Combination of Wireless sensor network and artificial neural network : A new approach of Modeling

THÈSE

présentée et soutenue publiquement le 12 Décembre 2013

pour l'obtention du

Doctorat de l'Université de Toulon

(spécialité électronique, génie informatique, et traitement du signal)

par

Yi Zhao

Composition du jury

Président :	M. le Professeur M. Cotsaftis Ecole Centrale d'Electronique
Rapporteurs :	M. le Professeur M. A. Aziz Alaoui Université du Havre M. le Professeur Cyrille Bertelle Université du Havre
Examinateur :	M. le Professeur E. Moreau Université de Toulon
Directeurs :	M. Jean Marc Ginoux (MCF, HDR) Université de Toulon M. Valentin Gies(MCF) Université de Toulon

---

Laboratoire PROTEE

École doctorale 548: Mer et Sciences, Université de Toulon (UTLN)

Mis en page avec la classe thloria.

Remerciements

Je tiens à exprimer mes sincères remerciements et témoigner de ma grande reconnaissance à mon directeur et co-directeur de thèse M. Jean Marc Ginoux et M. Valentin Gies qui m'ont accueilli et soutenu pendant ces trois ans, qui m'ont encadré et motivé dans ce travail de recherche.

Je souhaite exprimer également ma gratitude envers tous les membres du jury. Merci particulièrement à M. le Professeur Michel Cotsaftis, qui a accepté de présider le jury de cette thèse. C'est un honneur que MM. les Professeurs M.A. Aziz Alaoui et Cyrille Bertelle aient été les rapporteurs de ma thèse. Je souhaite exprimer également ma gratitude envers tous les autre membres du jury, à la fois pour leur intérêt vis-à-vis de mes travaux de recherche, ainsi que pour les remarques et les discussions stimulantes lors de la soutenance.

Je remercie également M. Patrice Bendahan, Xavier Nesi et Christophe Lephay et les autres personnages de l'equipe ESPHI pour les passionnants échanges.

Je voudrais remercier M. le Professeur Yves Lucas et M. Stéphane Mounier de m'avoir accueilli au sein de laboratoire PROTEE. Merci pour leur soutien pendant ces trois ans.

Je voudrais remercier MM. les Professeurs Pascal Vincent, Stéphane Holé et Dr. Olivier Couture qui m'ont accueilli et soutenu pendant mon Master à UPMC (Université Paris VI) et ESPCI Paris Tech.

Je remercie également la Région PACA et l'entreprise ESPHI d'avoir financé ce travail, et je voudrais aussi remercier sincèrement à tous ceux qui ont contribué de près ou de loin à cette thèse et leur exprimer ma gratitude pour l'intérêt et le soutien qu'ils m'ont généreusement accordé.

Je remercie profondément mes parents, pour leur dévouement incessant durant ces années.

Enfin, l'itinéraire scientifique qui m'a amené à ce manuscrit et il aurait été inconcevable sans l'engagement de la France, de son Université et de ses collectivités territoriales que je remercie tout particulièrement.

à Mes parents ;
à Fenghuan Ma ;

Résumé

Face à la limitation de la modélisation paramétrique, nous avons proposé dans cette thèse une procédure standard pour combiner les données reçues a partir de Réseaux de capteurs sans fils (WSN) pour modéliser a l'aide de Réseaux de Neurones Artificiels (ANN). Des expériences sur la modélisation thermique ont permis de démontrer que la combinaison de WSN et d'ANN est capable de produire des modèles thermiques précis. Une nouvelle méthode de formation "Multi-Pattern Cross Training" (MPCT) a également été introduite dans ce travail. Cette méthode permet de fusionner les informations provenant de différentes sources de données d'entraînements indépendants (patterns) en un seul modèle ANN. D'autres expériences ont montré que les modèles formés par la méthode MPCT fournissent une meilleure performance de généralisation et que les erreurs de prévision sont réduites. De plus, le modèle de réseau neuronal basé sur la méthode MPCT a montré des avantages importants dans le multi-variable Model Prédictive Control (MPC). Les simulations numériques indiquent que le MPC basé sur le MPCT a surpassé le MPC multi-modèles au niveau de l'efficacité du contrôle.

Mots-clés: Réseaux de Capteurs sans fil (WSN), Réseaux de neurones artificiels (ANN), Modélisation, Formation croisée de Multiple source (MPCT) méthode, Les économies d'énergie, les performances de généralisation, Control Prédictif.

Abstract

A Wireless Sensor Network (WSN) consisting of autonomous sensor nodes can provide a rich stream of sensor data representing physical measurements. A well built Artificial Neural Network (ANN) model needs sufficient training data sources. Facing the limitation of traditional parametric modeling, this paper proposes a standard procedure of combining ANN and WSN sensor data in modeling. Experiments on indoor thermal modeling demonstrated that WSN together with ANN can lead to accurate fine grained indoor thermal models. A new training method"Multi-Pattern Cross Training" (MPCT) is also introduced in this work. This training method makes it possible to merge knowledge from different independent training data sources (patterns) into a single ANN model. Further experiments demonstrated that models trained by MPCT method shew better generalization performance and lower prediction errors in tests using different data sets. Also the MPCT based Neural Network Model has shown advantages in multi-variable Neural Network based Model Predictive Control (NNMPC). Software simulation and application results indicate that MPCT implemented NNMPC outperformed Multiple models based NNMPC in online control efficiency.

Keywords: Wireless Sensors Networks(WSN), Artificial Neural Networks (ANN), Modeling, Multi-Pattern Cross Training(MPCT) method, Energy efficiency, Generalization performance, Model Predictive Control.

