**Technical University of Cluj-Napoca**

**Faculty of Engineering**

# HABILITATION THESIS

# Solutions regarding Intelligent Embedded Systems for Active and Assisted Living

**Assoc. prof. Ștefan Vasile Oniga, Ph.D.**

**2016**

# Table of Contents

## Rezumat

Lucrarea de față prezintă activitatea de cercetare științifică și didactică efectuată după susținerea publică a tezei și obținerea titlului de doctor al Universității Politehnica din Timișoara (2005). In aceasta perioada am participat la mai mult de 15 contracte, granturi de cercetare și educaționale. Ca rezultat al activității de cercetare am publicat în aceasta perioadă un număr de 68 articole științifice din care 3 în reviste cotate ISI cu factor de impact, 25 în volumele unor conferințe indexate ISI și 32 articole indexate BDI. De asemenea în aceasta perioadă am publicat un capitol de carte legat de tema tezei în Springer Lecture Notes in Computer Science cu titlul ”Application Possibilities of Hardware Implemented Hybrid Neural Networks to Support Independent Life of Elderly People”, 2 cărți in edituri naționale și am elaborat 3 materiale didactice disponibile în format electronic.

Prima secțiune a lucrării intitulat Realizări științifice prezintă contribuțiile aduse în domeniul dezvoltării sistemelor dedicate inteligente pentru o viață activă și asistată (AAL). Aceste contribuții sunt grupate în trei direcții de cercetare care sunt prezentate in câte un capitol distinct din prezenta teză:

• Implementarea rețelelor neuronale artificiale în circuite programabile de tip FPGA și dezvoltarea de aplicații bazate pe aceasta metodă.

• Sisteme e-Health și sisteme dedicate pentru o viață activă asistată (AAL) cu cele trei subdirecții:

- Soluții de implementare pentru sisteme ambientale inteligente
- Sisteme senzoriale purtabile pentru monitorizarea activității și a stării de sănătate
- Roboți pentru asistență personală

• Contribuții la dezvoltarea de noi metode pentru recunoașterea activității prin combinarea tehnicilor de extragere a caracteristicilor și utilizarea rețelelor neuronale pentru recunoașterea tiparelor.

Capitolul 1 al tezei prezintă principalele contribuții legate de prima direcție de cercetare, cea de dezvoltare a metodei propuse în teza de doctorat privind implementarea rețelelor neuronale artificiale în circuite programabile de tip FPGA și dezvoltarea de aplicații bazate pe aceasta metodă. Sunt prezentate pe rând realizările privind implementările de rețele neuronale (feed-forward, publicate în [6], [7] și [8], rețele competitive [9] și [10], rețele neuronale cu învățare on-chip [8], [11], [13], [14], [15], [16]. Aplicațiile rețelelor neuronale implementate hardware pentru recunoașterea gesturilor mâinii au fost publicate în [26], [27], [28], [29]. Alte aplicații dezvoltate se referă la un sistem olfactiv artificial prezentat în [30], interfețe inteligente om-mașina [32]

respectiv senzori inteligenți [6]. Alte contribuții legate de acest subiect sunt: implementarea funcției sigmoid in circuite de tip FPGA [19] și un nou simulator pentru circuite neuronale Feed-Forward [134]. Dintre acestea 8 sunt lucrări indexate ISI și 8 indexate BDI.

Contribuțiile aduse la tema a doua cu privire la sistemele dedicate inteligente pentru viață activă și asistată au fost dezvoltate în trei sub-direcții prezentate în Capitolul 2: Sisteme ambientale inteligente, rezultatele fiind publicate în [34], [38], [43], [45], [46], Sistem senzorial purtabil pentru monitorizarea activității și a stării de sănătate [33], [46], [47], [48], [54], [55], [57], [58], [59] și respectiv dezvoltarea de roboți pentru asistență personală [72], [73], [74], [75], [76], [77]. Patru dintre aceste publicații sunt indexate ISI și 12 indexate BDI.

In direcția a treia de cercetare contribuțiile aduse sunt legate de modelarea sistemelor de recunoaștere a activității umane în Matlab [47], [48], [80] și [81]; recunoașterea activității umane și a stării de sănătate [111], [112] și respectiv contribuții privind optimizarea metodelor de recunoaștere a activității umane [113], [114]. Aceste realizări au fost diseminate prin 2 articole în reviste ISI, 3 in publicațiile unor conferințe indexate ISI (una in curs de publicare).

Contribuțiile științifice au o vizibilitate bună fiind citate în mai mult de 270 de lucrări și cărți, dintre care 52 de citări independente în reviste sau conferințe cotate ISI, cărți și teze de doctorat.

Secțiunea a doua a lucrării prezintă succint evoluția mea profesională, sunt enumerați principalii colaboratori, contribuțiile aduse prin inițierea și coordonarea de programe de studii în țară și străinătate, introducerea de noi cursuri de nivel licență, masterat și doctorat, inițierea de colaborări academice și de cercetare cu parteneri interni și internaționali, realizări în activitatea didactică și de îndrumare a proiectelor de diplomă și de disertație și doctorat. Sunt prezentate de asemenea granturile și proiectele de cercetare pe care le-am condus sau în care am activat ca membru în echipa de cercetare.

Secțiunea a treia dedicată Planului de evoluție și dezvoltare a carierei schițează preconizata mea evoluție în cariera universitară. Aceasta indică obiectivele urmărite și acțiunile asociate pe privind componenta didactică respectiv componenta de cercetare. Sunt prezentate direcțiile de cercetare fundamentală și aplicativă pe care mi le propun, perspectivele cu privire la creșterea vizibilității rezultatelor cercetării, creșterea impactului publicațiilor științifice, participarea la proiecte internaționale respectiv conducerea de proiecte educaționale și de cercetare.

Partea finală include o listă de referințe cu 136 titluri, în care se regăsesc și 48 din publicațiile mele în tematica abordată în lucrare.

## Abstract

This work presents my research activity since I publicly defended my thesis and I obtained the Ph.D. title from the "Politehnica" University of Timisoara (2005). During this time I participated in more than 15 research and educational contracts and grants. As a result of my research activity I published in this period a total of 68 scientific papers including 3 in ISI journals with impact factor, 25 in conference proceedings indexed by ISI and 32 articles indexed in prestigious international databases. Also I published a book chapter related to the subject of my thesis in Springer Lecture Notes in Computer Science with the title "Application Possibilities of Hardware Implemented Hybrid Neural Networks to Support Independent Life of Elderly People", 2 books in national publishing houses and I have also developed 3 teaching materials available in electronic format.

The first section of the manuscript entitled "Scientific achievements", reviews my contributions to the development of intelligent embedded systems for active and assisted living (AAL).

These contributions are grouped into three research directions, which are presented each in a separate chapter of this thesis:

- Hardware implementation of artificial neural networks (ANN) using field programmable gate arrays (FPGAs) and applications development based in this method.
- E-Health and Ambient assisted systems, grouped around three sub-topics:
  - Ambient intelligent systems
  - Wearable systems for activity and health monitoring
  - Robots for personal assistance
- Contributions to development of new methods for activity recognition using combination of features extraction techniques and the use of neural networks for pattern recognition.

Chapter 1 presents the main contributions of the thesis related to the first direction of research, the development of the method proposed in my PhD work on the implementation of artificial neural networks in programmable circuits (FPGAs) and development of applications using this method. There are presented the achievements on the implementation of Feed-Forward (FF) neural networks reported in [6], [7], and [8], competitive networks [9] and [10], neural networks with on-chip learning [8] [11], [13], [14], [15], [16]. Hardware implemented neural network applications for hand gestures recognition were published in [26], [27], [28], [29]. Other applications developed refer to an artificial olfactory system presented in [30], intelligent man-machine interfaces [32] or intelligent sensors [6]. Other contributions related to this subject are: sigmoid function hardware implementation on FPGA circuits [19], and a new C++ implemented feed forward neural network

simulator [134]. Among this articles 8 are ISI indexed and 8 indexed in other international databases.

My contribution to the second theme were related to three sub-topics: Ambient intelligent systems results being published in [34], [38], [43], [45] [46], wearable systems for activity and health monitoring [33], [46], [47], [48], [54], [55], [57], [58], [59] and respectively development of robots for personal assistance [72], [73], [74], [75], [76], [77] respectively. Four articles are indexed by ISI and 12 in other international databases.

In the third direction of research my contributions are related to modeling in Matlab environment the human activity recognition systems [47], [48], [80], [81]; Human activity and health status recognition [111], [112] and studies regarding optimal recognition methods of human activity [113], [114]. This achievements were disseminated trough 2 ISI indexed journals and 3 at conferences with proceedings indexed by ISI (one accepted for publication).

My scientific contributions have a good visibility being cited in more than 270 scientific papers and books, 52 citations being in ISI indexed journals or conferences, books or PhD. thesis (excluding auto-citations).

The second section describes my professional achievements, presenting my main collaborators, my contributions regarding initiations and coordination of new study programs in Romania and abroad, development of new courses at B.Sc., M.Sc., and Ph.D. level, initiation of academic and research collaborations with Romanian and international partners, achievements in teaching activities and supervision of thesis at B.Sc., master, and PhD level. The grants and research projects that I leaded or I was involved in as a member of the research team are also presented.

Part three dedicated to the career evolution and development plan outlines my vision regarding the evolution of my academic career. It indicates the objectives pursued and the associated actions on the didactic component and the research component respectively. The fundamental and applied research direction that I envision, the perspectives on increasing the visibility of my research results, increasing the impact of my scientific publications, leading or participating into international educational and research projects, are also being presented here.

The final part includes a list of references with 136 titles which lists 48 of my publications related to the topics of this thesis.

# I. Scientific achievements

This habilitation thesis summarizes my scientific and academic activity, since 2005 when I got the Ph.D. title from the "Politehnica" University of Timisoara (domain: Electronic Engineering and Telecommunications), following the public presentation of the Ph.D. thesis entitled: "Sensorial system for hand gesture recognition using artificial neural networks"/ "Sistem senzorial pentru recunoașterea gesturilor unei mâni utilizând rețele neuronale artificiale".

My main scientific research directions and scientific achievements since 2005 can be grouped under the following thematic research directions:

- Hardware implementation of artificial neural networks (ANN) using FPGA circuits and applications development based in this method.
- E-Health and Ambient assisted systems, grouped around three sub-topics:
  - Ambient intelligent systems
  - Wearable systems for activity and health monitoring
  - Robots for personal assistance
- Development of new methods for activity recognition using combination of features extraction techniques and the use of neural networks for pattern recognition.

The main results of my research activity in the mentioned research directions are presented in a next part. Contributions are grouped and then presented within sub-themes.

**Scientific publications**

The result of the research activity within this period has been disseminated in a total of 70 scientific publications including:

- One book chapter related to the subject of this thesis in *Springer Lecture Notes in Computer Science* with the title "Application Possibilities of Hardware Implemented Hybrid Neural Networks to Support Independent Life of Elderly People";
- 3 ISI indexed journal articles with impact factor (one in Measurement Science Revue – Impact Factor 0.989 and two in Elektronika ir Elektrotechnika - Impact Factor 0.561;
- 25 papers in ISI indexed conference proceedings (IEEE Conference on Cognitive Infocommunications - CogInfoCom, IEEE International Conference on Intelligent Engineering Systems - INES, IEEE International Symposium on Signals, Circuits and Systems - ISSCS, IEEE International Symposium for Design and Technology in Electronic Packaging - SIITME, IEEE International Conference on Optimization of Electrical and

Electronic Equipment – OPTIM, IEEE International Carpathian Control Conference - ICCC, IEEE International Spring Seminar on Electronics Technology – ISSE, IEEE International Conference on Automation, Quality and Testing, Robotics – AQTR;

- 32 articles indexed in other prestigious international databases IEEE Xplore, Scopus, Inspec, Google Scholar, Mathematical Reviews, Zentralblatt Math, Ebsco, Proquest, Index Copernicus.

**International committees**

- More than 40 Program Committees of International Conferences including:
  - Embedded World Conference, Nuremberg, Germany 2014, 2015, 2016;
  - International Conference on Hybrid Artificial Intelligent Systems, HAIS, Salamanca, Spain 2013;
  - IEEE International Carpathian Control Conference, ICCC, 2010, 2012, 2014-2016;
  - International Conference on Soft Computing Models in Industrial and Environmental Applications, Burgos, Spain, 2015;
  - International Conference on Recent Achievements in Mechatronics, Automation, Computer Sciences and Robotics, MACRo, 2015;
  - International Symposium for Design and Technology in Electronic Packaging, SIITME, 2007-2014;
  - International Conference: Building Services, Mechanical and Building Industry Days, 2009-2011;
  - International Symposium on Applied Informatics and Related Areas, AIS, 2010;
  - Regional Conference on Embedded and Ambient Systems, RCEAS, Budapest, 2007;
  - International Workshop of Electromagnetic Compatibility, CEM 2007.
- Chair and co-chair:
  - Conference co-chair - Embedded Systems Design and Applications. 2008-2010;
  - Conference co-chair - Conference on Embedded Systems and Wireless Sensors Networks Design and Applications, ESWSDA, 2012;
  - Session chair - Regional Conference on Embedded and Ambient Systems, 2007.
- Reviewer for:
  - Infocommunications Journal, ISSN 2061-2079, (Scopus, Inspec, Compendex);
  - Carpathian Journal of Electronics and Computer Engineering, CJECE (BDI);
  - International Review of Applied Sciences and Engineering, IRASE (BDI);
  - Embedded world Conference;

- IEEE International Carpathian Control Conference (ISI indexed proceedings);
- International Conference on Recent Achievements in Mechatronics, Automation, Computer Sciences and Robotics, MACRo;
- International Conference on Soft Computing Models in Industrial and Environmental Applications.

The complete list can be found as well on my personal web page.

**International conferences and events**

**Founder and organizer** of a series of 6 international conferences

- Embedded Systems Design and Applications - ESDA, Baia Mare, Romania (2007-2010);
- Conference on Embedded Systems and Wireless Sensors Networks Design and Applications, 2011-2012, Debrecen Hungary.

Also I was involved in organization of several other e.g., symposiums, workshops, etc.:

- 13th International Symposium for Design and Technology of Electronics Packages - SIITME, Baia Mare, Romania, 2007
- Embedded Systems Design and Applications - Summer school Baia Mare, Romania 2007
- Workshops and Information days in collaboration with industrial partners: Analog Devices, National Instruments, Digilent Inc., Agilent Technologies, Mentor Graphics.

**National and international cooperation**

In research directions of this thesis I worked in joint research projects and/or published papers alongside prestigious professors from the:

- University Politehnica of Bucharest (Prof. Mircea Bodea and Prof. Dan Claudius, Ph.D. supervisors)
- Technical University of Cluj-Napoca (Prof. Corneliu Rusu, Prof. Eugen Lupu and Prof. Gheorghe Sebestyen, Ph.D. supervisors)
- Politehnica University of Timisoara (Prof. Virgil Tiponuț - Ph.D. supervisor)
- Beijing University of Posts and Telecommunications (Prof. Yongjiang Guo, Ph.D)
- University of Debrecen (Prof. Janos Vegh and Gyorgy Terdik, Ph.D. supervisors)
- Anglia Ruskin University, Cambridge (Prof. Marcian Cârstea and Dr. Alin Tisan).

During my research activity I established a significant network of collaborators from abroad, e.g., Prof. Andrei Vladimirescu - University of California at Berkley, Clint Cole - Washington State University and Digilent Inc. USA, Dr. Tom O'Dwyer and Dr. Stefan Marinca - Analog Devices, Prof. Yongjiang Guo - Beijing University of Posts and Telecommunications, Prof. Marcian Cârstea and Dr. Alin Tisan - Anglia Ruskin University, Cambridge, UK, Prof. Ján Turán - University of Technology, Košice, Slovakia, Prof. Gyorgy Terdik, Sztrik Janos and Prof. Husi Géza - University of Debrecen, Prof. Vegh Janos and Assoc. Prof. Vásárhelyi Janos - University of Miskolc, Olli Väänänen - Jyväskylä University of Applied Sciences, Finland, Andre Fiselier – Saxion University of Applied Sciences, Netherland, Cathal McCabe – Xilinx University Program, Prof. Florin Breaban - Université d'Artois – IUTB.

**Ph.D. student coordination**

During this period I collaborated with Ph.D. students who developed or are developing their doctoral thesis Daniel Mic (supervisor Prof. Emil Micu), Attila Buchman (supervisor Prof. Șerban Lungu), Alin Tisan (supervisor Prof. Lelia Feștilă), Ciprian Gavrincea (supervisor Prof. Virgil Tiponuț), Claudiu Lung, Ioan Orha, Sebastian Sabou (supervisor Prof. Dorin Petreuș).

Several PhD thesis were developed based partially on my method of hardware implementation of artificial neural networks:

- D. Mic, Contributions to the development of a hardware - software integrated environment for controlling electric motors, Brasov: Teza de doctorat Facultatea de Inginerie Electrică si Stiinta Calculatoarelor, Universitatea Transilvania din Brasov, 2007 [118];
- Tisan, Contributions to the analysis, synthesis and implementation of applications with intelligent sensorial systems: The electronic nose, Technical University of Cluj Napoca, 2009 [117];
- C. Gavrincea, Research regarding the implementation of a neural network used to process signals generated by the muscular and nervous system, “Politehnica” University of Timisoara, 2009 [120];
- C. Lung, Theoretical and experimental contributions to implementetion of intelligent embedded systems, Technical University of Cluj Napoca, 2013. [119];
- I. Orha, Human activity recognition and physiological parameters monitoring systems, Technical University of Cluj Napoca, 2015 [121].

I am currently the **Ph.D. supervisor** at University of Debrecen of the Ph.D. student József Sütő who is developing his thesis having the subject Activity recognition systems.

I am also currently leading the Intelligent Embedded Systems research center from Faculty of Engineering, where a research team consisting of researchers and students from the Technical University of Cluj-Napoca undertake both fundamental and applied researches. The scientific objectives are correlated with the development of intelligent embedded systems using field programmable gate arrays that have learning capabilities and adaptive behavior.

**Projects and strategic programmes**

For these research directions I lead and I was member of national research grants as follows:

- "Dezvoltarea parteneriatelor la nivel național și internațional, în domeniul Sisteme Dedicate, în vederea organizării de manifestări științifice și pregătirea de proiecte comune în programul cadru 7 UE"/"National and international partnership development in the field of Embedded Systems, with the aim of organizing scientific meetings and drafting projects cooperatively in the EU framework program - PASED" (CEEX-M3 PASED – contract no.: 253/01.08.2006). Duration: 2007-2010. *Role:* **director**. Financing institution: Romanian Ministry of Education and Research.

  ***Ph.D. student members of the research team***: Alin Tisan and Ciprian Gavrincea. ***Partners in this projects***:

  - University Politehnica of Bucharest (Prof. Mircea Bodea, Prof. Dragoș Burileanu, Prof. Dan Claudius, etc.)
  - Technical University of Cluj-Napoca (Prof. Corneliu Rusu, Prof. Eugen Lupu, Prof. Radu Arsinte, etc.).

  In the framework of this project we organized a series of scientific events: summer school, workshops, conferences, entitled "***Embedded systems design and applications***".

  ***Chairman:***

  Prof. Andrei Vladimirescu - University of California at Berkley, 2007, 2010
  Prof. Ján Turán - University of Technology, Košice Slovakia, 2008
  Prof. Corneliu Rusu - Technical University of Cluj-Napoca, 2009

  ***Keynote and Plenary speakers:***

  Prof. Andrei Vladimirescu - University of California at Berkley
  Prof. Mircea Bodea, Prof. Dan Claudius - University Politehnica of Bucharest
  Prof. Corneliu Rusu - Technical University of Cluj-Napoca
  Clint Cole – Washington State University and CEO Digilent Inc. USA
  Dr. Tom O'Dwyer - Analog Devices

Prof. Jan Turan - University of Technology, Košice Slovakia

Prof. Vegh Janos and Prof. Husi Geza - University of Debrecen,

Prof. Adam Tihamer and Assoc.Prof. Jozsef Vásárhelyi - University of Miskolc etc.

***Industrial partners:***

- Analog Devices, (Dr. Tom O'Dwyer, Dr. Marinca Stefan)
- National Instruments
- Texas Instruments
- Mentor Graphics,
- Digilent Inc. USA, etc.

***PhD forum organized free of charge for students from:***

- University Politehnica of Bucharest
- Technical University of Cluj-Napoca
- "Politehnica" University of Timisoara
- "Gheorghe Asachi" Technical University of Iasi
- Tyndall National Institute, Cork, Ireland
- University of Debrecen, Hungary
- University of Technology Kosice, Slovakia

- "Mediu integrat pentru deplasarea asistată a persoanelor cu handicap vizual" / "Integrated environment for assisted movement of visually impaired persons". Duration 2005-2007. Role: **researcher (**director Prof. Virgil Tiponuț). Financing institution: CNCSIS no. 639/2005. Ph.D. student involved: Gavrincea Ciprian.

- "Human activity recognition system using artificial neural networks" - Joint research project with Beijing University of Posts and Telecommunications, School of Science. Role: **Director**. Duration 25.05.2015 - 19.06.2015. Chinese partner: Prof. Yongjiang GUO, Ph.D. Financing institution: Balassi Institute, Campus Hungary Higher Education Staff Short Term Mobility, B2/4R/16048; 2015.

- "Future Internet Research, Services and Technology; Subproject: Internet of Things". TÁMOP-4.2.2.C-11/1/KONV-2012-0001 Hungary. Duration 2012-2014. Financing institutions: Hungarian National Development Agency. Grant awarded: 5.121.700 €. Project supported by the European Union, co-financed by the European Social Fund.

  Beneficiary: University of Debrecen. Consortia partners: Inter-University Cooperative Research Centre; Hungarian Academy of Sciences. Institute for Nuclear Research; etc.

**Role: scientific manager of the research theme**: "ICT tools for smart homes and assisted living for elders". The research group involved among others:

- Ph.D. student József Sütő,
- Researcher from Anglia Ruskin University: Dr. Alin Tisan
- Researchers from Technical University of Cluj-Napoca: Prof. Gheorghe Sebestyen, and Dr. Claudiu Lung, [135].

Besides the projects listed in the above section related to the theme of this thesis, I am/was involved in several other international and national research projects such us:

- Developing of a biophysical system based on zeolites microorganisms-vegetal species for ecoremediation of tailing ponds coming from gold-silver preparation industry - ZEMIP Consortium coordinator: dr. eng. Leonard Mihaly Cozmuta Marian - Chemistry-Biology Department North University of Baia Mare. Funding source: PNCDI 2, Partners: North University of Baia Mare, Johannesburg University South Africa. 2009-2011.
- Smart functions of packages containing nano-structured materials in food preservation – SMARTPACK, ERA-NET-MicroNanoTechnologies (MNT-ERANET), project 7-065, Director: dr. Anca Peter, Funding source: UEFISCDI. 2012-2015.
- Rehabilitation of tailing ponds by applying of amendaments and cultivation of vegetal species with high adaptability to heavy metals – RIVAM, Coordinator of consortium: dr. eng. Leonard Mihaly Cozmuta - Chemistry-Biology Department North University of Baia Mare. Funding source: CNMP - PNCD 2. 2008-2011.
- Bioaccumulation of heavy metals in soil-vegetable-human chain - BIOMEG, Coordinator of consortium: Associate prof. dr. Camelia Varga - Chemistry-Biology Department North University of Baia Mare, Funding source: CNMP - PNCD 2. 2008-2011
- Monitoring of soil microbiota action as it utilization in ecological rehabilitation of tailing ponds – AMSREI, Coordinator of consortium: dr. Monica Marian - Chemistry-Biology Department North University of Baia Mare, Funding source: CNMP - PNCD2. 2007-2010.

**The main contributions** that will be described in the next chapters and the corresponding publications are listed below:

- Hardware implementation of various neural networks
  - Implementation of Feed-Forward (FF) ANNs [6], [7], [8]
  - Optimizing FPGA Implementation of FF Neural Networks [7]
  - Competitive ANN [9], [10]
  - Self-Organizing Maps (SOM) network [11]

  - On-chip learning [8], [11], [13], [14], [15], [16]
  - Digital implementation of the sigmoid function for FPGA circuits [19]
  - A new C++ implemented feed-forward artificial neural network simulator [134]
  - Artificial neural networks application in pattern recognition: hand gesture recognition [26], [27], [28], [29], artificial olfaction system [30], intelligent Human-Machine Interface [32], smart sensors, smart devices [6].
- Intelligent embedded systems for daily life assistance
  - Ambient Intelligent systems [34], [38], [43], [45], [46]
  - Wearable systems for activity and health monitoring [33], [46], [47], [48], [54], [55], [57], [58], [59]
  - Development of an assistive robot [72], [73], [74], [75], [76], [77].
- Development of methods for activity recognition using neural networks
  - Contributions to the modeling of activity recognition systems [47], [48], [80], [81]
  - Human activity and health status recognition [111], [112]
  - Studies regarding optimal recognition methods of human activity [113], [114].

I would like to mention that several articles I authored or coauthored are not being referred in this thesis since they do not relate to the thematic directions mentioned above. I am referring here to:

- 11 publications in ISI indexed journals or conferences [123]-[133],
- 7 publications indexed in international databases addressing industrial electronics, power systems modelling, electric machines control, etc.

# 1 HARDWARE IMPLEMENTATION OF NEURAL NETWORKS USING FPGA CIRCUITS AND APPLICATIONS DEVELOPMENT BASED IN THIS METHOD.

## 1.1 Introduction

Development of a smart devices that behave and can also be operated naturally and also can be used without prior knowledge or settings of their parameters are nowadays requirements. For this reason many research groups worldwide try to develop adaptive and intelligent systems using Artificial Intelligence (AI) techniques based on biologically inspired intelligent algorithms such as: evolutionary computation, swarm intelligence, artificial immune systems, fuzzy systems, artificial neural networks, etc. [1]. Computational Intelligence (CI) is a sub-branch of AI which deals with the study of adaptive systems capable of intelligent behavior in a complex and constantly changing environment. The artificial neural networks are inspired from biological neural networks and can learn from examples to solve increasingly algorithmic models that are hard to describe using precise mathematical models. They proved their efficiency for example in tasks like pattern association, pattern recognition, control, etc.

My researches during my Ph.D. studies were related to the ANNs hardware implementation in FPGA circuits. During this period I developed a new method using Xilinx System Generator, an add-on for Matlab/Simulink which adds Xilinx blockset to Simulink library [2], [3]. Using this method I developed an HW/SW environment and a new library with ANNs' specific blocks that can be used for easy design of ANNs in order to add learning and adaptive behavior to a more complex system. As a proof of concept I designed and tested a Sensorial system for hand gesture recognition using artificial neural networks [4].

The following two subparagraphs present briefly the method developed in my Ph.D. work.

An ANN can be trained either offline or on-chip. I adopted for my Ph.D. thesis first approach, i.e. to train the network in Matlab with Neural Network Toolbox, to save the weights in a file and then load them automatically in hardware weight memories for hardware implementation. Another approach is to use on-chip learning. Both methods have advantages and drawbacks. For the former one is no need for computer and Matlab to train the ANNs, the training stage could take only a few minutes, and the ANNs could be easily adapted to new input conditions. But the hardware implementation of some training algorithm could be difficult, and the space occupied by hardware needed to implement the learning phase leads to space limitation for the circuit that implements the ANN [5]. The space limitation is always an issue in HW implementation of the ANNs because of the lack of a great number of hardware multipliers. This is why we have opted in most cases

for offline training of the network. A new ANN can be designed, instantiated and parameterized, connecting together the blocks from our library together with generic building blocks. This way different types of ANNs can be implemented, choosing the best suitable network for the given application.

### 1.1.1 Neurons hardware model

One of the first implementations that we made was the well-known McCulloch-Pitts model of the neuron, presented in Figure 1. The central element of the model is the summing junction having as inputs the weighted inputs of the neuron. It could be modeled using a hardware multiplier and an accumulator. The hardware multiplier blocks existing in Xilinx FPGAs can be used for an efficient implementation of the neuron's multiplier. We used one multiplier per neuron implementing in this way a neuron parallelism. Figure 2 shows the same neuron modeled using hardware blocks from the developed library.

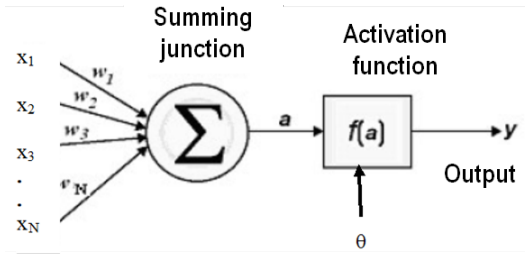



Figure 1. McCulloch-Pitts model of the neuron

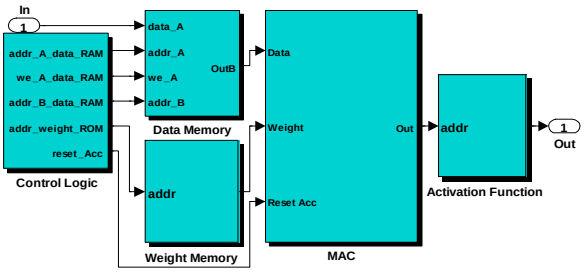



Figure 2. Neuron block hardware model [29].

$$y(x) = f(a-\theta) = f\left(\sum_{i=1}^{N} w_i x_i - \theta\right) \quad (1)$$

### 1.1.2 ANN implementation using System generator

We developed an ANN library using System generator tool. It contains all necessary blocks for rapid prototyping of ANN (MAC blocks, activation function blocks, weight memory block, and control logic block). A new ANN can be designed instantiating, parameterizing and connecting together the newly created blocks and generic building blocks. In this way different types of ANNs can be implemented, choosing the best suitable type for the given application.

With the method described above we have designed several types of ANNs such us Feed- Forward Back Propagation networks trained using correlation or Levenberg-Marquardt algorithm [6], [7], [8], Competitive network [9], [10], Self-Organizing Maps (SOM) network [11], and also we implemented some hybrid networks.

A selection from these works is presented in the next sections.

## 1.2 Implementation of FF ANN with neuron parallelism

The Feed-Forward Artificial Neural Networks are among the most widely employed artificial neural networks in applications that have as goal to establish an association between a set of input models and a set of target models. An FF type network has a layer organized structure. Each layer receives signals only from the layer situated immediately beneath it and sends signals only to the layer found immediately above it [12].

The output value of each neuron is determined by the equation:

$$y_j = f\left(\sum_{i=k}^{N} w_{ij}\, x_i - \theta\right) \tag{2}$$

The implementation of FF ANN with neuron parallelism represents a compromise between speed performance, on one hand, and the resources needed to implement and flexibility on the other hand.

### 1.2.1 Training the ANN

Training a neural network is achieved by using a set of input data and the corresponding target data. The ANN modifies its weights in order to learn correct associations between the inputs and the outputs. There are several methods of training an ANN but widely used are the descending gradient and the correlation method or algorithm.

No matter what the chosen training algorithm is, it can be executed either in real time on the chip, or offline. In this research we opted for the second training method, due to the advantages related to the simplicity of the implementation and to the versatility, so much needed in the prototype development phase.

### 1.2.2 Hardware implementation of a FF ANN with neuron parallelism

The structure of the neuron is similar with the type presented in Figure 2 but contains only a MAC block and a memory to store the weights. The data memory and the logic control block are common for all neurons. In this way we implemented neural layers with different number of neurons. The results were presented in [7]. Figure 3 presents a neural layer composed of 7 neurons. Due to the fact that the neurons in the next layer do not need all the data simultaneously, the multiplexer transmits the output data of the ANN layer in a sequential manner. As a consequence, the activation function block can be common to all the neurons in the layer. The data memory, of dual port RAM type, has M+1 locations, where M is the input vector size, while 1 location is needed for the bias. The memory is filled with the data present at the input of the ANN as the data is entered and it is read at a M+1 times greater speed, so that while the last element of the input

vector is filled, at the output B of the memory the M inputs are available, along with the bias. The command block coordinates the input data and the weights reading, while providing the command signals for the MAC operation.

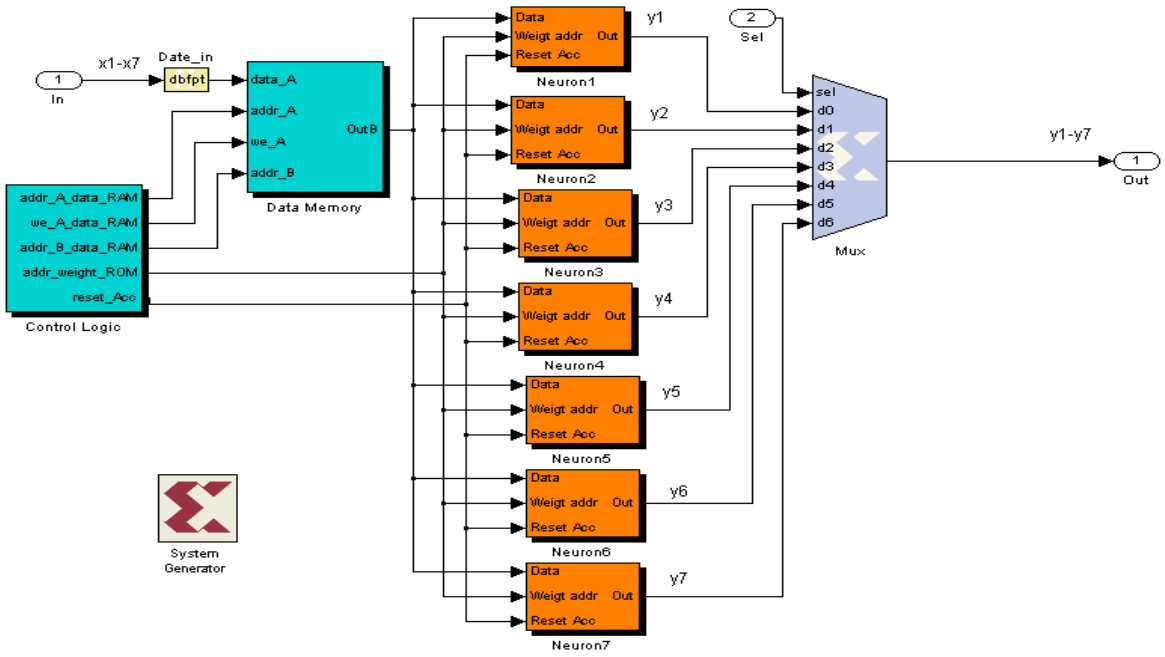



Figure 3. The hardware model of a neuronal layer composed of 7 neurons [7].

### 1.3 Implementing the ANN with a correlation learning rule

This section presents the results of the in depth study of FPGA implementation of Feed-Forward Neural Networks regarding error reduction as a function of number of bits used for weights representation, originally presented in [7]. Xilinx block parameters influence on resources occupied by the network and the maximum working frequency is also studied.

Correlation learning rule is one of the simplest learning methods used for ANN training. For test purpose we used this rule to train an artificial neural network composed of a layer of 7 neurons. For a pair consisting of the training signal $x_i$ on the *i*-th input and the desired output $t_j$, we can determine the resulting weight from *i*-th input to *j*-th neuron:

$$\Delta w_{ij} = x_i t_i \tag{3}$$

This training algorithm usually starts with initialization of weights to zero. The weights obtained are loaded into a newly created network using the Neural Network Toolbox. The results of the simulation, compared to the target vector allow us to determine the errors of the network. Thus, the ANN trained through the correlation algorithm, having the weights represented in double precision, generates at the output 3 erroneous vectors as a response to the 15 test vectors applied

at the input, which is a recognition rate of 80%. The weights calculation precision must be as high as possible in order to obtain ANN convergence and correct synaptic weights and thus less quantization errors in the final implementation. This is why the learning stage is best suited to the software implementation, using a floating point representation, which leads to a convergence and an accuracy of the network. The precision of the weights in the propagation phase can be lessened the error resulting being negligible. In order to calculate the errors of the network at different representation precisions of the weights, we designed a Matlab code. Reducing the weights' representation precision and simulating again the network for another set of 150x7 input and test vectors we obtained the following graphic of errors as function of number of bits for the software model of the ANN.

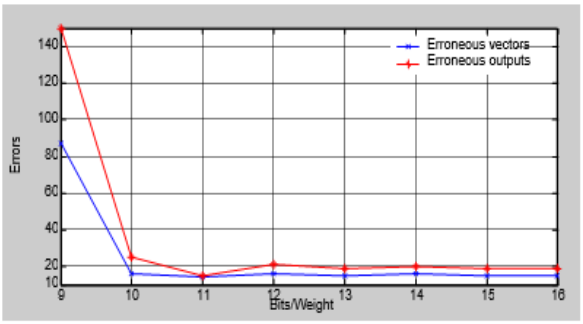



Figure 4. The errors of the ANN software model

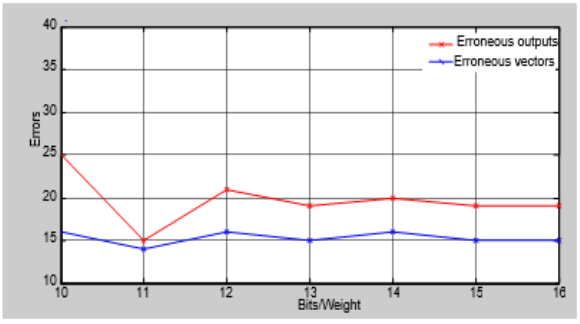



Figure 5. The errors of the ANN hardware model

It can be noticed that the number bits used for representation of the weights could be as low as 11 and even so the correct association of the vectors will be in this case 90,66%. For the hardware model (System Generator), using different representation precisions for the weights the same number of errors is obtained as in the case of software simulation. Figure 5 present a detail for the weights representation precision between 10 and 16 bits/weight. The minimum number of errors is obtained for an 11 bit precision in the representation of the weights. The next table presents the results of the hardware implementation of the network, for different weights representation precisions.

Table 1.
ANN performances and resources for different precisions [7].

| No. of bits / weight | Erroneous vectors | Slices | Max frequency (MHz) |
|---|---|---|---|
| 8 | 145 | 336 | 117.976 |
| 9 | 87 | 364 | 114.151 |
| 10 | 16 | 378 | 113.362 |
| **11** | **14** | **399** | **112.583** |
| 12 | 16 | 420 | 111.815 |
| 13 | 15 | 441 | 111.092 |
| 14 | 16 | 455 | 110.377 |
| 15 | 15 | 483 | 109.705 |
| 16 | 15 | 497 | 109.074 |

### 1.3.1 Optimizing FPGA Implementation of Feed-Forward Neural Networks

Upon identifying the optimal representation precision for the weight from the errors point of view, we can pass on to the optimization of the network implementation parameters, that is: the minimization of the resources used and the maximization of the working frequency. In order to achieve this, a study of the two parameters as a function of miscellaneous factors was conducted. We analyzed the influence of factors as: the number of bits the multiplier uses, the way quantization is made within the multiplier and convertor blocks and the way overflow is treated in the multiplier, accumulator and convertor blocks. The number of bits used by the multiplier greatly influences the resources used up by the ANN. The number of bits necessary to achieve maximum precision in the multiplier is given by the relation:

$$nbm_{\max} = n_i + n_p \tag{4}$$

Where $n_i$ and $p_i$ represent the number of bits utilized for date and weights representation. For the example used, $nbm_{max}$= 8+11=19. Even though apparently reducing the number of bits of the multiplier leads to a reduction of the resources used by it, in reality this doesn't happen in any situation. If we use a number of bits smaller than the one recommended by the equation (4) decisions about the way quantization is achieved must be taken (by truncating or rounding) respectively how an overflow is treated (saturation or warp). Using the saturation (in case of an overflow) and rounding (in the case of quantization) options requires supplemental calculus logic thus reducing speed performances.

The minimum number of bits necessary to represent correctly the integer part of the output of the multiplier can be determined using the relation proposed by the author:

$$nbm_{\min} = \text{ceil}(\log_2(\text{round}(\max(x_{\max} * \max(W), \text{abs}(x_{\max} * \min(W)))))) + 1 \tag{5}$$

where $x_{max}$ represent the maximum value possible for an input, W is the weight matrix, and 1 bit is needed for the sign representation. In the case of our test vectors and coefficients, resulted $nbm_{min}$= 5. The number of bits for the fractional part are kept equal to the number of bits used to represent the weights, i.e. 11. Thus, the representation precision of the output of the multiplier can be modified between the maximum Fix_19.11 and the minimum Fix_16.11.

In order to determine the number of bits that are to be used in the accumulator, the following equation was determined:

$$nba_{\max} = (n_i + n_p) + ceil(\log_2(M + 1)) \tag{6}$$

where M represents the number of neuron synapses. The maximum and respectively the minimum value at the output of the accumulator are given by the equations:

$$Y_{acc\_\max} = \max(W * X) \tag{7}$$

$$Y_{acc\_\min} = \min(W * X) \tag{8}$$

The number of bits used to represent the integral part can be reduced up to:

$$nba_{\min} = \text{ceil}(\log_2(\text{round}(\max(\max(W * X), \text{abs}(\min(W * X)))))) + 1 \tag{9}$$

The minimum number of bits used by the accumulator in limited by the constraint imposed by the Xilinx block for which the number of bits of the output cannot be lower than the number of bits of the input.

A synthesis of the results obtained following the implementation of the ANN for miscellaneous settings of the multiplier, accumulator and convertor blocks is presented in Table 2:

Table 2. Dependency of resources used and work frequency upon the parameters of the component block [7].

| | **Multiplier Bits** | **Multiplier Quantization** | **Multiplier Overflow** | **Accumulator Bits** | **Accumulator Overflow** | **Converter Quantization** | **Converter Overflow** | **Slices** | **F.max. (MHz)** |
|---|---|---|---|---|---|---|---|---|---|
| 1 | $nbm_{max}$=19 | - | - | $nba_{max}$=22 | saturation | rounding | saturation | 473 | 112.583 |
| 2 | -//- | - | - | $n_i+n_p$=19 | saturation | rounding | saturation | 430 | 115.018 |
| 3 | -//- | - | - | -//- | wrap | rounding | saturation | 353 | 150.421 |
| 4 | -//- | - | - | -//- | wrap | truncating | saturation | 354 | 149.970 |
| 5 | -//- | - | - | -//- | wrap | truncating | wrap | 304 | 179.122 |
| 6 | -//- | - | - | -//- | wrap | rounding | wrap | 269 | 179.122 |
| 7 | $nbm_{min}$=16 | truncating | saturation | -//- | wrap | rounding | wrap | 350 | 162.364 |
| 8 | -//- | rounding | saturation | -//- | wrap | rounding | wrap | 350 | 162.364 |
| 9 | -//- | rounding | wrap | -//- | wrap | rounding | wrap | 269 | 179.122 |
| 10 | -//- | truncating | wrap | -//- | wrap | rounding | wrap | 269 | 179.122 |
| 11 | -//- | truncating | saturation | -//- | saturation | rounding | saturation | 521 | 106.216 |
| 12 | -//- | rounding | wrap | $nbm_{min}$=16 | wrap | rounding | wrap | 269 | 184.680 |
| 13 | -//- | truncating | wrap | $nbm_{min}$=16 | wrap | truncating | wrap | 283 | 184.680 |
| 14 | $nbm_{max}$=19 | - | - | $nba_{max}$=22 | wrap | rounding | wrap | 269 | 174.344 |

Line 11 of the table above represents the most unfavorable case. Even though the number of bits used for the multiplier is lower than the one corresponding to the maximum precision, the supplemental logic required in order to implement the saturation option is costly regarding resources usage and working frequency. The most advantageous case in absolute values is presented in line 12. Comparing the performances of this setting with those in line 6, it is noticeable that the resources used are the same, even if the multiplier and the accumulator work using a smaller number of bits (16 compared to 19), but the work frequency increases. Even so,

the most advantageous settings are those in line 14, because the resources used are minimal, even at the maximum precision for the multiplier and the accumulator, so that no supplemental calculations are necessary in order to determine the minimum number of bits for the two blocks. The speed performance decrease is not noticeable.

