and free software with a subtle difference between the two. In spite of this, both have clearly become a revolution from many different points of view as they have enabled people from everywhere to work together with the simple and naive objective to spread knowledge and improve living conditions. In the latest times, a new paradigm is pushing, the Open-source Hardware (OSHW). OSHW is to extend the philosophy of Open-source Software (OSS) to further boundaries. Indeed, OSHW takes advantage of latest development to create communities of developers not only to develop software but also around the hardware running that software (i.e. the specific electronic devices). Essentially, this has helped to enlarge the basis of low-level software developers (a.k.a. firmware) and spread around the basis of electronics for a widespread audience.

The realization of this project has had very positive effects on my technical background complementing my previous knowledge of computer science. This thesis introduced me in the field of Open-source hardware OSHW, while this new paradigm is making its first steps. This made my introduction more motivational, if possible, since I was able to see how this paradigm evolved and reinvented itself in a short period of time. In this manner, iCognos relies on the usage of different OSHW projects which are deliberately on the top of the wave in this moment. Indeed, projects like Raspberry Pi, ePuck or Arduino have still room to grow, though they have become very popular. The understanding of this novel paradigm and manner of approaching to the firmware/hardware development has been very enriching and motivational for me and my development.

Additionally, iCognos has also helped me to understand the concept of cross-compiling through the most popular computer architectures. I must confess that the process of understanding the different compilers for different architecture was not an easy task (in fact, the hardest part of the thesis), though very interesting for me.

Finally, I would like also to highlight the use of the operating system GNU/Linux with a low-level purpose, that is, being in contact with kernel modules and making use of communication protocol like Bluetooth or RS-232 within GNU/Linux. This has helped me to better understand the architecture of an operating system and make the link between the basic electronic components (e.g microcontroller, resistors and capacitor) and more complex system such as those based on RTOS or full-functional OS such as GNU/Linux in which is based the R-PI distribution.

7.2 Future Work

Since the objective of this thesis has been to generate a platform to show the potential of cognitive rehabilitation through telerobotics, it is obvious that there is a lot of future work possible.

The ePuck, as explained, contains much more sensors and actuators as the once used in this thesis. To show more possibilities of a telerobotic platform, the rest of the robots features could have been controlled and included in the prototype. For example, the infrared proximity sensors or the highly interesting 3D accelerometer could have been included in iCognos. This feature were avoid in order to perform a first proof-of-concept keeping the development simple which forced to use the simplest sensors available. Furthermore, control of the system was made with a commercial controller, the WiiMote, giving flexibility to the system but attaching the final prototype to a third party component with a closed source philosophy limiting the space for improving the user experience. To solve this issue, a custom controller or an open-source hardware controller could be developed or used which might ultimately be more suitable for the rehabilitation purposes enabling, for instance, physical rehabilitation.

From the user validation, I got additional ideas for future improvements. Most of the test testing subjects agreed that a level system to determine tasks difficulty would be a nice feature. Indeed, the ability to configure the level of the cognitive stimulation task is called to be a positive advance in order to adapt the task to each user but also for motivational purposes: the user will advance levels while improving his/her cognitive skills.

To wrap up this master thesis, this paragraph summarises an analysis on how a real cognitive rehabilitation tool could be accomplished. A specialized institute in cognitive disorders needs to be contacted to get a proper and professional definition and objectives of every task/assignments. With these inputs, better and more innovative cognitive tasks could be developed following their recommended guidelines. Once those tasks were implemented, the validation with real end-users could begin and more feedback could be collected, and the tool could be improved even further. This methodology of development is called user-centre design (UCD) which definition describes that the end-users needs to be involved in whole life-cycle of the development: from the requirements definition to the quality assurance. UCD's goal is to ensure the proper alignment of both technical and non-technical, in our case the end-users, counterparts to reach a satisfactory outcome. In the end, after being completed all these stages, the tool presented here, iCognos, could be converted in a real product and marketing of it could begin.