Table des matières

Part II Artificial Neural Networks and Wireless Sensor Network

Part IV Results and Perspectives

General introduction

Since the year of 1960, discussions have been made on the subject of non-parametric techniques against parametric techniques [1]. As a matter of fact, the discussions have motivated researcher and scientists from different domains including finance, economics, pattern recognition, modeling, signal and image processing etc.
These preliminary discussions revealed their methodological differences : in parametric techniques, a mathematical model (or statistical model) was developed for the problem from a statistical, geometrical, physical, or phenomenological perspective. Although a relatively small number of unknown model parameters exist, they can be estimated from the data. These models are based on predefined rules as mathematical equations which can give a clear definition of the problem. It explicitly defines step-by-step tasks to achieve the desired results. This can be an ideal way to model a phenomenon when the rules relating to this problem is well known.
In some practical cases, where the rules are either unknown or they are extremely difficult to be mathematically presented. Parametric modeling techniques do not meet these requirements and are not suitable as an appropriate solution [2, 3].
Artificial Neural Networks (ANN) is a typical nonparametric modeling technique. It is extremely useful in situations where the rules of the phenomena are either unknown or are too difficult to discover. Moreover, it is considered as an appropriate solution for applications where the precision of traditional techniques cannot meet the requirements when a phenomena is too complex to be modeled with the well-defined rules [4].

1 Wireless Sensor Networks(WSN)

Wireless Sensor Networks (WSN) is considered as a significant technology ever since its birth, A wireless sensor network (WSN) consisting of autonomous sensor nodes can provide a rich stream of sensor data representing physical measurements. It can be described as a network of distributed self-powered nodes that could sense or exchange with environment. The main advantage of WSN is that it could be easily and rapidly installed and gather information for a long period of time, providing an enormous quantity of sensor data. WSN based applications have shown a rapid growth in a variety of fields, including target tracking and surveillance, natural disaster relief, health monitoring, environment exploration and geological sensing, etc [5].

So far, most of the sensor networks deployed involve a relatively limited number of sensor nodes. They are usually connected to a central processing unit where all signal processing is performed [6]. On the contrary, the WSN is a wireless distributed network, in which the signal processing is often done with acquisitions.
To better understand the necessity of deploying WSN in real applications, some description and statement should be made :

- Wireless
  Cabled sensors networks work perfectly when nodes can be wired to stable energy sources and reliable infrastructure of communications. However, in many practical applications, the monitored target-area does not equipped any of these, for example, when the monitoring target is a group of wild animals in the nature. Therefore sensor nodes should rely on local, finite, and relatively small energy sources as well as wireless communication channels, this would open a new door for wider applications with mobility and Autonomy.

- Distrubuted sensing
  When a precise location of the interest-area is unknown in a surveillance zone, WSN allows a distribution of more autonomous sensors in the place closer to the wanted monitoring area. Instead of using only one or few sensors, this gives more signal to noise ratio (SNR) and better opportunities for the line of sight. SNR can be addressed in some cases by the deployment of a high sensitivity sensor, however, the line of sight of and more generally disturbance of noise cannot be processed by the deployment of a sensor with high sensitivity. Thus, distributed sensing provides more robustness under different environmental conditions.

- Distributed processing
  It may be considered reasonable that in the cabled sensor networks, data can be communicated back to a central processing unit. However, for the sensor nodes of WSN, there are two main barriers : first, the finite energy budget is the first primary constraint. RF (Radio Frequency) Communication makes the main energy consumer. Secondly, most wireless sensor network defined limited data transmission rate. Thus, we need to process data as much as possible inside the network to reduce the energy consumption as well as the number of bits transmitted, particularly over longer distances.

2 Artificial Neural Network(ANN)

Computer has become an necessary tool in engineering. Engineers have used various computer applications to improve their efficiency and performance. Ever since early 70s, Artificial Intelligence (AI) have been implemented by engineers to perform specialized tasks design.

Although computers are involved in a variety of engineering activities, currently, the main software applicable areas are with well-defined rules, such as the sophisticated analysis, graphic and CAD applications, etc. However, where there are no defined rules or heuristics, the use of computer is very limited.

Artificial Neural Networks (ANN) are another AI application that has recently been widely used to model nonlinear system, or system with unknown dynamics in many different domains of science and engineering [7]. ANN has been found to be extremely useful in situations where the rules are either unknown or are very difficult to explicit. Some of the main attributes of ANN can be listed as follows :

- ANN can learn and generalize from examples to produce practical solutions to problems.
- They can perfectly cope with situations where the input data of the network is unclear or incomplete.
- ANN are able to adapt solutions in time and to compensate from changing circumstances
- The data for training an ANN can be theoretical data, experimental data or empirical evidence based on reliable experiences.

ANN can be considered as a good generalized approximator based on the experience of a set of training data, it contains no explicit rules. Although it may not have the exact formality as the traditional parametric approaches, it is still A powerful tool that can produce perfect approximations when formal traditional solutions has difficulties with insufficient knowledge of the problem. It is chosen for applications where precision of traditional techniques can not meet the requirement that the problem is too complex to model with rules [4].

3 Combination of WSN and ANN in modeling

So far, WSN and ANN together have hardly been used in modeling. However, we think there are two main advantages of applying WSN and ANN in modeling and system identifications. First, the nature of WSN and ANN make them a combination : WSN could be easily and rapidly implemented, providing a huge quantity

of sensor data. These data sources can be essential for the ANN to identify a fine grained model. Second, they have high practical values : traditional mathematical modeling, Take the thermal modeling of building rooms as an example, approaches like [8] are usually established with well defined equations, they are usually used in general simulations instead of practical applications. It is mainly because these models are based on elements such as room thermal capacitances/resistance, airflow rate, heat transfer coefficient, heat gain coefficient, etc. These parameters are difficult to measure precisely in buildings. Also, the dynamic behavior of some phenomenon is very complex [1], It is nearly impossible to obtain an accurate mathematical model with limited number of system parameters. The WSN system, on the contrary, is highly transplantable as it could be quickly equipped under any environmental surroundings to gather real-time thermal data. Additionally, with its self-adaptive learning and mapping ability, ANN can directly simulate the relations between the modeling object's inputs and outputs. Based on the two reasons, it seems that the combination of WSN and ANN can be a valuable and reasonable solution for modeling.