Further optimization specific to the synthesis and implementation process can be conducted. The synthesis can be area optimized, fact that will lead to a further decrease of the resources utilized. For example, for the settings in line 12, the implementation occupies 255 slices. Also if an optimization of the instantiated primitives is performed, the working frequency can increase to 186.916 MHz. Out of the implementation report we can conclude that it is possible to implement 40 neurons using the 40 dedicated multipliers available in the device. The maximum number of connections per second that can be executed by the 40 multipliers available in the device is 7.477 GMAC/s.

The ANN can be implemented also using distributed resources available, without using the dedicated multipliers. In this case the maximum working frequency is 155.892 MHz.

For a combined implementation of the MAC blocks using all the 40 dedicated multipliers as well as the distributed logic from within a Virtex-II XC2V1000 device a maximum number of 101 neurons can be implemented, using up almost all the resources.

The maximum working frequency in this case is 155,892 MHz, so the 101 neurons could achieve 15.75 MMAC/s.

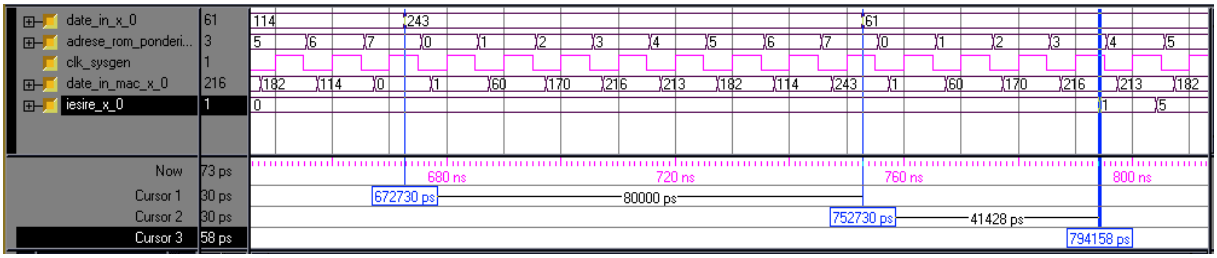



Figure 6. The simulation of the ANN hardware implementation [7].

A significant reduction of the resources used can be obtained if the latency time of the multipliers is reduces from 3 to 1. In this case, the necessary resources decrease from 28 to 12 slices and the time it takes to calculate the answer also decreases to 2 clock cycles.

### 1.3.2 Conclusions

The research presented in this paragraph pursued the study of the effects that the parameters of blocks dedicated to neural network design have on the performance of a neural network. The results gathered represent an important support for the designers of neural networks implemented

in FPGA. We have implemented FF ANN with one and two layers. The networks are trained using two different methods, correlation and Levenberg-Marquardt learning rule. This paragraph presents the results obtained with supervised correlation training method.

Studying the effect of the number of bits used to represent the weights proved that 11-12 bits is usually sufficient in order to achieve the same precision as the one achieved by the software model that utilizes a 64 bit representation for the weights. Another conclusion refers to the importance of the way weights precision decrease is achieved. Through adequate decrease, we can obtain the same precision as for the software model, as shown in Figure 5.

The errors were also studied function of the test vectors and the network neurons. An important space was dedicated to the influence of the parameters of the component blocks on the resources utilized and on the maximum work frequency. The latency time of the multiplier and the way quantization is achieved, respectively the way overflow is handled have the major influence over the resources utilized. It was proven that the most useful options are the latency time 1, the quantization by truncation and wrap for the overflow. Thus, the resources necessary for a single MAC were reduced from approximately 60 slices to 12 slices.

### 1.4 Implementation of a FF-BP ANN

The training algorithm with errors backpropagation involves changing of the weights to minimize the cost function. The standard training algorithm has disadvantages like the slow convergence that is dependent on network parameters. The fast algorithms converge to the optimum values in fewer epochs. Their disadvantage is the complexity of the calculations, so fewer driving epochs does not mean necessarily a shorter time. One of the fast training algorithm is the Levenberg-Marquardt (LM), which was used in my work for FF-BP ANNs training. Levenberg-Marquardt algorithm, as the cvasi-Newton method, eliminates the necessity of hessian calculus (the second order partial derivate of the cost function) that is used in the Newton method. The cvasi-Newton method approximates the hessian of cost function with the matrix presented by the relation (10):

$$H = J^T J \tag{10}$$

The gradient is calculated as in (11).

$$g = J^T e \tag{11}$$

where J is the Jacobian matrix which contains first order partial derivate of the errors with respect to the weights, and **e** is the network's error vector.

$$\mathbf{J(x)} = \begin{bmatrix} \frac{\partial e_{1,1}}{\partial w^1_{1,1}} & \frac{\partial e_{1,1}}{\partial w^1_{1,2}} & \cdots & \frac{\partial e_{1,1}}{\partial w^1_{S^1,R}} & \frac{\partial e_{1,1}}{\partial b^1_1} & \cdots \\ \frac{\partial e_{2,1}}{\partial w^1_{1,1}} & \frac{\partial e_{2,1}}{\partial w^1_{1,2}} & \cdots & \frac{\partial e_{2,1}}{\partial w^1_{S^1,R}} & \frac{\partial e_{2,1}}{\partial b^1_1} & \cdots \\ \vdots & \vdots & & \vdots & \vdots & \\ \frac{\partial e_{S^M,1}}{\partial w^1_{1,1}} & \frac{\partial e_{S^M,1}}{\partial w^1_{1,2}} & \cdots & \frac{\partial e_{e_{S^M,1}}}{\partial w^1_{S^1,R}} & \frac{\partial e_{e_{S^M,1}}}{\partial b^1_1} & \cdots \\ \frac{\partial e_{1,2}}{\partial w^1_{1,1}} & \frac{\partial e_{1,2}}{\partial w^1_{1,2}} & \cdots & \frac{\partial e_{1,2}}{\partial w^1_{S^1,R}} & \frac{\partial e_{1,2}}{\partial b^1_1} & \cdots \\ \vdots & \vdots & & \vdots & \vdots & \end{bmatrix} \tag{12}$$

The weight update could be done using (13):

$$w_{k+1} = w_k - [J^T J + \mu I]^{-1} J^T e \tag{13}$$

where I is the identity matrix and μ is the control parameter. The network could be trained using Neural Network Toolbox or a Matlab script.

### 1.4.1 Implementation of a FF-BP ANN with one layer

In [6] we presented a software implementation of the ANN training phase respectively a hardware implementation for the propagation phase. The neural networks implemented were trained through several training methods. One of the training methods that were tested is the Levenberg-Marquardt method that is part of the fast training algorithms for a FF-BP ANN.

The weights of the trained ANN can be saved in a weights file. After training, the network can also be loaded in Simulink for simulations. This model can serve as standard for the hardware model that will be developed using System Generator. In order to implement an ANN with neuron parallelism, a few particularities must be observed. For this, the structure of the neuron presented in Figure 2 is modified in such a manner that it will contain only one MAC block and one weights memory. The structure of the ANN is similar with that presented in Figure 3.

For the implementation, the representation precision for the weights, which were calculated in double precision, is reduced. To modify the way weights precision representation is reduced, before loading them into the ROM memory, an explicit reduction of precision can be made. We can use for example equation (14) to represent weight in fixed point representation.

```
q = quantizer( 'fixed',  'fix',  'saturate', [nb bp]);
pond=round(q, ponderi);
```
(14)

where q represents the quantization parameters.

The quantization is achieved by rounding to the closest value to zero, expressed on the number of bits specified by "nb" of which "bp" are the bits used to express the fractional part. The overflow is treated by saturation. Testing various solutions for reducing precision we obtained the results presented in Table 3.

Table 3. Number bits used in weights implementation [6].

| **Quantization** | **Number of erroneous vectors** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| No. of bits / Method | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| round | 109 | 50 | 44 | 60 | 37 | 39 | 41 | 41 | 41 |
| fix | 123 | 78 | 19 | 22 | 35 | 37 | 38 | 39 | 41 |
| ceil | 150 | 150 | 141 | 101 | 77 | 53 | 49 | 48 | 46 |
| floor | 150 | 147 | 82 | **10** | 23 | 24 | 30 | 38 | 41 |
| convergent | 109 | 50 | 44 | 60 | 37 | 39 | 41 | 41 | 41 |

The minimum number of errors (10) was obtained for the rounding to the closest value to minus infinite (floor), for an 11 bit representation of the weights. The precision of the hardware implemented network becomes in this way 93.33%.

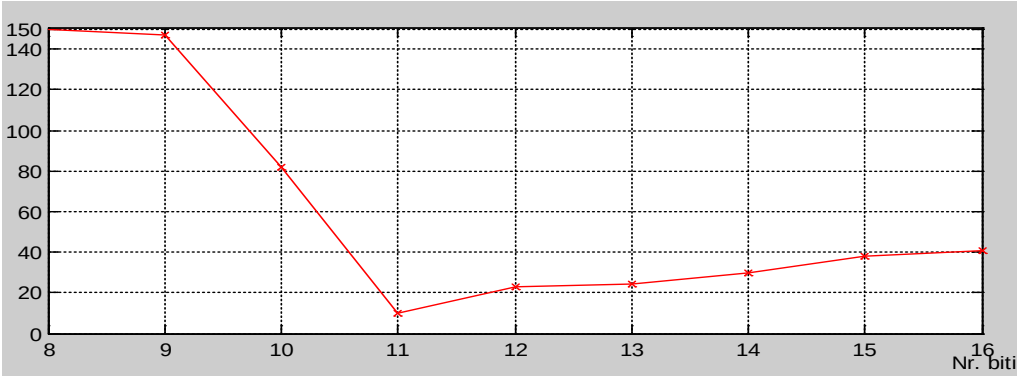



Figure 7. Errors of the 1x7 FF-BP-LM ANN implemented with weights correction [6].

A part of the post-implementation simulation with ModelSim using the testbench and the test vectors generated by the Simulink/System Generator model is presented in Figure 8. The frequency of the clock used is 100 MHz. The resources used by a neuron are 14 slices, a dedicated multiplier and a Block RAM memory, while the maximum working frequency after the optimization is 186,916 MHz.

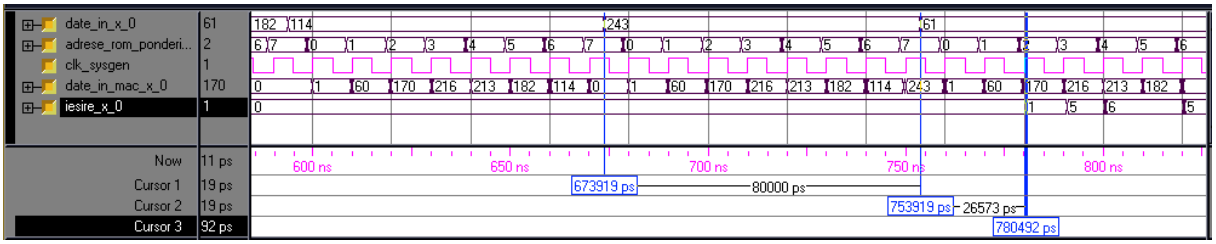



Figure 8. Post implementation simulation of the FF-BP-LM 1x7 ANN.

## 1.5 Implementing a competitive ANN

The competitive and the self-organizing neural networks, represent one of the most interesting types of the artificial neural networks. The way information is organized in the human brain inspired these types of ANN. The task of the competitive networks is to classify input models. Similar models are classified into the same class, represented by the same output unit. Each neuron will become specialized in recognizing certain features of the input data.

Following is presented the successful implementation of some simple competitive neural networks used in model classification tasks in FPGAs [9], [10]. The network design was carried out using the System Generator software, which was also used to generate the VHDL code for the network.

### 1.5.1 The structure of competitive ANNs

Self-organizing neural networks are characterized by the fact that they actually learn unsupervised to discover features, regular patterns and correlations of the input data. The neurons of the simple competitive networks are arranged in a one-dimensional output layer, totally connected to the neurons of the input layer through some stimulating weights. They are permanently in a competitive state, at a certain point in time only one being active.

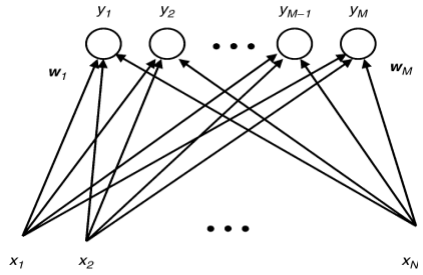



Figure 9. ANN with simple competitive learning.

Each neuron has as many input connections as the number of attributes used in classifying. Starting with a set of initials weights, randomly generated, the training procedure consists in finding the neuron that has the closest weights compared to the input vector and declaring that neuron as winner. There are two methods of determining similarity. The first determines the net output of *k*-th neuron according to the equation:

$$net_k = \sum_{i}^{N} w_{ki} x_i \tag{15}$$

The neuron for which the output has the highest value is declared the winner. The second method calculates the distance between the input vector and the weights vector, the most commonly used being the Euclidian distance, as in the following equation:

$$net_k = \sqrt{\sum_i^N \left[x_i - w_{ki}\right]^2} \tag{16}$$

The neuron for which the equation (16) has the smallest value will win the competition.

The learning process consists of changing the weights according to the equation:

$$\Delta w_{ji} = \eta(x_j - w_{ji}) \tag{17}$$

For the j neuron which won the competition, and

$$\Delta w_{ki} = 0, \tag{18}$$

for the neuron k $\neq$ j. η represents the learning rate.

Through this, the weights vector of the winning neuron, j, gets closer to the pattern present at the input. Upon the completion of the training, the classification process consists of calculating the distance between the input vector and each neuron and declaring the input as pertaining to the class represented by the winning neuron.

The training phase of the ANN was software implemented, while the propagation phase was implemented hardware. We will present below the way the network was trained and subsequently implemented hardware in the propagation phase.

### 1.5.2 Training a simple competitive ANN

In order to train a competitive ANN, a simple ANN was created by using Matlab code or by using the graphical interface Network/Data Manager. After training the network with a set of 10x15 test vectors and simulating it, we get the results presented next:

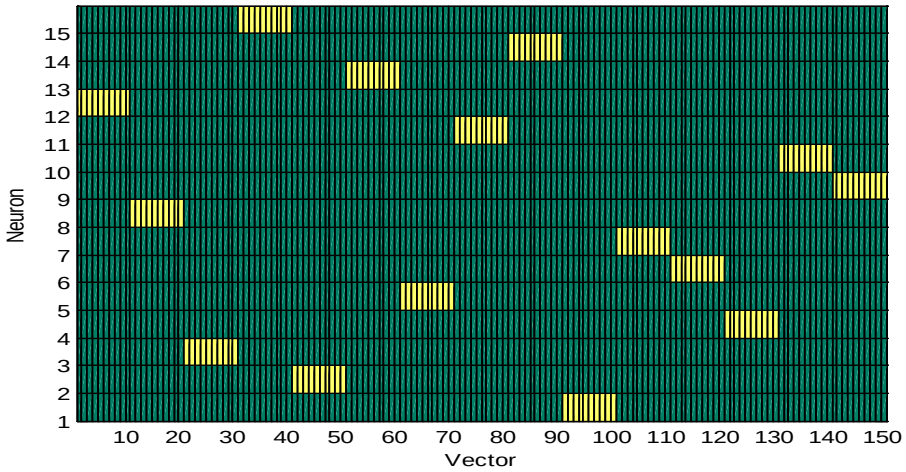



Figure 10. The results of the competitive ANN simulation [10].

It is noticeable that, at a certain point in time, only one neuron is active and that the 15 sets of 10 vectors are each divided into 15 different classes. The first set of 10 vectors is part of class 12, the second set of class 8, and so on. The results of the simulation being as expected (the classification in 15 different classes) the weights of the network are saved at this point.

### 1.5.3 The hardware model for the propagation phase of a competitive ANN

We implemented the competitive ANN using Xilinx blocks. Calculating the Euclidian distance using equation (16) is difficult because of the square root operation that is hard to implement hardware. Due to the fact that the comparison between the neurons can also be achieved by using the square of the distances between the input vectors and the corresponding weights, implementing the square root calculation is not necessary, so the equation (16) can be replaced by:

$$y_k = \sum_{i}^{N} \left[ x_i - w_{ki} \right]^2 \tag{19}$$

To achieve this, the structure of the neuron will contain a memory for the weights, a block used to calculate the $x_i$-$w_{ki}$ subtractions, a MAC block used to calculate the sum of the squares of these subtractions and, finally, the block of the competitive activation function, as presented in Figure 11. The activation function block is common for all the neurons in the output layer. The Negdist block consists of an AddSub Block, part of the Xilinx Blockset and a MAC block created by the author.

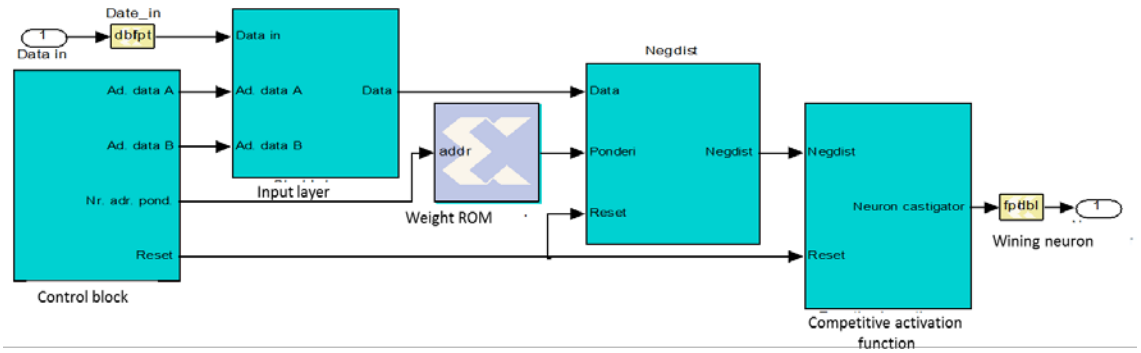



Figure 11. The hardware model of the competitive ANN [10].

The Addsub block is used to calculate de $x_i$-$w_{ki}$ subtractions, while the MAC block determines the sum of the squares of these subtractions. The activation function block, presented next, is composed of two parts. A first part determines the minimum of the $y_k$ type of inputs of the neurons that form the output layer, and the second part determines the position of the neuron which has the lowest output.

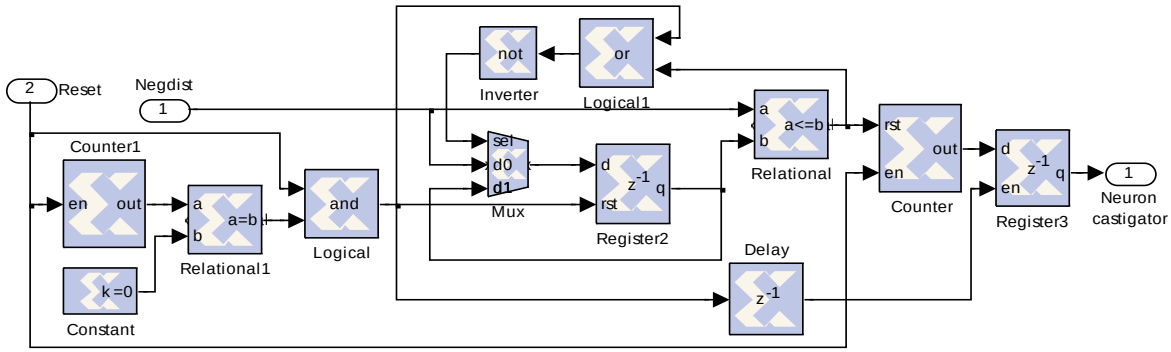
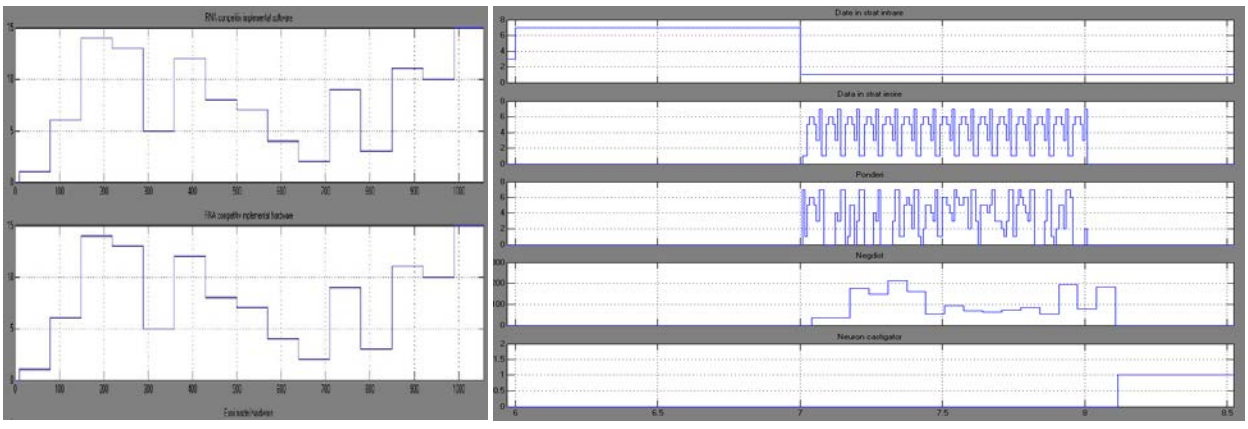



Figure 12. The competitive function activation block [10].

### 1.5.4 Implementing a competitive ANN with layer parallelism

In order to implement a competitive ANN with layer parallelism, a single processing element per layer is required. To verify the performance of a competitive ANN with layer parallelism, a network with an $n_1$=7 neurons input layer and an $n_2$=15 neurons output layer was implemented. The hardware model has the structure presented in Figure 11. The competitive ANN with layer parallelism consists of a control block, an input layer made with a memory of type Block RAM, a weights memory common to all the neurons, the simplified Euclidian distance calculus block and the competitive function activation block, also shared by all the neurons. The architecture is the same, no matter how many neurons are on the input, respectively on the output layer. The only differences are the parameters of the blocks, the capacity of the memories and the computational speed. The results of the network simulation are presented next.



a) b)

Figure 13. Software and hardware simulation of the competitive ANN model [10].

The upper graphic, in Figure 13.a represents the classification in the 15 classes of the 15x10 input vectors, performed with the software model while the lower graphic presents the classification made by the hardware model. In Figure 13.b, a detailed view of the hardware model simulation is presented. Following the application of a vector at the input of the input layer, this is applied to

the output layer at a frequency of $(n_1+1)n_2$ times higher due to the fact that the negdist block must execute $n_1+1$ computations for each of the $n_2$ neurons.

As we can see, there is no difference between the classification made by the hardware model compared to the one made by the software implementation. Simulating the hardware model using another set of 15x10 test vectors creates the same results, without any errors.

#### 1.5.4.1 Results

Table 4 shows the implementation report of the competitive ANN into a Virtex II XC2V1000.

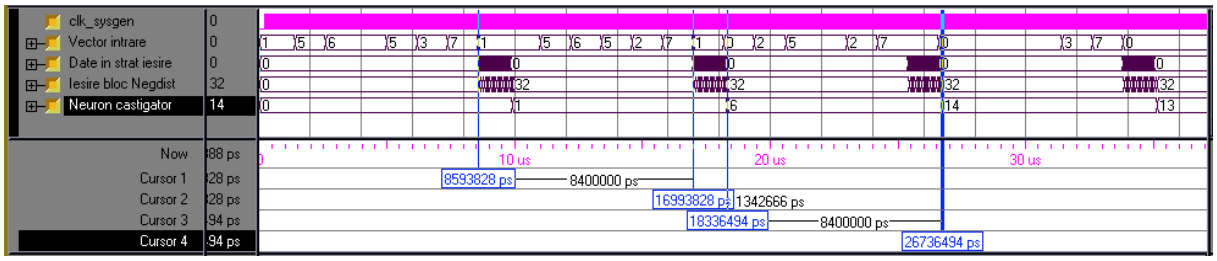
Table 4. Resources used by the competitive ANN [10].

| | Command block | Input layer | Weights Memory | Dist. calc. block | Activation function | TOTAL |
|---|---|---|---|---|---|---|
| **Slices** | 16 | 3 | 0 | 15 | 34 | 68 |
| **Flip-flops** | 15 | 4 | 3 | 25 | 27 | 74 |
| **RAM blocks** | 0 | 1 | 1 | 0 | 0 | 2 |
| **Mem. tables** | 24 | 0 | 0 | 12 | 40 | 76 |
| **Multipliers** | 0 | 0 | 0 | 1 | 0 | 1 |

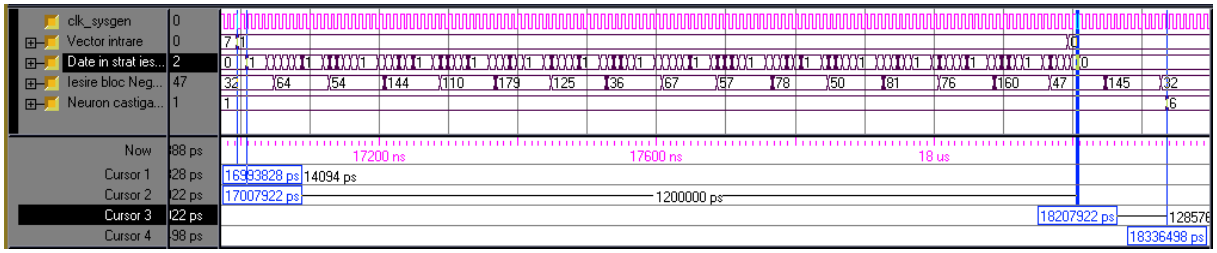
The results of the post-implementation simulation, presented in Figure 14, confirm the correct functioning of the implemented circuit. The elements of the input vector are fed to the input layer in a sequential manner with a frequency of 1/Ts, where Ts represents the duration of a vector element equal to:

$$T_s = (n_1+1)n_2T_{clk} = 1200 \text{ ns} \qquad (20)$$

for a clock signal frequency of 100 MHz.




a) Detail of the first 4 vectors




b) Detail on the computation times for the second test vector

Figure 14. The post implementation simulation of the ANN with layer parallelism [10].

The input layer successively transmits these elements to the output layer with a frequency of (n1+1)n2/Ts, allowing it to perform the (n1+1)n2 calculations. The neuron that wins the competition is determined after 128 ns. The maximum frequency of the clock signal resulted from the synthesis report is 136 MHz.

### 1.5.5 Implementing a competitive ANN with neuron parallelism

A competitive ANN with neuron parallelism consists of a control block, an input layer and an output layer. The output layer consists has a number of neuron equal to the number of classes in which we would like to classify the inputs. It also has an activation function calculus block, common to all the neurons. The neuron consists of a weights memory specific to each neuron and a calculus block used to calculate the simplified Euclidian distance. This block calculates the differences between the data and the weights, while the multiplication block calculates the square of these differences, the accumulator block sums up the squares of the differences, the results being stored temporally in the output register. In order to calculate the smallest value and determine the neuron which has the minimum output value, a block that calculates the competitive activation function has been implemented. The block is similar to the one in Figure 12. Because there is only one such block, the neuron outputs are fed in sequence to its input by the means of a 15 input multiplexer.

#### 1.5.5.1 Results

The results of the implementation of the competitive ANN with neuron parallelism in a FPGA XC2V1000 device were published initially in [9] and are presented in the Table 5.

Table 5. The resources utilized by the competitive ANN with neuron parallelism [9].

| | Com. blk. | Input Layer | 15 Neurons (res/neuron) | Neuron outputs multipliers | Activation function | TOTAL | % of XC2V1000 |
|---|---|---|---|---|---|---|---|
| Slices | 19 | 0 | 12 | 43 | 29 | 271 | 5.27 |
| Flip-flops | 17 | 0 | 18 | 2 | 23 | 312 | 3.05 |
| RAM blocks | 0 | 1 | 1 | 0 | 0 | 16 | 40 |
| Mem. tables | 29 | 0 | 12 | 81 | 35 | 325 | 3.18 |
| Multipliers | 0 | 0 | 1 | 0 | 0 | 15 | 37.5 |

The resources used in the implementation of a neuron are only 12 slices, 1 RAM block and one dedicated multiplier. The total of resources necessary to implement the competitive ANN with 15 neurons represents only a small percentage of the total resources available in the circuit, except the memory blocks and the dedicated multipliers which are used in 40 and respectively 37.5 percent. In order to implement more than 40 neurons, the multiplying block can be implemented using distributed logic instead of the dedicated multipliers, their number being limited to 40. In this case, the resources per neuron increase to 22 slices, 30 flip-flops, 1 RAM block, 31 LTUs, 0

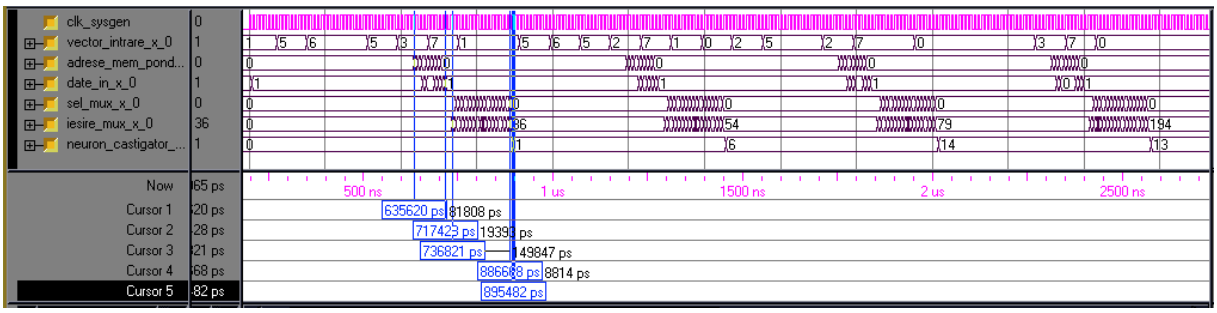
multipliers. Also, such memories can be implemented using the distributed memory available in the device (160 Kb) instead of using the block RAM memories. The necessary resources increase in this case with 3 slices, 3 flip-flops and 3 LUTs. We can estimate the maximum number of competitive neurons that can be implemented into this device, as in the following table:

Table 6. Resources utilized by the maximum number of neurons that can be implemented in the XC2V1000 [9].

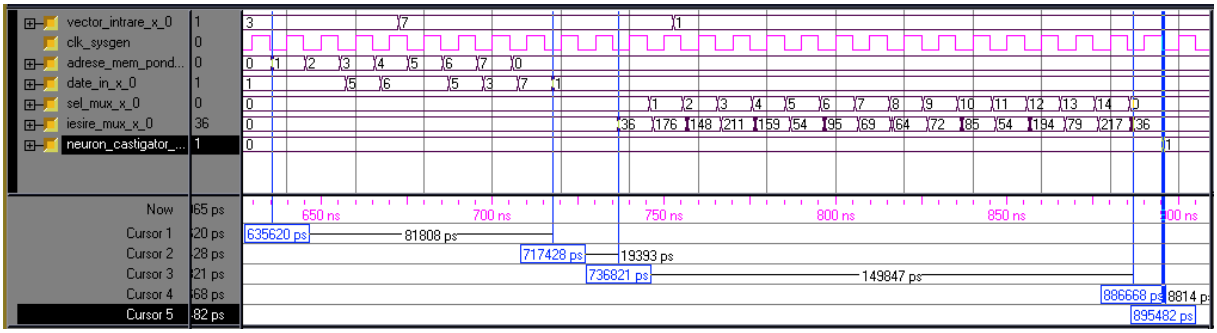
| | ANN 15 neurons ver. 1 | ANN 40 neurons ver. 1 | Total XC2V1000 | Available | ANN 15 Neurons ver. 2 | No of neurons | TOTAL 40 neurons, ver. 1+ 141 neurons, ver. 2 |
|---|---|---|---|---|---|---|---|
| Slices | 271 | 723 | 5120 | 4397 | 466 | 141 | 5103 |
| Flip-flops | 312 | 832 | 10240 | 9408 | 640 | 221 | 6848 |
| RAM blocks | 16 | 40 | 40 | 0 | 0 | - | 40 |
| LUTs | 325 | 867 | 10240 | 9373 | 655 | 215 | 7024 |
| Multipliers | 15 | 40 | 40 | 0 | 0 | - | 40 |

From the calculations we can estimate that the maximum number of neurons, with the corresponding circuitry, that can be implemented in the device chosen is around 181. Of these, 40 will be implemented with dedicated multipliers and 141 using distributed logic.

The results of the functional simulation of the implemented network, more precisely a detail on the first 4 test vectors and over the computational time intervals for the answer for the first test vector are presented in Figure 15.a and respectively Figure 15.b.




a) Detail of the first 4 vectors




b) Detail on the computation times for the first test vector

Figure 15. Functional simulation of the competitive ANN with neuron parallelism [9].

The simulation ran with a 100MHz clock signal. The elements of the input vector are written in the memory of the input layer in a sequential manner, at equal time intervals.

$$Ts = (n_1+1)T_{clk} = 80 \text{ ns} \quad (21)$$

The duration of the input vector reading is 560 ns.

The input layer transmits in sequence these elements, plus the bias, to the output layer, at time intervals of $Ts/(n_1+1)$. Each of the $n_2$ neurons of this layer makes $(n_1+1)$ computations. The transfer of the outputs to the block that computes the competitive function activation is done at 15 clock intervals. The neuron that wins the competition is determined after 8.8 ns. The results of the simulation show that the first test vector from the figure [1 5 6 6 5 3 7] was assigned the first class, the second vector [1 1 5 6 5 2 7] to the 6th class, and so on. The maximum frequency of the clock signal resulted from the synthesis report is 123.7 MHz. The maximum frequency with which the input vector elements can be sampled is:

$$Fs \max = Fclk \max/(n_1+1) = 15{,}45 \text{ MHz} \quad (22)$$

### 1.6 On-chip learning

As mentioned above the first implementations of ANN were SW/HW meaning that the learning phase is done in software using Matlab and the propagation phase was implemented in hardware (FPGAs). Thereafter we designed several ANNs with on-chip learning [8], [11], [13], [14], [15], [16]. This section describes my scientific contribution in this field. These contributions are related with the collaboration with Alin Tisan during his Ph.D. Some of this will be presented in next subparagraphs.

#### 1.6.1 Hardware implementation of a MLP network with on-chip learning

The multilayer perceptron (MLP) network generally gives quickly results, is efficient with information processing, and learns by presenting examples. The backpropagation learning algorithm consists of four different steps: (forward) propagation, error check, back propagation (BP) and weights update. First, the input values are propagated forward, using equations (23) and (24), to obtain the actual outputs.

$$net = bias + \sum_{k=1}^{N} w_k x_k \quad (23)$$

$$o = output(net) = \frac{1}{1+e^{-net}} \quad (24)$$

Second step requires calculating the average of the total as a sum of squared individual errors.

$$E = \sum_p e^p = \frac{1}{2}\sum_p (t^p - o^p)^2 \tag{25}$$

where O is the actual output and T is the desired value for a given pattern p.

Next, if the total error is greater than a given threshold the value of the weights must be adjusted in order to minimize the error function. This minimization is done in idea to make changes in the weight proportional to the negative of the derivative of the error - backpropagation step, [17], [18]

In the backpropagation step, the first thing to do is to obtain the gradient of the output neurons. This is done through:

$$\delta = -\frac{\partial E}{\partial net_k} = -\frac{\partial E}{\partial O_k}\frac{\partial O_k}{\partial net_k} \tag{26}$$

For

$$f'(net) = -\frac{\partial O_k}{\partial net_k} \tag{27}$$

and

$$\frac{\partial E}{\partial O_k} = -(T_k - O_k) \tag{28}$$

we obtained

$$\delta_k = (T_k - O_k) f'(net) \tag{29}$$

where O is the actual output, T is the desired value and f'(net) is the first derivative of the activation function and k is the number of the output neurons.

In the case of the sigmoid function the derivative is given by:

$$f'(net) = f(net)(1 - f(net)) = O(1 - O) \tag{30}$$

This value is then propagated backwards in order to obtain the gradient for each of the hidden layer neurons. For a given hidden neuron, its gradient is given by

$$\delta_j = f_j'(net_j)\sum_{k=1}^{K} \delta_k w_{kj} \tag{31}$$

The last step consists of updating the weights in order to minimize the error.

In the case of the neurons from the output layer the formula that gives the change of corresponding weight is:

$$\Delta w_k = \eta \delta_k y_j \tag{32}$$

where is the output of the neuron j from the previous layer.

In the case of the neurons from the hidden layers the change of corresponding weight is:

$$\Delta v_k = \eta \delta_j z_j \tag{33}$$

where $z_i$ is the output of the neuron i from the previous layer.

Because the BP learning algorithm is an iterative process the new epochs are repeated until the network is trained enough, i.e., until the error between the actual and the desired outputs is lower than a given threshold.

For the implementation we have adopted a node parallelism that requires one multiplier per neuron and so all neurons can work in parallel [8]. Our model has two major blocks: a logic control block and a processing block. The control logic block manage the control signal of the processing bloc in order to initialize and command the processing components.

The processing block is designed to calculate the neural output, the weights according to learning rule adopted, in this case Delta rule, and to update these weights. This block consist of the artificial neuron and the logic for on-chip learning algorithm.

The structure of the artificial neuron consist in one memory block, for data samples, one MAC unit and an activation unit, Figure 16.

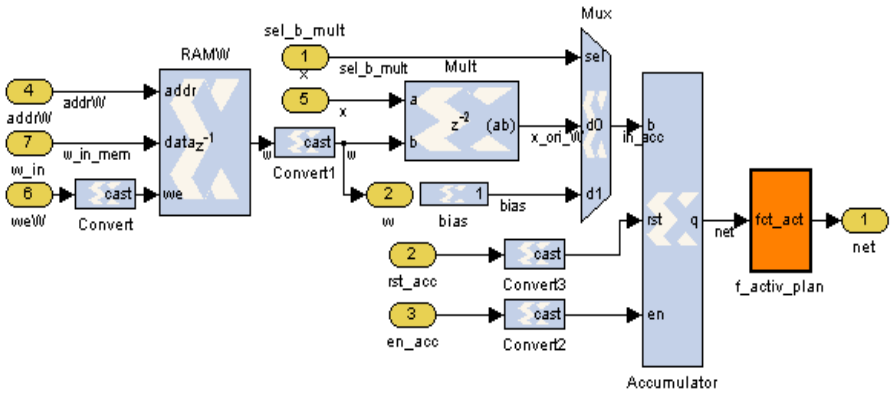



Figure 16. Architecture of the neuron [8].

For implementation in FPGA of the activation function was used the PLAN approximation. The PLAN approximation uses digital gates to directly transform from x to y. The calculations need

only be performed on the absolute value of the input x. In Figure 17 is shown the hardware architecture of the PLAN approximation of sigmoid function. The architecture includes beside comparators, multipliers, multiplexers and adders blocks, a black box that contains a VHDL file that control the multiplexer outputs data.

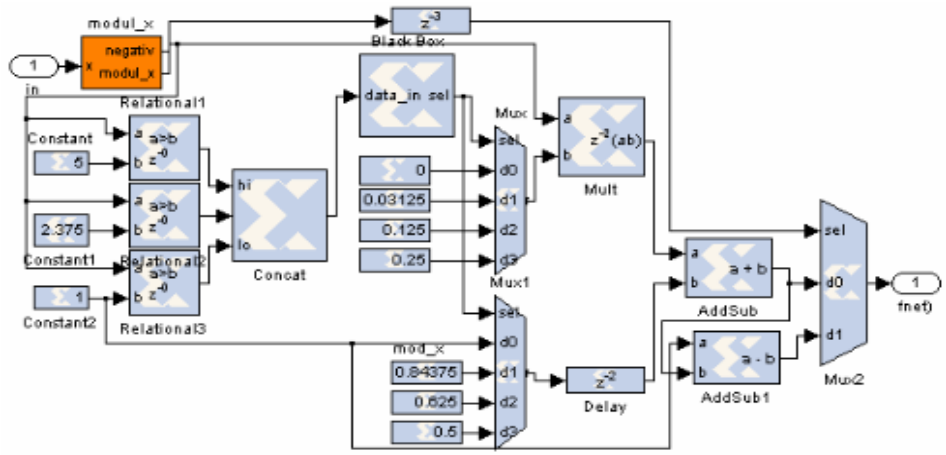



Figure 17. The hardware architecture of the PLAN approximation of sigmoid function [8].

**Error check block**

Another important block is the Error check block that deals with the calculus of the total error, the comparing with the given threshold Emax and the calculus of the error signal of the *k*-th neuron, (delta_k), Figure 18.

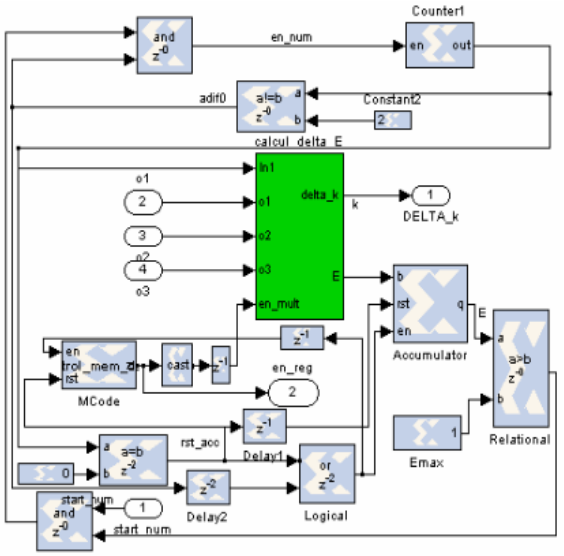



Figure 18. Hardware architecture of the Error check block [8].

**Calculus block of the output layer weights**

This block deals with the calculus of the new weights of the output layer. The block implements the calculus formula of the weights according to (32).

**Calculus block of the hidden layer weights**

This block contains in fact other three blocks that will calculate the weights according to the above formulas equations (31), (32) and (33), Figure 19:

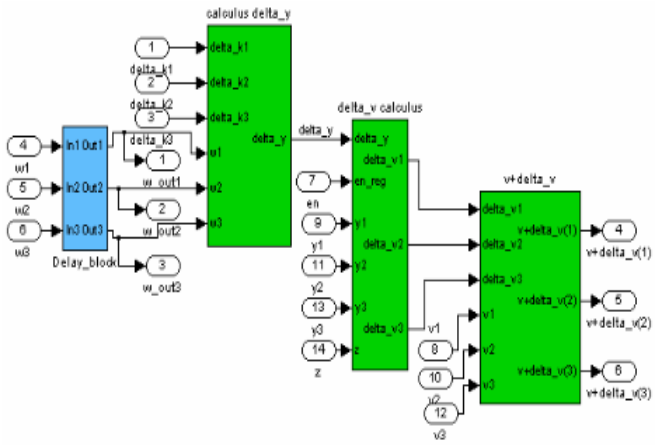



Figure 19. Block configuration of the Calculus block of the hidden layer weights [8].