8 Bibliography

- [1] Bloom's Taxonomy: A classification of thinking organized by level of complexity. Knowledge, comprehension, application, analysis, synthesis, and evaluation are the six levels. Retrieved Feb. 2009 from Macomb Intermediate School District: <http://www.misd.net/gifted/terms.htm>
- [2] Cost of disorders of the Brain in Europe, *European Journal of Neurology*, Volume 12, September 2005
- [3] Abreu BC & Togliola JP(1987) "Cognitive rehabilitation: a model for occupational therapy"
- [4] Mid-term evaluation of the European Action Plan 2003-2010 on Equal Opportunities for People with Disabilities. Center for Strategy & Evaluation Services. June 2009
- [5] Communication from the commission to the European parliament, the council, the European economic and social committee of the regions. Horizon 2020 - The Framework Programme for Research and Innovation. Brussels, 30/11/2011
- [6] M. Roussos, Issues in the Design and Evaluation of a Virtual Reality Learning Environment, M.Sc. Thesis, University of Illinois at Chicago (1997).
- [7] K. Wada, T. Shibada, T. Musha, S. Kimura. Robot Therapy for elders affected by dementia. *EMBS* July 2008 http://www.pulse.embs.org/Past_Issues/2008July/Wada.pdf
- [8] H.I. Krebs, PhD; B.T. Volpe, MD; M.L. Aisen, MD; N. Hogan, PhD "Increasing productivity and quality of care: Robot-aided neuro-rehabilitation". *Journal of Rehabilitation Research and Development*; Vol. 37 No. 6, November/December 2000
- [9] B.T. Volpe, MD et al. "A novel approach to stroke rehabilitation: Robot-aided sensorimotor stimulation". *Neurology* May 23, 2000 vol. 54 no. 10 1938-1944
- [10] Pal Robotics, The REEM Robot - <http://pal-robotics.com/robots>
- [11] The Q.bo project - <http://openqbo.org/wiki/doku.php>
- [12] The therapeutic robot Paro - <http://www.parorobots.com/>
- [13] Gert Kwakkel, PhD,^{1,2} Boudewijn J. Kollen, PhD,³ and Hermano I. Krebs, PhD. "Effects of Robot-assisted therapy on upper limb recovery after stroke: A Systematic Review" *Neurorehabil Neural Repair*. Author manuscript; available in PMC 2009 August 23.
- [14] Peter S. Lum, PhD, Charles G. Burgar, MD, Peggy C. Shor, OTR, Matra Majmundar, OTR, Machiel Van der Loos, PhD "Robot-Assisted Movement Training Compared With Conventional Therapy Techniques for the Rehabilitation of Upper-Limb Motor Function After Stroke". 2002 by the American Congress of Rehabilitation Medicine and the American Academy of Physical Medicine and Rehabilitation
- [15] *Occupational Therapy and Mental Health*. Jennifer Creek, DipCOT and Lesley Lougher, BScSoc, DipCOT. Churchill Livingstone, 2008
- [16] Culture, Cultural Competency and Occupational Therapy: a Review of the Literature. *The British Journal of Occupational Therapy*, Volume 66, Number 8, 1 August 2003 , pp. 356-362(7)

- [17] "The MIT Press | 50 years of publishing". Mitpress.mit.edu. Retrieved 2012-10-25.
- [18] Erik Rubow, "Open source hardware". November 20, 2008
- [19] "Open Source Hardware definition" - <http://freedomdefined.org/OSHW>
- [20] Philippe Torrone "Million dollar baby – Businesses designing and selling open source hardware, making millions." Makezine May 2006
- [21] The Open-source Hardware Beagleboard project - <http://www.beagleboard.org>
- [22] The Raspberry PI Foundation - <http://www.raspberrypi.org>
- [23] The Arduino Project <http://www.arduino.cc/>
- [24] e-puck: The educational robot - <http://www.e-puck.org>
- [25] Jin-Shyan Lee, Yu-Wei Su, and Chung-Chou Shen. A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi. The 33rd Annual Conference of the IEEE Industrial Electronics Society (IECON). Nov. 5-8, 2007, Taipei, Taiwan
- [26] Comparison of wireless technologies. Future Electronics - http://www.future-electronics.com/wp-content/plugins/fe_downloads/Uploads/Comparison%20of%20Wireless%20Technologies.pdf
- [27] The comparison of Wi-Fi, Bluetooth and ZigBee - <http://www.sena.com/blog/?p=359>
- [28] Raspberry Pi User's Guide. Eber Upton and Gareth Halfacree. Wiley Publications.
- [29] Inside the Raspberry PI - http://elinux.org/RPi_VideoCore_APIS
- [30] Raspberry PI: Bare-metal programming - <http://kariddi.blogspot.com.es/2012/08/raspberry-pi-bare-metal-part-1-boot.html>
- [31] Wikipedia Article – Debian - <http://en.wikipedia.org/wiki/>
- [32] The Qt Project. <http://qt-project.org/downloads>
- [33] The CIFS Protocol. https://wiki.samba.org/index.php/LinuxCIFS_utils
- [34] The Cwiid project - <http://abstrakraft.org/cwiid/>
- [35] The FreeRTOS project - <http://www.freertos.org/>

ANNEX A : GNU Free Documentation License

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.
<<http://fsf.org/>>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.
<<http://fsf.org/>>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document",

below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited

only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be

listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the

various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.