1. Here, the word "complex" refers to the complexity of the physical phenomenon, it differs from the definition of complexity in mathematics. The mathematical conception on complexity can be found in the following works [9, 10, 11].

4 Outline of this Thesis

This thesis is structured in four parts. In the first part of this thesis, we highlighted the limitations of traditional modeling methods by establishing and analyzing a mathematical thermal model of a building room. Instead ,we proposed to use WSN and ANN combined solution in building thermal modeling. The second part mainly introduces the concept and development of ANN as modeling tools and the WSN as our platform data acquisition. The wireless protocol and the design of embedded systems of our WSN system is systematically presented in this part. Also, we conducted experiments to show that our developed WSN system is reliable for applications outside or inside the building. we introduced the software we developed for combining ANN and WSN in modeling, the main functions and features of this software has been highlighted.
In Part 3 of this thesis, we have shown the design of all our experiences in modeling the thermal response of a real building room with ANN and WSN. To improve the ANN thermal models' generalization performance, we proposed a new training method called "Multi-Pattern Cross Training". It is able to merge knowledge from different training data patterns into a single ANN model. Thus, by exploring the general behavior of the phenomenon, it can be properly used to build a more complete model. We then presented the concept of Model Predictive Control (MPC) in which we focused on the neural network based MPC in the control of non- linear processes. The necessity to use the MPCT method in MPC for optimizing the control efficiency is outlined in Part 3.
The results of WSN based ANN Modeling and the modeling performance with MPCT method are presented and discussed in Part 4, It is marked that MPCT method can be used to build models with better generalization performance, it also shows that MPCT method can improve the efficiency in Neural network based MPC. Some of our major embedded codes of the WSN system and is presented in the appendix of this thesis.

Première partie

The dynamic thermal behavior of Buildings

1

The phenomenon involved in building

1.1 Building Science : Thermal Phenomena

Buildings of residential and commercial facilities take about 40% of total energy consumption in Europe and USA. In China, residential urban energy consumption has tripled between 1996 and 2008 [12]. As a matter of fact, the energy consumption varies between different types of buildings. For example, buildings more than 30 years old consume from 300 to $400 kWh/m^2/year$, while modern buildings consume approximately from 150 to $200 kWh/m^2/year$ [13]. The improvement of energy efficiency in these old buildings can bring considerable environmental and financial benefits.
In order to find the best strategy to achieve energy efficiency, a basic understanding on the thermal behavior of buildings is necessary.

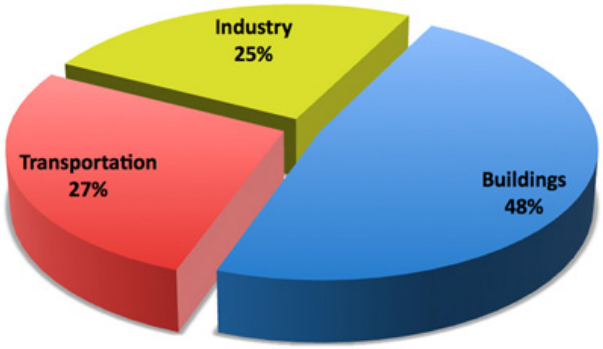


FIGURE 1.1 – Energy use of buildings in the United States (architecture2030.org)

1.2 Physical phenomenon involved in buildings

The three main physical phenomena involved in the thermal response of buildings are :

- Radiation : transferring heat from one object to another by electromagnetic waves (without contact).
- Conduction : the heat spreads within the material (solid, liquid, gas).
- Transport : The heat transfer between the air and the solid material resulting from the displacement of the particles (from the air) on the interface.

For a homogeneous material, we have the following equations to describe the behavior of heat transfer through it. If the thickness of the material is T, the thermal resistance R considering the conductivity of the material λ is expressed by :

$$R = \frac{T}{\lambda} \tag{1.1}$$

The heat transfer coefficient U [2] is an indicator of energy efficiency. The value of U is expressed as the inverse of R :

$$U = \frac{1}{R} = \frac{\lambda}{T} \tag{1.2}$$

For example, for a concrete wall ($\lambda = 1.75W/mK$) with a thickness of $0.3m$, its thermal resistance has the value :

$$R = \frac{0,3}{1,75} = 0,171m^2K/W \tag{1.3}$$

For a wall built of different materials, the overall thermal resistance is the sum of the resistances, and if we take into account the internal and external convection, we have here the overall thermal resistance :

$$R = \sum_{i=1}^{n} \frac{T_i}{\lambda_i} + \frac{1}{h_{in}} + \frac{1}{h_{out}} \tag{1.4}$$

The coefficients h_{in} and h_{out} are due to the internal and external convection respectively.

Another important concept is the thermal inertia also known as thermal mass, it is a term used to describe the ability of materials to store heat (thermal storage capacity). The fundamental characteristic of any thermal mass materials is its ability to absorb, store or release heat.