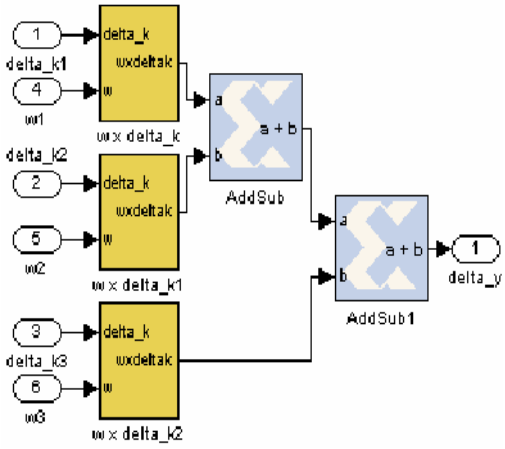



Figure 20. Hardware implementation of the Delta_ y calculus block [8].

All blocks implied in the calculus are resizable and can be changed by the parameters given from outside.

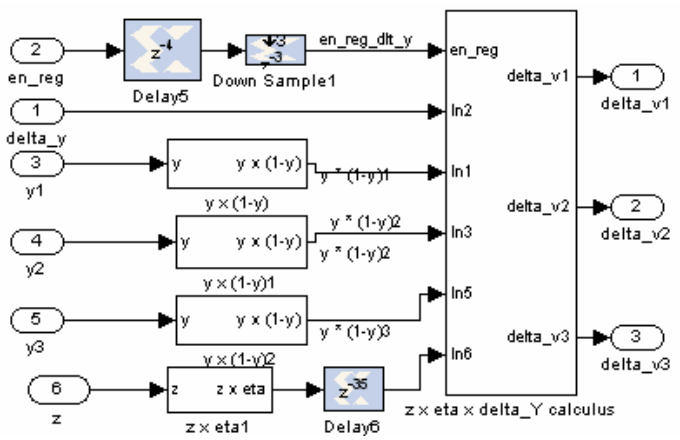



Figure 21. Hardware implementation of the Delta_ v calculus block [8].

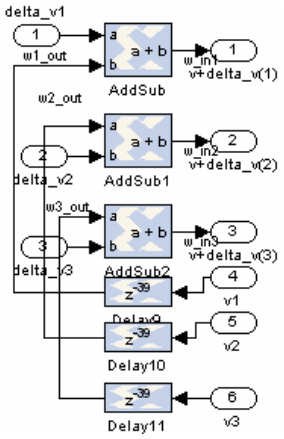



Figure 22. Hardware implementation of the changed weights [8].

The above figures presents the implementation of the following formulas:

$$\delta_y = y_j(1-y_j)\sum_{k=1}^{K}(T_k - O_k)(1-O_k)O_k w_{kj} \qquad (34)$$

for the Delta_y calculus block, Figure 20.

$$\Delta v = \eta \delta_{ok} z_i \tag{35}$$

for the Delta_v calculus block, where $z_i$ are the values of the input pattern, Figure 21, and

$$v^{new} = v^{old} + \Delta v \tag{36}$$

for the calculus of the updated weight, Figure 22.

The implementation resources were estimated using Simulink Resource Estimator Block:

- 790 Slices
- 1241Flip-flops
- 1222 LUTs
- 8 IOBs

### 1.6.2 Architecture and Algorithms for Synthesizable ANNs with On-Chip Learning

The goal of this research [16] was to design a powerful and flexible hardware platform to allow backpropagation ANNs of different size to be efficiently computed.

We have developed an extendable digital architecture for the implementation of a backpropagation neural network with on-chip delta rule learning using FPGAs and a design methodology that allows the system designer to concentrate on a high level functional specification. For this reason we have created a new library Simulink block set constituted by Simulink Xilinx blocks, MCode blocks and VHDL blocks. With these new blocks, the designer will be able to develop the entire neural network by parameterize the ANN topologies as number of neurons and layers. The implementation goal is achieved using the Mathworks' Simulink environment for functional specification and System Generation to generate the VHDL code according to the characteristics of the chosen FPGA device.

In order to minimize the error function, the value of the weights must be adjusted proportional, accordingly with the negative of the derivative of the error. The computing of the new weights calls the resolving of the gradient of the error equations. The corresponding weight value of the k neuron with the output of the output layer and the j neuron, from the hidden layer, must be adjusted with (37)

$$\Delta w_{kj} = -\eta \frac{\partial E}{\partial w_{kj}} \tag{37}$$

That can be done by (38):

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial net_k}\frac{\partial net_k}{\partial w_{kj}} = -\delta_{ok} y_j \tag{38}$$

That means (39)

$$\Delta w_{kj} = \eta \delta_{ok} y_j \tag{39}$$

The error signal for the kth neuron, can be calculated from (40)

$$\delta_{ok} = -\frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial net_k} = -\frac{\partial\left[\frac{1}{2}\sum(t_k - o_k)^2\right]}{\partial o_k}\frac{\partial o_k}{\partial net_k} = (d_k - o_k) o_k (1 - o_k) \tag{40}$$

Thus the weight of the neurons from the output layer will be calculated with the formula:

$$w_{kj}^{new} = w_{kj}^{old} + \eta (d_k - o_k) o_k (1 - o_k) y_j \tag{41}$$

where $w_{kj}^{old}$ and $w_{kj}^{new}$ are the weights before and respectively after the calculus, η is learning rate, *o* is the output value and *d* is the target value. In the case of the hidden layer, layer "n-1", (where the "n" layer is the output layer) the new weights, are calculated according to the formula (42):

$$v_{ji}^{(n-1)new} = v_{ji}^{(n-1)old} + \eta f(net_i)^{n-1}\left(1 - f(net_j)^n\right) f(net_j)^n \sum_k \delta_k w_{jk} \tag{42}$$

In this way, using (42), we can calculate the weights of the neurons from all hidden layers.

Because the BP learning algorithm is an iterative process the new epochs are repeated until the network is trained enough, i.e., until the error between the actual and the desired outputs is lower than a given threshold.

The most difficulty parts in FPGA implementation of the MLP network are the sigmoid function and the calculus algorithm (Delta rule) of the weights. All blocks are designed in Simulink environment and uses basic computing Xilinx blocks, VHDL code incorporated into design by a black box HDL and MCode blocks.

The architecture of a neuron is similar with that presented in Figure 16 and for the activation function was used the PLAN approximation. The architecture of the block that calculates (42) is presented in Figure 23 and is composed of a multiplier and adder blocks, a block for data parallelization, a delay block and a block for computing of the new weight.

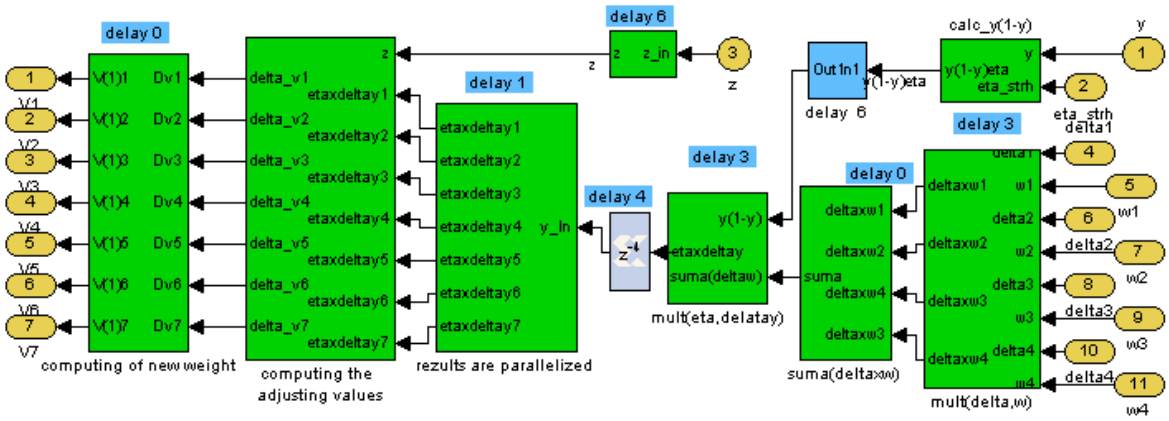



Figure 23. The architecture of the calculus block of new weights for the hidden layers [13].

The overview architecture of the proposed neural network is presented in Figure 24 and includes one input layer with seven neurons, one hidden layer with seven neurons and an output layer with four neurons.

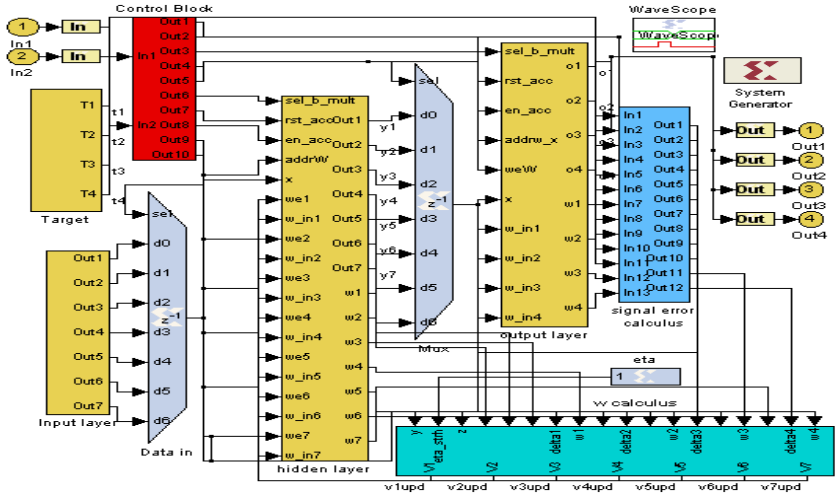



Figure 24. The architecture of the neural network [16].

The design is implemented into XC3S200FT256 FPGA and occupies 956 slices, 1550 FFs, 1246 LUTs and 30 multipliers, representing about third of resources. Regarding the dynamic performances, the propagation phase requires 13 clock periods (given by the sum of the number of the input neurons and a standard delay of 6). The learning phase required time to update the first layer neurons memory (the longest calculus way) with the new weights is 53 clock periods and is given by the sum of time for propagation phase and the time to calculate all new weights.

## 1.7 Digital implementation of the sigmoid function for FPGA circuits

The sigmoid activation function, presented in Equation (43) is one of the possible activation function used in artificial neural networks. We made a research regarding some possibilities of hardware implementation using FPGA circuits of the sigmoid activation function [19]. This work has aroused interest in the scientific community, gathering 22 independent ISI citations.

$$output = \frac{1}{1+e^{-\left(b+\sum_{k=1}^{N} w_k x_k\right)}} \tag{43}$$

Three previously published piecewise linear and one piecewise second-order approximation are analyzed from point of view of hardware resources utilization, induced errors caused by the approximation function and bits representation, power consumption and speed processing. The hardware implementation implies important hardware resources consumptions [20]. To reduce the needed resources some approximation methods could be adopted. The principal classical methods to digitally implement the activation function are Look-up tables and truncation of the Taylor series expansion. The second method can further be divided in: sum-of-steps, piece-wise linear, combination of the previous, or others. The best results reported in the literature show errors of 8% to 13.1% for sum-of-steps approximations and ± 2.45% to ± 1.14% [21], [22], for piece-wise linear approximation. Also, there are approximations with lower errors, but uses floating-point multiplications, which makes it far too complicated for a practical FPGA implementation. The hardware implementations of the sigmoid approximations are done in System Generator, part of the Simulink/Matlab environment. In the following, the hardware resources consumption and the generated errors for different FPGA implementation of five approximations proposed in literature are analyzed.

### 1.7.1 Look-up Tables implementation

The hardware resources of an FPGA circuit utilized for Look-up Tables implementation are small if we use Block RAM implementation. Next table presents the hardware resources utilization of the 4VSX35 FPGA circuit.

Table 7. Hardware resources utilization of the 4VSX35 FPGA circuit [19].

| Logic Utilization | Used | Available | Utilization |
|---|---|---|---|
| Slices | 0 | 15,360 | 0% |
| LUTs | 0 | 30,720 | 0% |
| BRAMs | 1 | 192 | 0.52% |
| DSPs | 0 | 192 | 0% |
| Total equivalent gates | 196.60 | 3.5M | 3.7% |

### 1.7.2 A-law approximation

The method is proposed by Myers and Hutchinson in order to obtain a modified curve with the gradient of each linear segment expressed as a power of two. In this way the multipliers are replaced with shifters. The curve is approximated by seven segments and its breakpoints are presented in Table 8.

Table 8. Breakpoints of a-law based sigmoid approximation [19].

| x | -8.0 | -4.0 | -2.0 | -1.0 | 1.0 | 2.0 | 4.0 | 8.0 |
|---|---|---|---|---|---|---|---|---|
| y | 0.0 | 0.0625 | 0.12 | 0.25 | 0.75 | 0.87 | 0.937 | 1.0 |

Figure 25 presents a comparison between sigmoid function and the A_law approximation and Figure 26 the errors of this method for different number of bits used for representation [19].

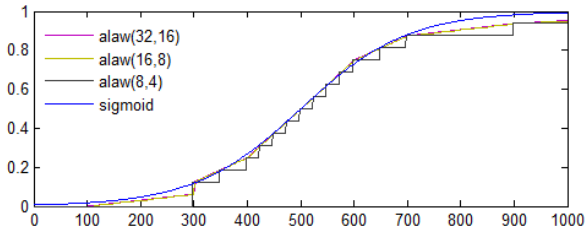



Figure 25. Comparative representation of the sigmoid function and A_law approximation [19].

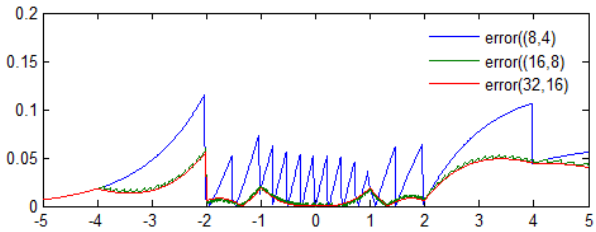



Figure 26. Errors introduced by the A_law approximation for different bits representation [19].

It can be conclude that the maximum error introduced by A_law approximation is 5.63% and the mean error is 0.6335%. These errors increase with decreasing of the bits representation: 11.51 % in case we use 8 bits for integer and 4 bits for fractional part representation.

The hardware implementation of the A_law function is presented in Figure 27.

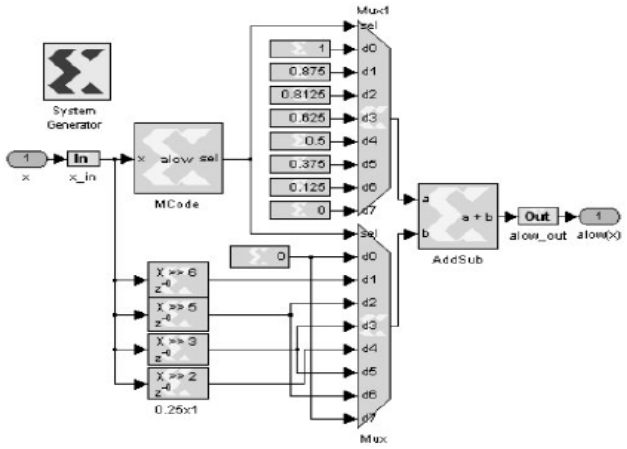



Figure 27. Hardware architecture of the A_law approximation [19].

All the used blocks are part of the System Generator library (Xilinx Blockset). The hardware resources utilized are resumed in Table 9.

Table 9. Resources utilization of the 4VSX35 FPGA circuit for hardware implementation of the A_law approximation [19].

| **Logic utilization** | **Used** | | | **Available** |
|---|---|---|---|---|
| | (32,16) | (16,8) | ( (8,4) | |
| Slices | 185 | 74 | 23 | 15,360 |
| LUTs | 101 | 40 | 16 | 30,720 |
| BRAMs | 0 | 0 | 0 | 192 |
| DSPs | 0 | 0 | 0 | 192 |
| Total equivalent gates | 1,653 | 1,440 | 204 | 3.5M |

### 1.7.3 Alippi and Storti-Gajani Approximation

The Alippi and Storti-Gajani Approximation lies on the selecting of a set of breakpoints of the first derivate and setting the function as sum of power of two's numbers. For the reason that the sigmoid function has a symmetry point at coordinates (0, 0.5) only half pairs of x-y will be calculated, [21]:

$$y_{x>0} = 1 - y_{x\le 0} \tag{44}$$

Taking in consideration only the negative numbers, negative x-axis, and defining the integral part of x as INT(x), the decimal part of x with its own sign, denoted FRAC(x), is defined as:

$$FRAC(x) = x + \left|INT(x)\right| \tag{45}$$

Therefore, the expression of the Alippi function may be defined as:

$$Alippi(x) = \begin{cases} 1 - \dfrac{1/2 + FRAC(-x)/4}{2^{|INT(x)|}} & for\ x > 0 \\ \dfrac{1/2 + FRAC(x)/4}{2^{|INT(x)|}} & for\ x \le 0 \end{cases} \tag{46}$$

Implementation of the Alippi function is made in the System Generator Toolbox Figure 28.

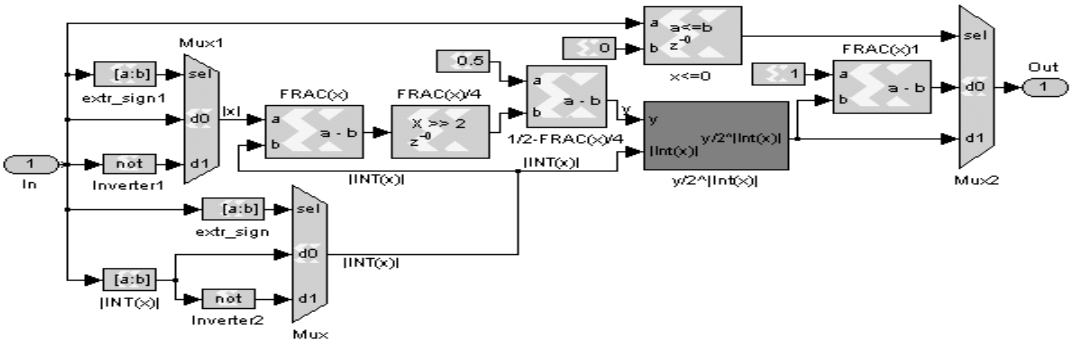



Figure 28. Hardware architecture of the Alippi function approximation [19].

The resources utilized are resumed at 8 shift registers used for dividing the Frac(x) signal to 4 and for shifting it with a number of Int(x) bits, 4 multiplexers, 3 subtraction blocks, 2 slice blocks and 1 comparator block. The numerical representation used for signals processing, i.e. number of bits and binary point are set in a pop-up parameters window as global variables. For each blocks in part the binary point is customized function of the maximum value of the function. In this way it can be obtained maximum performance at a given number of bits.

In order to evidence the errors introduced by the Alippi approximation, in Figure 29 are shown the sigmoid function beside Alippi function hardware implemented and in Figure 30 errors of this approximation as function of number of bits used for representation [19].

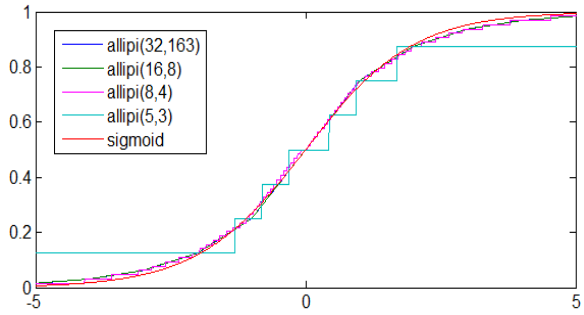



Figure 29. Comparative representation of the sigmoid function and Alippi approximation [19].

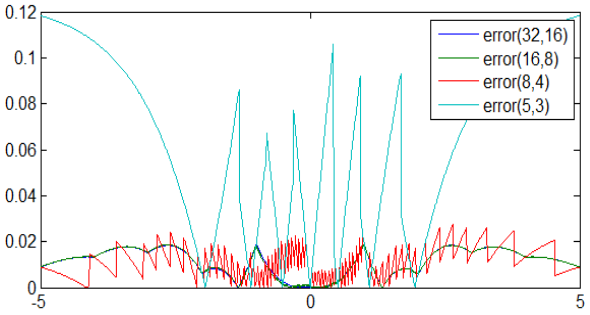



Figure 30. Errors introduced by the Alippi approximation for different bits representation

Analyzing the results presented in Figure 30, can concludes that minimum error is of 1.89 %, and the maximum error is of 11.83%. In order to evidence the total equivalent gates used, function of number of bits allocated for hardware implementation of the Alippi function, we generated an ISE implementation report that is presented in Table 10.

Table 10. Resources utilization of the 4VSX35 FPGA circuit for hardware implementation of the ALLIPI approximation [19].

| **Logic utilization** | **Used** | | | | **Available** |
|---|---|---|---|---|---|
| | (32,16) | (16,8) | (8,4) | (5,3) | |
| Slices | 127 | 67 | 36 | 15 | 15,360 |
| LUTs | 218 | 110 | 56 | 24 | 30,720 |
| BRAMs | 0 | 0 | 0 | 0 | 192 |
| DSPs | 0 | 0 | 0 | 0 | 192 |
| Total equivalent gates | 1812 | 912 | 456 | 219 | 3.5M |

### 1.7.4 PLAN Approximation

The PLAN approximation (Piecewise Linear Approximation of a Nonlinear function) was proposed by Amin, Curtis and Hayes–Gill [23]. The PLAN approximation uses digital gates to compounds the sigmoid function. The approximation of the sigmoid function is presented in Table 11. The advantage of this approximation is giving by the fact that instead of multiplication, shifting operations are required.

Table 11. PLAN approximation [19].

| Condition | Approximation |
|---|---|
| $\lvert x\rvert \geq 5$ | $y = 1$ |
| $2.375 \leq \lvert x\rvert < 5$ | $y = 0.03125 \cdot \lvert x\rvert + 0.84375$ |
| $1 \leq \lvert x\rvert < 2.375$ | $y = 0.125 \cdot \lvert x\rvert + 0.625$ |
| $0 \leq \lvert x\rvert < 1$ | $y = 0.25 \cdot \lvert x\rvert + 0.5$ |

Figure 31 shows the PLAN approximation compared with sigmoid function and Figure 32 shows errors of this approximation as function of number of bits used for representation [19].

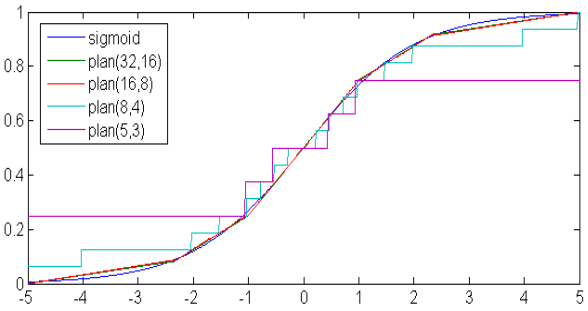



Figure 31. Comparative representation of the sigmoid function and PLAN approximation [19].

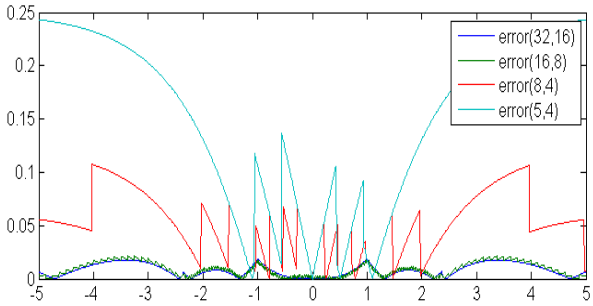



Figure 32. Errors introduced by the PLAN approximation for different bits representation.

The PLAN function approximation implementation is presented in Figure 33.

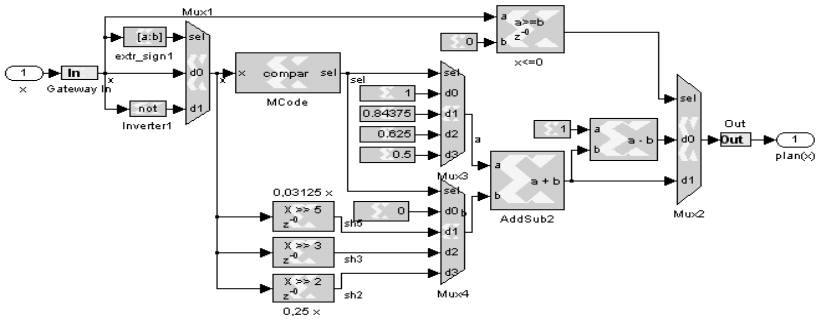



Figure 33. Hardware architecture of the PLAN function approximation [19].

In Table 12 are presented the resources utilized in function of numerical representation for hardware implementation of the PLAN approximation.

Table 12. Resources utilization of the 4VSX35 FPGA circuit for hardware implementation of the PLAN approximation [19].

| **Logic utilization** | **Used** | | | | **Available** |
|---|---|---|---|---|---|
| | (32,16) | (16,8) | (8,4) | (5,3) | |
| Slices | 109 | 44 | 24 | 11 | 15,360 |
| LUTs | 138 | 67 | 32 | 11 | 30,720 |
| BRAMs | 0 | 0 | 0 | 0 | 192 |
| DSPs | 0 | 0 | 0 | 0 | 192 |
| Total equivalent gates | 1164 | 561 | 270 | 114 | 3.5M |

### 1.7.5 Piecewise second-order approximation

The sigmoid function can also be implemented as a piecewise second-order approximation. This kind of approximation implies using of generic square functions that involve using of the multiplications blocks. In order to avoid this, Zhang, Vassiliadis and Delgado–Frias have presented a second-order approximation scheme defined in the interval (-4, 4) that requires only one multiplier, (Zhang approximation), [24]. There are reported others second-order approximation, but the hardware resources needed are three times higher, [25].

$$Alippi(x) = \begin{cases} \frac{1}{2}(\frac{x}{2^2}-1)^2 & for \ -4 > x < 0 \\ 1-\frac{1}{2}(\frac{x}{2^2}-1)^2 & for \ 4 > x \geq 0 \end{cases} \quad (47)$$

Hardware architecture of the Zhang function approximation is presented in next figure.

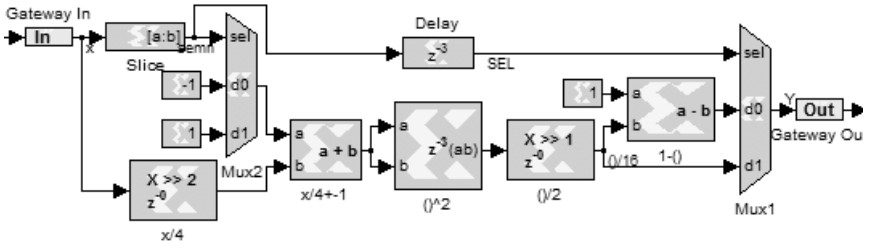



Figure 34. Hardware architecture of the Zhang function approximation [19].

A comparative representation of the sigmoid function and Zhang approximation is presented in Figure 35. Analyzing the errors introduced by the Zhang approximation it can be saw that the maximum error induced in function of the used numerical representation are from a maximum16.95% to a minimum 2.24%. The mean of absolute errors is distributed as between 7.21% and 1.18%.

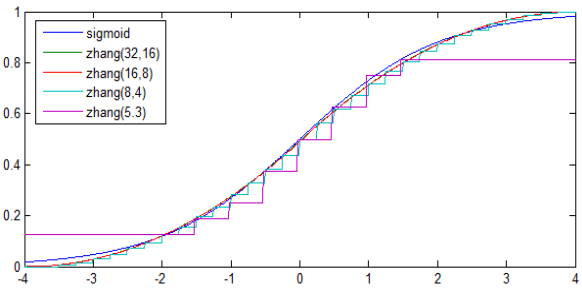



Figure 35. Comparative representation of the sigmoid function and Zhang approximation [19]

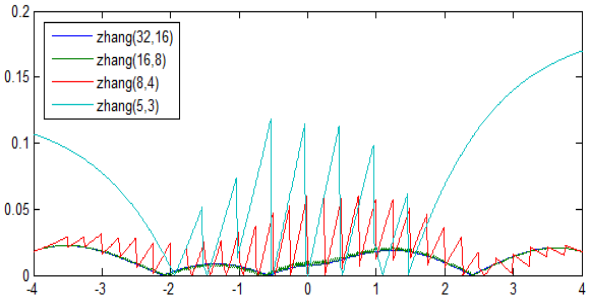



Figure 36. Errors introduced by the Zhang approximation for different bits representation

The resources utilized for implementation of the Zhang function are shown in Table 13.

Table 13. Resources utilization of the 4VSX35 FPGA circuit for hardware implementation of the ZHANG approximation [19].

| **Logic utilization** | **Used** | | | | **Available** |
|---|---|---|---|---|---|
| | (32,16) | (16,8) | (8,4) | (5,3) | |
| Slices | 93 | 29 | 18 | 10 | 15,360 |
| LUTs | 86 | 46 | 26 | 14 | 30,720 |
| BRAMs | 0 | 0 | 0 | 0 | 192 |
| DSPs | 0 | 0 | 0 | 0 | 192 |
| Total equivalent gates | 1169 | 513 | 309 | 201 | 3.5M |

### 1.7.6 Conclusions

Analyzing the report of the introduced errors vs. utilized resources, we get the conclusion that the best approximation method is the PLAN function in the case in which the number of the artificial neurons hardware implemented that use sigmoid function as fire function is larger than the number of the BRAM blocks available in the FPGA circuit. In the case in which the number of artificial neurons is lower than the total BRAM blocks available in the FPGA circuit the best way to approximate the sigmoid function is Lookup Tables method. The results of hardware implementations of the considered approximation functions are shown in Table 14.

Table 14. Errors and resources utilization of the 4VSX35 FPGA circuit for hardware implementation of the Sigmoid approximation [19].

| Approximation function | Maximum error (%) | Mean error (%) | Total equivalent gates count for design |
|---|---|---|---|
| Lookup Table | 0 | 0 | 131.072 |
| A-low | 5.63 | 0.63 | 411 |
| Allipi | 1.89 | 1.11 | 877 |
| Plan | 1.89 | 0.63 | 351 |
| Zhang | 2.16 | 1.10 | 314 |

All the approximation functions were grouped into a library in Simulink/System Generator environment and used as fired function for the neurons of an artificial neural network hardware implemented. In order to underline the main characteristics of the developed firing blocks, such as the hardware resources utilization, processing frequency and power consumption, hardware implementations of neurons with different firing approximation function were made. The FPGA circuit used for implementation was 4VSX35. The reports are presented in next table.

Table 15. Resource distributions for hardware implementation of artificial neuron with different firing function [19].

| Resources distribution | Neuron lookup_table | Neuron f_zhang | Neuron f_alippi | Neuron f_Alow | Neuron f_plan | Available resources 4VSX35 |
|---|---|---|---|---|---|---|
| Slices | 5 | 28 | 59 | 29 | 12 | 15.360 |
| LUTs | 0 | 25 | 89 | 31 | 10 | 30.720 |
| RAMBs | 2 | 1 | 1 | 1 | 1 | 192 |
| DSPs | 1 | 2 | 1 | 1 | 1 | 192 |
| Total equivalent gates count for design | 131.120 | 65.898 | 66.418 | 65.911 | 65.680 | 3.5 M |
| Max frequency (MHz) | 227.790 | 255.860 | 290.613 | 268.168 | 234.467 | - |
| Estimated power consumed | 609 | 608 | 607 | 607.02 | 608 | - |

The maximum frequency supported by the hardware implementation of the approximations reported in the literature shows much smaller performances comparative with the same approximation but hardware implemented with the proposed method described in this paper.

The results obtained in this research are means to be guidance tool in selecting and using of most appropriate approximation function of the sigmoid function. The design of the all firing function considered is made by the authors in Simulink/System Generator environment in order to create a library of components used to develop neurons and neuron networks with different topologies. Finally, the library of the neuronal components represents a useful instrument for an easier designing of a neural network system.

## 1.8 Artificial neural networks application in pattern recognition

As mentioned earlier one of the possible applications of the ANN is the pattern recognition task. This feature is very useful in applications related to support independent life of elderly people like in: hand posture recognition, activity pattern recognition, and health state pattern recognition. In this way the system could adapt to elderly or persons with disabilities and special needs. In the next section I will present some applications developed for hand postures and activity recognition using hardware implemented neural networks.

Between tested applications are different pattern recognition systems for hand gesture recognition [26], [27], [28], [29], artificial olfaction system [30], [31], [134], intelligent Human-Machine Interface [32], smart sensors, smart devices [6], etc.

### 1.8.1 Hand postures recognition system implementation using hybrid ANN

In a previous work we developed a system for hand posture recognition based on data acquired from a sensorial data glove with optical fiber bend sensors. After we tried several neural networks architectures we concluded that the best performing was a hybrid neural network composed of a FF-BP and a competitive network.

Next architecture that we experimented is a hybrid, two levels architecture composed from:

- Preprocessing FF-BP ANN
- Gesture classifying Competitive ANN

***Implemented function***

First layer implements the function described in Equation (48):

$$y_j^{(1)}(x) = f\left(\sum_{i=1}^{M} w_i^{(1)} x_i^{(1)} - \theta^{(1)}\right) \tag{48}$$

where j =1,2,..., $N_1$, represents neurons on first network, ($N_1$=7 in our case), and i=1,2,...M, (M=7) is the number of inputs of the neurons. In case of linear activation function:

$$y_j^{(1)}(x) = \sum_{i=1}^{M} w_i^{(1)} x_i^{(1)} - \theta^{(1)} \tag{49}$$

Net output of the neurons of the second network is given by equation (50):

$$net_k = \sqrt{\sum_{j}^{N_1} \left[ y_j^{(1)} - w_{kj}^{(2)} \right]^2} \tag{50}$$

where k =1,2,..., N2, (N2=15) represents neurons of second network. Simplifying Equation (50) to an equivalent hardware friendly form we obtained:

$$net_k = \sum_{j}^{N_1} \left[ y_j^{(1)} - w_{kj}^{(2)} \right]^2 \tag{51}$$

So the final form of the net output is given by Equation (56):

$$net_k = \sum_{j}^{N(1)} \left[ \left( \sum_{i=1}^{M} w_i^{(1)} x_i^{(1)} - \theta \right) - w_{kj}^{(2)} \right]^2 \tag{52}$$

The activation function of the neurons on second network is the competitive activation function. The designed system recognizes 100% of the training set vectors. The system was tested with other two sets of test vectors representing static gestures formed by the same user, the success rate being also of 100%. The results of a test that used gestures coming from another user gave a recognition rate of 96.67%.

***Hardware modeling of the hand postures recognition system***

The hardware design of the hand postures recognition system is an implementation of the (52) made with blocks from the ANN library created by us. The model for the recognition system was created in System Generator.

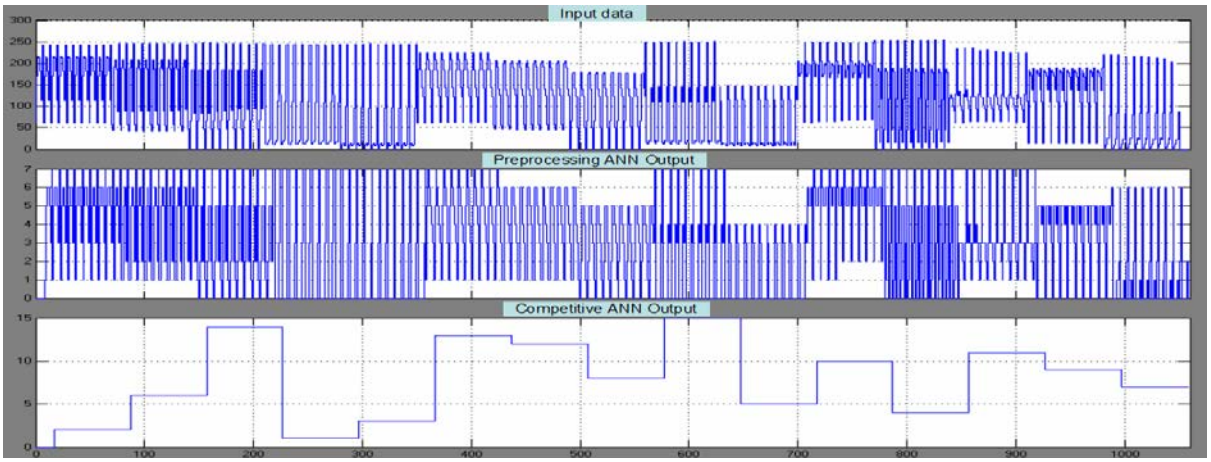



Figure 37. Simulation of the HW model for the postures recognition system [26].

Figure 37 presents the results of the simulation of the hardware model for the recognition system, when at the inputs the training set of vectors was applied, set composed of 10 vectors for each of

the 15 postures. The postures recognition system was implemented in several FPGA devices. For example, to take advantage of the dedicated multipliers from within the Virtex devices, the XC2V1000 device was used. A real time simulation of the system, in which the input vector elements are being entered at a frequency of 100 Hz and with a clock signal of Fclk = 10 MHz, is presented in Figure 38.

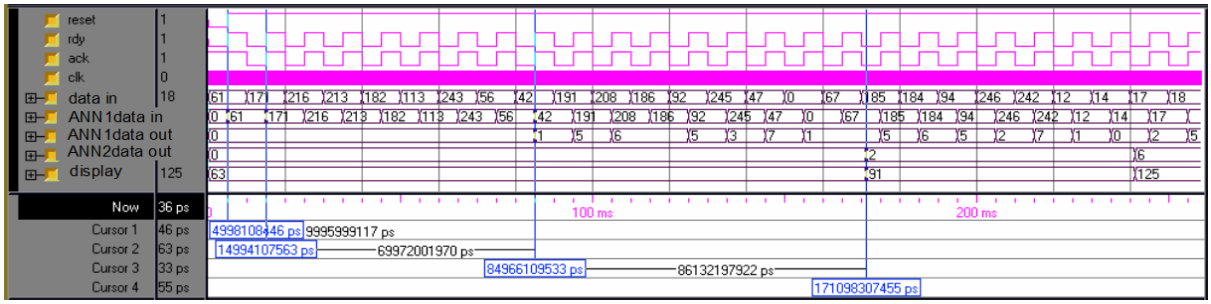



Figure 38. The real time response of the postures recognition system [26].

The static gesture recognition system, was designed and tested based on my new ideas. We proposed the use of an ANN to preprocess data instead of the classical pre-processing methods. Also we proposed the use on simple competitive networks to classify postures. And last but not least, the hardware implementation of the ANN using FPGA devices must be mentioned, which makes the system usable to create a portable recognition system. All ideas proposed where successfully verified using the hardware/software testing environment developed.

***Possible applications***

From amongst the simpler usage possibilities we can mention the computer input devices that allow the manipulation of virtual objects or be used in interactive presentations. The user that wears such a glove can point to something on the image displayed by the computer can go to the next image, etc. More important applications are in the rehabilitation of impaired patients, with different types of handicaps. The system can be used for example in assisting patients in recovering after an accident, when verification and even a quantization of the patient's ability to perform certain postures are necessary.

### 1.8.2 Artificial olfaction system with hardware on-chip learning neural networks

In this research [30], we demonstrated de possibility to use of the hardware implemented ANNs for the recognition of the type of coffee presented in a test chamber. Data acquisition was achieved through the PC-MIO-16E-1 acquisition card and a virtual instrument, developed in Labview, for signal pre-processing and data logging into text files. The patterns presented (the type of coffee) have been recognized through neural networks. In order to select the ANN with the highest accuracy in recognizing the coffee type, several different ANNs were simulated. The pattern

recognition module is a neural network hardware implemented in an FPGA circuit, with on-chip learning controlled by generic block units described in VHDL code, MCode and Xilinx blocks. In order to design and implement the neural network, the Simulink environment was used for functional specification, System Generator to generate the VHDL code according to the characteristics of the chosen FPGA device and ISE Xilinx to simulate the design at different stages of implementation and to generate the configuration file.

The adopted model of the artificial olfaction system consists of: seven gas sensors (TGS842, TGS826_1, TGS826_2, TGS2600, TGS2601, TGS2602, TGS2620) chosen to respond to a wider spectrum of odors, a temperature sensor (LM35) and a humidity sensor (SY-HS-230) (all mounted in a gas test chamber), a harvesting chamber, three pumps for gas transportation, circuits for sensors conditioning and pumps command, a data acquisition board (PCI-MIO-16E-1), a pattern recognizer module hardware implemented in FPGA (Virtex-4 SX MB - 4VSX35) and a user interface developed in LabVIEW.

Considering the feature extraction methods reported in literature, the heuristic method was adopted, with the following selected features: average value, maximum value, integral of the function, integral of the absorption time function, maximum slope of the absorption time function, maximum slope of the desorption time function and time of maximum slope of desorption function. All features were extracted using an application developed in Matlab.

The developed artificial olfaction system (AOS) is an instrument that tries to mimic the human olfaction system. In this way, the AOS is both a chemical sensing and a data analysis system, developed in order to discriminate between different simple or complex odors. Considering the important advantages of the hardware implementation of the ANNs over computer simulated ones, explained by small-size, low power consumption and the full exploitation of the parallel operation of the neurons (which gives the possibility to achieve a very high speed of information processing), the hardware implementation of ANN based pattern recognition system seems to be a natural choice for an electronic nose. An extendable digital architecture design methodology has been developed, for the implementation of different ANN topologies such as backpropagation neural network with on-chip delta rule learning, which allows the system designer to concentrate on a high level functional specification.

The proposed model is constituted by neuronal blocks, organized in hidden and output layers, the control blocks that control and command the behavior of the neurons elements (one MAC unit, a RAM memory block for weights storage) and one HDL block that implement a specific firing

function (Figure 39). The firing function adopted is a sigmoidal one and for its implementation a PLAN approximation function was assumed [19]. Due to the node parallelism of the neurons, for each layer in part the designer has to set the layer parameters: weights initialization and the depth of the memory.

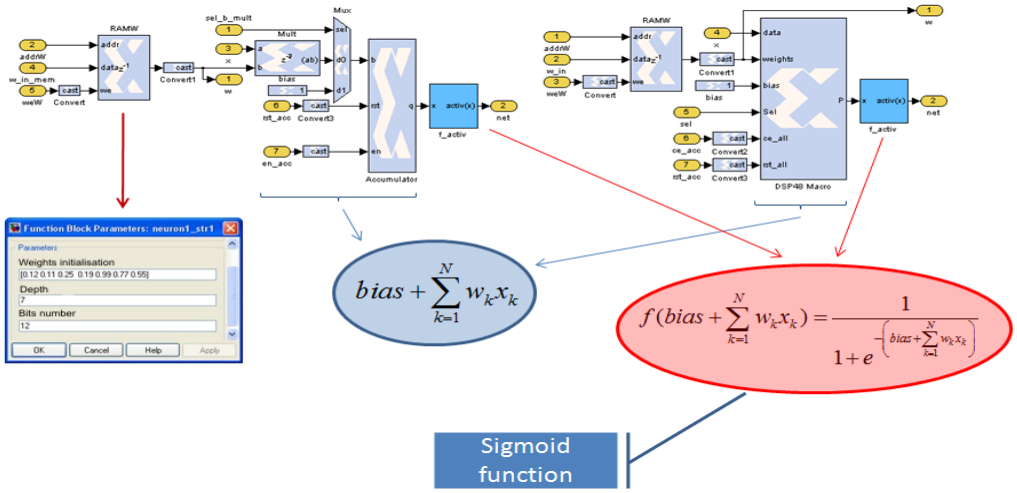



Figure 39. Hardware model of the FF-BP neuron [30].

The overview architecture of the proposed FF-BP neural network used for the pattern recognition module of the artificial olfaction system is presented in Figure 40. It includes as inputs one input layer with seven neurons, one hidden layer with seven neurons and an output layer with four neurons. The blocks involved are: a control block of the neuronal system, a neuronal layer block, errors and weights computation of the hidden layer block, target and input blocks and an output layer weights computation block.

Considering that for a specific network it is possible to make an aprioristic estimation of the utilized resources expressed in terms of RAMs, LUTs and DSP blocks, the total number of the neurons that can be implemented into 4VSX35 circuit is 120, with 80 neurons in the hidden layer and 40 neurons in the output layer.

In order to find the best FF-BP topology with minimum resource utilization, a series of different FF-BP ANN topologies were implemented. In the case of an FF-BP ANN having a topology of 56-56-4 neurons, meaning 56 neurons on the input layer, 56 neurons on the hidden layer and 4 neurons on the output layer, which processes an input vector that comprises 56 components (8 features per gas sensors and. having 7 gas sensors).

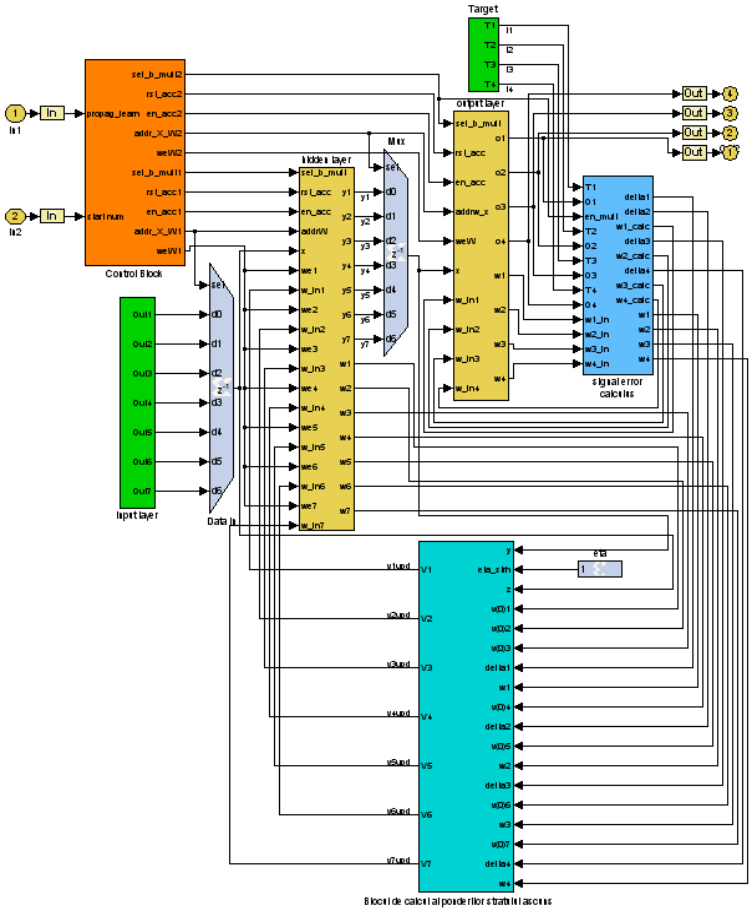


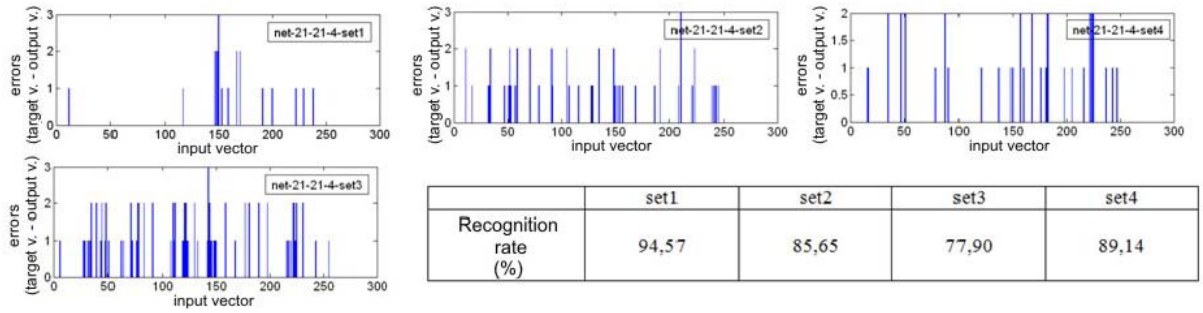

Figure 40. The architecture of the proposed FF-BP ANN [30].

An analysis of recognition rates versus data training with different features was made considering a smaller topology (21-21-4 neurons). It can be clearly seen in Figure 41 that the recognition rate depends on input vector dimensionality and its features component. In this case, the recognition varies from 95% to 75%. The considered features are:

- average value (seti1),
- maximum value (set-i2),
- integral of the function (set-i3),
- integral of the absorption time function (set-i4),
- maximum slope of the absorption time function (set-i5),
- maximum slope of the desorption time function (set-i6),
- time of maximum slope of absorption function (set-i7)
- time of maximum slope of desorption function (set-i8)

| | set-i1 | set-i2 | set-i3 | set-i4 | set-i5 | set-i6 | set-i7 | set-i8 |
|---|---|---|---|---|---|---|---|---|
| Recognition rate (%) | 87,20 | 90,31 | 75,19 | 78,68 | 74,03 | 52,32 | 36,82 | 41,08 |

Figure 41. Recognition rate for different features [30].




| | set1 | set2 | set3 | set4 |
|---|---|---|---|---|
| Recognition rate (%) | 94,57 | 85,65 | 77,90 | 89,14 |

Figure 42. Recognition rate for different sets of data training [30].

More details are available in the Ph.D. thesis of Alin Tisan [134], publicly defended in 2009.

## 2 INTELLIGENT EMBEDDED SYSTEMS FOR DAILY LIFE ASSISTANCE

The aim of this research is to design embedded systems that can help the elderly or persons with different type of disabilities to take advantage for as long as possible of living in familiar surroundings. The target is to bring together the smart home technologies with mobile assistive robots and to build the so called “Intelligent Environment”. These environments include multi-sensor networks, smart digital appliances, smart monitoring systems, assistive robots, etc. In 2050, 37% of EU inhabitants will have more than 60 years, and ratio between 65 years old person and an active one will greater than one. This is why will be a huge demand on nurses/care givers. The lack of enough human caregivers, could be compensated with the widespread of intelligent assistive home environment, the smart wearable monitoring devices and the use of assistive robots. We have contributed with researches in all this directions aiming to develop and integrate these three major components: intelligent ambient systems, the health and activity monitoring, and recognition system, and robots providing personal assistance.

Our studies related to intelligent ambient system, consisting in gesture based remote control, monitoring modules, smart medication dispenser etc., were published in [33], [34], [38], [43], [45], [46].

For activity and health state recognition we have developed several modules for vital parameters monitoring (temperature, heart rate, acceleration) and we have published the results in [46], [47], [48], [54], [55], [57], [58], [59].

Our achievement related to development of an autonomous robot as part of our assistive system have been presented in several articles and at international conferences [72], [73], [74], [75], [76], [77].

## 2.1 Intelligent embedded systems for Ambient assisted living

Nowadays there is a growing interest in the development of embedded systems, able to easily integrate a variety of I/O devices and sensory interfaces using different protocols. It is desirable that these interfaces have the capacity to learning and adaptive, which can be obtained for example by using neural networks. It is also necessary to integrate the output interfaces, using an equally large variety of protocols. Among possible applications we can mention intelligent computer peripherals enabling people with any kind of handicap to use computer, to communicate, as well as any kind of industrial or domestic device. Main applications considered for development of such smart devices are in the monitoring, domotics, automotive, prosthetic and automation field where the trend is to produce easy-to-use devices with embedded intelligence and a completely different philosophy from that of personal computers.

Our contributions are related to the development of a platform that could integrate several smart devices, which can be used to assist elderly or sick people's every day independent activities using and developing the latest assistive technologies.

### 2.1.1 Adaptive Hardware-Software Co-Design Platform for Fast Prototyping of Embedded Systems

The goal of the work presented in [34] was to develop a hardware-software co-design platform that can easily integrate input and output devices using a huge number of communication protocols. The platform permit the fast development of smart interfaces using:

- Application specific sensors,
- Hardware IP modules that can be easily connected,
- VHDL modules that can manage sensors,
- Artificial Neural Networks used for example for pattern association or recognition.

Using this framework development of a new smart devices needs only the design and synthesis of new VHDL drivers for the new sensors, peripherals, actuators and new application-specific ANNs. The smart devices must have multisensory interfaces and have a reduced development time. Other features that have to provide are as those presented in [35] and [36]:

• The possibility of reusing HDL blocks developed. These are application modules that can perform specific and common sub-tasks. These components must have a specific well defined common interface for any kind of specific behavior.

• The possibility to easy change of the sensors, I/O devices, due to use of HDL/software drivers for any common devices class. HDL drivers must have an interface defined for any function/device.

• Intelligent behavior concentrated in one or more HDL Neural Network modules, tailored for hardware implementation, are key elements for this framework. Any application of a new smart device should use these “neural engines” to add adaptive and learning capability.

#### 2.1.1.1 Platform Architecture Analysis and Design

The platform developed in order to provide a fast prototyping environment for adaptive embedded systems is shown in Figure 43, and it was developed to facilitate the use of co-design techniques.

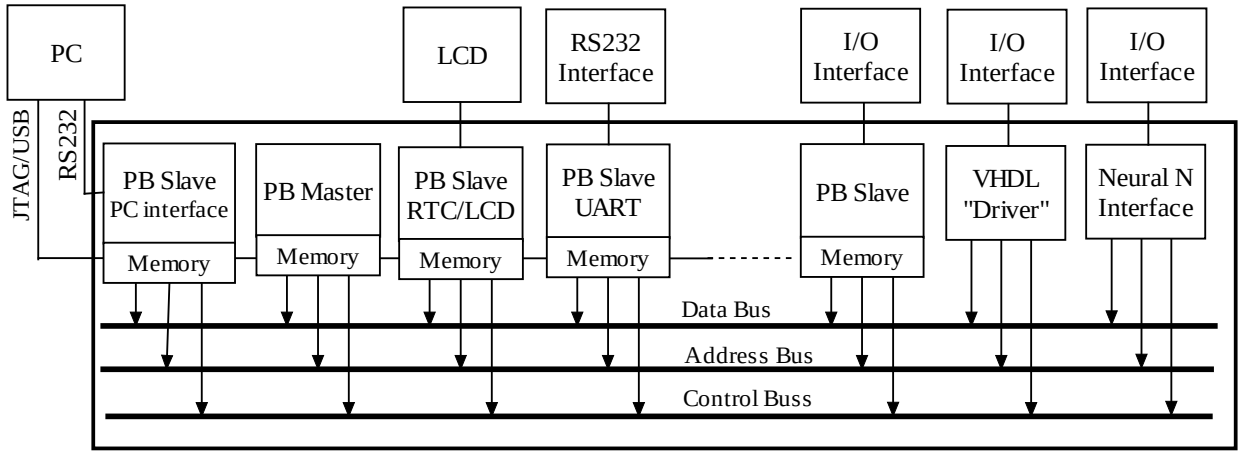



Figure 43. The hardware-software co-design platform bloc diagram [34].

The main components of the platform are several modules consisting of a set of processor – memory. From many possibilities, we have opted for a pair of a Xilinx 8-bit, soft-core PicoBlaze™ (PB) processor and a Block ROM that contains the software code for PicoBlaze. We chose the PicoBlaze because of its simplicity and possibility to implement a large number of PB cores in a FPGA circuit. It occupies just 96 Spartan-3 Slices which is less than 3% of the XC3S500 device [37]. The newer KCPSM6 is even smaller occupying only 26 slices and is is working at higher frequency (105-238 MHZ). The KCPSM assembler generates a HDL file from rom.psm template file, which can be used for the ROM program memory implementation and simulation.

The PicoBlaze-memory pairs have different roles and operate independently until they need to exchange information with other modules or I/O devices. The access to the common bus and input/output devices is arbitrated by a VHDL Connection matrix block according to predetermined priorities (Figure 44). The “Connection matrix” block allows 8 inputs and 8 outputs and it connects an internal block to an I/O interface.

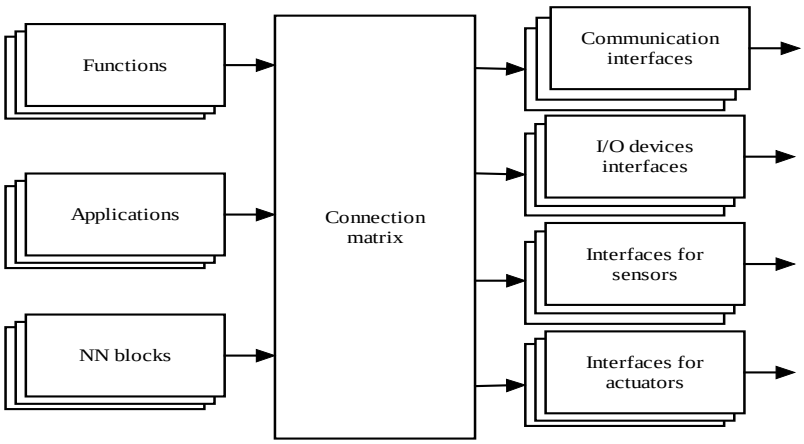



Figure 44. Connection matrix block [34].

*Common Interface Architecture Definition*

For date transfer between several modules and/or I/O interfaces it is necessary to implement standard rules for communication. For example sensors could communicate using simple one or two wires protocols, I2C, SPI, etc. Also actuators and some communication modules could be controlled using one of the previous protocols.

*Specific Interface Architecture Definition*

This step refers to the design of specific interfaces for any specific function or for a specific device. For example many sensors/actuators could communicate using a standard UART port but they could have also specific rules (protocols). For example (Figure 45) a RFID reader and a X10 device could use a standard UART port for communication, but each has specific protocols to initiate the communication or to answer a request.

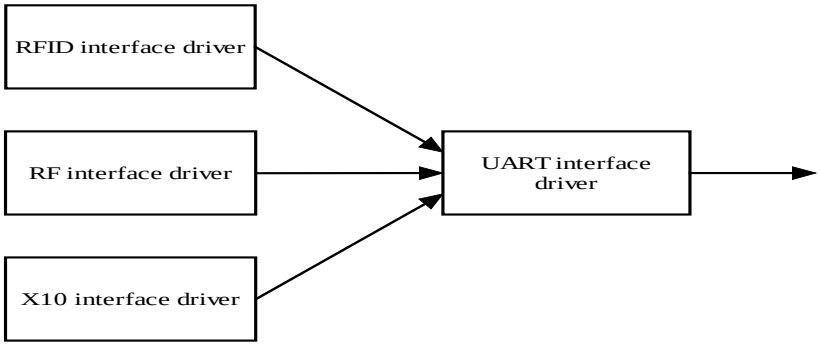



Figure 45. Specific and common levels of drivers [34].

### 2.1.1.2 Platform components development

Platform components development requires the following steps:

- **Identification of some common types of input/output devices and sensors.** Some of the most used input/output devices and sensors in low cost general purpose FPGA boards are: serial

port, keyboard, mouse, VGA monitor, LCD, LED, seven segment display, switches, push buttons, temperature sensor, RF interface, IR interface, etc.

- **Development of class drivers for common I/O devices, sensors and actuators.** The "drivers" are implemented using a PicoBlaze or a VHDL module. Devices used in the test applications were: UART interfaces, LCD, rotary encoder, etc.

For any device class we defined the interface and then the functionality that should be supplied. This functionality must be a minimum common set of functions that all devices in that class should provide: this is the key that enables exchangeability of devices (different models and different constructors) in the same class.

For example the LCD class driver has the same architecture, the difference is the processor program memory. The program for every driver can be assembled and simulated with PicoBlaze tools. The content of program ROMs can be changed individually.

- **Development of specific modules for specific I/O devices, sensors and actuators.** We developed some specific modules for specific interfaces, protocols, functions. Such an example is the implementation of the X10 protocol. X10 is an open, international industry communication standard, between electronic devices used in home automation. X10 uses as medium of propagation for signaling and control, the existing household power grid. We have used a commercially available RS232 to X10 interface for signal injection in power grid. For communication with RS232/X10 interface we used an UART interface module written in VHDL. For X10 specific protocol implementation we have created a specific module using the same pair of PB-ROM, and a specific program that runs on processor.

- **Development of basic function modules.** These modules are blocks that implement common functions such as FFT, or other resources used by several modules, e.g. a real time clock.

- **Development of VHDL Neural Networks.** As has been stated, using of neural networks blocks is essential to enable the system to adapt to changes in input signals. For neural networks implementation we have chosen the method presented in [3] and [6].

- **Simulation and test of all modules.** Each module was simulated and implemented independently and then together with other modules. Figure 46 presents the simulation of the LCD class driver.

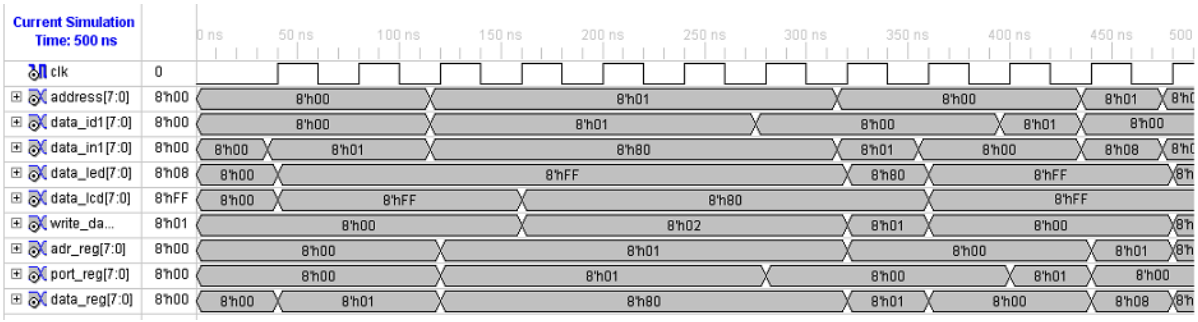



Figure 46. Simulation of LCD class driver [34].

### 2.1.1.3 Testing the development platform

We chose to test the implementation of a monitoring and control system for a smart house. Control and monitoring platform for a smart house has to be able to receive and generates signals from and for a variety of devices using various protocols (Figure 47).

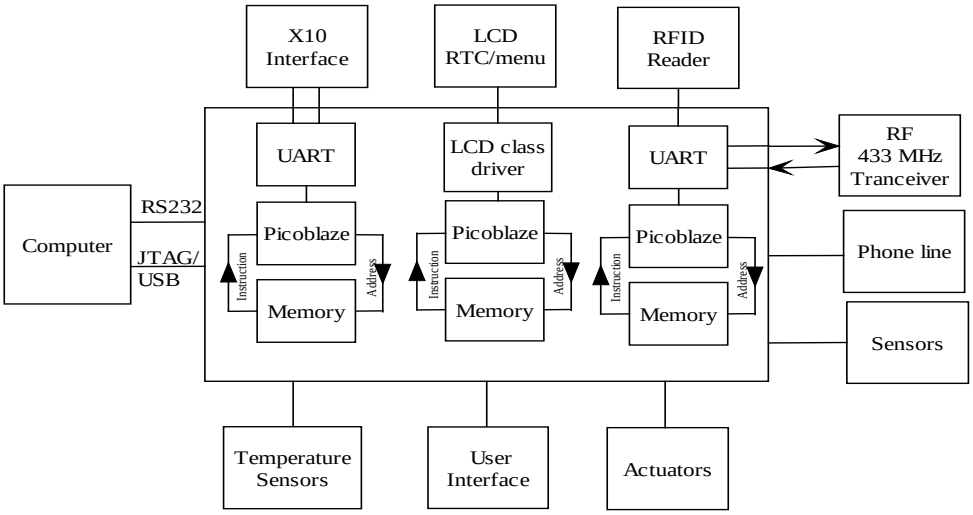



Figure 47. Control and monitoring platform for a smart house [34].

Also, one of the main requirements for the system is to enable easy expansion and easy change of the architecture and functions incorporated.

We have implemented a series of drivers for I/O devices (keyboards, switches, push buttons, LCD, 7-segments display, LEDs), sensors and actuators, communication protocols (UART, PS2), functions (RTC, X10, RFID, wireless) and neural network blocks. The experimental setup has as central unit a S3E Starter Board with Spartan 3E FPGA. It connects various modules.

- Smart house Kit (command module, lamp modules, appliance modules, transceiver, remote control)
- „Home Guard” alarm KIT
- RFID Reader/Writer module - APSX RW-310 RFID:
- RTX-MID-3V 433.93 MHz transceiver module-compatible with RS232 @9600bps.

#### 2.1.1.4 Conclusions

The main contribution consist on the definition of a common development platform with the flexibility offered by programmable logic circuits, having the capability to be reconfigured in the system, and adaptability offered by the use of neural networks blocks.

We have designed and successfully tested a multiprocessor architecture that can be easily extended. The framework that we have developed has the following characteristics:

- It is based on low cost general purpose FPGA boards without specific components / interfaces
- Independence from hardware platform
- Reusability of components with standard behavior-dependent interfaces
- Exchangeability of sensors, I/O devices thanks to common standardized device class VHDL drivers
- Learning and adaptive behavior concentrated in one or more VHDL Neural Network modules, tailored for hardware implementation
- No need of hardware design and fabrication
- Fast development of new smart devices needs only design of new application-specific VHDL modules or a new program that runs on a PicoBlaze processor.

The main advantage of the solution adopted i.e. using an FPGA platform is that it allows easy reconfiguration of the system making it suitable for such an application that requires frequent changes by adding new resources and adapt to changes and new requirements.

### 2.1.2 Alternative control method of the smart house natural gestures

In [38] we investigated the possibility of implementation of a gesture controlled smart house. The smart house has been present as a concept for more than a century. There are two particular types of smart houses, the green house that tries to minimize the impact on the environment and the house assisting the elderly. There are two turning points for the introduction of domestic technology: the introduction of electricity and the introduction of information technology [39]. The smart house concept arose from the convergence of domestic technologies and dreams of total control. Intelligent behavior was simulated by trying to create a network of devices that communicate with each other [40]. The core of the smart house is trying to make it safer, more convenient and economically friendly. One of the first technologies that made the realization of a smart house possible is the X10 protocol which is a standard data communication protocol for the Power Line Carrier that use high frequency 120 KHz signal for data transmission [41]. This technology offered the freedom to control almost any appliance even from outside the house. The contemporary smart house, according to David Heckman, represents the convergence of three

areas: robotics, artificial intelligence and media convergence [42]. At present, the smart house is a common item that is widely available to the general public.

The control methods commonly available in a smart house usually includes the use of touch screens, infrared remote control or mobile devices. All these control methods have advantages and drawbacks. The use of touch screens or even standard PC and LCD has the advantage of using standard components for the control system and the common use of the internet but has a major drawback: in order to control the house from any point you need to be near a control box, meaning that the whole house needs to be fitted with control points. The classic infrared remote control is suitable to control a relative small number of devices. Although there are a number of improved remote control devices, they tend to be complicated and are not suitable for elderly people, impaired people or children. The use of mobile devices to control the smart house implies the use of the internet. The main advantage of this method of control is that it uses standard smart phones or even tablet PC that can easily be replaced because the mobile device only displays a simple web page which can be controlled by the user. The main disadvantage is that it tends to be a rather complicated method of control and even expensive.

#### 2.1.2.1 Gesture based smart house control system

A gesture based control system is a classic control system in which the input device has been changed with gesture recognition. The wide range of gesture capable sensing devices makes this control system easy to build. Because the main interface between human and machine is gesture, the communication is carried out easily and in an intuitive manner.

The control system developed by us is composed by two subsystems that communicate via radio waves and a localization identification system. The first subsystem is a bracelet that acquires dynamics and motion data of the hand. These data is transmitted to the second subsystem alongside with data regarding the user position, data that have been received by infrared from the location identification system. The second subsystem is the control box on which the data processing takes places. According to data received the approximate hand movement is determined alongside with the user position. These hand movements allow the user to navigate into a menu displayed on a screen. The menu entries are real world actions that the system can undertake. These menu entries are stored on a SD card in order to be easily modified.

The user has the possibility of controlling electronic/electric devices only with hand gestures. This implementation supports the recognition of four basic hand movements: up, down, left, right. The main system can be used in two main operation modes: on screen control and direct control. The

on screen control operation mode implies the use of a screen on which different action to control objects is displayed. The actions are organized under the form of a menu in which the user can navigate using simple hand movements. The control actions are selected via hand moments only. The second control operation mode allows the user to control different objects without the use of a display system.

#### 2.1.2.2 The bracelet

This subsystem detects the hand movements using a three axes accelerometer. The accelerometer chosen was ADXL345 from Analog Devices due to its 13 bits resolution and the ability to measure acceleration up to 16 g. The user location is determined by receiving information from the location identification system using an infrared module. The infrared module chosen was TSOP31238 at 38 KHz and has built-in filters and preamps. The microcontroller used is an Atmega168 from Atmel. The data is sent to control box using a low power AT86RF212 on a PmodRF1 board radio transceiver. The bracelet hardware components are presented in Figure 48 and its software flowchart in Figure 49.

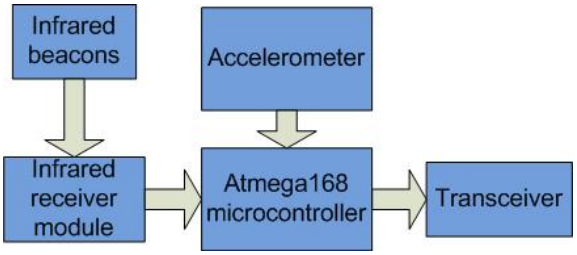



Figure 48. Bracelet components [38].

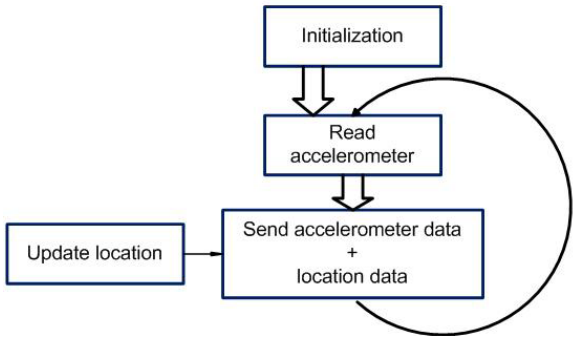



Figure 49. Bracelet software flowchart [38].

#### 2.1.2.3 The control box

The components of the control box are shown in Figure 50. The control box is built around a FPGA board. This assures easy upgrade and a cheap yet powerful platform for data processing. The data are received using an identical radio transceiver PmodRF1. The FPGA board has a VGA output that is used for displaying the menu. Any VGA compatible monitor can be used and with an external converter is possible to display the menu even on a regular TV. The menu entries are stored on a SD card for easy access, the SPI protocol being used for access. The menu entries permit the user to control real objects, each menu entry representing a possible action or a category of actions. The menu entries are stored under a XML similar format.

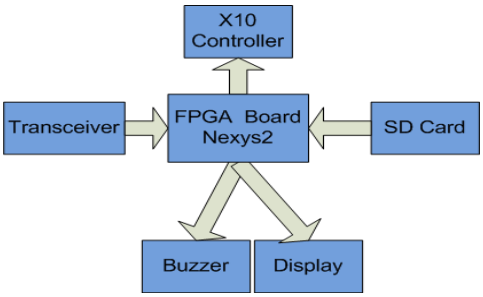



Figure 50. Control box components [38].

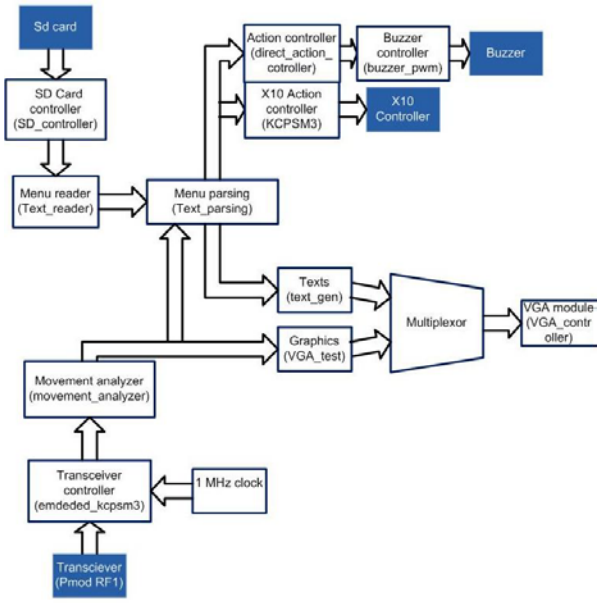



Figure 51. Control box software modules [38].

The FPGA board has a build-in serial port that is used for communicating with an X10 standard controller. The controller used is a CM11 from Marmitek. Via the serial interface the system communicated X10 command that must be retransmitted by the controller via the existing electrical grid. This way any X10 compatible receivers can be used in order to control any household appliance without a direct connection with the control box. There is a possibility to use direct control for the devices that are near the control box. For testing purposes, a personal alarm system was implemented to demonstrate the capability of the system to directly control a device.

The control box in programmed mainly in VHDL language using XILINX ISE development environment. The Picoblaze microcontrollers were programmed using assembly language. The software modules in the control box are presented in Figure 51.

#### 2.1.2.4 Gestures recognizing algorithm

In order to determine the movement identification method, a series of data from the accelerometer have been captured and plotted into graphs. For example in Figure 52 a right movement graph is presented. Based on this graphs, the method of movement identification was chosen. In this case, due to the necessity of recognizing only four basic gestures, a compare based algorithm is used. This algorithm works well if the number of gestures that has to be identified is small, but if the number of gesture increases, a more powerful algorithm needs to be used in order to ensure a precise identification. The system can also be controlled by hand tilting only, giving persons with a low degree of mobility a possibility to use the system.

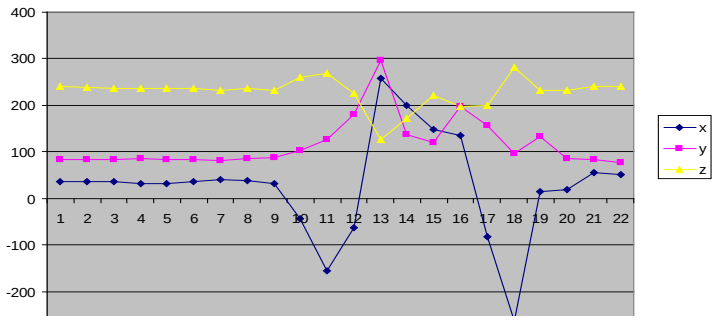



Figure 52. Right movement captured data [38].

### 2.1.3 Automated system for evaluating health status

In the present world, millions of people die every year due to lack of information about their health. Increased costs in the healthcare system could be reduced, if it would give more attention to disease prevention through regular assessment of health status and their treatment in the early stages. As a result of this situation, many researchers are trying to develop technologies to improve the quality of human life. Flexible technologies that are easy to access, such as intelligent clothing, watches, head bands, shoes and belts, will be the future in monitoring and treating disease. They could help medical personnel with more relevant information and establishing a better medication to patients. Most people shows an opening for these technologies and medical centers expect to begin to use technology to improve their services. Today's sensors can provide real-time data or collect and retain data until the synchronization with other systems. Multiple sensors fusion technology is still new and developing, but could offer many benefits for healthcare applications. Sensor fusion can provide improved measurement accuracy and at the same time provide more results in less time. At the same time, using Web browsers, medical personnel can easily monitor their patients in real time, or even to examine the results and to compare them between certain days or hours. By comparing the results of different days on a web diagram, usually doctors can diagnose and recommend accurate medication for the patient disease. In this way the workload of medical personnel could be reduced only by offering new tools that they could use to make patients' lives easier and better.

A modern medical system that could replace some of the existent medical equipment and facilitate the work of medical personnel should include the following features:

- fusion of multiple sensors in a fixed or portable device to provide wireless connectivity;
- be a single device that can be attached to any patient for easier use;
- allow remote patient monitoring;
- store information on a website and allow to be accessed by medical personnel;
- display measurements in real time;

- save data in a database;
- allow the generation of statistics to analyze data;
- allow the generation of graphical data analysis;
- detect emergencies and alerting medical personnel.

Our research reflected this trend, to help gather information to assess health and provide information to properly trained personnel in diagnosing patients. We can use this information, to monitor in real time the state of a patient, or to get sensitive data in order to be subsequently analyses for medical diagnosis.

In this research [43] we have proposed a system for automatic recording of the main physiological parameters of the human body: body temperature, blood pressure, respiratory rate, electrocardiogram (ECG), skin resistance, etc. To realize this system, we have developed a program that can read and automatically save in a file, the data from specific sensors. Further it is possible to interpret the results, by comparing them with known normal values and thus offering the possibility for a primary health status diagnosis by specialized personnel. The data received from the sensors is taken by an interface circuit, provided with signal conditioning (filtering, amplification, etc.). Data acquisition is controlled by a microcontroller development platform. The data are transferred to a PC, using serial communication. The whole process of health assessment is commissioned by a new program developed by us in the Python programming language. The program provides automatic recording of the aforementioned parameters in a predetermined sequence, or if you want only certain parameters are registered.

The data received from the sensors is taken by an interface circuit, provided with signal conditioning (filtering, amplification, etc. To implement the proposed system we used the e-health platform from Libelium Company [44]. The structure of the system that we used is shown in Figure 53.

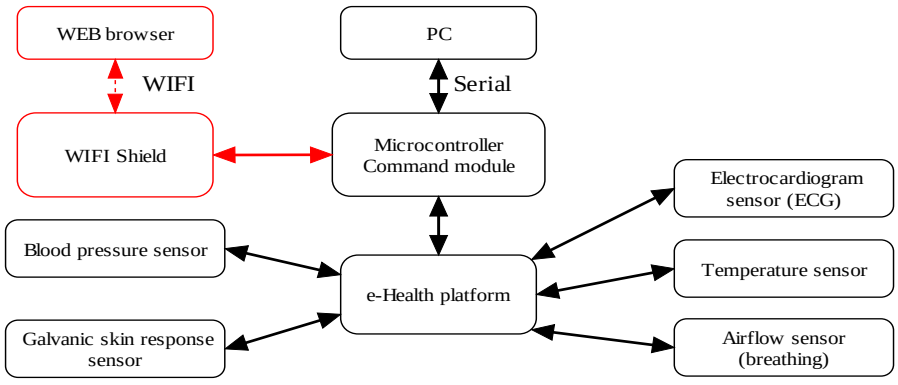



Figure 53. System structure.

The data of physiological parameters gathered can be wirelessly sent, using any of the 6 connectivity options available: Wi-Fi, 3G, GPRS, Bluetooth, 802.15.4 and ZigBee depending on the application. If we want to send photos and videos of the patient to a medical diagnosis center in real time, for image diagnosis, a camera can be attached to the 3G module.

#### 2.1.3.1 System software

Our program, developed in Python programming language, automates the process of sensory data acquisition. The program allows recording data from the five sensors, in a predetermined sequence or choosing the desired parameter(s) from the menu. The system allows recording sensor data in files for storage and further processing or real-time tracking of the parameters provided by sensors. The system is open, being able to add new sensors by modifying the interface and software properly. Some sensors require an initial calibration to increase the accuracy of data reading and modifying the microcontroller software parameters after some preliminary measurements.

### 2.1.4 Advanced Medication Dispenser

A medication dispenser is a perfect system for medicine administration because it tries to eliminate, or at least minimize, the error that may appear in the process of organizing and/or dispensing medicine. Medication dispensing is an important activity that can have major implications if done improperly. Dispensing must be done in the correct time interval, at the correct user, with the correct drug and dose. We proposed in [45] a smart medication dispenser that can satisfy these needs and provide a mechanism for supervision. In order to ensure that the dispensing process is error free, the concept of a new smart medication container is used. A smart medication container is “smart” as it holds the medication dispensing parameters for the drugs it contains: dispensing time and date and name. The medicines are dispensed only to the correct person; the identification being done via RFID tags. Based on this information, the actual dispensing is done. The device can be monitored via an Ethernet network and offers information regarding the drugs that the user must take, or taken, via a webpage. The supervisor can verify in real-time if the user has taken the drugs and which drugs wasn’t taken.

#### 2.1.4.1 Hardware design

The medication dispenser presented in Figure 54 is built using a PIC32 microcontroller on a ChipKit Max32 board. The wireless communication is assured through the 802.11 WiFi shield. This creates a device that can be monitored from a local area network, and with the help of a VPN connection, the device can be accessed from any location with internet access.

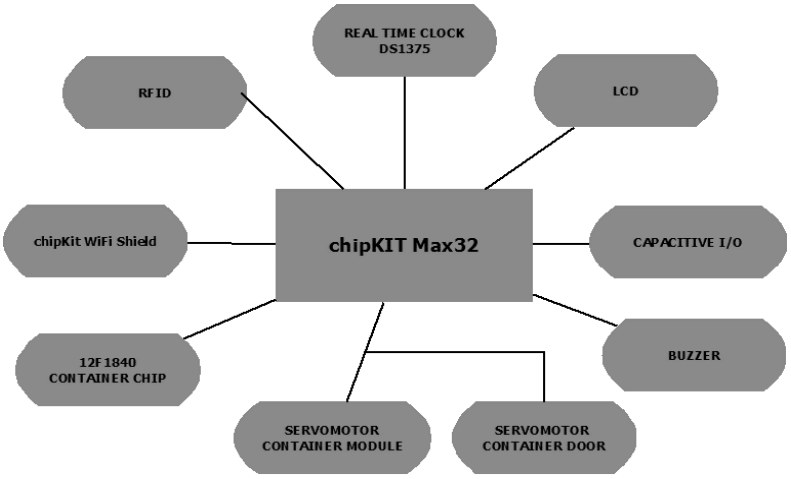



Figure 54. The medication dispenser hardware components [45].

The component that keeps track of time is a real time clock (Maxim DS1375), and it is responsible for dispensing the drugs at the correct time. It was chosen because it has an I2C interface and does not require a crystal.

An ID-12 Innovations RFID module ensures that only the authorized person can take the drugs. It was chosen because it supports ASCII, Wiegand26 and Magnetic ABA Track2 data formats. This module can operate at distances up to 12+ cm, being perfect for reading a RFID card.

The container is fitted with a PIC 12F1840 microcontroller responsible with the storage of the medicine list. Besides the name of the drugs, this microcontroller stores the date and time of actual dispensing. A microcontroller was chosen to act as a bridge for communication and to allow the authentication of the container. In order to keep the part count low, the internal EEPROM memory was used to store the data.

An LCD module is used to display information to the user and two capacitive button modules to acquire information from the user. A buzzer is fitted in order to allow an acoustic alarm when the patient needs to receive the medication.

#### 2.1.4.2 Software design

***A. The PIC 12F1840 software***

The software on this microcontroller acts as a bridge between the serial interface and the EEPROM internal memory, allowing storing values sent by serial interface directly in EEPROM and their retrieval. This storing method was chosen in order to increase security, this implementation allowing password protection over the data. On the EEPROM, a container start date is stored, along with 24 drug information. Each drug data contains the time difference from the last drug that must be administrated and the drug name.

### *B. The ChipKit Max32 software*

The program for the ChipKit MAX32 is created around the Ethernet (TCP/IP and UDP) libraries, written in such a way that any function tries to return as soon as possible, in order to avoid bottlenecks. This behavior allows a fast response to any external event, including external interrupts. Figure 55 shows the medication dispenser software flowchart.

The Ethernet library creates a web server, which allows a remote person to connect to the device and monitor the medication dispensing activity. The supervisor can access the devices web page from any location, via internet. The supervisor must be authenticated in order to be able to see the medicine dispensing related information.

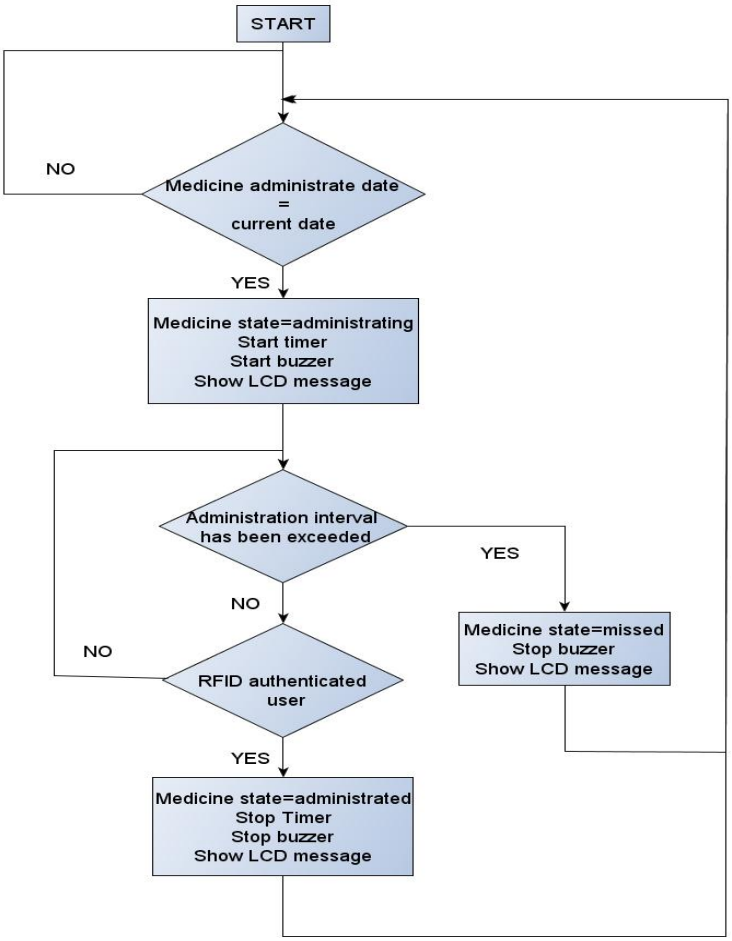



Figure 55. The medication software flowchart [45].

This project offers a complex assistance and medicine dispensing system. Due to the security checks used, RFID identification, the system is able to avoid potential critical medicine administration errors, and to ensure that the medication has been taken. It offers more features than other related products, like patient supervision via internet and almost instant setup. The system is open to extension and integration with the smart house and a wide range of sensors.

### 2.1.5 Microcontroller based health monitoring system

Nowadays, the health monitoring is an outstanding research area. Throughout the world lots of researchers and health institutions deal with health monitoring. The collected information sets about a patient helps the doctors in the examination of health state changes. Currently, there are many health monitoring devices and applications (e.g. intelligent watches and mobile applications) but most of those devices are expensive and difficult to acquire. Our goal is to develop a generally applicable network monitoring system which is available for everybody.

In our work [46] we developed a simple health monitoring system which is well applicable at home or in an institution. The system provides a good opportunity to gather and store health information about one or more observed patients. The collected information can help analyze the patient's state of heath. In the regularly gathered information the analyzer methods can search patterns which refer to symptoms. Therefore doctors can prevent or treat the initial illnesses.

On Figure 56 the simplified structure of the system can be seen. The system includes one Java server and lots of microcontroller based clients.