2. It refers as the U-value in U.S.

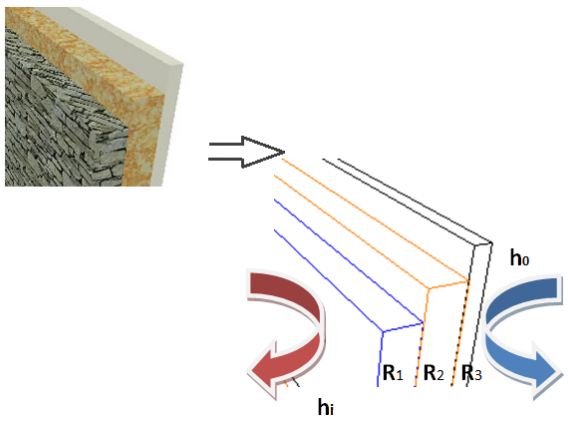


FIGURE 1.2 – Thermal resistance of a wall

Adding thermal mass in building's insulated envelope can effectively reduce the extreme temperatures inside the building, making the most moderate average internal temperature and comfort for habitants. Normally, building materials which can hold a heavy store of heat contain high thermal mass. On the contrary, the materials which is not capable of storing heat has lower thermal mass. In warm weather, the thermal mass at an initial temperature lower than the ambient air acts as a heat sink. By absorbing heat from the atmosphere, the temperature of the internal air reduce during the daytime. By releasing heat from the materials, the temperature of internal air drops slower. Thermal mass is particularly effective in areas where there is a large difference between the daytime maximum temperature and minimum night temperature.
The other important element is the solar radiation see Fig. 1.4, the heat inputs are generally two ways :

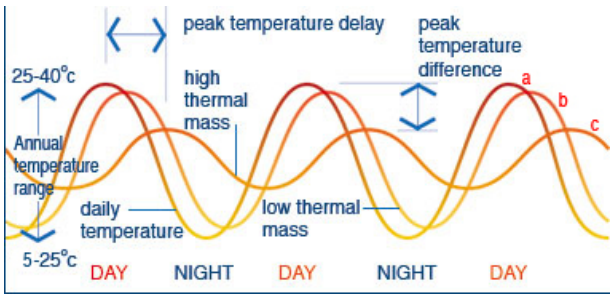


FIGURE 1.3 – Temperature variations in the use of thermal mass

– Absorbed on the surfaces of wall, solar radiation is converted to heat stored in the walls.

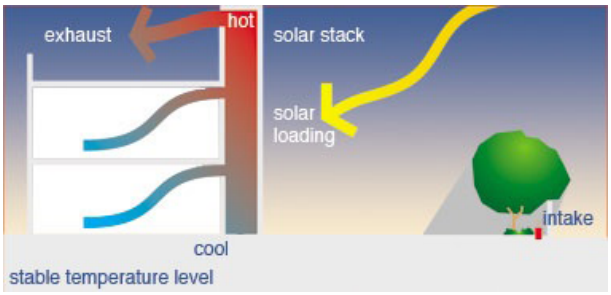


FIGURE 1.4 – Solar radiation and indoor temperature

– The absorbed solar radiation through the transparent window is converted into heat in the ambient air.

2

Mathematical model of a room

2.1 Thermal modeling of building

Based on the description made in the previous chapter, the thermal response of building is related to many nonlinear and time varying effects such as the heat transport, convection, solar radiation, the interior charges, lighting equipment, circulation inside and outside air etc. The building room is a very complex thermal system, it is almost impossible to build a precise mathematical model with a limited well characterized parameters.
Fig. 2.1is presented as a mathematical model of the room E106 in the campus of Université de Toulon (UTLN), this room is equipped with a simple air conditioning as internal heating system and a vent pipe.

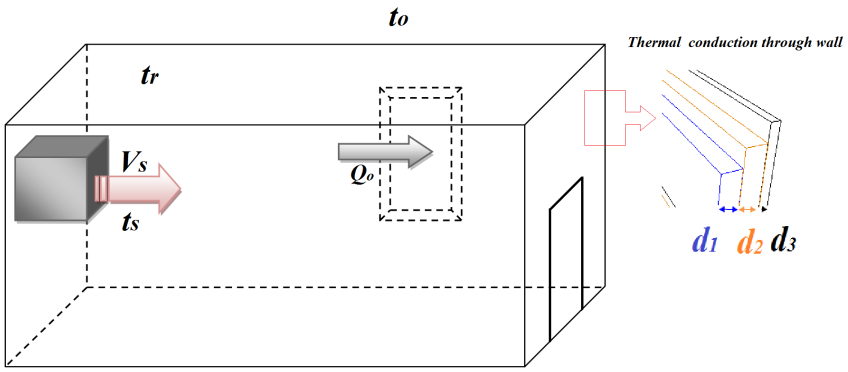


FIGURE 2.1 – Thermal model of room E106 in University of Toulon

2.1.1 Effective Indoor Thermal Time Constant(EITTC)

The thermal time constant (TTC) can be used to quantify the thermal inertia of a building. In simpler words, it describes how the temperature varies under different thermal excitations. In previous works, the building TTC are calculated mostly with the thermal resistance and thermal capacitance of the building fabrics [14][15], Thus, these thermal constants are usually used to simulate the thermal response of buildings under environmental excitations. They can be very difficult to be used directly to characterize the indoor temperature changes in response to existing indoor heating/cooling system. Some of the thermal time constant are calculated based on major simplifications that all heat transport phenomena in building are linear [16] which can be limited facing the complex dynamic behavior of buildings. Thus, we proposed in this thesis a new concept : the Effective Indoor Thermal Time Constant (EITTC). By taking into consideration of both existing heating/cooling system's performance and the building room's thermal property. The EITTC can be used directly to characterize the indoor thermal response under existing indoor heating/cooling excitation and different outdoor conditions.