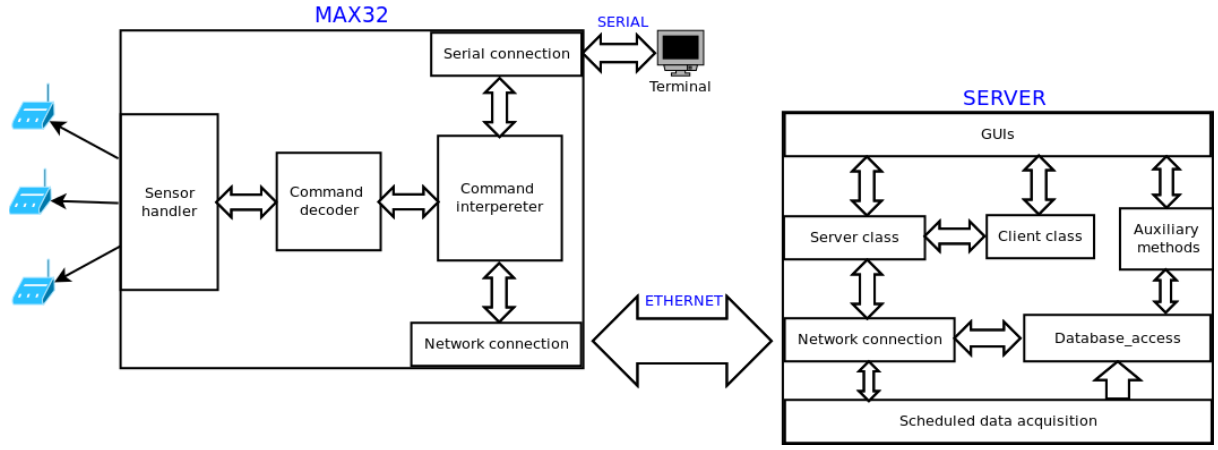



Figure 56. The simplified structure of the system [46].

**Server side.** On the server side, a Java based program handles the incoming information and stores it in a MySQL database. The user can reach the logged clients and queries data sets from the database. The server is responsible for data evaluation based on statistical methods.

**Client side.** Clients are responsible for sensor control and data transmission. The client supports two data acquisition possibility. The first is the regular data acquisition. In this case, data collection happens at adjustable periods. In the second case data collection depends on the patient decision. The regular data acquisition is suited for wireless sensors while the second option is mainly suited for wired sensors.

#### 2.1.5.1 The Client side

The clients are based on a Chipkit Max32 board with Microchip PIC32MX795F512 microcontroller. Two additional modules were attached to the base board: a network and an IO shield. The network module ensures the Ethernet communication. It contains a SMSC LAN8720 10/100 Mbit Ethernet PHY which covers the first layer of OSI model. The IO module provides the user interface of the client. It comprises an OLED display, 8 LEDs, 4 push buttons, 4 slide switches, a temperature sensor and some other elements.

Every client controls some essential sensors such as blood pressure, body temperature, pulse and oxygen in blood sensors. Depending on the type of the sensor the used communication protocols can be different. Therefore, it is recommended to separate those functions which are responsible for a given task into classes or header files. The client supports two communication possibilities: network and serial. The server and clients can communicate via network. The serial communication is necessary when the user would like to set the client parameters (ID, static IP, etc.). Those parameters are individual and can be set via serial communication only. In addition, serial communication is sometimes useful when the user want to test the client. Actually, clients can interpret 16 commands. Moreover, the administrator similarly can test the client from the remote server and can require data acquisition. The serial connection is available from a simple terminal if the appropriate port and baud rate (115200 baud) are set.

#### 2.1.5.2 The Server side

The server program was created in Java programming language. It is responsible for:

- network management
- data acquisitions
- database management
- data evaluation
- user interface

The server software is very user-friendly and provides high-level of controllability. The program ensures graphical user interfaces (GUI) to all features that facilitate the usage. On the server the users can see the logged clients and can communicate with them. Thanks to the command interpreter and decoder the client is controllable from the server by commands. On the server the “network” GUI is responsible for all network properties. The “network” GUI indicates the base parameters of the selected client and contains a command line area where users can type commands. The “network” GUI can be seen on Figure 57.

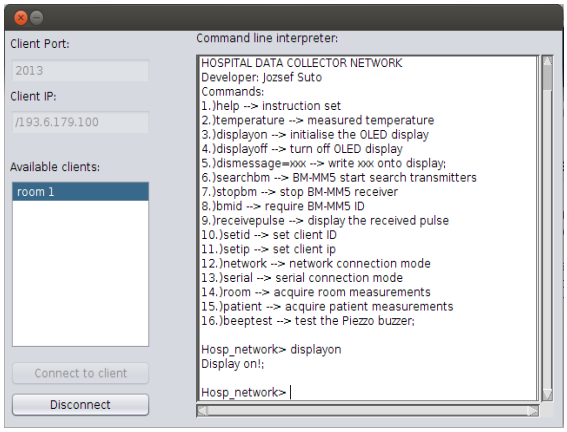



Figure 57. The “network” GUI [46].

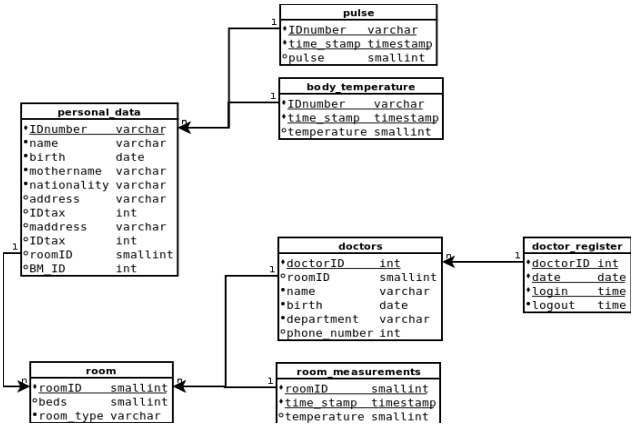



Figure 58. Database scheme.

**The database** was created in MySQL (Figure 58). The measured information sets (blood pressure, pulse, etc.) are stored in distinct tables. The tables have a composite primary key where the first element is a timestamp and the second is the serial number of the applied sensor. In this case, sensors should be unique. Before the client sends the packet, it inserts the serial number of the sensor. If an information packet arrives to the server, the server attaches a timestamp to the information and inserts it into the appropriate table.

**Data evaluation possibilities**. Currently, the server supports some elementary data evaluation possibilities. The data analysis is based on statistical methods. For instance: mode, median, mean, standard deviation. Under Linux from a Java program the developer can call shell script which contains one or more embedded GLE (Graphics Layout Engine) scripts. GLE is responsible for the “analysis” GUI. This GUI displays the evaluated data graphically.

**To sum up.** There are many health monitoring devices and applications but most of those are not generally applicable. In contrast, the presented system solves some essential tasks and provides versatile application possibilities. Consequently, an expanded and improved version of the system is applicable generally. In order the system to be usable in a hospital, clients and server should handle the highest layer (application) of TCP/IP model because the system has to pay attention to data protection. In the future the system will be extended with some new sensors and analysis methods (mainly neural network based).

## 2.2 Wearable systems for activity and health parameters monitoring

The monitoring systems for human activity and (or) health, which process data from different types of sensors worn by the assisted people have become increasingly popular. Using data acquired we can draw some conclusions about the activities of the monitored person as well as on his health and mental status.

We have developed tested and compared several wearable modules for some vital parameters' monitoring, e.g., temperature, heart rate, acceleration, etc. The acquired data were logged locally or on a remote server for advanced processing. The data were used for initial training of the ANN and later for real time recognition of the activity and health status of the patient. Our research work was focused on the design and test of several monitoring systems and recognition systems using artificial neural networks for pattern recognition.

Our contributions to this subject were disseminated through the following publications:

- Microcontroller based health monitoring system [46],
- Human Activity Monitoring System Design Implementation and Test [47],
- Human activity recognition using neural networks [48],
- Wireless data acquisition system for IoT applications [54],
- Wearable sensors network for activity recognition using inertial sensors [55],
- Activity recognition using an e-Textile data acquisition system [57],
- Wearable sensors network for health monitoring using e-Health platform [58],
- Real time human activity monitoring [59].

Also we investigated the possibility of monitoring human activity using portable devices [33].

An overview of these works is presented in the next two subparagraphs.

### 2.2.1 Design of a human activity and/or health monitoring system that process data from different types of sensors

We started the development of the prototypes using off the shelf modules in combination with modules developed by us. In the beginning we used the Chronos watch from Texas Instruments (TI) as acceleration data source combined with a chest belt from BM Innovations as heart rate data source. The receiver were built-up from a ChipKit Max32, a Wi-Fi shield and a communication shield that holds the BM receiver and the TI access point.

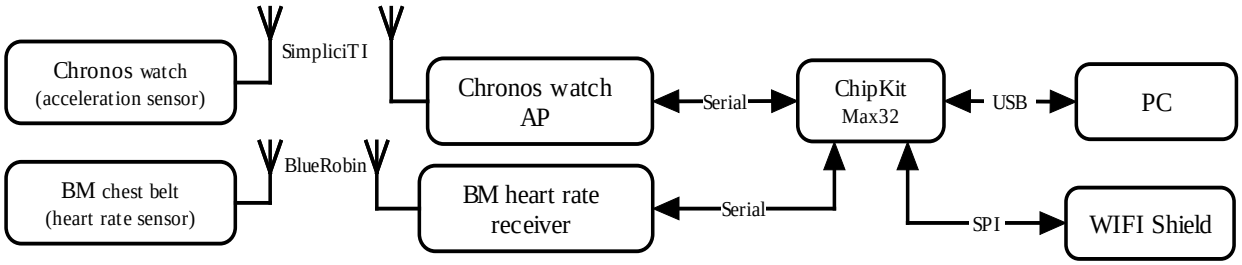



Figure 59. Activity/health monitoring system.

- **The ChipKit™ Max32** is based on the Microchip PIC32MX795F512 32-bit microcontroller, compatible with the most Arduino™ microcontroller board shields [49].
- **WiFi shield**, allows to connect the platform to remote locations or to interface a webserver hosted in MAX32 board with client applications, via internet. This board is develop based on Microchip MRF24WB0MA WiFi module [50].
- **eZ430-Chronos watch** is a complete CC430-based wireless development system which provides useful and complete hardware/software resources to implement wireless smart watch applications. It is based on the CC430F6137 <1 GHz RF SoC. The eZ430-Chronos has a 96 segment LCD display, an integrated pressure sensor and 3-axis accelerometer [51]. Also offers temperature and battery voltage measurement and is complete with a USB-based CC1111 wireless interface to a PC.
- **Chronos access point** was connected to microcontroller using a VDIP1 [52] host USB controller in order to be controlled using the UART or SPI interface on the VNC1L device.
- **BM-CS5R chest strap** is used as heart rate sensor and a data transmitter using BlueRobin™ data transmission technology [53].
- **Three different wireless protocols**: SimpliciTI for communication between Chronos watch and its access point, BlueRobin for communication with the heart rate belt and WiFi for communication with the gateway unit.

A newer version of the communication shield that was developed could receive acceleration data from 2 or three Chronos watches, a heart rate monitor chest belt and has an incorporated Bluetooth module. Also the shield holds an SD card interface for storing the received data and a RTC module for time stamping the received data (Figure 60).

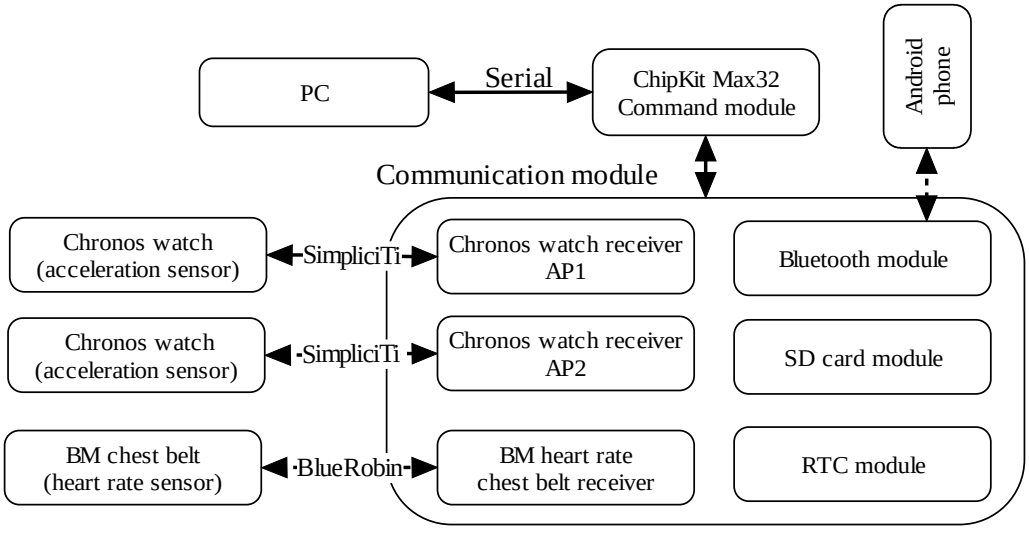



Figure 60. Activity monitoring system [111].

Max32 board use serial communication with VDIP module and Chest belt receiver, so two serial interfaces. Data received from those modules are sent to PC for data logging and advanced processing in Matlab. Some commands are implemented to establish communication between the chest belt and MAX32 board.

In the main loop of the program, data from the chest belt are requested and tested if they are valid, by testing the RSSI communication parameter. Also, acceleration data from the Chronos watch are requested and tested to be valid by testing the parameter contained in the header of data string. After validation of this parameter, all data are sent to PC.
The resulting data matrix contains in the columns data from heart rate sensor, V(hr), and data from acceleration sensors V(a1x), V(a1y), V(a1z), V(a2x), V(a2y) and V(a2z):

$$\text{data} = [V^T(\text{hr}), V^T(\text{a1x}), V^T(\text{a1y}), V^T(\text{a1z}), V^T(\text{a2x}), V^T(\text{a2y}), V^T(\text{a2z})] \quad (53)$$

or in extended form

$$\text{data} = \begin{bmatrix} hr_1 & a_{1x1} & a_{1y1} & a_{1z1} & a_{2x1} & a_{2y1} & a_{2z1} \\ hr_2 & a_{1x2} & a_{1y2} & a_{1z2} & a_{2x2} & a_{2y2} & a_{2z2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ hr_n & a_{1xn} & a_{1yn} & a_{1zn} & a_{2xn} & a_{2zn} & a_{2zn} \end{bmatrix} \quad (54)$$

where n is the number of acquired data samples.

***Wireless transmission of acquired data***

A wireless data transmission was implemented using the WiFi shield that is used to send data to a remote PC, hosting a database, or a webpage to display charts or information based on this data. Wireless communication implementation starts with specific library inclusion and TCP connection parameter definition. TCP communication parameters includes the SSID of the gateway wireless router, in this case is “first”, IP address, gateway IP, subnet mask, DNS and port number used to server connection. Wireless network connection process is started with proper security configuration parameter. Network password is stored in a constant character string. The connection process continues with connection status check and a TCP connection to a web server with stored parameters (IP address, gateway address. In this case connection is made to an Automatic Packet Reporting System (APRS) server. Connection to APRS server requires a specific IP address and port number and also a login process. After a successful login operation, data can be sent to the database for processing. APRS system allows to display object parameters and to place him on map. Data for object placement on map it can be obtained from a GPS or they can be set manually. In this project these data are fixed because are used only for tests. More details and an example of data transmitted, to APRS server is presented in [54].

### 2.2.1.1 Data aquisition system using inertial sensors

In these researches [55] we have tested many configurations using up to three MPU-9250 inertial sensors and using as main module a Waspmote board and XBee communication modules. Waspmote is a development platform equipped with an ATmega 1281 microcontroller that could be connected using 10 different communication protocols: 802.15.4, ZigBee, WiFi, Bluetooth, GPRS, 3G, RFID, NFC, 868MHz and 900 MHz. It has many incorporated interfaces and sensors: two serial interfaces GPS, SPI, SD card slot, RTC, temperature sensor and accelerometer. The programming environment is the Waspmote IDE. The data acquisition system using two inertial sensors is presented in Figure 61.

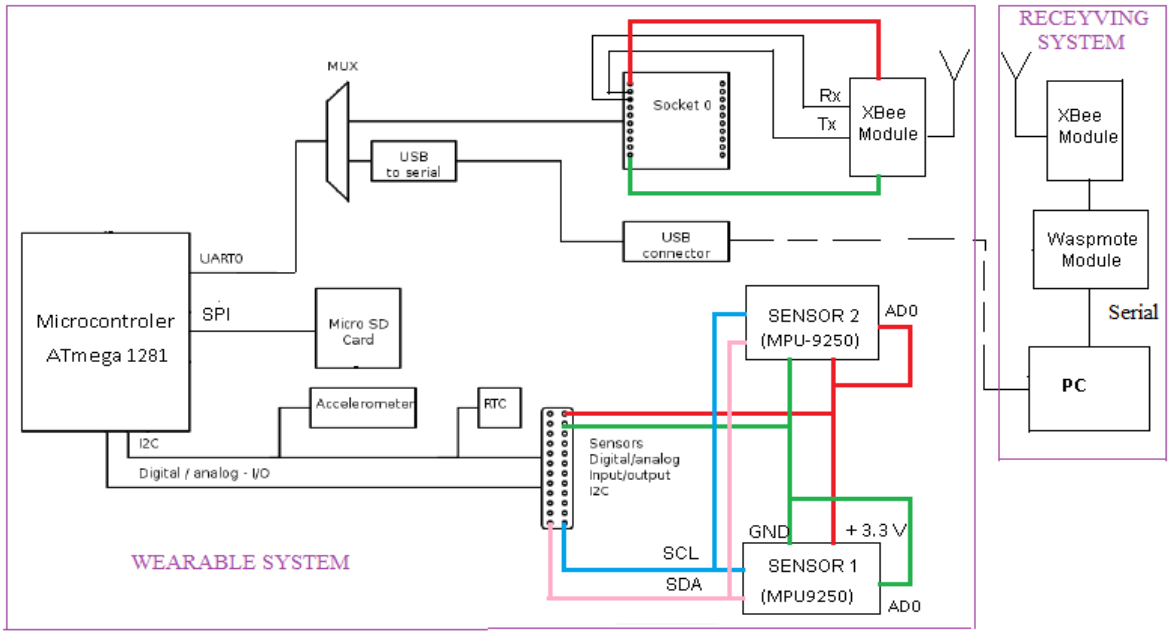



Figure 61. Data acquisition system using two MPU-9250 sensors [55].

The MPU-9250 sensor is a 9DOF (Degree of Freedom) inertial sensor that contains a 3-axis accelerometer a gyroscope a magnetometer and a Digital Motion Processor (DMP). It is also called nine-axis MEMS motion tracking device. MPU-9250 has three 16-bit analog-to-digital converters (ADCs) for each sensor. Communication with all registers of the device is performed using either I2 C at 400 kHz or SPI at 1MHz [56]. The acquired data structure from one sensor is presented bellow

$$\text{data} = \begin{bmatrix} a_{1x1} & a_{1y1} & a_{1z1} & \theta_{1x1} & \theta_{1y1} & \theta_{1z1} & m_{1x1} & m_{1y1} & m_{1z1} \\ a_{1x2} & a_{1y2} & a_{1z2} & \theta_{1x2} & \theta_{1y2} & \theta_{1z2} & m_{1x2} & m_{1y2} & m_{1z2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{1xn} & a_{1yn} & a_{1zn} & \theta_{1xn} & \theta_{1yn} & \theta_{1zn} & m_{1xn} & m_{1yn} & m_{1zn} \end{bmatrix} \quad (55)$$

where $a$ is the acceleration, $\theta$ is the rotation angle, $m$ the magnetization and $n$ the number of samples acquired.

#### 2.2.1.2 Other monitoring systems

We implemented and tested many types of activity and health monitoring systems as for example: a data acquisition system using e-textile sensors [57], a wearable sensors network for health monitoring using e-Health platform [58], etc.

Latter we also designed a wearable watch sized, low consumption, acceleration sensor tag (Figure 62). It sends the 3 axis acceleration data of the body part on which is placed. The device is composed by an ADXL350 acceleration sensor from Analog Devices, a CC2541 low power SoC for Bluetooth low energy (BLE) applications, from Texas Instruments and a TPS61220 Step-Up (Boost) converter. The tag is powered by a single coin cell battery (CR2032).

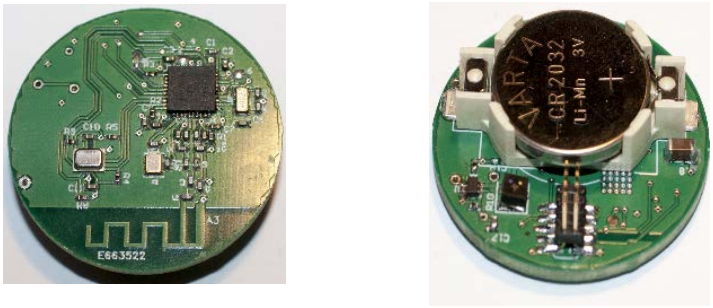



Figure 62. Acceleration sensor tag [111].

### 2.2.2 Real time human activity monitoring

The aim of this study [59] is to present a method to the activity classification. In addition, an objective of this research is to create a real time system which monitors the daily activities of an observed patient. The observed patient should wear the device during the day. Therefore, the device have to be portable and small. One of the most important criteria of the device is the cost. Since the system consists only one data collector device, the cost of the system is low. Obviously, more data collector devices and sensors require higher cost. Moreover, another criteria is the independence. This means that, hospital environment is not necessary to the observation. The system facilitates the self-care and enhance the independence of the patients against the public health systems.

In the system the data collector device is a Raspberry Pi (RPi) with an ADXL345 3-axis accelerometer and a Roving RN-171 WiFi module. The device was fixed into a thin plate which provides stable position on the chest. Using the Linux operation system on the RPi, the developer can easily create high level programs.

In contrast, our method present the following advantages:

- Works in real time. The disadvantages of most analysis techniques is the offline mode. In this cases, the analysis take place after the data acquisition on a computer by some well-known software (Matlab, Labview, etc.).
- Data evaluation is made on the same device that performs the data acquisition. For this, the two tasks are running simultaneously in parallel on the device.
- A particular pattern recognition technique was tested. This simple technique runs an ideal pattern through the time varying signal and calculates the shifted and summarized square error (SSE). Every activity has an own rhythm which describes a periodical pattern. If an algorithm can recognize the patterns, it can define the current movement.

On the RPi a C++ program performs the data acquisition and data evaluation. In the C++ code the POSIX thread or Pthread library allows parallel programming [60]. Pthread is a set of C programming types and procedure calls. The data acquisition and the analysis are independent tasks which can be executed concurrently. The analyzer program uses two threads. The main thread collects the measured acceleration components and an auxiliary thread performs the data analysis. Figure 63 illustrates the parallelized data collection and evaluation.

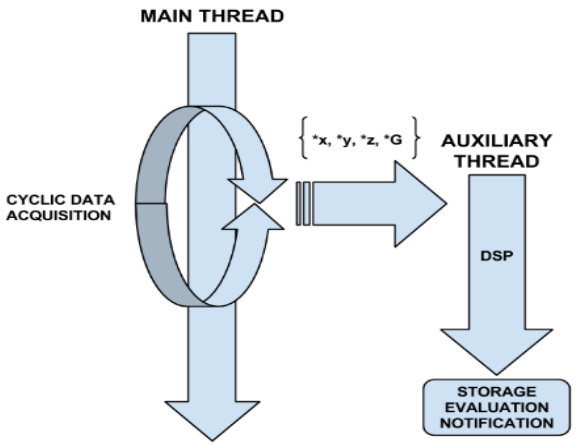



Figure 63. The structure of parallel operation [59].

While the device is active, the data acquisition function runs continuously and stores the collected data in buffers. Currently, the data acquisition frequency is 100 Hz and the buffer size is $2^8$. Consequently, the analyzer method splits the continuous signal to short parts (about 2.5 seconds long) and tries to decide the current activity. When the buffers are full, the auxiliary thread starts and gets a void* structure which includes the buffers. Pthread permits to pass only one argument to the new thread, therefore every argument have to be embedded into a structure. The auxiliary

thread will call the digital signal processing (DSP) and pattern recognition functions. Furthermore, the auxiliary thread is responsible for the storage and notification. If the data analysis finishes, the decision about the movement will be stored in a file and sent to a server in IP packet. A strict constrain exists, namely that the auxiliary thread has to be faster than the data acquisition process. Thereby, the program can avoid thread collision. The *x, *y, *z are buffers which contain the x, y, z components of the accelerometer and *G buffer includes the normalized acceleration magnitude.

#### 2.2.2.1 Description of the method

The activity analysis is based on time-frequency signal processing. First, the algorithm categorizes the acquired signal in the frequency domain. In the next step, according to the frequency category, it will search the possible patterns which belong to the assigned category. The detected pattern will identify the movement type.

Figure 64 illustrates the flowchart of the applied algorithm. On Figure 64 there is an *unknown activity* state. It means that, if the examined signal contains an incomprehensible sequence, the algorithm will not give a decision.

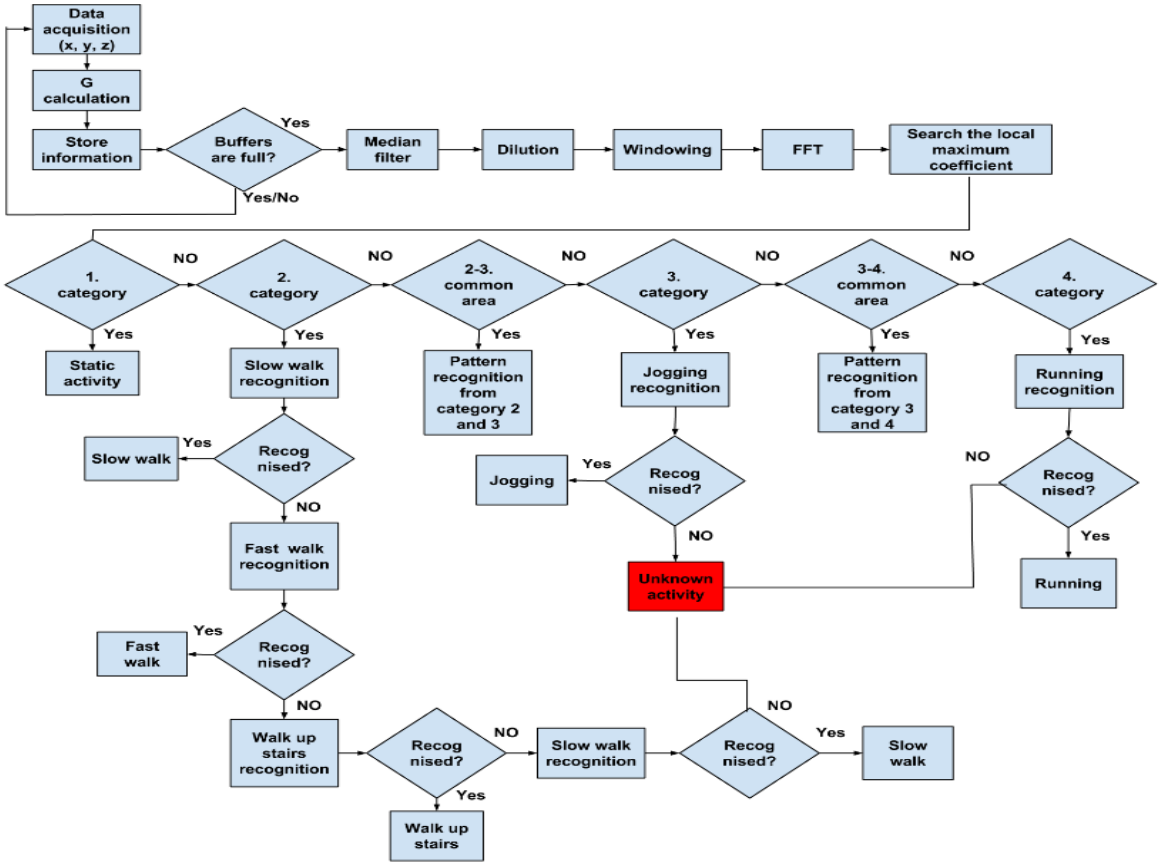



Figure 64. The flowchart of the algorithm [59].

During the acquisition process, the program calculates the normalized acceleration magnitude (G[i]) from the collected parameters (x, y, z).

$$G[i] = \frac{\sqrt{x[i]^2 + y[i]^2 + z[i]^2}}{1g}$$

where 1g depends on the resolution of the accelerometer. The G characterizes the change in the movement, thus it will be the key in the analysis. Firstly, the frequency coefficients of the G signal was calculated. Before the fast Fourier transformation (FFT), the signal was diluted and windowed with a Blackman-Nuttall window in order to minimize the leakage and separate the closely spaced frequencies. The length of the signal influences the frequency resolution. Consequently, if the frequency resolution is higher, then the frequency categorization is easier. The real signal size is $2^8$ which will be diluted to $2^9$ with zeros. After the dilution the extended signal was multiplied with the window function.

In order to the auxiliary thread can analyze the G signal faster than the data acquisition (less than 2.5 seconds), we created an optimized FFT algorithm to calculate the frequency coefficients [61]. Since, the signal length is fix and the FFT runs periodically in the program therefore worth to store the "twiddle-factors" into memory as predefined constants [138]. Generally the radix-2 FFT can be written as,

$$X(k) = E(k) + W_N^k O(k) \qquad k = 0,1, \dots \frac{N}{2} - 1$$

$$X(k) = E\left(k - \frac{N}{2}\right) + W_N^k O\left(k - \frac{N}{2}\right) \qquad k = \frac{N}{2}, \dots, N - 1$$

In the equation E(k) and O(k) contain the frequency coefficients to the even and odd elements and $W_N^k$ for k = 0, ...,N −1 is the N'th root of unity. If we take advantage of the relations between $W_N^k$ factors (56), we will save memory because it is enough to store one-fourth of factors.

$$W_{NIm}^{k+\frac{N}{4}} = -W_{NRe}^k$$

$$W_{NRe}^{k+\frac{N}{4}} = W_{NIm}^k$$

$$W_N^{k+\frac{N}{2}} = -W_N^k \qquad (56)$$

In the above formulas *Re* and *Im* refer to the real and imaginary parts of a complex number. The Euler's formula (57) allows the decomposition of the $W_N^k$ factors into real and imaginary parts thus the real and imaginary parts of the factors will be stored separately in the program.

$$W_N^k = \cos\left(k\frac{2\pi}{N}\right) - isin(k\frac{2\pi}{N}) \quad (57)$$

After the frequency coefficients are available, the algorithm searches the maximum value inside a specified frequency interval. The frequency categories bound an interval between 0.8 Hz and 3.8 Hz. The maximum value defines the activity category. Table 16 contains the applied frequency categories and an approximate frequency interval. Actually, the data is stored in binary form thus the frequency intervals are based on the indexes (bins) of the frequency vector.

Table 16. Frequency categories [59].

| Categories | Min. frequency | Max. frequency |
|---|---|---|
| 1 | 0.0 Hz | 0.8 Hz |
| 2 | 0.81 Hz | 2.25 Hz |
| 2-3 | 2.26 Hz | 2.5 Hz |
| 3 | 2.51 Hz | 2.65 Hz |
| 3-4 | 2.66 Hz | 2.8 Hz |
| 4 | 2.81 Hz | 3.8 Hz |

According to the category, the appropriate pattern detection functions will be used on the G signal. Therefore, the number of operations greatly decreases. Unfortunately, between some categories there is a narrow overlapping. In that case, if the maximum coefficient is in the common area of two categories, then the algorithm will search each patters which belong to the two adjacent categories. Figure 65 illustrates the frequency spectrum of some main activities.

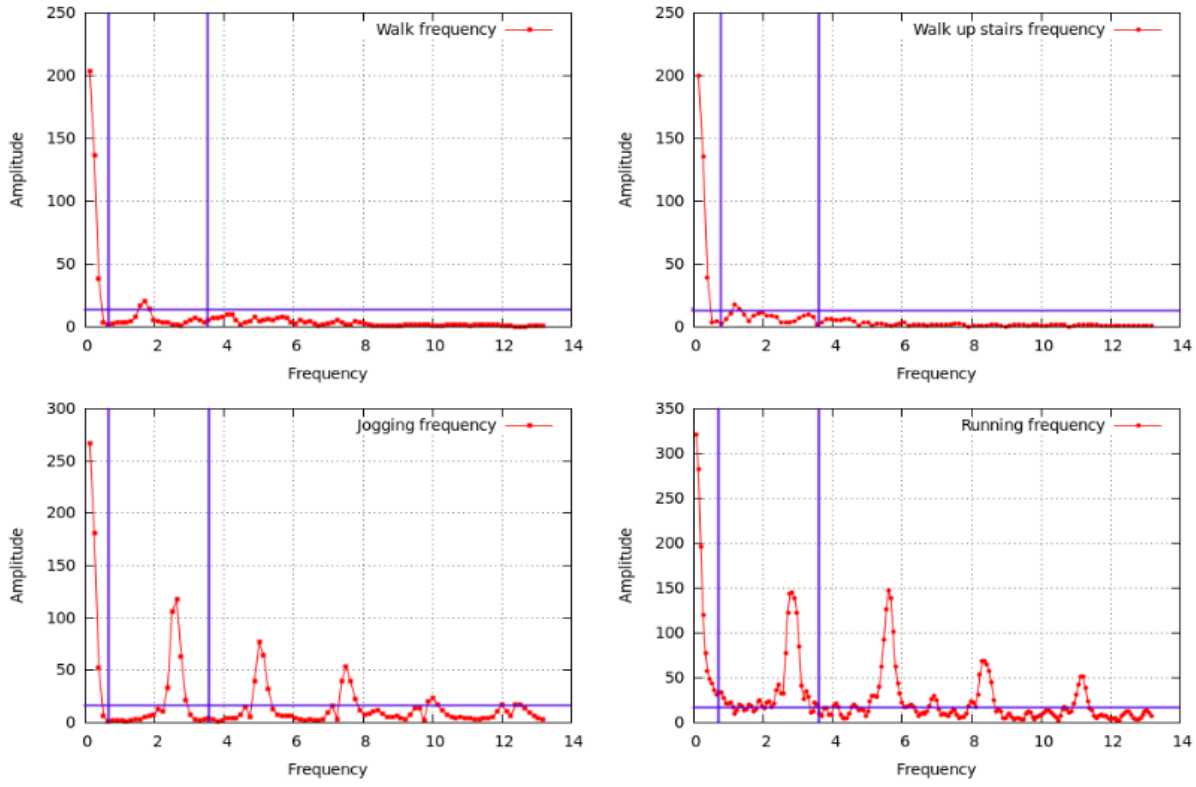



Figure 65. Frequency spectrum of some main activities [59].

The maximum frequency coefficient should higher than an appropriate amplitude limit. In this case, a reliable limit is between 14.5 and 16.5. On the above figure the blue lines indicate the frequency interval and the amplitude limit. If the maximum is lower than the limit, then the movement is static. According to the position the algorithm concludes the frequency category.

#### 2.2.2.2 Pattern recognition

The pattern recognition starts with a median filtering with a seven samples length window to reduce the noise on the G signal before the recognition [62], [63]. In the literature, one of the most common pattern recognition technique is the correlation [64].

$$(f \cdot g)[n] \coloneqq \sum_{m=0}^{N-1} f[m]g[m+n]$$

where f and g are real vectors. However, in some cases the correlation does not provide acceptable result. Consequently, we developed a simple and individual pattern recognition method. The method is a combination of the square error and the correlation. It can be characterized as a shifted and summarized square error (SSE).

$$SSE(f,g)[n] \coloneqq \sum_{m=0}^{N-1} (f[m] - g[m+n])^2$$

In both cases an ideal pattern (or kernel) with a special shape will pass through the examined signal. In a correlated signal the peak(s) are higher when the similarity between the pattern and the examined signal is large. However, in the SSE the low parts indicate the high similarity. The difference between the correlation and the SSE can be visualized with a simple example. The example compares two similar activities: walking and the climbing of stairs. In the example the climbing stairs pattern was used on the two activity sequences. In ideal case, if the sought pattern does not appear in the signal, all resulted points have to be less or higher as a well-defined limit, according to the applied technique.

Figure 66 presents the result where the blue lines are imaginary limits to the two pattern recognition techniques. As the example shows, the correlation cannot separate the activities with the used kernel. For instance, if we use the 76 (blue line) as limit, the same number of points will be higher than the limit in both cases. In contrast, the SSE separates the activities well. On Figure 66 a lot of points belong under 5 of the climbing stairs sequence while in the walk sequence every point is higher than 5.

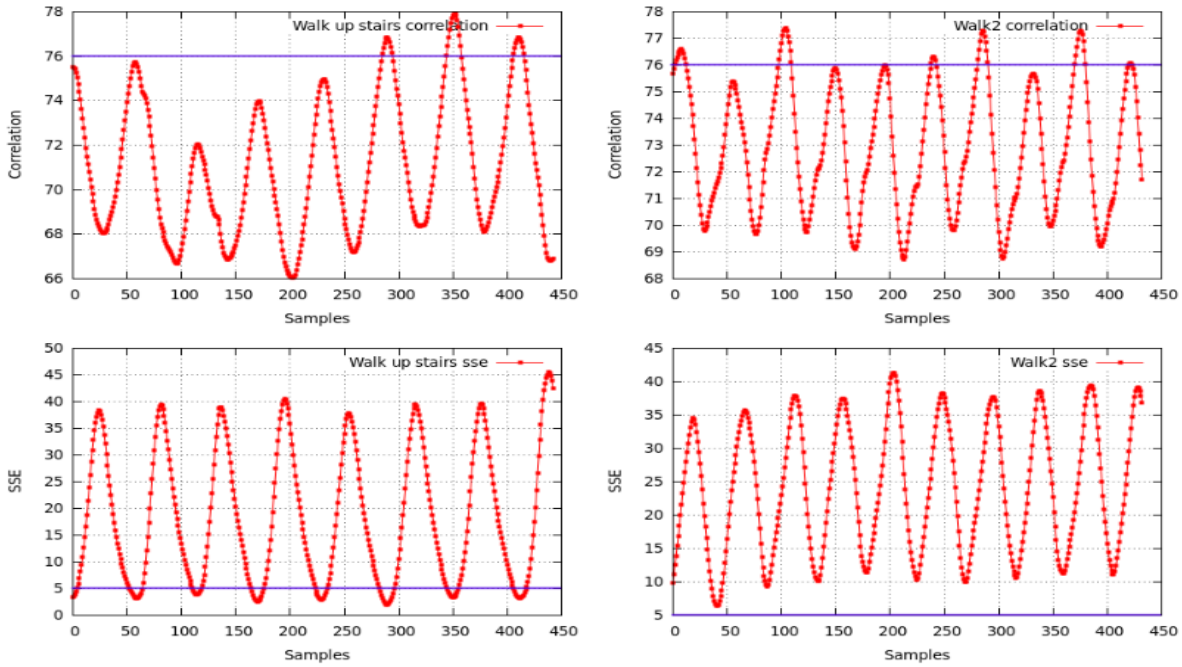



Figure 66. Comparison of the two pattern recognition techniques [59].

#### 2.2.2.3 Conclusion

The presented technique is well applicable for activity recognition. If an activity describes an individual and periodic acceleration change then the presented recognition algorithm will find the patterns. As Figure 64 shows, if we know the frequency interval and the described pattern of an activity, we can easily past the new activity into the method. Obviously, an unexpected event such as a fall is similarly recognizable because a fall (regardless of the direction) has a suddenly ascending and then decaying acceleration fluctuation.

The ability to evaluate the movement types provides an exceptional source of knowledge for doctors to diagnose patients. The movement analysis is a useful aid to detect potential causes of gait and lifestyles abnormalities [65]. Human movement is researched in relation to a lot of diseases, such us reduced mobility disorders (stroke, obesity), sclerosis and Parkinson's disease [66], [65], [67]. The presented solution to the activity recognition problem belongs to this branch of research which is expected to be determinative in the following years.

### 2.3 Development of an assistive robot providing personal assistance

The increasing number of the elderly population along with increased costs related to assistance of their independent living at home leads to extreme challenges that could be solved only by using assistive technologies in general and assistive robots in particular. In this way elderly citizens would need less human assistance or need human assistance at a much later stage in their aging process. The acceptance of technology still remains a delicate subject.

Major universities and research centers have research related to assistive robots, healthcare and life assistance [68], [69], [70], [71]. This research presents a work in progress, namely the development of an assistive assembly consisting of an assistive and telepresence robot platform together with the related components and services presented in several articles and at international conferences [72], [73], [74], [75], [76], [77].

A definition of terms assistive and social robot could be find in [71], [78], and [79]. The authors have made also a classification of the robots with special focus on assistive and social robots.

We aim to develop a complex autonomous robot as a part of our assistive system for elderly local and remote assistance. The robot must pay close attention to the elderly person, try to learn his behavioral patterns, give him verbal advices, could control the room temperature, and could alert the family or doctor in case of the assisted person is not feeling well. An important role of the robot is to entertain, playing the favorite music, video or photo slideshow, or displaying favorite internet sites. It can respond to question regarding calendar (day, month and year), temperature indoor and outdoor, daily expected events and meetings. It should behave like a sensitive family member and react to the problems of the patient.

The main functions required for the assistive robot are:

- Autonomous movement in patient's home
- Possibility of remote control over internet, telepresence (AV streaming)
- Patient physiological parameters monitoring, an abnormal state detection, emergency call
- Falling detection (using acceleration sensors)
- Message sending to physician or family members
- Phone book with pictures and automatic dialing (landline and mobile phones)
- Daily program assistance: reminder for taking pills, drinking water, etc.
- Controlling through RF/IR/Bluetooth of the smart home devices (lights, heating, blinds, door opening, etc.)
- Shopping list
- Entertainment
  - Audio files playback (music)
  - Video files playback
  - Photo slideshow

### 2.3.1 Assistive robot architecture

Figure 67 shows the architecture of the assistive platform. It is composed of several modules responsible with different implemented function:

- Android user interface and robot manual control

- Robot platform command module with microcontroller
- Communication modules (Bluetooth, 3G/GPRS)
- Motors drive modules
- Obstacle avoidance modules
- Movement mapping modules
- Power supply module

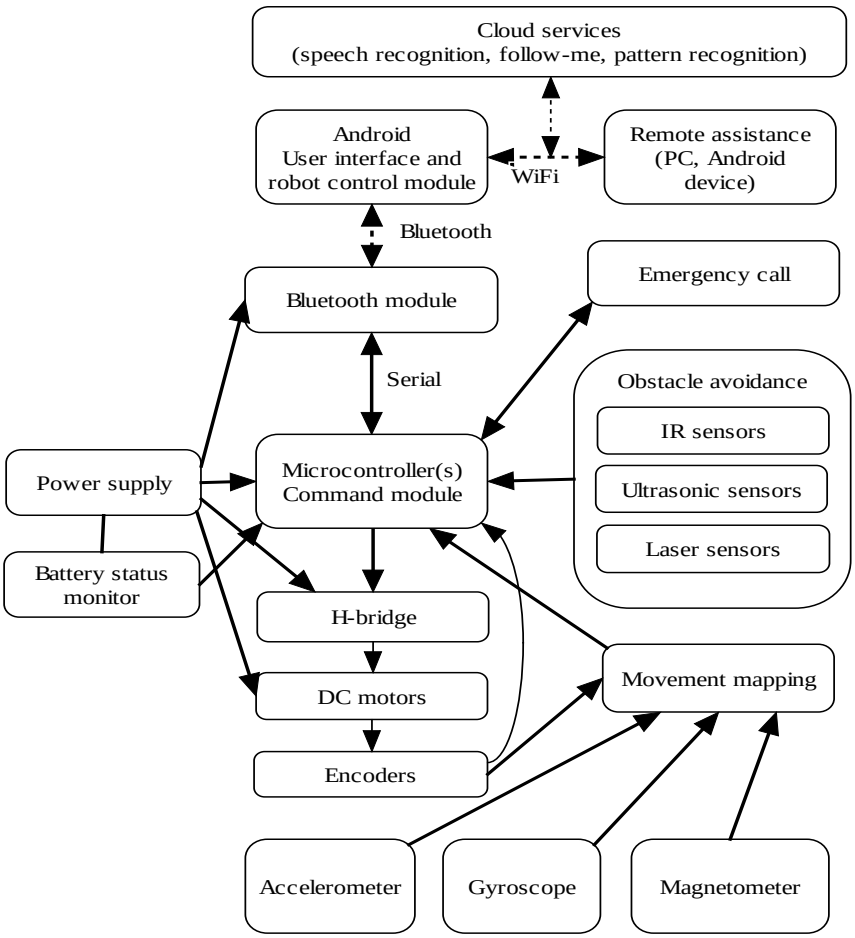



Figure 67. Assistive robot platform.

- **User interface** assures the communication with the robotic platform. The main interface is a Tablet PC running Android OS. The web interface running on it allows remote control of the robot and remote assistance of the assisted person. We have also tested the implementation of a web server in the microcontroller.
- **PC application.** Another possibility that we tried was the development of the remote assistance interface as an application running on PC. For control of the robot using postures/gestures we can use an intelligent bracelet or watch.
- **The command module** is equipped with one 32-bit Microchip® PIC32MX795F512 microcontroller on a chipKIT Max32™ board. The microcontroller operating frequency is 80 MHz. Max32 board has: 128K RAM, 512K Flash and 83 I/O ports.

This microcontroller was programmed with the MPIDE but can be also programmed with the MPLAB IDE because it has a Microchip microcontroller. The board controls the motors via H-bridges. Commands are adapted using feedback from encoders regarding the exact movement of wheels. The board could receive commands from the Tablet PC using Bluetooth communication, a Chronos watch developed by Texas Instruments via SimpliciTI RF protocol or via WiFi. The robot platform operating diagram is presented in Figure 68.

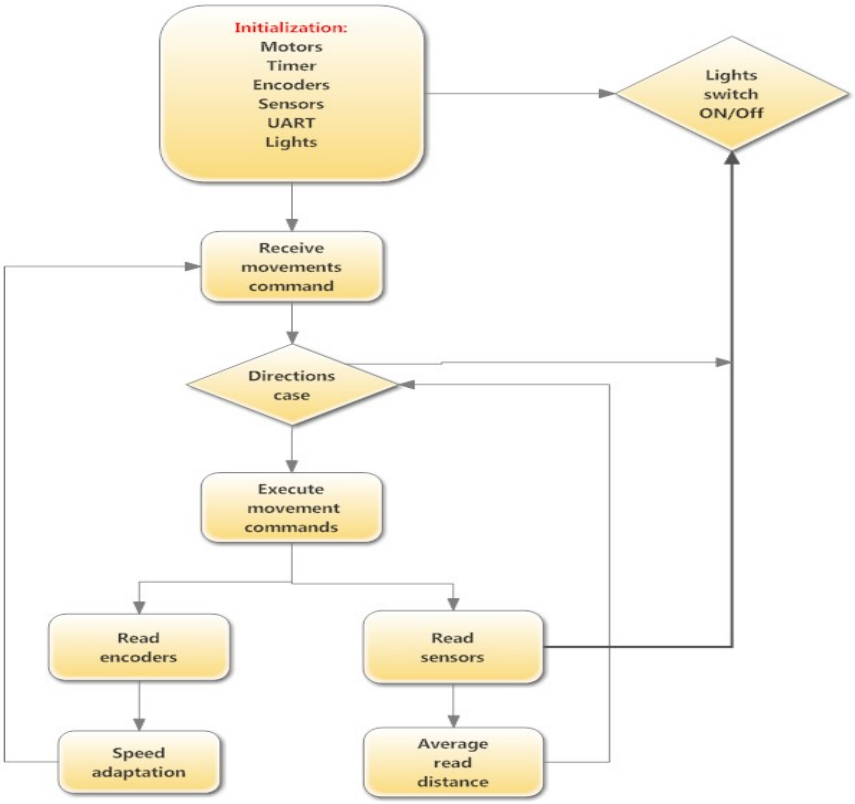



Figure 68. Operating diagram.

### 2.3.2 Android command center

We use a Tablet PC as a user interface due to its touch screen, the built-in sensors and the relative low price. There are many useful free applications on Google Play Store and the application development of new applications on Android OS is quite easy and fast.

In automatic mode the tablet is used for remote assistance. For this purpose we have implemented a server that is responsible for streaming the video and audio signal of the incorporated camera and microphone. Also it receives commands for robot movements and transmits this command to robotic platform via Bluetooth. The remote user can connect to the server using a web browser and the known IP, port, user and password.

Other functions implemented in Android are:

- Speech recognition (using Google speech recognition) and Text-to-Speech, for interaction with the user in the diagnosis and alert functions.
- Medication reminder is an important function which allows creation of a list of medications and to choose the intervals to be administered. The application shows the current medication and alerts when most be taken.
- Emergency Skype call of the designated person.
- Display of the house plan and robot movement.

In Figure 69 can be seen the services that are already implemented or are in plan to be implemented in next period.

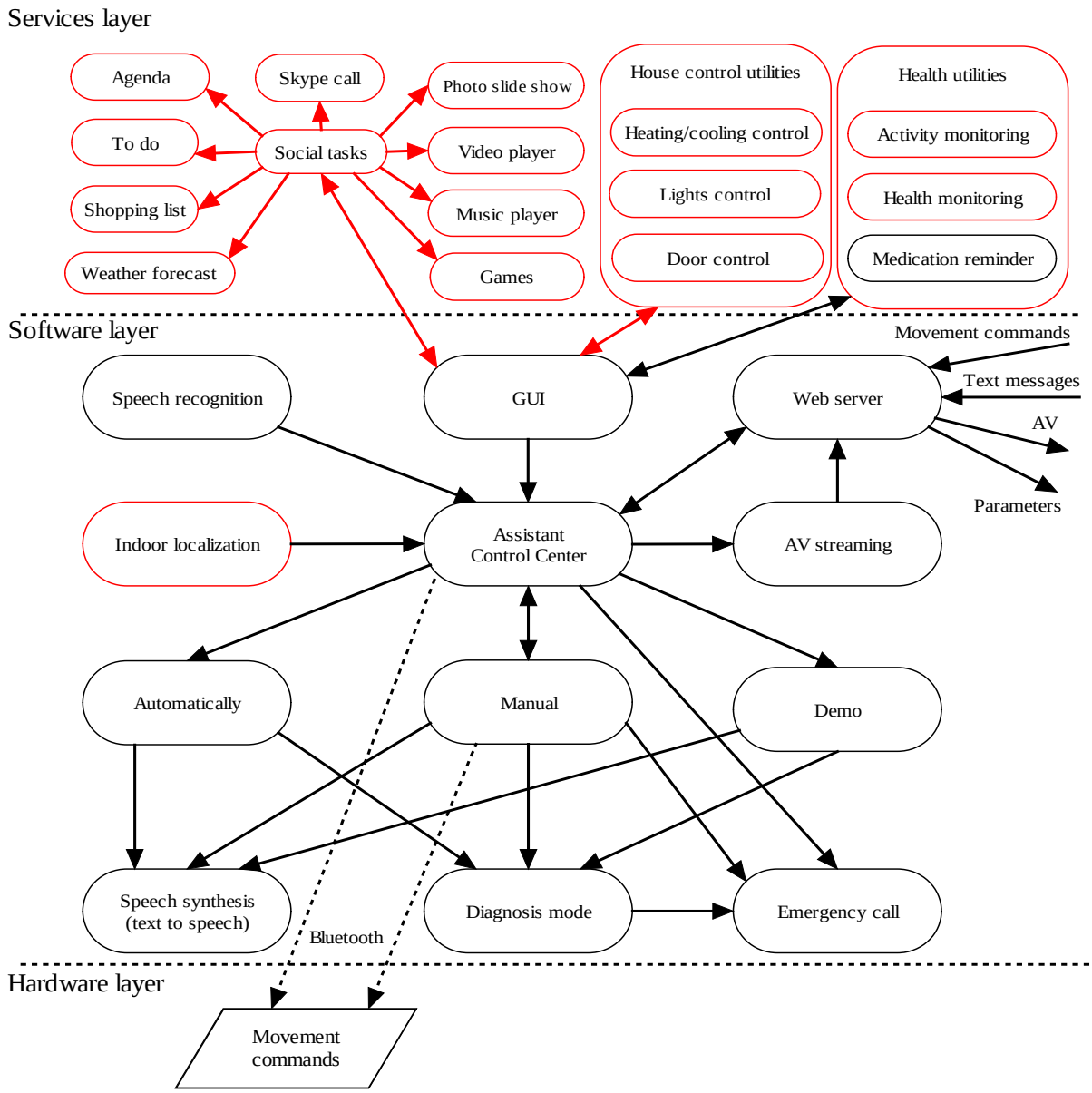



Figure 69. Functions implemented on Android command center [72].

The structure of the application is very complex, it contains more than 100 java files and more than 10 libraries. The MainActivity.java file contains the main activity that initialize the components and the interfaces. The interfaces extend the Fragment class. The structure of the interfaces accessible for the user is shown in next diagram:

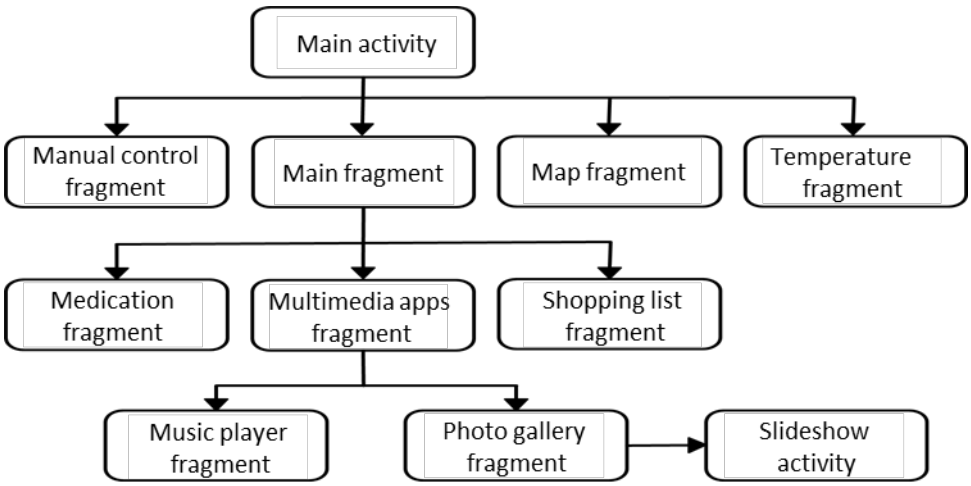



Figure 70. The structure of the application.

## 3 DEVELOPMENT OF METHODS FOR ACTIVITY RECOGNITION USING NEURAL NETWORKS.

Our research related to activity recognition were conducted in parallel in several directions. One of the direction was the development of a Matlab model of activity recognition system that use artificial neural network in order to recognize activity or health status of the patient and trigger alert signals in case of unusual state detection [47], [48], [80], [81]. Another direction was the development of hardware implemented real-time recognition system. Data provided by data acquisition system were used, on the one hand to train the artificial neural network and on the other hand to recognize the activities or health status. We simulated in Matlab several recognition systems for arm posture, body postures and for usual activities, like: lying on various sides, sitting, standing, walking, running, descending or climbing stairs etc.

The recognition system should use an algorithm that is capable to learn, generalize and adapt and also to tolerate the inherent errors (noise). From the possible biologically inspired algorithms we opted for the artificial neural networks. In the process of ANN design, the number of input neurons is given by the number of input data channels and the number of output neurons is given by the number of activities to be recognized. Finding a neural network model with good performance for a given application which is also easy to implement in hardware is not exactly an easy task. Only after several simulations of different ANN models we have opted for a Feed-Forward

Backpropagation ANN that give good results and also is relatively easy to implement in hardware using microcontrollers or FPGAs [7], [48], [59]. We have made many simulations in order to find the optimal number of hidden layers and number of neurons per hidden layer(s). Also we conducted studies regarding the proper activation function and best performing training function. We concluded that good results could be obtained with two-layer FF-BP network, with sigmoid activation function. We have chosen Levenberg-Marquardt training method because on the one hand it is the fastest backpropagation algorithm offered by Matlab and on other hand it gives goods results. For performance evaluation we used the mean squared error (MSE) function.

### 3.1 Methods used for activity recognition

We can find a huge variety of activity recognition methods in the literature. A survey of Human Activity Recognition using Wearable Sensors is presented in [82]. The methods used for activity recognition generally have two phases. First phase is features extraction and selection and second is the classification phase. For feature extraction the most used methods could be on the one hand statistical methods (max, media, variance, standard deviation, energy) called also time-domain features [83], and on the other hand could be extracted frequency-domain features using FFT [84], [85], [86] or time-frequency features using wavelet filter banks [87], [88], [89], [90]. A comparison between this methods are presented in [91], [92] and [93].

Regarding the classification, in literature are presented many methods which include: data mining algorithms such as Naïve Bayes classifier [94], [95], Bayesian Networks Classifiers [96]. Other encountered methods of classification are Support Vector Machines (SVM) [97], [98], Decision Trees classifiers [84], k-nearest neighbor (kNN) classifier [91], hidden Markov models [99], [100], and neural networks [93], [101]. A comparison between classifications methods are presented in [102] and [103].

### 3.2 Designing the recognition system based on ANNs simulated in Matlab

The Matlab implementation of the desired recognition method is considered the first step before migrating towards the more dedicated hardware-software environment implemented in the FPGA circuits.

The recognition system is tailored for meeting the expected recognition rate accuracy of at least 90%. It basically consists of an artificial neural network modelled in Matlab and a series of pre-processing modules (filtering, normalization, feature extraction).

Considering the nature of the data provided by the sensors used in the body posture recognition system a recognition system should use an algorithm that is capable to deal with nonlinear data, is able to reconfigure, learn and generalize and also to tolerate the inherent errors (noise). The most important recognition algorithms developed so far have a biological influence and can be grouped in the following categories: artificial neural networks, fuzzy systems, genetic algorithms, neuro-fuzzy systems. All the enumerated systems share these main characteristics: parallel computation, learning, generalization and flexibility. In the following, the artificial neural networks are analyzed.

Finding the right topology of the ANN for a specific application is a challenging task and only through several simulations of different ANN topologies, in terms of numbers of neurons per layers and numbers of hidden layers, can be found the optimum one. In our case after several tries we obtained good result using a two-layer feed-forward network, with sigmoid activation function. The training function adopted was the Levenberg-Marquardt optimization (trainlm). The training method is the fastest backpropagation algorithm offered by the Matlab. The performance function is set to the MSE function. It measures the networks performance according to the mean of squared errors.

### 3.2.1 Arm posture recognition

The first recognition experiments were made for 6 arm postures as are presented in Table 17. Acceleration data are supplied by TI Chronos smart watch wore on the wrist.

Table 17. Arm postures to be recognized [48].

| | |
|---|---|
| 1. arm downwards | 4. arm horizontal backward |
| 2. arm upwards | 5. arm horizontal forward rotated upwards |
| 3. arm horizontal forward | 6. arm horizontal forward rotated downwards |

A sample of data acquired from acceleration sensor is presented in Figure 71.

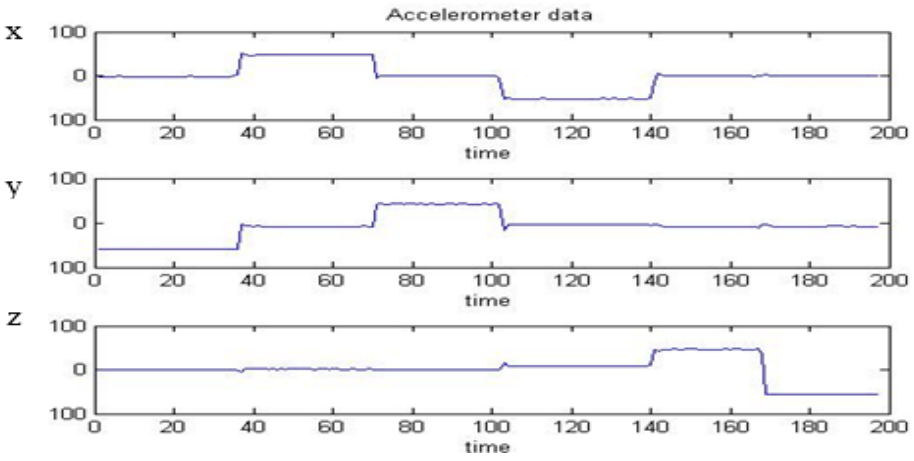



Figure 71. Sample of data acquired from acceleration sensor for the 6 arm postures [48].

The ANN model is presented in Figure 72. As the input data vector was formed from 3 sets of data, one for each axes of the accelerometer, the neuron numbers in the input layer was set to 3. Considering that there are 6 postures to be recognized, six neurons have been placed on the output layer. For the neuron from the hidden layer have been tried different numbers. The recognition rate was 100 % on the data used for training.

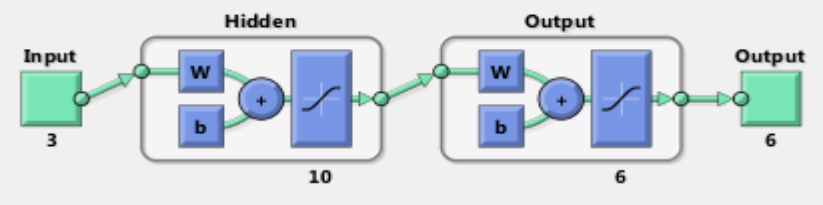



Figure 72. ANN used for arm posture recognition.

#### 3.2.1.1 ANN Matlab design

The Matlab code used for designing, training and simulating the network consists of the following main parts:

- *data load*: data is imported in the Matlab environment;
- *data extraction per axis*: the imported data is a repetitive column of values acquired from the 3 axes of the accelerometer. The z axis data has to be corrected therefore an offset value is used.
- *2's complement conversion*: the values sent by the acquisition system are coded in 2's complement and therefore a conversion function has been used.
- *function for converting the 2s complement* represented on n bits to decimal
- *data filtering*: the data bulk acquired from the accelerometer needs filtering for smoothing its shape and making it more recognizable. The filtering consists in mediating the values over a predefined window of 5 samples.
- *normalization function* (to be applied when exists vectors with different value ranges, such when the heart pulse rate vector data is used). This function can be skipped when the Matlab's *mapminmax function* is used in the pre-processing stage
- *randomize the values* used for training. Use indexes to keep the learning – training data pairs intact
- *create the corresponding target data*
- *create the ANN*. The work flow for generating a neural network design has several stages, which are resumed below
  - assign the input and target data vectors (the pairs are used to train the netwok)
  - set the neural network topology based on the input vector and output vector dimensionality and on chosen neurons in the hidden layer.
  - for a better recognition rate data is pre-processed. Analysing the data used for training the *'removeconstantrows'* and *'mapminmax' (to avoid the transfer function saturation)* functions have been considered for pre-processing.

- Data set for training is split up in 3 subsets for training, validation and testing. For dividing the data into the subsets the *divideram* function was used. It randomly divides the input data into the subsets using the division parameters set by the user, for this case 70/15/15.
- The training function adopted was trainlm that updates the weights and biases values of the neural network according to Levenberg-Marquardt optimization.
- The performance function is set to the MSE function. It measures the networks performance according to the mean of squared errors.
- Training the network

- Plots: the data/accelerometer axis acquired from the sensors may displayed on graphs
- Other parameters may be displayed

#### 3.2.1.2 Correlation of the accelerometer data with data from the heart sensor

In the subsequent experiments, we defined 10 hand postures combined with some activities to be recognized. In this case data from the accelerometer has been correlated with the one from the heart rate sensor. The measurements are and displayed in Figure 73. The feature extracted from heart rate data was the effort, which was calculated as a difference between the present sample and the sample acquired 30 samples ago (the window width is 30).

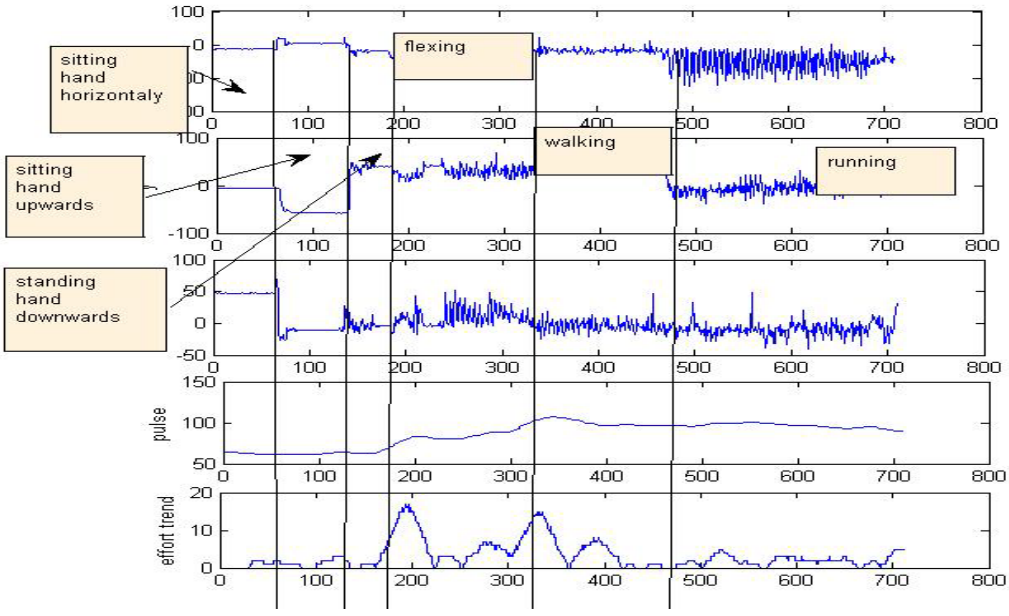



Figure 73. Heart monitor and accelerometer data correlation.

The topology that has been chosen has just one hidden layer and with 4 neurons on the input layer, 12 neurons on the hidden layer and 10 on the output layer. These numbers have been dictated by the dimensionality of the input vector (which is four: 3 sets of data from the 3 axes and one set from the heart rate sensor). The output neurons follows the “one to n” coding adopted for showing the recognized posture (there are 10 posture defined).

### 3.2.2 Body posture recognition

The next step toward activity recognition was the recognition of 5 body postures: sitting, prone, supine, left lateral recumbent and right lateral recumbent. Figure 74 shows acceleration data for the 5 body positions defined above obtained using as acceleration data source a Chronos device fixed on chest. We have modelled an ANN with 10 neurons on hidden layer and 5 neurons on the output layer. Simulation results of this network presented in Figure 75. The recognition rate was 99.96%, MSE = 3.6747e-004.

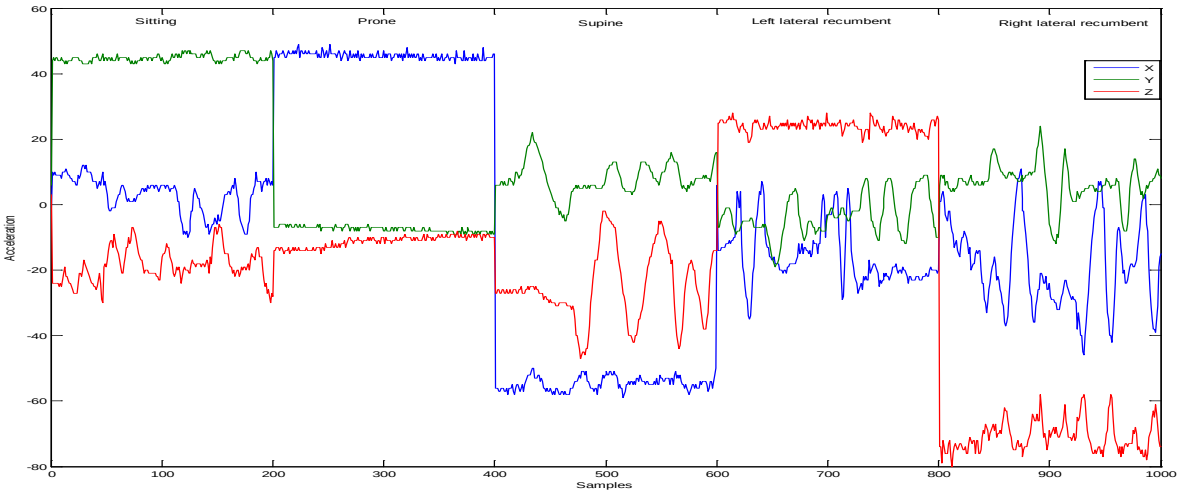



Figure 74. Acceleration data for 5 body positions [48].

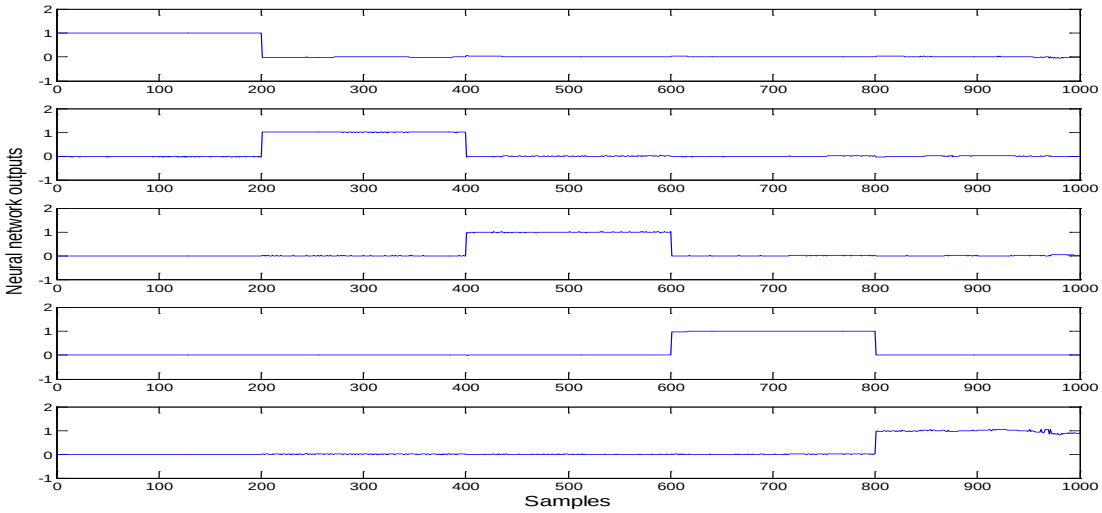



Figure 75. ANN output for the 5 body postures [48].

### 3.2.3 Activity recognition

A new data set was acquired and includes six postures and four activities (Figure 76). The data set was split up in 3 subsets for training, validation and testing. The numbers of neuron on the input layer vector is four (3 sets of data from the 3 axes of the accelerometer and one set from the heart rate sensor). The output layer has 10 neurons because there are 10 postures/activities to be recognized as follows:

1. Standing
2. Supine
3. Left lateral recumbent
4. Right lateral recumbent
5. Prone
6. Walking (forward)
7. Walking (backward)
8. Running (forward)
9. Running(backward)
10. Seating

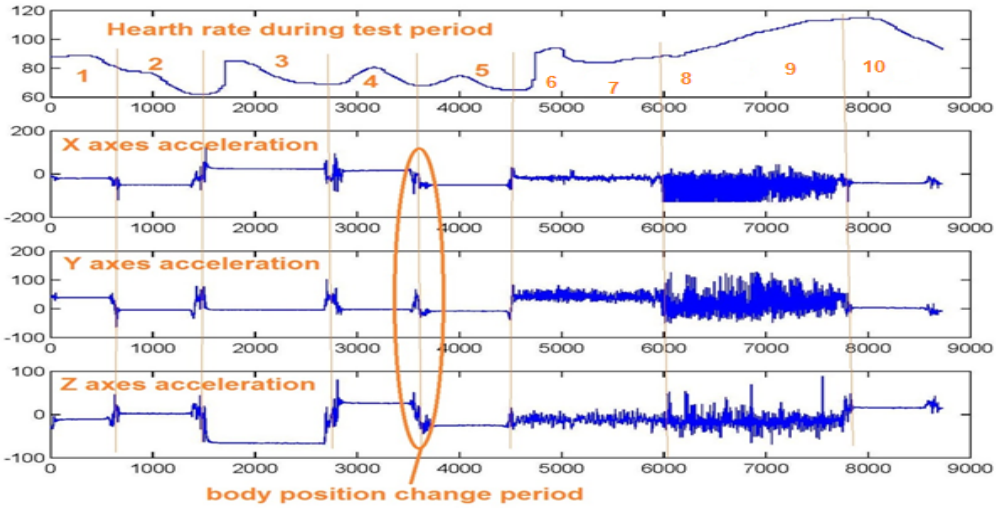



Figure 76. Data acquired for the 10 activities [48].

After several tests we found that for the hidden layer 10 neurons seems to be enough for a good recognition rate. The testing sets were chosen from the training set and was setup from 10 times 500 consecutive vectors representing each activity to be recognized. So the first neuron will recognize first 500 sample and so on. The recognition / errors are displayed in Figure 77. Total number of errors was only 456, meaning a 99,088% recognition rate.

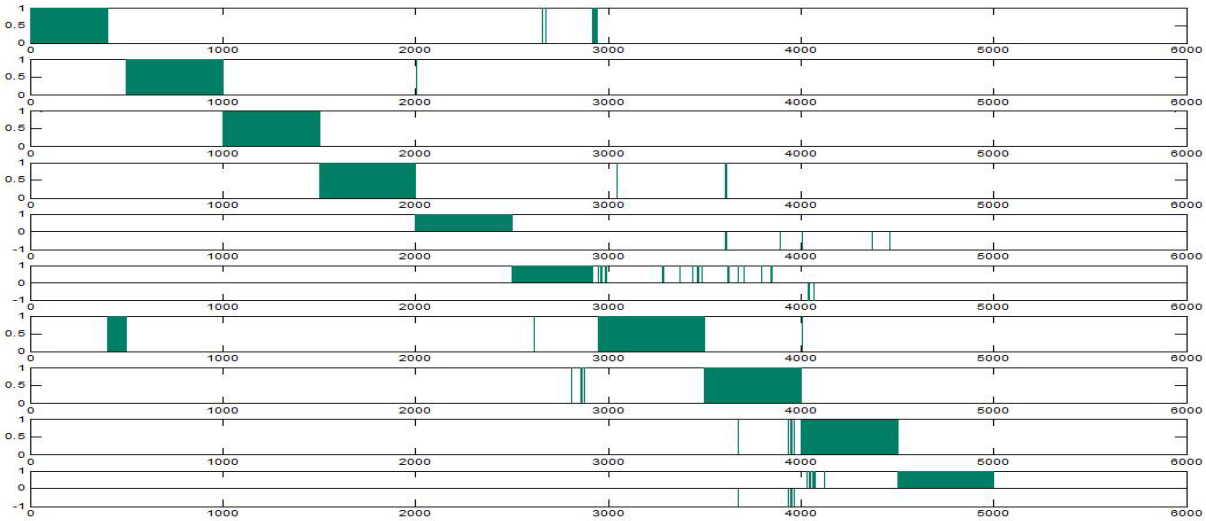


Figure 77. Output layer neurons' response to the testing set [48].

For differentiating the dynamic activity from static, requires some processing on the raw data set. The literature reports the using of features that include Mean value, Variance, Correlation

coefficients, Energy, Frequency-Domain Entropy, Cepstral Coefficients, Log FFT Frequency Bands, etc. [84], [85], [92], [97], [104], [105], [106]. Other authors report the use of the autoregressive (AR) modelling coefficients, the Signal Magnitude Area (SMA) and Tilt Angle (TA) as key features to distinguish dynamic activity from static [107], [108], [109], [110].

We have chosen to use standard deviation values for differentiating the dynamic activity from static. The results are shown in Figure 78 for each of the accelerometer's axes and for the pulse rate. The standard deviation was calculated over a window of 50 samples (it is considered that the data acquisition was made at 20 Hz and within 1 sec there is at least 1 step made). Analyzing the figure can be seen that by setting the right threshold (red line) for the standard deviation, the static – dynamic postures discrimination can be easily differentiated.

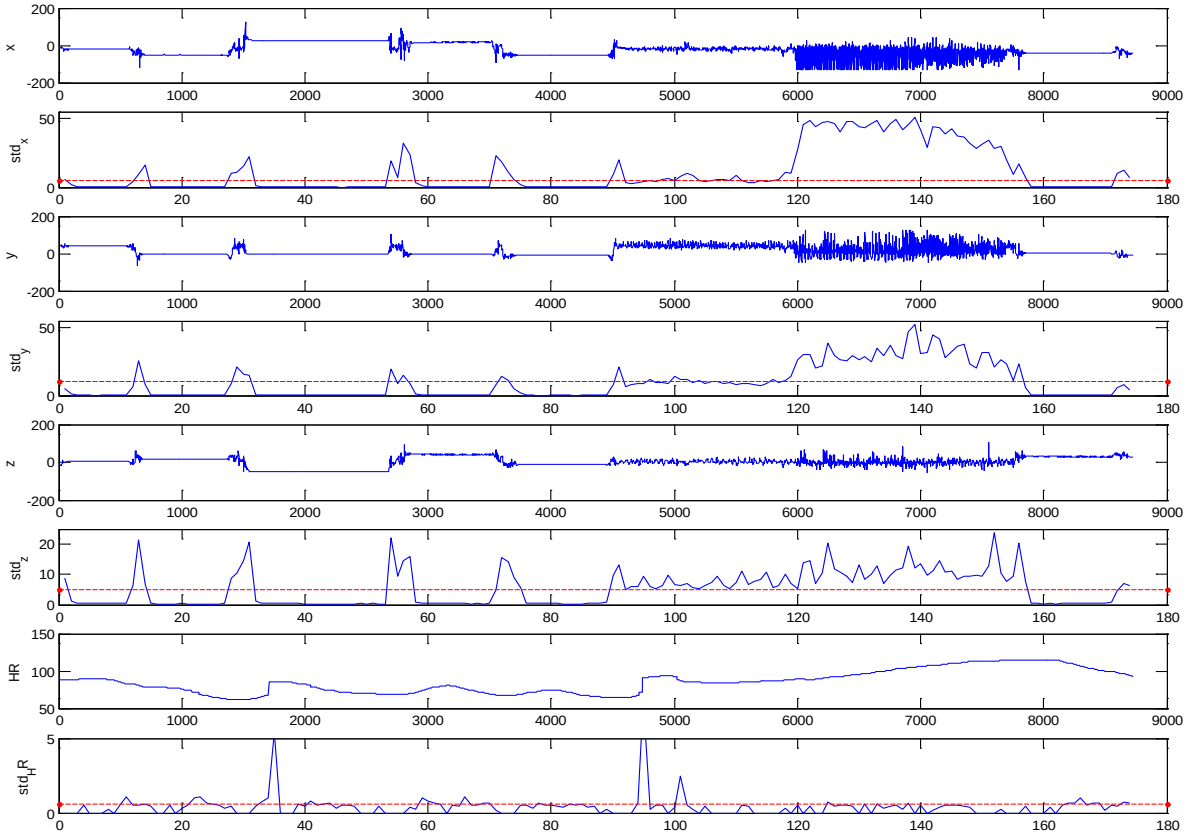



Figure 78. Differentiating dynamic activity from static using standard deviation [48].

For recognizing the walking and running activities, further relevant features have to be extracted from raw data set. Also using some processing algorithms on can determine further properties related to the activities, as for example using FFT transform we can determine the stepping rate. Figure 79 shows the initial acceleration signal (a) FFT transform of the signal for walking period (b) and the FFT transform of the signal for running. The FFT transform was used to determine the stepping rate of a person as the most dominating frequency in the acceleration signal's spectrum and was approximately 1.4 respectively 2.2 steps/second.

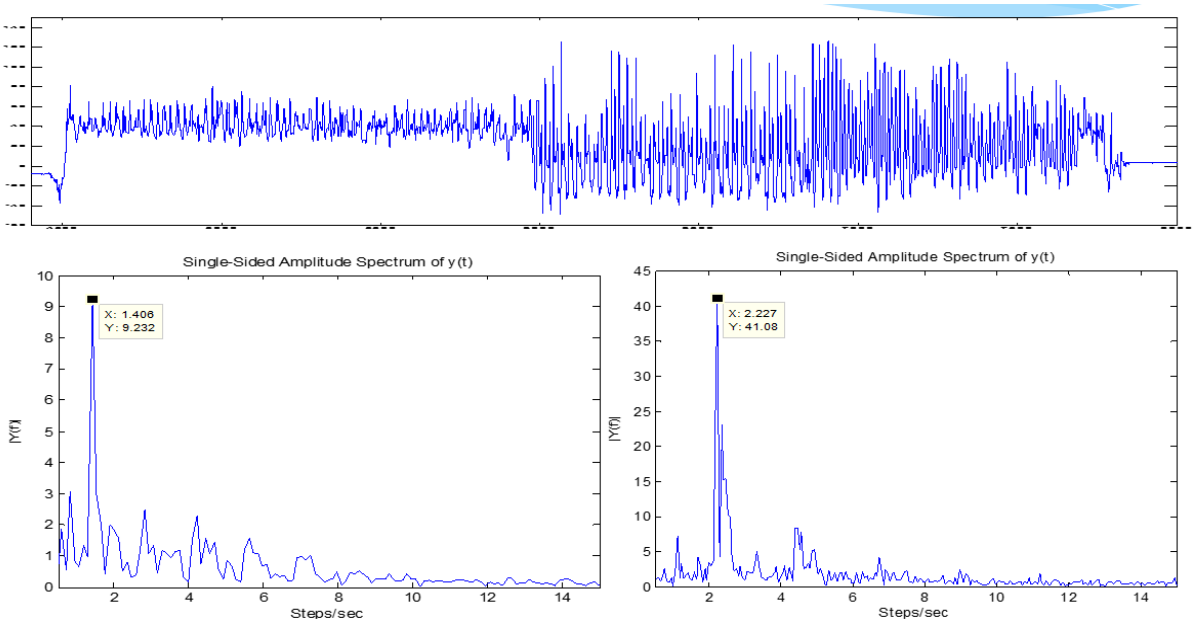



Figure 79. Using FFT for stepping rate evaluation [48].

### 3.3 Human activity and health status recognition

The aim of this work presented in [111] was to make a complex analysis of the different factors that may influence the recognition rate of the human activity using wearable sensors data. The first research was made with the aim to analyze the influence of number of sensors and their placement on the recognition rate. The results were presented in [112]. In our experiments for recognition of predetermined positions of the human body, we use a body area network consisting of three accelerometer sensors placed on the human body. We have established 17 common activities to be recognized (Table 18).

Table 18. Activities to be recognized [111].

| | |
|---|---|
| 1. Standing | 10. Left bending |
| 2. Sitting | 11. Right bending |
| 3. Supine | 12. Squats |
| 4. Prone | 13. Standing up/Sitting down |
| 5. Left lateral recumbent | 14. Falls |
| 6. Right lateral recumbent | 15. Turns left and right |
| 7. Walking | 16. Climbing stairs |
| 8. Running | 17. Descending stairs |
| 9. Bending forward | Transitions |

Using a 27 samples/second rate we acquired 600 samples for each activity, from three acceleration sensor placed on the chest.

The research was conducted in several direction. One direction of research was the design and test of several Matlab ANN models for activity recognition in order to find the best performing architecture, as reported in [48].

Using a two layer architecture we obtained a recognition rate above 95%.

Another research direction was related to the necessary preprocessing of raw data aiming to have a better recognition rate. As it is presented in the literature, the data can be preprocessed to obtain new features as Mean value, Variance, Energy, Correlation coefficients, Frequency-Domain Entropy, Log FFT Frequency Bands, etc. [84], [85], [104], [97], [92], [105]. After several simulations we find that the standard deviation could be used with very good results as a supplementary input data for the neurons. In the training phase of the ANN we tried to calculate the standard deviation over all the samples belonging to an activity (row 2 in Table 19) or over a window with different width (rows 3-6 in Table 19.). X-Acc, Y-Acc. and Z-Acc. represent the row acceleration data while X+Y+Z-Acc. is the sum. Std_w600(X+Y+Z-Acc.) is the standard deviation over all samples belonging to one activity while Std_w50(X+Y+Z-Acc.) is the standard deviation over a window of 50 samples. The difference between rows 3-6 consist in the threshold level (0.5, 0.6, 0.7 and 0.8) for the step activation function used in the output layer. The results are shown in Figure 80.

Table 19. Recognition rates as function of inputs [48].

| | **ANN input data** | |
|---|---|---|
| 1. | X-Acc, Y-Acc, Z-Acc, X+Y+Z-Acc | 95.44% |
| 2. | X-Acc, Y-Acc, Z-Acc, X+Y+Z-Acc, Std_w600(X+Y+Z-Acc) | 96.28% |
| 3. | X-Acc, Y-Acc, Z-Acc, X+Y+Z-Acc, Std_w50(X+Y+Z-Acc)$^1$ | 98.06% |
| 4. | X-Acc, Y-Acc, Z-Acc, X+Y+Z-Acc, Std_w50(X+Y+Z-Acc)$^2$ | 98.07% |
| 5. | X-Acc, Y-Acc, Z-Acc, X+Y+Z-Acc, Std_w50(X+Y+Z-Acc)$^3$ | 97.81% |
| 6. | X-Acc, Y-Acc, Z-Acc, X+Y+Z-Acc, Std_w50(X+Y+Z-Acc)$^4$ | 96.28% |

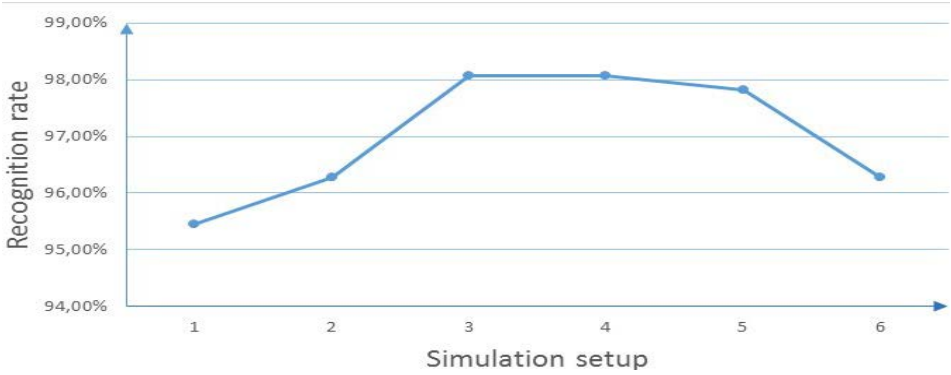



Figure 80. Recognition rate using data from one acceleration sensor and 6 different preprocessed input signals [111].