2.1.2 Mathematical model of room E106 in UTLN

In the mathematical model proposed by J. Florez and G.C. Barney [16], the thermal time constant T_r is described as :

$$\frac{dT_r}{dt} + \frac{T_r}{\tau_r} = \frac{Q_h}{C_r} + \frac{T_o}{C_r R_o} + \frac{T_f}{C_r R_f} \tag{2.1}$$

where T_r is the room temperature, τ_r is the room thermal time constant. Q_h is heat source in the room, T_o is the out door temperature, C_r is the thermal flow store of the room, R_0 is the linear thermal dissipator from the room to exteriors, T_f is the fabric (wall, roof) temperature, and the linear thermal dissipator from room to the fabric is R_f. The thermal time constant of the room can be written as :

$$\tau_r = \frac{C_r R_f R_o}{R_f + R_o} \tag{2.2}$$

where τ_r is the room thermal time constant, C_r is the thermal flow store of the room, R_0 is the linear thermal dissipator from the room to exteriors, and the linear thermal dissipator from room to the building fabrics is R_f. This equation (Eq.(2.2)) is derived from a simplified linearized mathematical model of a room, it is much a general expression of room thermal constant. It can not be used directly to describe the indoor thermal response.

So, in order to characterize precisely the thermal response of the building room, in this thesis, a more detailed mathematical thermal model of building room is established :

According to the law of conservation of energy, the mathematical expression of this room model reads :

$$m_n c_n \frac{dt_n}{dt} = \rho_s C_s G_s (t_s - t_n) + \frac{t_0 - t_n}{R_1} + \frac{t_0 - t_n}{R_2} + \frac{t_0 - t_n}{R_3} + \frac{t_h - t_n}{R_4} + Q_d \quad (2.3)$$

where $m_n c_n$ is the heat capacity of indoor air $(kJ/^\circ C)$, t_s is the output air temperature $(^\circ C)$ from the air-conditioner, G_s is the volume of heated output air per second(m^3/s) ; t_n, t_h, t_0 are the indoor temperature, the adjoining room temperature and outdoor temperature (°C). ρ_s is the density of heated output air(kg/m^3) ; C_s is the specific heat capacity of heated output air $(KJ/(kg^\circ C))$; R_1, R_2,R_3,R_4 are the thermal resistance of walls, windows, roof and the partition wall to the next door(k/kW). Respectively, $\frac{t_0-t_n}{R_1}$, $\frac{t_0-t_n}{R_2}$, $\frac{t_0-t_n}{R_3}$ and $\frac{t_h-t_n}{R_4}$ are the thermal conduction (kW) on every part of the room. Q_d is the sum of heat (kW) emitted by the indoor equipments and human bodies.
The incremental equation derived from Eq. (2.3) can be expressed as follows :

$$\tau \frac{d\Delta t_n}{dt} + \Delta t_n = K_1 \Delta G_s + K_2 \Delta Q \quad (2.4)$$

where τ is the EITTC (s), K_1 is the amplification factor of the indoor temperature caused by heated air $(^\circ C/(m^3/s))$; K_2 is the amplification factor of indoor thermal perturbation $(^\circ C/kW)$; ΔQ is the total change of the indoor heat (kW). If the temperature of heated air is defined as t_s, we have :

$$K_2 = \frac{1}{\rho_s C_s G_s + \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} + \frac{1}{R_4}}, \quad (2.5)$$

$$\tau = \frac{m_n c_n}{\rho_s C_s G_s + \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} + \frac{1}{R_4}} = K_2 m_n c_n, \quad (2.6)$$

$$K_1 = \frac{\rho_s C_s (t_s - t_n)}{\rho_s C_s G_s + \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} + \frac{1}{R_4}} = K_2 \rho_s C_s (t_s - t_n), \quad (2.7)$$

$$\Delta Q = \Delta Q_d + \frac{\Delta t_0}{R_1} + \frac{\Delta t_0}{R_2} + \frac{\Delta t_0}{R_3} + \frac{\Delta t_h}{R_4} \quad (2.8)$$

After performing a Laplace transformation of equation (2.4), we have the following equation :

$$\tau S \Delta t_n(S) + \Delta t_n(S) = K_1 \Delta G_s(S) = K_1 \Delta G_s(S) + K_2 \Delta q_f(S) \quad (2.9)$$

If we define the delay between the thermal influences and indoor temperature variations as T_R, the transfer function is obtained between the amount of heated air and the changes of indoor temperatures read :

$$\frac{\Delta t_n(S)}{\Delta G_s(S)} = \frac{K_1 e^{-T_r s}}{T_1 S + 1} \tag{2.10}$$

In the same way, the transfer function between the thermal interference and indoor temperature reads.

$$\frac{\Delta t_n(S)}{\Delta q_f(S)} = \frac{K_2 e^{-T_r s}}{T_1 S + 1} \tag{2.11}$$

If we consider that the volume of heated air G_S is constant, we can find the differential equations on the thermal response of the building room :

$$\tau' \frac{dt_n}{dt} + t_n = K_1' t_s + K_2' qf \tag{2.12}$$

In this equation, T_1',K_1',K_2' are the same parameters as in Eq. (2.4).

$$\tau' = \frac{M_c}{\rho_s c_s G_s + \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} + \frac{1}{R_4}} = K_2' M_c, \tag{2.13}$$

$$K_1' = \frac{\rho_s c_s G_s}{\rho_s c_s G_s + \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} + \frac{1}{R_4}} = K_2' \rho_s c_s G_s, \tag{2.14}$$

$$K_2' = \frac{1}{\rho_s c_s G_s + \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} + \frac{1}{R_4}}, \tag{2.15}$$

$$qf = q_n + \frac{t_0}{R_1} + \frac{t_0}{R_2} + \frac{t_0}{R_3} + \frac{t_n'}{R_4} \tag{2.16}$$

And if we add the delay T_r', we have the transfer function between the temperature of the heated air and the temperature of the room :

$$\frac{t_n(S)}{t_s(S)} = \frac{K_1' e^{-T_r' s}}{T_1' S + 1} \tag{2.17}$$

and the transfer function between the thermal interference and indoor temperature :

$$\frac{t_n(S)}{q_f(S)} = \frac{K'_2 e^{-T'_r s}}{T'_1 S + 1} \tag{2.18}$$

If the volume of heated air is considered as constant, we have

$$G_s = G_{s0} \tag{2.19}$$

where G_{s0} is the volume of heated air at the start time.
If we compare from Eq. (2.4) to Eq. (2.12), we have :

$$T_1 = T'_1; \quad T_r = T'_r; \quad K_1 = \frac{t_{s0} - t_{n0}}{G_s 0} K'_1; \quad K_2 = k'_2 = \frac{1}{\rho_s c_s G_{s0}} K'_1 \tag{2.20}$$

2.1.3 Limitation of the established mathematical model

Two common problems with this type of models are : 1. the model is based on simplifications and it does not have the ability to provide accurate predictions in real time and it is hard to precisely measure all the parameters used in this model. 2. these models is only applicable for the specific modeled room, when the room changes, all the parameters and equations should be redefined. Thus, they are not practical.

For the first problem, as we explained in the introduction : for this type of mathematical modeling, the basic problem is that the number of parameters increases with the increase of problem's complexity [3]. If we want to build a precise mathematical thermal model of the room, we have to take all the physical phenomenon involved into the models, which means all the heat transferring, conduction, radiation. Furthermore, The thermal response of the room is related to the outside temperature, solar radiation, the interior equipment, lighting, human activity and the opening and closing of the door, etc. It is almost impossible to establish a precise mathematical model with a limited number of parameters. The complexity of this dynamic thermal response in the room will finally lead to a explosion of parameters in the correspondent mathematical model. This is the main reason that most of the mathematical room thermal models used in simulations are based on simplifications [16] below regardless of the fact that these simplification has weakened the accuracy of models :

- All the phenomena of heat transfer are considered linear.
- Consider the room is a unique capability and neglect the content inside the room.
- neglecting the air flow inside the room.
- Suppose that the internal temperature is homogeneous.

The accuracy of the model can not be guaranteed due to the parameters : for example, parameters such as t_0, t'_n and Q_n are time varying elements which is very difficult to be measured in real time. Furthermore, the most basic parameters like thermal resistance, thermal capacity thermal mass are also difficult to be measured precisely. In addition to the difficulty of measuring these parameters.
The second simple and practical problem is that these models are not reusable, if the room changed, all the parameters have to be measured again due to different building fabrics and geometries.
Facing these limitations, new modeling methods should to be explored.

2.2 Proposed new modeling method

In this thesis, the modeling solution combining WSN data and ANN is proposed to overcome the limitations of the traditional mathematical models introduced. ANN has been applied to improve the performance of WSNs [17]. However, We find that there is almost no previous research combined WSN data and ANN in modeling.
The two main advantages of applying WSN and ANN in building room thermal modeling. First, the nature of WSN and ANN make them a perfect combination : on one hand, WSN could be easily and rapidly implemented, providing a huge quantity of sensor data. These data sources in return could be essential for the ANN to identify a fine grained thermal model. Second, they have high practical values : as it is discussed, mathematical thermal modeling approaches are usually used in general simulations. They are hard to be applied in some practical applications. The WSN system, on the contrary, is highly transplantable as it could be quickly equipped in any buildings to gather real-time thermal data. Additionally, with its self-adaptive learning and mapping ability, ANN can directly simulate the relations between building thermal affecting factors (heating source, solar radiation, outdoor temperature) and indoor temperatures. Based on the two reasons above, we believe that the combination of WSN and ANN can be a valuable solution for indoor thermal modeling.

Admittedly, some existing thermal modeling techniques exist, however, previous studies outlined that ANN model outperformed Auto-Regressive(ARX) models in predicting the indoor temperature because the ANN models are more sensible to the nonlinearities of the thermal effects in buildings [18][19]. Further more, J.W. Moon has pointed out in his previous work [20] that ANN's adaptability makes it a more advantageous method in thermal control compare to Fuzzy method.
Different techniques have their own advantages and shortcomings, here, it is necessary to point out that the WSN and ANN combined modeling solution has a great advantage over the other modeling techniques : it is highly transportable and applicable under.
WSN as a very cheap, low-power, easy-to-use, miniature electronic devices can be installed quickly in most buildings while ANN's universal approximating ability makes it possible in generating adaptive thermal models under different environmental and indoor conditions. This means the same system can be easily deployed anywhere without modifications in the hardware or software. We think the practicality and wide-applicability of the proposed solution make it distinctive from other techniques.

Deuxième partie

Artificial Neural Networks and Wireless Sensor Network

3

Neural Network Basics

3.1 Neural network basics

To understand the methodology that we present in this thesis, a systematic understanding of the neural network is necessary. In this section, we will focus on the conceptions and the development of artificial neural networks (ANN).
Neural networks is a kind of computing architectures inspired by biological brain [21] [22]. This kind of architectures are commonly called "connectionist systems", they consist of interconnected and interacting components named nodes or neurons. Neural networks are not characterized by explicit representation of knowledge or well defined rules of the problem, no symbol or values that correspond directly to classes of interest. Knowledge is implicitly represented in the diagrams of interactions between network's components. Individual neuron in ANN emulate the biological neuron, it takes input data and make simple processing of it, selectively passing on the processed data to other neurons (Fig. 3.1) Before sending out