Another research was conducted in order to establish the number of sensors and their optimal placement. We acquired 600 samples for each activity, from three acceleration sensors placed on different parts of the body. One is placed on the right hand (Acc1), a second one above the right knee (Acc2) and the third one on the chest (Acc3). After a few first experiments it was obvious that the third accelerometer is difficult to wear and does not significantly improve the results. This

is why it wasn't used in further experiments. The results concerning recognition rates in different arrangements of sensors are summarized on Table 20. 2Acc is the setup with both sensors Acc1 and Acc2. For 2Acc configuration we present results for ANNs with one hidden layer with 20, 25 and 30 neurons and for an ANN having two hidden layers with 15 and 25 neurons respectively.

Table 20. Recognition rates as function of sensors arrangements [48].

| | **Acc1** | **Acc2** | **2ACC** | **2ACC** | **2ACC** | **2ACC** |
|---|---|---|---|---|---|---|
| | **20 neur.** | **20 neur.** | **20 neur.** | **25 neur.** | **30 neur.** | **15+25 neur.** |
| 1 | 99,97% | 100,00% | 100,00% | 99,98% | 99,91% | 99,98% |
| 2 | 100,00% | 99,96% | 99,93% | 100,00% | 99,99% | 99,53% |
| 3 | 99,94% | 100,00% | 100,00% | 99,52% | 99,95% | 100,00% |
| 4 | 98,99% | 99,63% | 99,47% | 99,98% | 99,76% | 99,98% |
| 5 | 99,51% | 99,69% | 99,46% | 99,32% | 99,61% | 99,88% |
| 6 | 99,51% | 100,00% | 99,46% | 99,68% | 99,64% | 99,51% |
| 7 | 95,73% | 99,14% | 99,53% | 98,02% | 99,54% | 99,41% |
| 8 | 97,73% | 99,02% | 99,51% | 99,51% | 99,51% | 99,75% |
| 9 | 97,57% | 95,29% | 94,34% | 97,25% | 99,16% | 98,39% |
| 10 | 96,83% | 95,19% | 96,61% | 98,63% | 97,62% | 98,58% |
| 11 | 94,28% | 95,35% | 98,25% | 98,35% | 97,79% | 99,09% |
| 12 | 99,01% | 97,21% | 98,61% | 98,78% | 98,32% | 99,75% |
| 13 | 97,51% | 97,48% | 97,97% | 99,00% | 98,91% | 99,93% |
| 14 | 96,41% | 96,71% | 97,30% | 96,79% | 97,57% | 97,86% |
| 15 | 97,23% | 97,61% | 98,87% | 98,59% | 98,69% | 98,81% |
| 16 | 95,77% | 98,05% | 98,61% | 99,09% | 98,88% | 99,01% |
| 17 | 98,47% | 99,09% | 98,86% | 99,09% | 99,24% | 99,16% |
| All | 97,91% | 98,20% | 98,63% | 98,92% | 99,06% | 99,33% |

Figure 81 shows the recognition rates of the static activities (Standing, Sitting, Supine, Prone, Left lateral recumbent, Right lateral recumbent) as a function of different sensors arrangements and the number of neurons on the hidden level of the neural network.

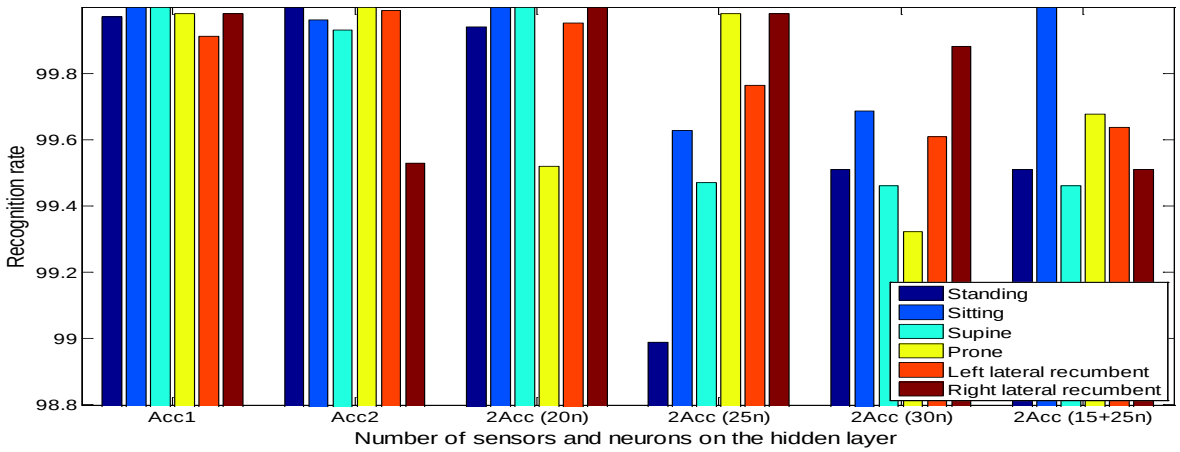



Figure 81. Recognition rates for static activities [111].

In Figure 82 we can see the recognition rates for selected dynamic activities (Walking, Running, Standing up/Sitting down, Falling, Climbing stairs, Descending stairs) as a function of different sensors arrangements.

Observing the results presented in Figure 81 and Figure 82 it can be concluded that overall recognition rate for the static activities is better than for dynamic activities.

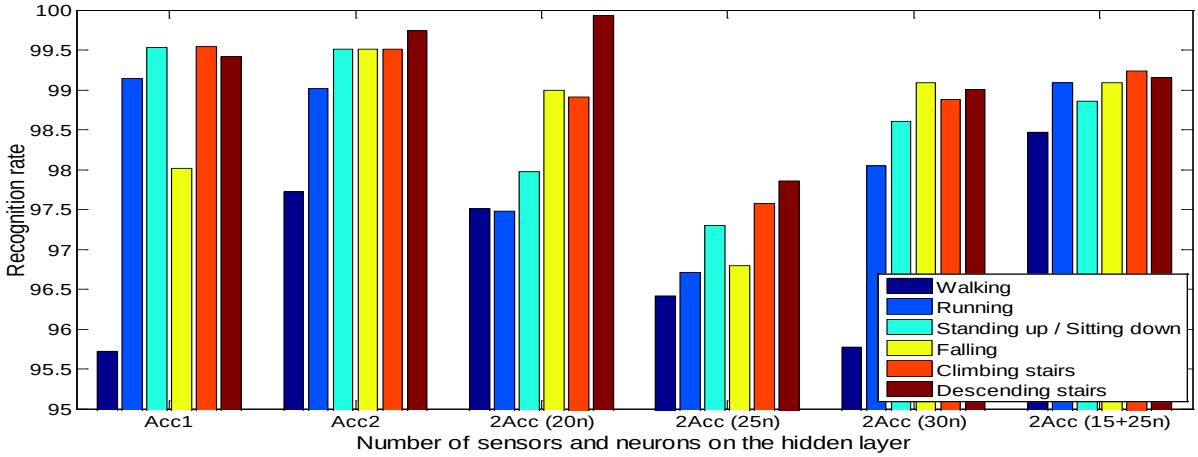



Figure 82. Recognition rates for selected dynamic activities [111].

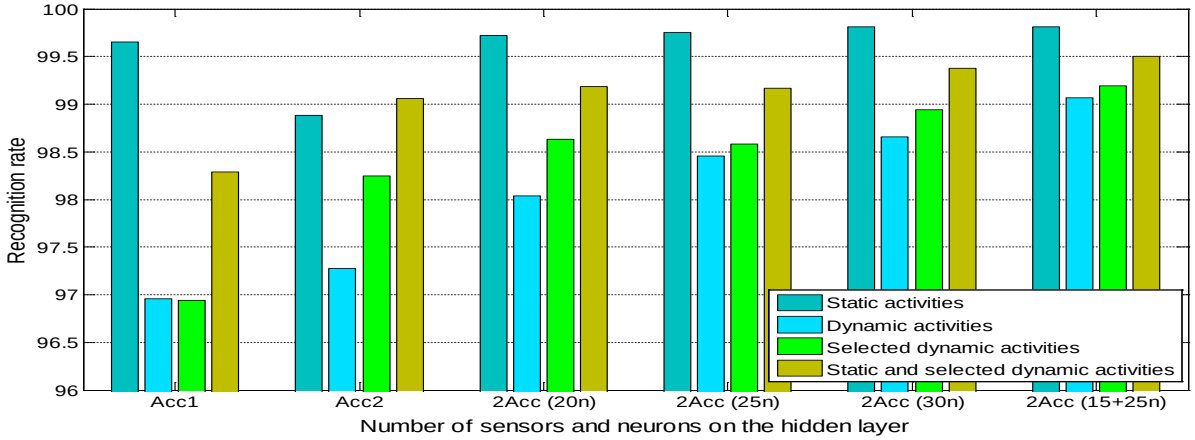



Figure 83. Comparison between recognition rates for static and selected dynamic activities [111].

Analyzing the results from the point of view of the sensors setup and the number of neurons of the neural network, it can be seen that for static activities the recognition rates are between 0.5% limits for all possible combinations. For all dynamic activities the best results could be obtained using the two accelerometers setup and an ANN with 2 hidden layers. For the selected dynamic activities we obtained good results even for the one accelerometer setup (Acc2) that implies that we can use a simpler artificial neural network with one hidden layer with only 20 neurons. This setup represents the best trade-off between recognition rate and the complexity of the recognition system.

The human activity and health parameters monitoring system was developed and optimized regarding good recognition rate using minimal resources. The use of ANN was found to be very effective even for architectures with one hidden layer with 20 neurons. It was demonstrated that even using a single 3-axis acceleration tag combined with proper signal preprocessing e.g., mean, standard deviation, etc. very high recognition rates can be obtained. Comparing our results with those presented in [23], [62], [66], [106], [137] we can conclude that our method give better results. As expected the recognition rate for the static activities was better than for dynamic activities. We performed also frequency domain analysis. FFT transform was used to determine the stepping rate in walking and running activities. We also implemented and tested a real time recognition system using Raspberry Pi mini-computer [59].

## 3.4 Studies regarding optimal recognition methods of human activity

In the research contract report [113] published partially also in [114], we have examined the influence of number of sensor nodes, sensor node type, pre-processing algorithms, classifier type and its parameters. We made a total of 1674 simulations on a publically released human activity database by a group of researcher from University of California at Berkeley.

The final purpose is to find the optimal setup for best recognition rates with lowest hardware and software costs. The hardware costs can be quantified by the number of sensor nodes required and the number of neurons of the neural network. Lower soft-ware costs means less necessary preprocessing needs.

### 3.4.1 WARD database

For our research we have used the WARD (wearable action recognition database) database provided by NSF TRUST (Team for Research in Ubiquitous Secure Technology) Center at University of California, Berkeley, University of Texas at Dallas, Tampere University of Technology, and Telecom Italia Laboratory [115], [116].

***Database description***

The database contains data supplied by wearable motion sensors network while performing a set of predetermined activities. The wireless sensor network is composed by sensor nodes equipped with one triaxial accelerometer and a biaxial gyroscope. The sensors are placed on both ankles, on waist and on both forearms. The database contains 13 activities recorded from 20 persons performed five times. These activities as described in [115] and [116] are:

1. Standing: standing still more than 10 seconds.
2. Sitting: siting still more than 10 seconds.
3. Lying: lying still more than 10 seconds.
4. Walk forward: walking straight forward more than 10 seconds.
5. Walk forward left-circle: walking in counter-clock circle more than 10 seconds.
6. Walk forward right-circle: walking in clock circle more than 10 seconds.
7. Turn left: staying at the same position and turning left more than 10 seconds.
8. Turn right: staying at the same position and turning right more than 10 seconds.
9. Go upstairs: going up one flight.
10. Go downstairs: going down one flight.
11. Jog: jogging straight forward more than 10 seconds.
12. Jump: staying at the same position and jump more than 5 times.
13. Push wheelchair: pushing a wheelchair more than 10 seconds.

The sampling rate was 20 samples /second. The author's intention was to create a database placed in the public domain with the aim to compare the results obtained in activity recognition using different recognition methods.

Equation (58) shows a record structure from one sensor node (k) at time t having 5 components three from accelerometer (x, y, z) and two from gyroscope (θ, ρ)

$$s_k(t) := (x_k(t), y_k(t), z_k(t), \theta_k(t), \rho_k(t)) \in \Re^5 \tag{58}$$

Next equation represents vector of all sensors data at time t.

$$s(t) := (s_1(t), s_2(t), ....s_L(t))^T \in \Re^{5xL} \tag{59}$$

In equation (60), $s_a$ collects all data corresponding to an activity of l samples.

$$s_a = (s(1), s(2), ....s(l))^T \in \Re^{5xLxl} \tag{60}$$

with a=1:13, and l being different for each activity.

The authors of the WARD database have published their results obtained using nearest-neighbor (NN) and distributed sparsity classifier (DSC) algorithms [115].

### 3.4.2 Our proposed method

We conceived a detailed analysis method of the factors that can influence human activity recognition, consisting of five steps that will be described below. Each step is implemented using a Matlab script. Each script will be called by a main Matlab code that generates the parameters for each evaluated case (from the total of 1674 considered cases).

***Extracting data from the database and reordering***

For an easier access to these data we created a Matlab script which extracts data corresponding to all activities performed by a subject in a recording session (in total of 13), ordering data corresponding to the five sensory nodes, of five data series each (acceleration 3 axes, gyroscope 2 axes) in 13 variable of $l_a$x25 columns, where $l_a$ is the number of samples corresponding to activity a, a=1:13. In this way we obtained 13 vectors of form (60). Finally we created a 5996x25 matrix representing acquired data from a subject during a recording session.

The database contains a totally of 20 x 5 matrix (number of subjects x number of recording sessions for each subject) available for simulations. During simulation, from this created matrices, can be automatically extracted only the appropriate columns and sensor nodes that need to be considered. In this way can be simulated all possible scenarios on the number of sensors and all their possible combinations of 1, 2 and so on up to 5 sensors. Total number of possible combinations is:

$$N=\sum_{x=1}^{5} C_5^x=31 \tag{61}$$

***Feature extraction***

In previous experiments presented in [48] we decided to for use the sum of accelerations on the 3 axes and standard deviation of this sum as supplementary input data for the recognition system, because in this way we obtained better recognition rates. Standard deviation was calculated on non-overlapping windows of one second. These vectors could be added or not to the data matrix as new columns according to the parameters set (corresponding to the desired experimental configuration). Since the experimental environment created allow us to run all the scenarios, we simulated all configurations with and without sum of acceleration on the 3 axes and with or without its standard deviation.

***Preparing data for training and simulation***

Next preparatory stage is to create data for training the neural network respectively data for testing the network. They consists in two sets of data: input data respectively target data target. The training data consisting of samples corresponding to the acceleration on the 3 axis from 1:5 sensors to which one can add data from the gyroscope, the sum of accelerations, and the standard deviation of the sum. In this way the simplest input data matrix has 5996 samples x 3 data, while the most complex contains 5996 samples x 35 data. Target data matrix has a fixed size 5996x13, each row contains a single one on the column representing the output corresponding to the activity to be

recognized. The rows of the two matrices containing training data are mixed synchronized. Data used for testing are in the same order as they were acquired to allow easier visual observation of simulation results.

***Creation of the network, training, simulation***

Artificial neural network, used by us for activity recognition was a feed forward backpropagation network with one hidden layer. The network is created automatically with settable parameters. Number of inputs depending on number of input data number is between 3 and 35, the number of outputs is 13 because this is the number of activities to be recognized. The number of neurons on the hidden layer is a parameter than can have three values 10, 20 or 30 for the networks with fewer inputs (simulations with 1:3 sensorial nodes), up to 50, 55, 60 in the case of networks for 5 sensorial nodes. We have chosen sigmoid function as activation function for hidden layer and linear function for the output layer. Training is done using LM algorithm.

After training the network is simulated in scenarios using three different output functions: round, compet and a threshold function with threshold 0.5.

***Post processing: error calculation, development of synthetic tables***

After simulation using Matlab scripts that we have created are automatically calculated the errors, confusion matrix and can be generated synthetic tables according to different parameters. Finally data are saved in files.

### 3.4.3 Simulation results

The purpose of the simulations was to observe the influence of various factors on the rate of recognition and identification of the solutions that require fewer hardware or software resources. For each possible combination of sensor nodes (a total of 31 combinations) we have imagined six configurations. In this way we created a total of 186 configuration and three different ANN were trained for each configuration. Each network was simulated with three output functions resulting a total of 1674 simulations results.

Table 21 presents the overall recognition rate obtained for Output function =1 (threshold). First four rows shows the parameters of the configuration. First column shows combination of the sensor nodes. Similar tables were created for the other two functions. In this table columns headers represents:

- g=1 simulation that use only acceleration data, g=2 acceleration + gyroscope date
- as=0 simulations without sum(Acc), as=1 simulations with sum(Acc) at the ANN input
- sd=0 simulations without Std(sum(Acc)), as=1 simulations with Std(sum(Acc)) at the ANN input
- nn= numer of neurons on hidden layer

Table 21. Overall recognition rate obtained for Output function =1 (threshold).

| g | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| as | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| sd | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| nn | 10 | 20 | 30 | 10 | 20 | 30 | 10 | 20 | 30 | 10 | 20 | 30 | 10 | 20 | 30 | 10 | 20 | 30 |
| 1 | 39,41 | 48,88 | 51,02 | 35,54 | 47,95 | 51,4 | 58,19 | 66,24 | 69,73 | 45,63 | 64,29 | 65,96 | 39,23 | 61,72 | 69,7 | 66,08 | 80,82 | 83,64 |
| 2 | 40,78 | 49,35 | 51,63 | 44,06 | 48,73 | 51,75 | 51,1 | 56,55 | 58,64 | 50,1 | 64,26 | 68,96 | 48,78 | 64,33 | 70,33 | 56,4 | 73,23 | 74,38 |
| 3 | 32,42 | 44,76 | 48,33 | 40,08 | 45,93 | 48,4 | 57,42 | 64,19 | 66,48 | 56,59 | 68,81 | 72,23 | 55,14 | 68,71 | 74,05 | 73,45 | 84,41 | 87,88 |
| 4 | 31,47 | 32,22 | 37,19 | 29,19 | 34,17 | 39,03 | 38,11 | 43,66 | 47,75 | 32,96 | 44,8 | 49,23 | 36,32 | 42,14 | 48,83 | 44,51 | 55,57 | 60,41 |
| 5 | 31,24 | 34,62 | 35,81 | 32,17 | 34,86 | 37,04 | 34,59 | 42,61 | 44,95 | 36,67 | 43,71 | 52,5 | 39,48 | 45,43 | 53,8 | 44,7 | 60,24 | 60,64 |
| nn | 20 | 25 | 30 | 20 | 25 | 30 | 20 | 25 | 30 | 20 | 25 | 30 | 20 | 25 | 30 | 20 | 25 | 30 |
| 12 | 61,26 | 60,74 | 60,96 | 56,9 | 59,69 | 62,34 | 81,15 | 81,69 | 82,77 | 72,4 | 75,07 | 78,87 | 72,45 | 77,38 | 76,55 | 87,63 | 90,33 | 92,21 |
| 13 | 61,82 | 60,51 | 61,66 | 60,27 | 62,34 | 64,78 | 78,9 | 85,17 | 86,59 | 79,2 | 82,39 | 86,51 | 81,95 | 82,25 | 86,41 | 94,85 | 95,93 | 95,51 |
| 14 | 50,5 | 50,87 | 58,39 | 45,91 | 50,95 | 60,47 | 76,02 | 82,3 | 83,01 | 65,49 | 74,55 | 78,62 | 67,44 | 74,87 | 78,82 | 89,03 | 90,66 | 89,71 |
| 15 | 48,9 | 54,64 | 59,26 | 51,48 | 54,02 | 56,95 | 77,37 | 80,19 | 81,49 | 73,75 | 78,97 | 81,02 | 73,05 | 78,24 | 81,2 | 91,43 | 91,83 | 92,73 |
| 23 | 55,04 | 58,47 | 61,51 | 56,95 | 56,72 | 60,29 | 70,36 | 82,04 | 77,59 | 82,92 | 85,47 | 86,12 | 80,7 | 85,79 | 83,54 | 93,25 | 93,9 | 94,43 |
| 24 | 55,12 | 54,57 | 57,25 | 51,73 | 58,62 | 57,82 | 66,08 | 68,81 | 73,1 | 67,13 | 71,1 | 75,12 | 70,7 | 71,45 | 75,02 | 80,12 | 80,25 | 82,35 |
| 25 | 52,87 | 53,57 | 57,82 | 52,42 | 55,27 | 57,74 | 67,28 | 72,67 | 72,78 | 75,72 | 80,2 | 81,55 | 76,88 | 81,27 | 83,32 | 87,91 | 88,94 | 90,23 |
| 34 | 49,8 | 52,65 | 53,62 | 49,22 | 54,45 | 54,34 | 74,6 | 74,38 | 75,75 | 75,03 | 78,14 | 77,7 | 74,13 | 77,3 | 78,27 | 89,19 | 90,93 | 90,96 |
| 35 | 48,12 | 51,48 | 54,25 | 49,3 | 51,7 | 54,4 | 76,42 | 77,43 | 78,85 | 80,62 | 83,61 | 82,97 | 82,19 | 84,06 | 84,32 | 94 | 95,36 | 94,95 |
| 45 | 39,14 | 41,63 | 44,06 | 40,19 | 43,46 | 46,43 | 58,69 | 64,39 | 66,76 | 59,54 | 66,73 | 68,38 | 57,79 | 67,36 | 69,06 | 81,47 | 82,86 | 82,22 |
| nn | 20 | 25 | 30 | 20 | 25 | 30 | 20 | 25 | 30 | 20 | 25 | 30 | 20 | 25 | 30 | 20 | 25 | 30 |
| 123 | 62,51 | 69,15 | 69,48 | 64,14 | 67,36 | 71,43 | 86,99 | 89,94 | 90,71 | 87,56 | 88,58 | 89,98 | 86,82 | 88,18 | 88,76 | 96 | 96,03 | 97,93 |
| 124 | 64,04 | 64,61 | 70,8 | 63,19 | 63,76 | 68,25 | 85,99 | 87,19 | 89,23 | 79,14 | 83,69 | 87,32 | 74,37 | 83,81 | 87,86 | 92,76 | 95,08 | 95,38 |
| 125 | 61,46 | 64,54 | 71,28 | 62,07 | 65,28 | 68,7 | 87,01 | 86,96 | 87,21 | 81,94 | 86,12 | 87,14 | 82,86 | 85,02 | 87,89 | 95,1 | 95,4 | 96,25 |
| 134 | 63,19 | 62,79 | 69,98 | 59,44 | 63,78 | 66,31 | 87,61 | 90,51 | 89,59 | 84,47 | 87,36 | 88,99 | 83,16 | 86,29 | 88,24 | 95,33 | 96,85 | 97,33 |
| 135 | 57,59 | 62,21 | 69,26 | 64,64 | 65,14 | 68,36 | 88,86 | 89,16 | 91,28 | 85,67 | 89,18 | 90,33 | 87,09 | 89,19 | 90,04 | 97,3 | 97,77 | 97,45 |
| 145 | 56,05 | 60,76 | 65,41 | 51,65 | 63,88 | 66,39 | 84,11 | 89,51 | 89,71 | 79,67 | 85,59 | 88,46 | 80,6 | 86,14 | 87,06 | 94 | 95,35 | 95,55 |
| 234 | 58,77 | 58,39 | 63,43 | 58,02 | 58,61 | 64,86 | 82,72 | 82,77 | 84,32 | 83,54 | 87,64 | 88,54 | 82,84 | 85,76 | 86,39 | 93,55 | 95,61 | 95,66 |
| 235 | 57,86 | 60,96 | 66,23 | 54,77 | 61,02 | 65,03 | 81,82 | 86,81 | 89,36 | 88,61 | 90,28 | 91,41 | 86,69 | 89,68 | 92,53 | 96,6 | 97,65 | 98,17 |
| 245 | 56,92 | 61,74 | 66,94 | 60,06 | 65,93 | 64,64 | 78,69 | 79,52 | 81,69 | 77,64 | 85,99 | 85,91 | 82,07 | 86,01 | 89,19 | 90,78 | 91,33 | 93,58 |
| 345 | 53,45 | 59,61 | 58,39 | 54,84 | 58,77 | 63,43 | 83,51 | 86,94 | 88,28 | 85,87 | 86,86 | 88,26 | 84,77 | 86,01 | 89,63 | 95,71 | 96,36 | 97,08 |
| nn | 30 | 40 | 50 | 30 | 40 | 50 | 30 | 40 | 50 | 30 | 40 | 50 | 30 | 40 | 50 | 30 | 40 | 50 |
| 1234 | 76,47 | 79,75 | 81,54 | 76 | 80,72 | 79,55 | 93,88 | 95,85 | 96,33 | 92,73 | 93,63 | 94,6 | 92,78 | 94,26 | 94,78 | 98,28 | 98,42 | 98,82 |
| 1235 | 73,7 | 79,72 | 82,17 | 73,7 | 79,37 | 84,01 | 92,44 | 94,5 | 94,98 | 93,3 | 95,01 | 96,48 | 93,26 | 94,71 | 96,3 | 99,42 | 98,92 | 99,3 |
| 1245 | 74,48 | 79,12 | 82,97 | 76,15 | 79,94 | 83,24 | 93,45 | 93,51 | 95,56 | 92,04 | 93,65 | 95,33 | 92,28 | 93,78 | 95,18 | 97,58 | 98,18 | 98,73 |
| 1345 | 78,07 | 76,47 | 80,22 | 72,83 | 80,14 | 83,21 | 93,46 | 96,55 | 96,58 | 93,7 | 95,21 | 96,38 | 93,53 | 95,41 | 96,33 | 98,08 | 99,02 | 98,88 |
| 2345 | 66,84 | 75,32 | 80,05 | 71,93 | 75,93 | 80,34 | 92,19 | 96,41 | 95,11 | 93,41 | 94,95 | 96,83 | 92,9 | 94,55 | 95,7 | 98,28 | 98,7 | 99,07 |
| nn | 50 | 55 | 60 | 50 | 55 | 60 | 50 | 55 | 60 | 50 | 55 | 60 | 50 | 55 | 60 | 50 | 55 | 60 |
| 12345 | 86,97 | 86,36 | 87,51 | 86,47 | 85,14 | 86,29 | 97,88 | 97,67 | 97,41 | 97,77 | 97,6 | 98,07 | 97,6 | 97,8 | 98,38 | 99,45 | 99,48 | 99,8 |

As was expected best recognition rates could be obtained using all 5 sensor nodes and taking into account also the gyroscope data. Adding in this case the sumAcc data and std(sumAcc) doesn't rises significantly the recognition rates (from 98.38% to 99.8%) and in this case is not necessary

because of their hardware and software costs for the computation. Using preprocessed data together with raw data increases the recognition rates for 3 and especially for 2 sensor nodes configurations. In this case calculation of derived features is preferable, because the use of fewer sensory nodes provides an increased comfort to wear and the system is more reliable.

We present below some of the results obtained for configurations with two sensory nodes. It can be observed that for two sensor nodes configurations we can obtain recognition rates over 95% for combinations 13 and 35 if we use not only the acceleration but also the gyroscope date, and only if we use as supplementary inputs for the neuron in both training and recognition phases the preprocessed data: sum of the acceleration on three axis and its standard deviation.

#### 3.4.3.1 Simulations based on the number of sensor nodes

Recognition rate depends on the number of sensors used, generally increasing with the number of sensors used. For some sensor nodes combination adopting some preprocessing we can obtain recognition rates over 90% even using only 2 sensor nodes. Table 22 shows recognition rates as a function of number of sensors without preprocessing and in two cases adding two simple preprocessed features: Sum(Acc) and std(Sum) respectively.

Table 22. Overall recognition as a function of number of sensors and their configuration [114].

| | Sensors configuration | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Preprocessing | 1 | 2 | 3 | 4 | 5 | 12 | 13 | 14 | 15 | 23 | 24 | 25 | 34 | 35 | 45 | 123 | 124 | 125 | 134 | 135 | 145 | 234 | 235 | 245 | 345 | 1234 | 1235 | 1245 | 1345 | 2345 | 12345 |
| ------ | 65,96 | 68,96 | 72,23 | 49,23 | 52,50 | 78,87 | 86,51 | 78,62 | 81,02 | 86,12 | 75,12 | 81,55 | 77,70 | 82,97 | 68,38 | 89,98 | 87,32 | 87,14 | 88,99 | 90,33 | 88,46 | 88,54 | 91,41 | 85,91 | 88,26 | 94,60 | 96,48 | 95,33 | 96,38 | 96,83 | 98,07 |
| Sum | 69,70 | 70,33 | 74,05 | 48,83 | 53,80 | 76,55 | 86,41 | 78,82 | 81,20 | 83,54 | 75,02 | 83,32 | 78,27 | 84,32 | 69,06 | 88,76 | 87,86 | 87,89 | 88,24 | 90,04 | 87,06 | 86,39 | 92,53 | 89,19 | 89,63 | 94,78 | 96,30 | 95,18 | 96,33 | 95,70 | 98,38 |
| Sum+Std | 83,64 | 74,38 | 87,88 | 60,41 | 60,64 | 92,21 | 95,51 | 89,71 | 92,73 | 94,43 | 82,35 | 90,23 | 90,96 | 94,95 | 82,22 | 97,93 | 95,38 | 96,25 | 97,33 | 97,45 | 95,55 | 95,66 | 98,17 | 93,58 | 97,08 | 98,82 | 99,30 | 98,73 | 98,88 | 99,07 | 99,80 |

#### 3.4.3.2 Simulations based on sensor nodes composition

Figure 84 shows the dependence between recognition rates and type of sensors used in nodes. The blue line represents results using only acceleration sensor and red line when we use both sensors. It is obvious that using both sensors we have a better recognition rate. The results depends also on the particularity of sensors placement. In this case the sensors on the lower part of the body were placed on the ankles and in this way they doesn't supply any information about body upper-body movement.

In case of a more suitable location, as presented in our previous work, on the thigh just above the knees, the sensor nodes can provide some information about upper–body movement too and in this way we obtained better results even without using gyroscope.

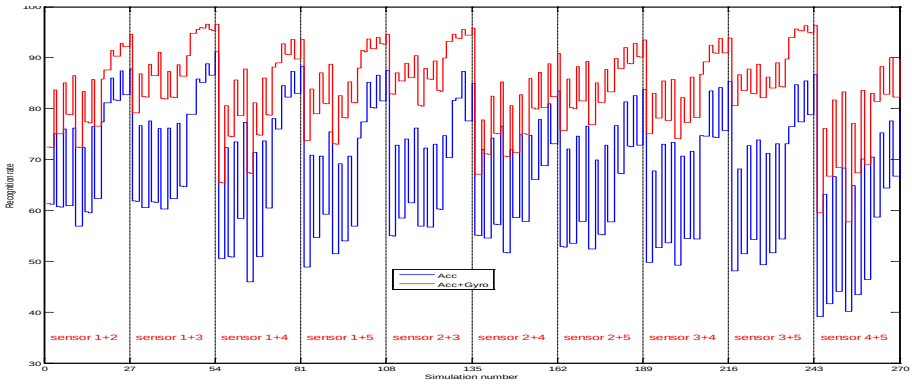



Figure 84. Comparison of recognition rates using Acceleration vs. Acceleration and Gyroscope [114].

### 3.4.3.3 The effect of adding as a supplementary input the sum of accelerations on 3 axes

Using sum of accelerations on the 3 axes, as a supplementary input for neural network doesn't increases the recognition rates as it can be observed in Table 21 comparing columns for as=0 with those for as=1 (for same g, sd, and nn). The results are presented

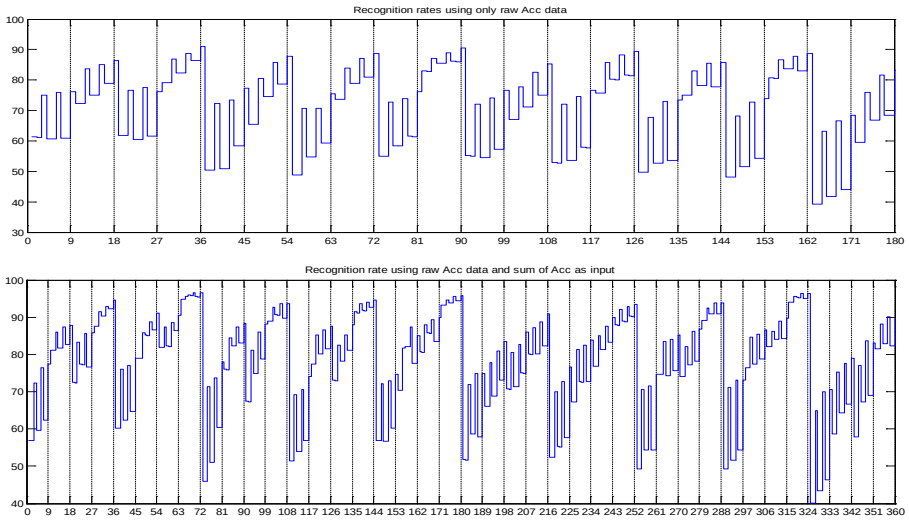



Figure 85. The effect of adding of sum(Acc) to the ANN input [114]

### 3.4.3.4 The effect of adding as input signal of the sum of accelerations on the 3 axes and its standard deviation

Adding standard deviation of sum(Acc) along with the raw data to the input of the ANN classifier significantly increases the recognition rate for the two sensor nodes configurations. In this cases the growth can be 15-20%. The growth is smaller for the configurations with a larger number of sensors.

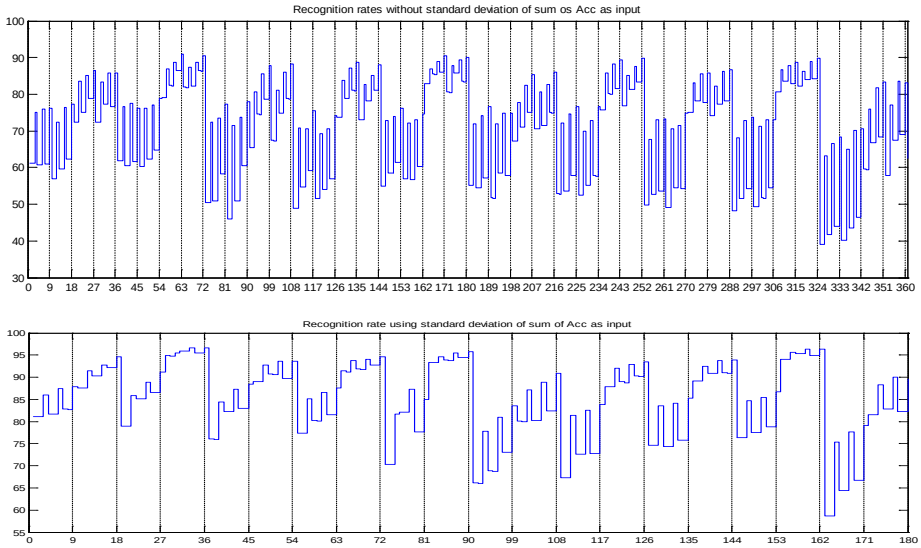



Figure 86. The effect of adding of Std(sum(Acc)) to the ANN input [113].

### 3.4.3.5 Simulations based on the number neurons of hidden layer (3 cases)

Increasing the number of hidden layer neurons generally result in increasing the rate of recognition, but the increase is not significant.

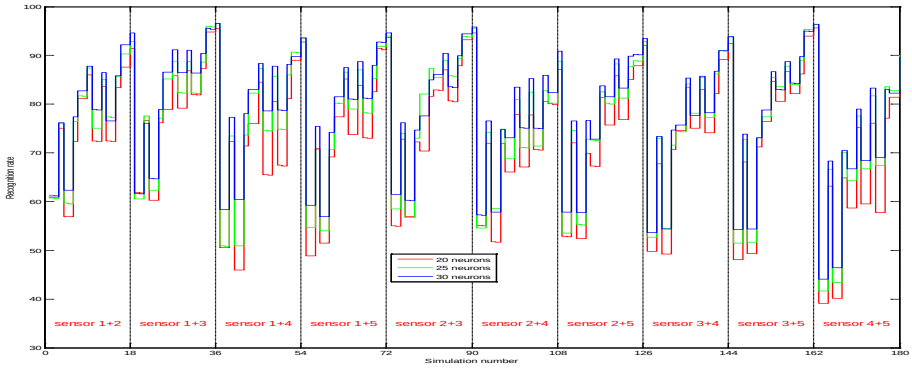



Figure 87. The effect of number of neurons on hidden layer on recognition rate [113]

### 3.4.3.6 Simulations based on activation function of output layer (3 types)

As it can be seen on figure 5 best results could be obtained for compet output function but the implementation of this function requires more hardware resources as the threshold output function. Round and threshold function give approximatively same results.

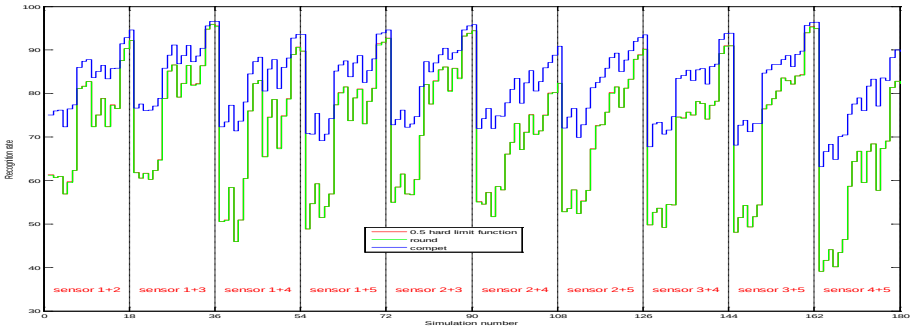



Figure 88. Recognition rate as a function of activation function of the output layer [113].

### 3.4.4 Conclusions

From a total of 1674 simulations performed if we consider only the cases when the recognition rates are over 90% still remain 467 cases. We try to identify those setups that provide better results like those presented in [115] using only 2 sensor nodes instead of 5, but adding sumAcc and std(sumAcc) as new preprocessed features to the input of the neural network used for human activity recognition.

Searching the trade-off between best recognition rate and less hardware resources needed and/or less preprocessing, we identified two setups with two sensor nodes, namely configurations with sensors 13 and 35. The methods for designing such a multi-sensor system required consideration of the following: sensors selection, placement of the sensors, selection of pre-processing algorithms, and classifier selection. Each factor impact has been investigated and we selected the most appropriate approach in order to achieve a trade-off between recognition accuracy and the hardware needed respectively preprocessing needed. The experimental results illustrate that the proposed multi-sensor system can achieve an overall recognition accuracy over than 97% without any preprocessing when we use 5 sensors like those described in [115]. We obtained a recognition rate over 95% even for some of the two sensors setup by adopting the sum of acceleration and standard deviation of the sum as preprocessed new features and using an ANN classifier. Table 23 presents comparison between results presented in [115] using nearest-neighbor (NN) and distributed sparsity classifier (DSC) algorithms for the configuration with 5 sensors (first row) and results obtained by us using an FF-BP ANN classifier with one layer of 50, 55 and 60 neurons respectively (rows 2-4). Columns A1-A13 present the results for the activities defined upper. Also we present in rows 5 and 6 results obtained for a 2 sensor configuration adding the selected new features to the input. Even in this case the recognition rate is over 95%.

Table 23. Comparison between results presented in [115] and results obtained by us [114].

| Recognition method/ sensors configuration/ number of neurons | Recognition rate/ Activity (%) | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 | A9 | A10 | A11 | A12 | A13 | All |
| DSC | 87,2 | 66,80 | 91,8 | 92,00 | 97,30 | 95,70 | 97,00 | 95,20 | 98,00 | 98,30 | 99,30 | 97,90 | 98,60 | 92,44 |
| ANN-12345-50 | 100 | 100 | 100 | 91,88 | 97,63 | 95,64 | 99,10 | 98,70 | 95,67 | 95,02 | 97,85 | 97,31 | 100 | 97,77 |
| ANN-12345-55 | 100 | 100 | 100 | 92,13 | 97,63 | 96,47 | 99,10 | 99,13 | 92,62 | 93,44 | 97,04 | 98,34 | 100 | 97,60 |
| ANN-12345-60 | 100 | 100 | 100 | 94,92 | 98,49 | 97,10 | 99,10 | 99,13 | 93,13 | 94,57 | 98,39 | 97,93 | 100 | 98,07 |
| ANN-13-25 | 100 | 100 | 100 | 82,23 | 92,90 | 92,32 | 99,55 | 97,18 | 88,80 | 95,93 | 96,24 | 98,55 | 98,25 | 95,93 |
| ANN-35-25 | 98,9 | 99,35 | 100 | 89,85 | 93,98 | 85,68 | 99,10 | 98,48 | 87,28 | 87,78 | 98,12 | 99,38 | 98,25 | 95,36 |

Even better results can be obtained using a better placement of the sensors. With a 2 sensor placed on the thigh and wrist and using only accelerometer data but adding preprocessed data we obtained very good results as we presented in [112].

## II. Professional and Academic Achievements

After receiving the Ph.D. in 2005 I became Associate professor at the Electrotechnical Department, Faculty of Engineering, North University of Baia Mare. Between 2007 and 2011 I was Head of Electronics and Computer Engineering Department. Since 2010 I am also Associate professor at University of Debrecen, Faculty of Informatics.

During my academic career I have taught several courses, seminars and laboratories for the undergraduate and master students and in addition I supervised several students for graduation thesis and master dissertations. I have taught several courses including: Digital design, Microprocessors systems, Embedded systems, Computer aided design, Electronics technology. At the University of Debrecen I teach Digital design with programmable logic, Digital technologies, Reconfigurable embedded systems, Simulation of digital circuits and HW-SW co-design. I also have classes at the doctoral school of the Faculty of Informatics entitled: Reconfigurable Embedded Systems based Cyber-Physical systems and I am supervisor of the Doctoral subject entitled: Artificial neural networks hardware implementation using reconfigurable devices.

For all these subjects I have contributed to the development of lecture notes, laboratory guides, I published two books and I built websites for the subjects. The teaching methods used are modern using PowerPoint presentations and Internet resources presentation combined with interactive discussions.

I contributed to the development and accreditation of the “Applied Electronics” major and I am the responsible person for this major. I periodically updated the curricula in accordance to the regulations and requirements of the employing companies.

I have developed, equipped and updated several electronic teaching laboratories such as: Analog electronic and Digital electronic laboratories. I contributed to equipping the laboratories by attracting donations obtained through grants that I applied for. I have obtained over time many donations consisting of measurement equipment, development boards, devices and software with a total value of over 20.000 euro.

I was member of three PhD thesis committees, one in Romania and two in Hungary:

- Candidate Liviu Petrean with the thesis: The security of computer application. Contributions to protection of the executable files (Securitatea aplicațiilor de calculator. Contribuții la protecția fișierelor executabile), Technical University of Cluj-Napoca, Romania, 2011;

- Candidate Márien Szabolcs with the thesis: Decision Based Interpretation of Object-Oriented Design Principles and Design Patterns, Decision Contraction Theory with Practice, University of Debrecen, Hungary, 2012;
- Candidate Szilagyi Szabolcs with the thesis: Performance Analysis of Infocommunication Networks, University of Debrecen, Hungary, 2015.

During this period I was also supervisor for more than 10 master level dissertations and over 50 B.Sc. level final year graduation project at Technical University of Cluj-Napoca and University of Debrecen.