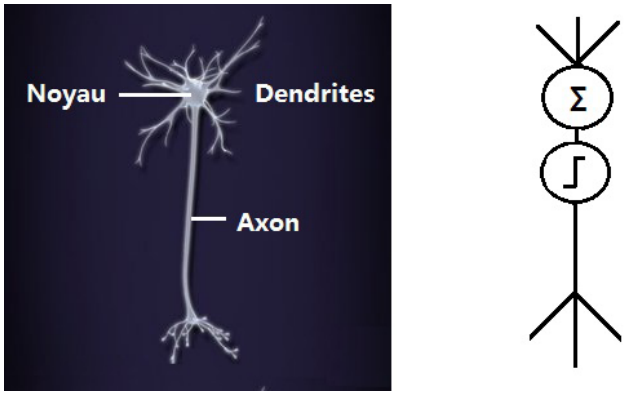


FIGURE 3.1 – Comparison between a biological neuron and an artificial neuron model

the processed data, each neuron use its "activation" function to format the data. Every neuron has its own weight values associated in the network, and these values

determine at what level the input data are related to output data. Originally, a raw neural network only has a determined structure, and the weight values are usually random. The training of the network is a procedure that the network gets the knowledge from the data of experiences. The iterative repeat of training allows the network to determine its best fitted weight values (i.e., in image processing, a neural network can be trained to classify different images. Weight values are defined during the training phase in which the network is taught to identify a image by the typical input image's characteristics.).

Once the neural network is trained, it can be used to process new data. Useful summaries of fundamental neural network principles can be found in the relative works of Rumelhart *etal.* [23], McClelland and Rumelhart [24], Rich and Knight [25], Winston [26], Luger et Stubblefield [22], Gallant[27] and Richards and Jia [28].

3.2 Model of McCulloch-Pitts

In the 1940's, McCulloch-Pitts network presented the beginning of Neural computing [29]. These connectionist networks are called "decision machines", as exemplified in Fig. 3.2. A set of inputs is multiplied by the weights, and the outputs are then determined by simple logical or mathematical operations. In this way the input values are related to output values. McCulloch-Pitts networks are binary ; only 0 or 1 is the acceptable input form, also it is the standard format of output. The mechanism of McCulloch-Pitts network is simple : if the sum of the products of the inputs associated with weights is greater than or equal to 0, the output returns 1, otherwise, 0. A threshold [3] should be exceeded or equalled in order to have an output of the system 1. The rules used in mapping the input values to the output values, is the activation function which means that this function is used to determine the rules for the output node. McCulloch-Pitts networks can be used in logical computations (see Fig. 3.2) and they contributed to inspire further research into connectionist models during the 1950's.

3.3 Perceptron

In the late 1950's, a connectionist system with limited learning ability is developed [30]. Rosenblatt created a new framework of neurons known as the perceptron (see Fig. 3.3). Like the network of McCulloch-Pitts, perceptron consists of binary activation functions and its binary output is determined by summing the products of its inputs and the weight values. Unlike the network of McCulloch-Pitts, a variable threshold value is used : if the linear sum of the input/weight products is greater than a threshold value (θ), the output of the system is 1, otherwise 0.

3. In McCulloch-Pitts network, the threshold is fixed at 0

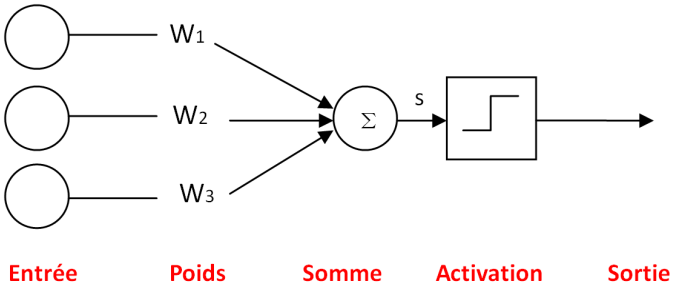


FIGURE 3.2 – McCulloch-Pitts' neuron model

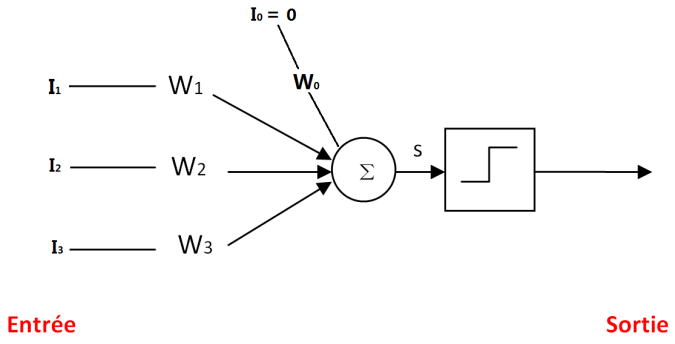


FIGURE 3.3 – A perceptron

3.4 The Delta Rule

Perceptron is a landmark on the road of Neural Networks. It is the beginning of making connectionist networks which is capable of mapping complex relations between inputs and outputs. By the end of 1950's, the connectionist community reached the conclusion that connectionist models need more flexible learning rules. Thus, mathematical derivation has been introduced to the activations of the network. The Delta Rule [4] was then invented in the early 1960's [31]. Although one may find this training rule is similar to the perceptron learning rule above [24]. The difference is that instead of using a threshold as the activation function, the delta rule uses a linear activation function for the output neuron. Further more, the differences of activations (network output) in every training iteration can be used to drive the learning. This is revolutionary comparing to all the connectionist's networks before. The delta rule makes it possible to create self-adaptive neural networks. The network itself can now begin to update its weights by reducing the difference between target and actual output activations.
The data's propagation through the network is presented below. Firstly, it will be normalized by the input layer neurons and multiplied by the associated weight. Then, the data are summed and reach to the output neurons where they are sent through an activation function, Finally, after the activation function, they become the output of the network.