I also mentored several student that have participated in national and international student contest and have obtained prizes. In 2015 I was awarded with the title of Best Adviser in the eleventh Edition of the Digilent Design Contest, EU Region, Cluj-Napoca Romania.

I founded and organized a series of 6 international conferences entitled Embedded Systems Design and Applications, Baia Mare, Romania (2007-2010) and Conference on Embedded Systems and Wireless Sensors Networks Design and Applications, 2011-2012, Debrecen Hungary.

I had more than 15 visits at prestigious universities and research centers around the world participating on professors workshops, giving talks and presenting my group research activity including: Beijing University of Post and Telecommunication, China; Saxion University Enschede and Saxion University Deventer Netherland; University of Cambridge Nanoscience Centre, University of Oxford, Begbroke Science Park, Imperial College/London Centre for Nanotechnology (LCN) and University of Sheffield, Kroto Research Institute and Centre for Nanoscience from England; Centre Technologic de Telecomunicacions de Catalunya, Barcelona, Spain; Sapienta University Roma, Italy; Budapest University of Technology and Economics, Ecole Supérieure d'Ingénieurs en Informatique de Luminy, Université de la Méditerranée, AIX-Marseille II, Institute Universitaire de Tehnologie de Bethune, France, etc.

I founded two scientific journals and I am Editor in chief of the Carpathian Journal of Electronics and Computer Engineering. The journal is indexed in international database. I am also member of the Editorial Committees of International Review of Applied Sciences and Engineering an international journals edited in Hungary. I am a reviewer of international journals and conferences including: Infocommunications Journal, International Review of Applied Sciences and Engineering, Embedded world Conference, International Carpathian Control Conference,

International Conference on Soft Computing Models in Industrial and Environmental Applications, etc.

I was a mentor in the Mentoring program for young teachers in the framework of Didatech project: "Lifelong learning and training for higher education teachers in the technical sciences and engineering fields", POSDRU/87/1.3/S/60891 and expert in the same project. Also I was expert collaborator in the POSDRU/5/1.5/S2 project entitled: "Doctorate in Schools of Excellence - Quality evaluation in universities and increasing the visibility of the scientific publishing".

I received the AGEPI medal of Moldavian National Agency for Intellectual Property for "Group of inventions presented at International trade for inventions and practical ideas, INVEST-INVENT SIR 21".

In 2016 I received from University of Debrecen, the "Faculty of Informatics Award Honours" for the outstanding academic and scientific achievements.

I am currently occupying the following administrative positions:

- member of the board of the Department (since 2007);
- member of the board of the Faculty of Engineering, (since 2006).

From these positions I contributed to the development of my department and the faculty through specific activities such as: involvement in accreditations procedures for the study programs supervised by my department.

## III. Career development plan

The previous sections related to my scientific, academic and professional achievements present my contributions on all these aspects. A right balance between these three components is essential for each teacher and I am going to concentrate my future efforts towards improving on each of these components.

**Didactic component**

After more than 23 years of teaching and research, during which I contributed to several areas of reconfigurable embedded systems design and application, intelligent systems design using hardware implemented artificial neural networks and more recently to human activity recognition methods, my personal objective is to coordinate PhD research in these fields.

As was mentioned earlier I closely mentored several doctoral students: Daniel Mic, Alin Tisan, Ciprian Gavrincea, Claudiu Lung, Ioan Orha and now I am supervising the PhD student Jozsef Suto at University of Debrecen and also several MSc. students working on their master thesis. The ability of leading PhD research plays a central role in my strategy for the evolution and development of the professional career. Obtaining the habilitation will allow me to take lead of PhD coordination in Romania also. I plan to recruit my best former graduate Master students who are willing to continue their studies with a PhD, after obtaining the official right of doctoral supervision. I will contribute with my knowledge, expertize and overview of the field.

Currently, I am involved as the mentor of a group of 5 students from 5 universities working on the theme proposed by me: “Case Study 6 – Innovative Solutions for Assistance of Active Daily Life at Home” in the INNOSOC (2015-1-HR01-KA203-013124) project funded through ERASMUS+ Key Action 2 Strategic Partnership Program.

The project goal is to set up a transnational multidisciplinary intensive study program in the field of innovations based on ICT targeting societal challenges defined by Europe 2020 and Horizon 2020 programs. The INNOSOC curricula, will be available as multilingual open educational resource (OER). Student projects will be based on the “blended” mobility approach and organized in two phases: (i) preparatory (virtual mobility); and (ii) execution phase (physical mobility). Physical mobility will be implemented through three two-week workshops hosted by partner universities in 2016 (Zagreb), 2017 (Valencia). Workshop participants will be professors (16 professors from 11 universities from 8 countries) and students (100 students from 11 universities

from 8 countries) from partner universities. I was selected to give lectures at the workshop organized in April 2016 at Zagreb.

The objectives of my career improvement plan concerning teaching activities are:

- Development of new books and teaching materials related to the courses taught, finalize the online supports for all courses and to periodically update the content of my courses, seminars and laboratory works;
- Permanently updating of my knowledge by regularly attending to conferences, seminars, workshops, and webinars organized by industry professionals or professional associations to improve my technical education and for learn about new technologies and tools;
- Participating in Erasmus+ mobility program at partner universities abroad, for a useful exchange of experience on the latest teaching methods;
- Continue using modern tools for teaching and evaluation, the use of teaching strategies focusing on competences training and the use of contents that combines theoretical and practical aspects;
- Continuing the coordination the of Students' research group on embedded systems topics, stimulating and guiding the students to participate in the competitions at national and international level like Digilent Design Contest, Xilinx Open Hardware, Texas Instruments Innovation Challenge: Europe; TI MCU Design Challenge, Interconnection Techniques In Electronics (TIE);
- Propose new challenging topics for graduation and Master thesis;
- Continue attracting students into research topics by involving them into research activities and projects.

**Research component**

It is encouraging in terms of my future plans to see that my themes of interest are overlapping with some of the hottest research topics today as Artificial intelligence and robotics, Bio-informatics biomedical engineering, and eHealth.

My goals related to the research activity are presented in the following section.

***Coordination of a research group***

My future research plan is to continue and to develop my research activity within the Intelligent embedded systems research center as its director and also to disseminate the scientific results in

ISI journals and proceedings of prestigious international conferences. Being aware of the importance of research funding and the existing difficulties another important aspect that I have in mind is to obtain funding for further research through participation in international and national research projects. Also, based on my very good collaboration history with major companies like: Texas Instruments, Analog Devices, Xilinx, Digilent Inc., Altera, Agilent Technologies and National Instruments, I plan to apply constantly for donation consisting in their newest products (boards, devices and software).

As leader of a similar laboratory at the University of Debrecen and as a natural continuation of work on the national CEEX excellence project that I have led “National and international partnership development in the field of Embedded Systems, with the aim of organizing scientific meetings and drafting projects cooperatively in the EU framework programs", I plan to settle joint research activities and submission of joint calls under the program H2020.

On medium term, my objective is to attract into joint research activities of my collaborators within the PASED project from Hungary and Slovakia and to find new collaborators in Romania, Hungary, Slovakia and Ukraine in order to establish a regional research network in embedded system design and their applications. This network should ensure the students and teachers mobility for research internships.

The universities and research centers from this region represent knowledge islands in the field of Embedded Systems, and it is necessary to conjugate our efforts in order to be able to integrate successfully into the European research system. The development of partnerships are aimed at creating a communication platform among the research groups and centers with complementary profiles in the field of designing and testing Intelligent embedded systems and the dissemination of the results. This will insure the concentration of resources and skills existent in different universities and research centers aimed at obtaining the critical mass for the creation of scientific community in the field of embedded systems design.

***Conducting fundamental and applicative research***

My future research will include both fundamental and applicative research projects. The dominant topics of fundamental research projects will be the learning systems, machine learning. The applied research will be grouped around the application possibilities of the hardware implemented ANNs and the development of the ambient intelligent systems.

The scientific objectives are correlated with the development of Reconfigurable intelligent embedded systems having learning capabilities and adaptive behavior. My research will be further focused on the following topics:

- **Implementation of artificial neural networks using reconfigurable devices**

The aim of the research is to continue the design of hardware implemented artificial neural networks using the latest reconfigurable hardware devices. For implementation we will test and compare the method developed using System Generator with the HDL description of the ANN and the new high level synthesis tools like Vivado Design Suite HLx. I intend to use for implementation the new hardware devices like the Xilinx All Programmable FPGAs and 3D ICs (the 7 Series, Ultrascale and Ultrascale+) or the Xilinx's All Programmable SoC (Zynq-7000 SoC, Zynq UltraScale+ MPSoC) to integrate the software programmability of a processor with the hardware programmability of an FPGA. In this way the training of the neural networks could be done using the on-chip Dual-core ARM® Cortex™-A9 MPCore™. The ANN could be implemented in the FPGA part to take advantage of parallel implementation of the neurons. The ANN implemented in this way will be tested for example in human activity pattern recognition.

- **e-Health and Ambient assisted living systems**

I intend to continue the research related to the development of devices and systems to support the elderly or disabled persons' everyday independent life, using the latest assistive technologies.

The intelligent assistive environment will include prototypes of multi-sensor networks, multi standard communications, smart digital appliances, or smart monitoring systems Related to this topics I intend to submit a project to one of the Calls of the Active and Assistive Living (AAL) program which funds projects in the field of ICT for active and healthy ageing. My research will continue in the following directions:

- Activity and health status integrated platform development

For human activity monitoring we will use multiple wearable sensor tags like the CC2541 power-optimized system-on-chip (SoC) that combines the performance of a Bluetooth low energy (BLE) transceiver with the industry-standard enhanced 8051 MCU. The tag contains a 6 axis motion sensor MPU6050 (acceleration + angular velocity) and BMP180 pressure and temperature sensor. I intend to acquire data using these tags and make publically available a large dataset with human activities in order to support reproducible research.

- Activity and health status recognition using neural networks modeled in Matlab

We started to develop our own methods related to HAR for feature extraction and feature selection on publically available datasets as WARD [116], using a combination of feature extraction and selection methods from repositories [122], and we plan to continue this research on our dataset too.

- Activity and health status recognition using hardware implemented ANNs

Following the successful completion of the first two goals we will use the results to implement in hardware a real-time human activity recognition system using neural networks. We aim to extend the recognition system to the health status recognition also.

- **Intelligent embedded systems design and applications**

We will continue to constantly improve the previously designed hardware-software co-design platform based on FPGA adding new functions and modules. For fast prototyping of reconfigurable embedded systems, we plan to design new IP blocks that can be easily connected and can manage basic behaviors of the I/O devices, sensors and actuators. The design of the adaptive interfaces using neural networks will represent also a research interest. The designed architecture allows the insertion of intelligent interfaces based on neural networks. This platform will be based on the low cost general purpose FPGA boards without a need for hardware design of the boards.

***Increasing research visibility***

This goal is very important for me and it can be achieved by:

- participating in the technical committees and organizing committees of the main scientific conference and events from the field, e.g., Embedded World Conference
- participating in the reviewing committees of the journals from the field
- disseminating the main research results by attending the major conferences
- collaboration with researcher from abroad in common projects, e.g., as result of participating in FIRST project my research theme was introduced in the “Report of the Future Internet research Coordination Center (FIIRC) Report 2014, Hungary.

**Increasing publications' impact**

One of the today criteria of evaluation of a researcher visibility is the number of articles published in international journals with impact factor and number of citations. My short term objectives related to this aspects are:

- **Significantly increasing the number of publications in high ranked journals and conferences**

  My objective is to publish at least 2 articles per year in high ranked journals and conferences. For example this year some publications are currently:

  - submitted being currently under review, e.g., Springer - Medical & Biological Engineering & Computing [81]
  - accepted e.g., IEEE - ICCCC2016, 6th International Conference on Computers Communications and Control [80]
  - under development, e.g., International Journal on Computers Communications & Control.

- **Increasing number of citations**

  Observing the trend of my citations according to Google Scholar: 11 in 2011, 25 in 2012 55 in 2013 and 74 in 2015 it can be estimated that the number of citations of my works will continue to grow. I expect to have around 75-100 citation per year.

**BIBLIOGRAPHY**


[1] A. P. Engebrecht, Computational Intelligence: An Introduction, second ed., John Wiley and Sons, 2007.

[2] S. Oniga and A. Buchman, "A New Method for Hardware Implementation of Artificial Neural Network Used in Smart Sensors," in *Proceedings of The 10th International Symposium for Design and Technology of Electronic Packages, SIITME 2004*, Bucharest, Romania, 2004.

[3] S. Oniga, "A New Method for FPGA Implementation of Artificial Neural Network Used in Smart Devices," in *International Computer Science Conference microCAD 2005*, Miskolc, Hungary, 2005.

[4] S. Oniga, Sistem senzorial pentru recunoașterea gesturilor unei mâni utilizând rețele neuronale artificiale (Sensorial system for hand gesture recognition using artificial neural networks), PhD. thesis, 2005.

[5] A. R. Omondi and J. Rajapakse, FPGA Implementations of Neural Networks, Springer, 2006.

[6] S. Oniga, A. Tisan, D. Mic, C. Lung, I. Orha, A. Buchman and A. Vida, "FPGA Implementation of Feed-Forward Neural Networks for Smart Devices Development," in *International Symposium on Signals, Circuits and Systems, ISSCS 2009*, Iasi, Romania, 2009.

[7] S. Oniga, A. Tisan, D. Mic, A. Buchman and A. Vida-Ratiu, "Optimizing FPGA Implementation of Feed-Forward Neural Networks," in *Proceedings of the 11th International Conference on Optimization of Electrical and Electronic Equipment, OPTIM 2008*, Brasov, Romania, 2008.

[8] A. Tisan, S. Oniga and C. Gavrincea, "Hardware implementation of a MLP network with on-chip learning," in *Proceedings of the 5th WSEAS Int. Conf. on Data Networks, Communications & Computers*, Bucharest, Romania, 2006.

[9] S. Oniga, A. Tisan, A. Buchman and C. Lung, "Hardware Implementation of Simple Competitive Artificial Neural Networks with Neuron Parallelism," in *Proceedings of Regional Conference on Embedded and Ambient Systems*, Budapest, Hungary, 2007.

[10] S. Oniga, A. Tisan, D. Mic, A. Buchman, C. Gavrincea and A. Vida, "Hardware implementation of simple competitive neural networks with layer parallelism," in *International Symposium for Design and Technology of Electronic Packages, SIITME 2007*, Baia Mare, Romania, 2007.

[11] A. Tisan, S. Oniga, C. Gavrincea and A. Buchman, "FPGA implementation of a Self-organized map with on-chip learning," in *Proceedings of the 11th International Conference on Optimization of Electrical and Electronic Equipment, OPTIM 2008*, Brasov, Romania, 2008.

[12] J. Torrensen and S. Tomita, "A Review of Implementation of Bakpropagation Neural Networks," in *Parallel Architectures for Artificial Neural Networks*, N. Sundararajan and P. Saratchandran ed., IEEE CS Press, 1998.

[13] A. Tisan, A. Buchman, S. Oniga and C. Gavrincea, "A Generic Control Block for Feedforward Neural Network with On-Chip Delta Rule Learning Algorithm," in *Proceedinggs of 30th International Spring Seminar on Electronics Technology, ISSE 2007*, Cluj-Napoca, 2007.

[14] A. Tisan, C. Gavrincea, S. Oniga and A. Buchman, "Methods for embedded systems design with on-chip learning neural networks," in *Proceedings of the International Symposium for Design and Technology of Electronic Packages, SIITME 2007*, Baia Mare, Romania, 2007.

[15] A. Tisan, S. Oniga and C. Gavrincea, "Hardware implementation of various neural network with on-chip learning," *Wseas Transactions on Signal Processing*, vol. 2, no. 10, pp. 1357-1363, 2006.

[16] A. Tisan, S. Oniga, A. Buchman and C. Gavrincea, "Architecture and algorithms for syntetizable neural networks with on-chip learning," in *Proceedings of the 2007 International Symposium On Signals, Circuits and Systems, ISSCS 2007*, Iasi, 2007.

[17] L. Yihua, "Neural Networks in Hardware: A Survey," Department of Computer Science, University of California, Davis, 2001.

[18] J. Zhu and P. Sutton, "FPGA Implementations of Neural Networks – a Survey of a Decade of Progress," in *Proceedings of 13th International Conference on Field Programmable Logic and Applications (FPL 2003)*, Lisbon, 2003.

[19] A. Tisan, S. Oniga, D. Mic and A. Buchman, "Digital Implementation of The Sigmoid Function for FPGA Circuits," *Acta Technica Napocensis – Electronics and Telecommunications*, vol. 50, no. 2, pp. 15-20, 2009.

[20] E. Won, "A hardware implementation of artificial neural networks using field programmable gate arrays," *Nuclear Instruments and Methods in Physics Research*, vol. 581, pp. 816-820, 2007.

[21] C. Alippi and G. Storti–Gajani, "Simple approximation of sigmoidal functions: realistic design of digital neural networks capable of learning," in *Proceedings of IEEE International Symposium on Circuits and Systems*, Singapore, 1991.

[22] H. Amin, K. M. Curtis and B. R. Hayes-Gill, "Piecewise linear approximation applied to nonlinear function of a neural network," *IEE Proceedings - Circuits, Devices and Systems*, vol. 144, no. 6, pp. 313-317, December 1997.

[23] H. Amin, K. M. Curtis and B. R. Hayes-Gill, "Piecewise linear approximation applied to nonlinear function of a neural network," *Circuits, Devices and Systems, IEE Proceedings-*, vol. 144, no. 6, pp. 313-317, December 1997.

[24] M. Zhang, S. Vassiliadis and J. G. Delgado-Frias, "Sigmoid generators for neural computing using piecewise approximations," *IEEE Transactions on Computers*, vol. 45, no. 9, pp. 1045-1049, 1996.

[25] N. Nedjaha, R. da Silvaa, L. Mourelleb and M. da Silva, "Dynamic MAC-based architecture of artificial neural networks suitable for hardware implementation on FPGAs," *Neurocomputing*, vol. 72, no. 10-12, pp. 2171-2179, 2009.

[26] S. Oniga, A. Tisan, D. Mic, A. Buchman and A. Vida, "Hand Postures Recognition System Using Artificial Neural Networks Implemented in FPGA," in *Preceedings of 30th International Spring Seminar on Electronics Technology, ISSE 2007*, Cluj-Napoca, Romania, 2007.

[27] S. Oniga and I. Orha, "Hardware implemented neural networks used for hand gestures recognition," *Carpathian Journal of Electronic and Computer Engineering*, vol. 4, no. 1, pp. 93-96, 2011.

[28] J. Suto and S. Oniga, "Testing artificial neural network for hand gesture," *Creative Mathematics and Informatics*, vol. 22, no. 2, pp. 223-228, 2013.

[29] S. Oniga and P. Pop-Sitar, "Application Possibilities of Hardware Implemented Hybrid Neural Networks to Support Independent Life of Elderly People," in *Hybrid Artificial*

*Intelligent Systems*, Vols. 8th International Conference, HAIS 2013, Salamanca, Spain, Springer Berlin Heidelberg, 2013, pp. 520-529.

[30] A. Tisan, M. Cirstea, S. Oniga and A. Buchman, "Artificial olfaction system with hardware on-chip learning neural networks," in *Proceedings of 12th International Conference on Optimization of Electrical and Electronic Equipment, OPTIM 2010*, Brasov, romania, 2010.

[31] E. Monmasson, L. Idkhajine, M. B. I. Cirstea, A. Tisan and M. Naouar, "FPGAs in industrial control applications," *IEEE Transactions on Industrial Informatics*, vol. 2, no. 2, pp. 224-243, 2001.

[32] S. Oniga, J. Vegh and I. Orha, "Intelligent Human-Machine Interface Using Hand Gestures Recognition," in *Proceedings of 2012 IEEE International Conference on Automation Quality and Testing Robotics (AQTR)*, Cluj-Napoca, Romania, 2012.

[33] G. Sebestyen, A. Hangan, S. Oniga and Z. Gál, "eHealth Solutions in the Context of Internet of Things," in *Automation, Quality and Testing, Robotics, 2014 IEEE International Conference on*, Cluj-Napoca, 2014.

[34] S. Oniga, A. Tisan, C. Lung, A. Buchman and I. Orha, "Adaptive Hardware-Software Co-Design Platform for Fast Prototyping of Embedded Systems," in *Proceedinngs of 12th International Conference on Optimization of Electrical and Electronic Equipment*, Brasov, 2010.

[35] L. Marchese, "ELAN (Easy Learning Adaptive Nimble), Project proposal, FP7 European Research Program," 2013. [Online]. Available: http://www.synaptics.org. [Accessed 12 02 2016].

[36] J. Fleischmann, K. Buchenrieder and R. Kress, "A hardware/software prototyping environment for dynamically reconfigurable embedded systems," in *Proceedings of the 6th international workshop on Hardware/software codesign (CODES/CASHE '98)*, Washington, DC, USA, 1998.

[37] Xilinx Inc., "PicoBlaze 8-bit Embedded Microcontroller User Guide," [Online]. Available: http://www.xilinx.com/support/documentation/user_guides/ug129.pdf. [Accessed 28 01 2010].

[38] S. Oniga, A. R. Osan and A. I. Alexan, "Alternative control method of the smart house: natural gestures," *Carpathian Journal of Electronic and Computer Engineering*, vol. 4, no. 1, pp. 97-100, 2011.

[39] R. Harper, Inside the smart home, London: Springer-Verlag, 2003.

[40] L. Bounegru, *Smart Houses: From Managing the House at a Distance to the management of Life Itself*, Amsterdam: University of Amsterdam, 2009.

[41] SmartHomeUSA, "HOW X10 WORKS," [Online]. Available: http://www.smarthomeusa.com/how-x10-works/#theory. [Accessed 12 2 2016].

[42] D. Heckman, A Small World. Smart Houses and the Dream of the Perfect Day, London: Duke University Press, 2008.

[43] I. Orha and S. Oniga, "Automated system for evaluating health status," in *Proceedings of IEEE 19th International Symposium for Design and Technology in Electronic Packaging (SIITME)*, Galati, Romania, 2013.

[44] "Tutorial: e-Health Sensor Platform for Arduino and Raspberry Pi [Biometric / Medical Applications]," Cooking Hacks, [Online]. Available: http://www.cooking-hacks.com. [Accessed 12 02 2016].

[45] A. Alexan, A. Osan and S. Oniga, "Advanced Medication Dispenser," *Carpathian Journal of Electronic and Computer Engineering,* vol. 6, no. 2, pp. 26-31, 2013.

[46] J. Sütő, S. Oniga and I. Orha, "Microcontroller based health monitoring system," in *2013 IEEE 19th International Symposium for Design and Technology in Electronic Packaging*, Galaţi, Romania, 2013.

[47] S. Oniga and et. al, "Human Activity Monitoring System Design Implementation and Test, Technical report: Future Internet Research, Services and Technology," Debrecen, 2013.

[48] S. Oniga and J. Sütő, "Human activity recognition using neural networks," in *Proceedings 15th International Carpathian Control Conference - ICCC 2014*, Velke Karlovice, Czech Republic, 2014.

[49] Digilent, Inc., "chipKIT™ Max32™ Board Reference Manual," 27 01 2015. [Online]. Available: https://reference.digilentinc.com/_media/chipkit_max32:chipkit_max32_rm.pdf. [Accessed 28 02 2016].

[50] Digilent, Inc., "chipKIT™ WiFi Shield™ Reference Manual," 23 01 2014. [Online]. Available: https://reference.digilentinc.com/_media/chipkit_shield_wifi:chipkit_wifi_shield_rm.pdf. [Accessed 28 02 2016].

[51] Texas Instruments Incorporated, "Chronos: Wireless development tool in a watch," [Online]. Available: http://www.ti.com/tool/ez430-chronos. [Accessed 28 02 2016].

[52] Future Technology Devices International Ltd., "VDIP1 Vinculum VNC1L Module Datasheet Version 1.02," 14 June 2011. [Online]. Available: http://www.ftdichip.com/Support/Documents/DataSheets/Modules/DS_VDIP1.pdf. [Accessed 3 March 2016].

[53] BM innovations, "Chest Strap BM-CS5SR," BM innovations GmbH, [Online]. Available: http://www.bm-innovations.com/index.php/chest_straps.html. [Accessed 28 02 2016].

[54] C. Lung, S. Oniga, A. Buchman and A. Tisan, "Wireless data acquisition system for IoT applications," *Carpathian Journal of Electronic and Computer Engineering,* vol. 6, no. 1, pp. 64-67, 2013.

[55] I. Orha and S. Oniga, "Wearable sensors network for activity recognition using inertial sensors," *Carpathian Journal of Electronic and Computer Engineering,* vol. 8, no. 1, pp. 3-6, 2015.

[56] InvenSense, Inc, "MPU-9250 Product Specification Revision 1.0," 2014. [Online]. Available: http://store.invensense.com/datasheets/invensense/MPU9250REV1.0.pdf. [Accessed 03 03 2016].

[57] I. Orha and S. Oniga, "Activity recognition using an e-Textile data acquisition system," in *Proceedings of the 2015 IEEE 21st International Symposium for Design and Technology in Electronic Packaging (SIITME)*, Brasov, Romania, 2015.

[58] I. Orha and S. Oniga, "Wearable sensors network for health monitoring using e-Health platform," *Carpathian Journal of Electronic and Computer Engineering,* vol. 7, no. 2, pp. 25-29, 2014.

[59] J. Sütő, S. Oniga and A. Buchman, "Real time human activity monitoring," *Annales Mathematicae et Informaticae,* vol. 44, pp. 187-196, 2015.

[60] D. Buttlar, J. Farrell and B. Nichols, PThreads Programming, O'Reilly Media, 1996.

[61] J. Suto, S. Oniga and H. Gy., "A simple fast Fourier transformation algorithm to microcontrollers and mini computers," in *Proceedings of IEEE 18th International Conference on Intelligent Engineering Systems (INES 2014)*, Tihany, Hungary, 2014.

[62] V. Lugade, E. Fortune, M. Morrow and K. Kaufman, "Validity of using triaxial accelerometers to measure human movement—Part I: Posture and movement detection," *Medical Engineering & Physics*, vol. 36, pp. 169-176, 2014.

[63] M. Kangas, A. Konttila, P. Lindgren, I. Winblad and T. Jamsa, "Comparison of low-complexity fall detection algorithms for body attached accelerometers," *Gait & Posture*, vol. 28, pp. 285-291, 2008.

[64] D. Kammler, A First Course in Fourier Analysis, Cambridge : Cambridge University Press, 2008.

[65] A. Godfrey, R. Conway, D. Meagher and G. Ólaighin, "Direct measurement of human movement by accelerometry," *Medical Engineering & Physics*, vol. 30, pp. 1364-1386, 2008.

[66] A. Godfrey, K. Bourke, M. Ólaighin, P. Ven de van and J. Nelson, "Activity classification using a single chest mounted tri-axial accelerometer," *Medical Engineering & Physics*, vol. 33, pp. 1127-1135, 2011.

[67] J. Kavanagh and B. Menz, "Accelerometry: A technique for quantifying movement patterns during walking," *Gait & Posture*, vol. 28, pp. 1-15, 2008.

[68] A. Lago, "DOMEO, Domestic Robot For Elderly Assistance. First," in *AAL Forum*, Lecce, Italy, 2011.

[69] V. Dupourqué, "DOMEO, an Open Robotic Platform for Cognitive and Physical Personalized Homecare Services," in *Workshop PAL*, Sophia Antipolis, France, 2011.

[70] E. Ackerman, "Suitable Technologies Introduces Beam Remote Presence System," *IEEE Spectrum Robotics News*, 26 September 2012.

[71] A. Mataric, A. Okamura and H. Christensen, "A Research Roadmap for Medical and Healthcare Robotics," Arlington, VA, 2008.

[72] S. Oniga, "ICT tools for smart homes and assisted living for elders," in *Proceedings of Advances in wireless sensors networks*, Debrecen, Hungary, 2013.

[73] A. Alexan, A. Osan and S. Oniga, "Personal assistant robot," in *Proceedings of 2012 IEEE 18th International Symposium for Design and Technology in Electronic Packaging (SIITME)*, Alba Iulia, Romania, 2012.

[74] A. Alexan, A. Osan and S. Oniga, "AssistMe robot, an assistance robotic platform," *Carpathian Journal of Electronic and Computer Engineering*, vol. 5, no. 1, pp. 1-4, 2012.

[75] I. Orha and S. Oniga, "Assistance and telepresence robots: a solution for elderly people," *Carpathian Journal of Electronic and Computer Engineering*, vol. 5, no. 1, pp. 87-90, 2012.

[76] J. Sütő, Á. Máté, J. Végh and I. Oniga, "Developing a general purpose data collector framework for robots," in *Proceedings of 13th International IEEE Carpathian Control Conference (ICCC)*, High Tatras, Slovakia, 2012.

[77] J. Sütő and S. Oniga, "Remote controlled data collector robot," *Carpathian Journal of Electronic and Computer Engineering*, vol. 5, no. 1, pp. 117-120, 2012.

[78] D. Feil-Seifer and M. Mataric, "Human-robot interaction," in *Encyclopedia of Complexity and System Science*, new York, Springer, 2009, pp. 4643-4659.

[79] M. Heerink, B. Kröse, V. Evers and B. Wielinga, "Assessing Acceptance of Assistive Social Agent Technology by Older Adults: the Almere Model," *International Journal of Social Robotics,* vol. 2, no. 4, pp. 361-375, 2010.

[80] J. Suto, S. Oniga and P. Pop-Sitar, "Comparison of Wrapper and Filter Feature Selection Algorithms on Human Activity Recognition," in *Proceedings of International Conference on Computers Communications and Control (ICCCC) 2016 [Accepted for publication]*, Băile Felix, Oradea, România, 2016.

[81] J. Suto and S. Oniga, "Feature analysis to human activity recognition," *Medical & Biological Engineering & Computing,* no. Submmited, 2016.

[82] O. D. Lara and M. A. Labrador, "A survey on human activity recognition using wearable sensors," *IEE Communications Surveys & Tutorials,* vol. 15, no. 3, pp. 1192-1209, 2013.

[83] U. Maurer, A. Rowe, A. Smailagic and D. Sieviorek, "Location and activity recognition using eWatch," *Ambient Intelligence in Everyday Life, Lecture Notes in Computer Science,* vol. 3864, pp. 86-102, 2006.

[84] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," in *Pervasive Computing, Lecture Notes in Computer Science, vol. 3001*, Springer Berlin Heidelberg, 2004, pp. 1-17.

[85] J. Parkka, M. Ermes, P. Korpipaa, J. Mantyjarvi, J. Peltola and I. Korhonen, "Activity classification using realistic data from wearable sensors," *Information Technology in Biomedicine, IEEE Transactions on,* vol. 10, no. 1, pp. 119-128, 2006.

[86] S. J. Preece, J. Y. Goulermas, L. P. J. Kenney and D. Howard, "A comparison of different feature generation methods in activity classification," in *Int. Conf. on Ambulatory Monitoring of Physical Activity and Movement (ICAMPAM)*, Rotterdam, 2008.

[87] M. Sekine, T. Tamura, M. Akay, T. Fujimoto, T. Togawa and Y. Fukui, "Discrimination of walking patterns using wavelet-based fractal analysis," *IEEE Trans Neural Syst Rehabil Eng.,* vol. 10, no. 3, pp. 188-196, 2002.

[88] M. Sekine, T. Tamura, T. Togawa and Y. Fukui, "Classification of waist-acceleration signals in a continuous walking record," *Medical engineering & physics,* vol. 22, no. 4, pp. 285-291, 2000.

[89] T. Tamura, M. Sekine, M. Ogawa, T. Togawa and Y. Fukui, "Classification of acceleration waveforms during walking by wavelet transform," *Methods of information in medicine,* vol. 36, no. 4-5, pp. 356-359, 1997.

[90] M. Nyan, F. Tay, K. Seah and Y. Sitoh, "Classification of gait patterns in the time-frequency domain," *Journal of biomechanics,* vol. 39, no. 14, pp. 2847-2856, 2006.

[91] S. Preece, J. Goulermas, L. Kenney and D. Howard, "A comparison of feature extraction methods for the classification of dynamic activities from accelerometer data," *Biomedical Engineering, IEEE Transactions on,* vol. 56, no. 3, pp. 871-879, 2009.

[92] U. Maurer, A. Smailagic, D. Siewiorek and M. Deisher, "Activity recognition and monitoring using multiple sensors on different body positions," in *Proceedings of International workshop on wearable and implantable body sensor networks (BSN'06)*, Cambridge, MA, 2006.

[93] J. Yang, J. Wang and Y. Chen, "Using acceleration measurements for activity recognition: An effective learning algorithm for constructing neural classifiers," *Pattern Recognition Letters,* vol. 29, no. 16, pp. 2213-2220, 2008.

[94] D. Lewis, "Naive (Bayes) at forty: The independence assumption in information retrieval," in *Proceedings of the 10th European Conference on Machine Learning*, Chemnitz, Germany, 98.

[95] L. Gao, A. Bourke and J. Nelson, "A System for Activity Recognition Using Multi-Sensor Fusion," in *33rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Boston, Massachusetts USA, 2011.

[96] D. Heckerman, "A tutorial on learning with Bayesian networks," in *Innovations in Bayesian Networks*, Springer Berlin Heidelberg, 2008, pp. 33-82.

[97] N. Ravi, N. Dandekar, P. Mysore and M. Littman, "Activity recognition from accelerometer data," in *In In Proceedings of the Seventeenth Conference on Innovative Applications of Artificial Intelligence(IAAI (2005)*, 2005.

[98] C. Doukas and I. Maglogiannis, "Advanced patient or elder fall detection based on movement and sound data," in *Proceedings of 2nd international conference on pervasive computing technologies for healthcare*, Tampere, Finland, 2008.

[99] N. Kern, G. Tröster, B. Schiele, H. Junker and P. Lukowicz, "Wearable Sensing to Annotate Meeting Recordings," *Personal and Ubiquitous Computing,* vol. 7, no. 5, pp. 263-274, 2003.

[100] D. Minnen, T. Starner, J. Ward, P. Lukowocz and G. Troster, "Recognizing and Discovering Human Actions from On-Body Sensor Data," in *Proceedings of IEEE International Conference on Multimedia and Expo, ICME 2005*, Amsterdam, 2005.

[101] A. Khan, Y.-K. Lee, S. Lee and T.-S. Kim, "A Triaxial Accelerometer-Based Physical-Activity Recognition via Augmented-Signal Features and a Hierarchical Recognizer," *IEEE Transactions on Information Technology in Biomedicine,* vol. 14, no. 5, pp. 1166-1172, 2010.

[102] I. Kouris and D. Koutsouris, "A comparative study of pattern recognition classifiers to predict physical activities using smartphones and wearable body sensors," *Technology and Health Care,* vol. 20, no. 4, pp. 263-275, 2012.

[103] L. Gao, A. Bourke and J. Nelson, "Evaluation of accelerometer based multi-sensor versus single-sensor activity recognition systems," *Medical engineering & physics,* vol. 36, no. 6, pp. 779-785, 2014.

[104] J. Lester, T. Choudhury, N. Kern, G. Borriello and B. Hannaford, "A hybrid discriminative /generative approach for modelling human activities," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2005.

[105] J. Lester, T. Choudhury and G. Borriello, "A practical approach to recognize physical activities," in *Pervazive Computing*, Vols. Lester, Jonathan, Tanzeem Choudhury, and Gaetano Borriello. "A practical approach to recognizing physical activities." Pervasive Computing. Springer Berlin Heidelberg, 2006. 1-16., Dublin, Springer Berlin Heidelberg, 2006, pp. 1-16.

[106] J. Wu, G. Pan, D. Zhang, G. Qi and S. Li, "Gesture Recognition with a 3-D Accelerometer," in *Ubiquitous Intelligence and Computing*, Brisbane, Australia, Springer Berlin Heidelberg, 2009, pp. 25-38.

[107] M. Mathie, A. Coster, N. Lovell, B. Celler, S. Lord and A. Tiedemann, "A pilot study of long-term monitoring of human movements in the home using accelerometry," *Journal of telemedicine and telecare,* vol. 10, no. 3, pp. 144-151, 2004.

[108] D. M. Karantonis, M. R. Narayanan, M. Mathie, N. H. Lovell and B. G. Celler, "Implementation of a real-time human movement classifier using a triaxial accelerometer

for ambulatory monitoring," *IEEE Transactions on Information Technology in Biomedicine,* vol. 10, no. 1, pp. 156-167, 2006.

[109] F. Allen, E. Ambikairajah, N. Lovell and B. Celler, "Classification of a known sequence of motions and postures from accelerometry data using adapted Gaussian mixture models.," *Physiological measurement,* vol. 27, no. 10, pp. 935-951, 2006.

[110] A. Khan, .. Lee and T. Kim, "Accelerometer signal-based human activity recognition using augmented autoregressive model coefficients and artificial neural nets," in *Preceedings of 30th Annual International IEEE Confernece Engineering in Medicine and Biology Society Conference*, Vancouver, British Columbia, Canada, 2008.

[111] S. Oniga and J. Sütő, "Activity Recognition in Adaptive Assistive Systems Using Artificial Neural Networks," *Elektronika ir Elektroteknika,* vol. 22, no. 1, 2016.

[112] I. Orha and S. Oniga, "Study regarding the optimal sensors placement on the body for human activity recognition," in *Procewdings of IEEE 20th International Symposium for Design and Technology in Electronic Packaging (SIITME)*, 2014.

[113] S. Oniga, "Optimal Activity Recognition Systems Using ANN, Technical report: Future Internet Research, Services and Technology," Debrecen, Hungary, 2015.

[114] S. Oniga and J. Sütő, "Optimal recognition Method of human Activities using artificial neural networks," *Measurement Science Review,* vol. 15, no. 6, pp. 323-327, 2015.

[115] A. Y. Yang, R. Jafari, S. S. Sastry and B. R. , "Distributed recognition of human actions using wearable motion sensor networks," *Journal of Ambient Intelligence and Smart Environments,* vol. 1, no. 2, pp. 103-115, 2009.

[116] A. Yang, P. Kuryloski and R. Bajcsy, "WARD: A Wearable Action Recognition Database," in *Proceedings of 27th Annual CHI Conference*, Boston, MA, USA, 2009.

[117] A. Tisan, Contributions to the analysis, synthesis and implementation of applications with intelligent sensorial systems: The electronic nose, Cluj Napoca, Romania: Unpublished Ph. D. Thesis, Technical University of Cluj Napoca, 2009.

[118] D. Mic, Contributions to the developement of a hardware - software integrated environement for controlling electric motors, Brasov: Unpublished Ph. D. Thesis, Facultatea de Inginerie Electrică si Stiinta Calculatoarelor, Universitatea Transilvania din Brasov.

[119] C. Lung, Theoretical and experimental contributions to implementetion of intelligent embedded systems, Cluj Napoca, Romania: Unpublished Ph. D. Thesis, Universitatea Tehnica din Cluj Napoca, 2013.

[120] C. Gavrincea, Research regarding the implementation of a neural network used to process signals generated by the muscular and nervous system, Timisoara: Unpublished Ph. D. Thesis, Universitatea Politehnica din Timisoara, 2009.

[121] I. Orha, Human activity recognition and physiological parameters monitoring systems, Cluj Napoca: Unpublished Ph. D. Thesis, Universitatea Tehnica din Cluj Napoca, 2015.

[122] Z. Zhao, F. Morstatte, S. Sharma, S. Alelyani, A. Anand and H. Liu, "Advancing feature selection research ASU feature selection repository. Technical Report,," Arizona State University, 2011.

[123] C. Gavrincea, A. Tisan, A. Buchman and S. Oniga, "Survey of wavelet based denoising filter design," in *Proceedings of the 30th International Spring Seminar on Electronics Technology*, Cluj-Napoca, Romania, 2007.

[124] D. Mic, A. Tisan, S. Oniga, C. Lung and S. Sabou, "The Development of a Simulink Library with FPGA Compatible Parametric Components for Electric Machines Control,"

in *Proceedings of the International Symposium on Signals, Circuits and Systems, ISSCS 2009*, Iasi, Romania, 2009.

[125] D. Mic, S. Oniga, E. Micu and C. Lung, "Complete Hardware / Software Solution for Implementing the Control of the Electrical Machines with Programmable Logic Circuits," in *Proceedings of the 11th International Conference on Optimization of Electrical and Electronic equipment, OPTIM 2008*, Brasov, Romania, 2008.

[126] D. Mic, E. Micu and S. Oniga, "Hardware and Software Co-Design Method for Implementation of Closed Loop Control for a Brushless DC Motor,," in *Proceedings of the 10th International Conference on Optimization of Electrical and Electronic Equipment, OPTIM 2006*, Brasov, Romania, 2006.

[127] J. Suto and S. Oniga, "A New Relation between “Twiddle Factors” in the Fast Fourier Transformation," *Elektronika ir Elektrotechnika*, vol. 21, no. 4, pp. 59-59, 2015.

[128] J. Sütő and S. Oniga, "FPGA implemented reduced Ethernet MAC,," in *Proceedings of the in 4th IEEE International Conference on Cognitive Infocommunications, CogInfoCom 2013*, Budapest, Hungary, 2013.

[129] A. Tisan, M. Cirstea, A. Buchman, A. Parera, S. Oniga and D. Ilea, "Holistic modeling, design and optimal digital control of a combined renewable power system," in *Proceedings of the 2010 IEEE International Symposium on Industrial Electronics, ISIE 2010*, Bari, Italy, 2010.

[130] R. Besenczi, M. Szilagyi, N. Batfai, A. Mamenyak, S. Oniga and M. Ispany, "Using Crowdsensed Information for Traffic Simulation in the Robocar World Championship Framework," in *Proceedings of the 6th IEEE Conference on Cognitive Infocommunications, CogInfoCom 2015* , Gyor, Hungary, 2015.

[131] S. Sebastian, S. Oniga and C. Lung, "Magnetic sensors in inertial navigation system," in *Proceedings of the 2014 IEEE 20th International Symposium for Design and Technology in Electronic Packaging (SIITME 2014)*, Bucharest, Romania, 2014.

[132] A. Buchman, S. Lungu, S. Oniga and A. Tisan, "Ultrasonic Echo Detection Experiments Using Large Beam Angle Transducers in Narrow Tubes," in *Proceedings of the 31st International Spring Seminar on Electronics Technology, ISSE 2008*, Budapest, Hungary, 2008.

[133] A. Buchman, A. Tisan and S. Oniga, "Ultrasonic Data Acquisition Signal to Noise Ratio Improvement," in *Proceedings of the 30th International Spring Seminar on Electronics Technology, ISSE 2007*, Cluj-Napoca, Romania, 2007.

[134] J. Sütő and S. Oniga, "A new C++ implemented feed forward artificial neural network simulator," *Carpathian Journal of Electronic and Computer Engineering*, vol. 6, no. 2, pp. 3-6, 2013.

[135] Z. Gal, B. Almasi, T. Daboczi, R. Vida, S. Oniga, S. Baran and I. Farkas, "Internet of Things: Application areas and Research Results of the FIRST Project,," *Infocommunications Journal*, vol. VI, no. 3, pp. 37-44, 2014.

[136] D. Mic and S. Oniga, "FPGA Implementation of a Digital Tachometer with Input Filtering," in *Proceedings of the International Symposium for Design and Technology of Electronic Packages, SIITME 2007*, Baia Mare, Romania, 2007.