$$y_j = f(\sum_{i}^{n} w_{ij} a_i) \tag{3.1}$$

where n is the total number of input layer neurons, w_{ij} represent the weight updating from the input layer neuron i to the output layer neuron j, a_i is the the activations function of the neurons in layer i, y_j is the network's output and f is the activation function of the output layer.
Given a training data pattern, the errors of the network is measured with the cost function (also called error function) (Fig. 3.4) [32]. The cost function is usually defined as the sum of the squares of the differences between all target outputs and actual network outputs (see Eq. (3.2)) The Delta Rule uses the gradient descent learning method to minimize the errors : the weights modified themselves along the direct path in weight-space, changes are applied to the weights proportionally to the negative direction of the derivation of the errors.

$$E = \frac{1}{2} \sum_{j=1}^{p} \sum_{n=1}^{m} (r_{jn} - y_{jn})^2 \tag{3.2}$$

4. Also called the Widrow and Hoff learning rule or the least mean square (LMS) rule.

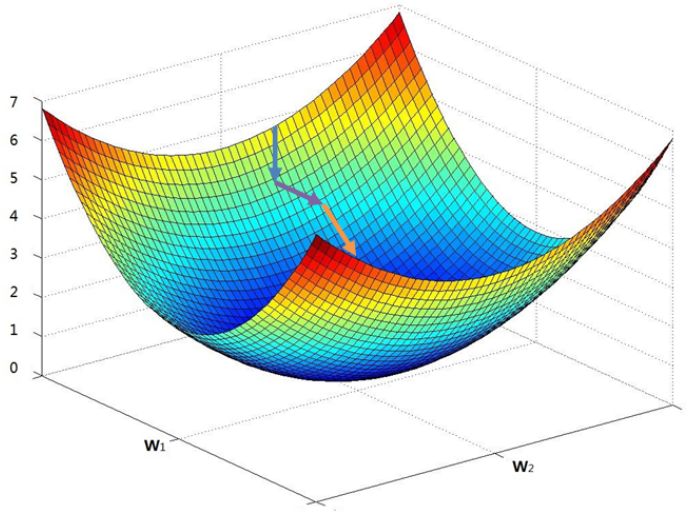


FIGURE 3.4 – The Delta rule in minimizing the error

where E is the total error, j is the number of all training patterns and n is the number of output neurons. A normalized form of E is given in the mean squared error (MSE) equation below :

$$MSE = E_m = \frac{1}{2pm} \sum_{j=1}^{p} \sum_{n=1}^{m} (r_{jn} - y_{jn})^2 \tag{3.3}$$

where P is the total number of training data patterns and m is the total number of neurons in the output layer, the minimization of errors can be made in this way, for a given weight c :

$$\frac{\delta E_m}{\delta w_{ij_c}} = \frac{\delta E_m}{\delta y_{j_z}} \frac{\delta y_{j_z}}{\delta w_{ij_c}} \tag{3.4}$$

where y_j sub z is the activation function for the node from the $w_i j$ sub c, it can be further expressed as :

$$\frac{\delta E_m}{\delta y_{j_z}} = y_{j_z} - r_{j_z} \tag{3.5}$$

and

$$\frac{\delta y_{j_z}}{\delta w_{ij_n}} = \frac{\delta}{\delta w_{ij_c}} \sum_{n=1}^{m} (w_{ij_n} y_{in}) = y_{i_c} \tag{3.6}$$

we have then :

$$\frac{\delta E_m}{\delta w_{ij_x}} = (y_{j_z} - r_{j_z})(y_{i_c}) \tag{3.7}$$

With the gradient descent learning method, the changes of weights must be proportional to the negative of Eq. (3.7), If a learning rate ϵ is introduced in Eq. (3.7), it can be re-written as :

$$\Delta w_{ij_x} = -\epsilon \frac{\delta E}{\delta w_{ij}} = \epsilon \delta a_{i_x} \tag{3.8}$$

Here, we take a simple example of using the Delta Rule to update weights of a neural network. The training data patterns which contain both inputs and outputs are presented in Table 3.1. The aim of training the network is to make it capable of labeling each of the four input cases in this table. This problem requires basically a network with four input nodes and one output node. If we define all the 4 weights associated with each input node are initially 0, and the learning rate ϵ is 0.25 During

TABLE 3.1 – Training data : Inputs and outputs samples

	Inputs				Outputs
Training data	+1	-1	+1	-1	1
	+1	+1	+1	+1	1
	+1	-1	-1	+1	-1
	-1	-1	-1	+1	-1

the training, each training data pattern flows through the network individually, and weights update according to the Delta Rule : when the first training data pattern is given to the network, the sum equals 0. Because the target output for this training data set is 1, thus, the error is $1 - 0 = 1$. By applying Delta Rule, (see Eq. (3.8)), the changes of all the four weights in the network are calculated, the results is 0,25 ; -0,25 ; 0,25 ; -0,25. As the initial value of weights are 0 ; 0 ; 0 ; 0. Thus, the weights become 0,25 ; -0,25 ; 0,25 ; -0,25. When the second training data pattern is presented, the update of the weights are calculated as 0,25 ; 0,25 ; 0,25 ; 0,25. So the weights are 0,5 ; 0 ; 0,5 ; 0 after the second round of training, respectively, after

the third and the fourth pattern presented, the weight changes from 0.25 ; 0.25 ; 0.75 ; -0,25 to -0,375 ; 0,375 ; 0,875 ; -0,375, At the end of the irrelative training procedure, the total mean square error is :

$$E_q = (1)^2 + (1)^2 + (-1)^2 + (-0,5)^2 = 3.25 \tag{3.9}$$

At the end of this first iteration, it may be hard to realize that the error minimized alone with the changes of weights. However, with iterative flow of training, the error reduces. In this example, at the tenth iteration of training, the network successes in classifying all data patterns. At this moment, we can say that the network is well trained and it can be used to classify the data pattern with similar rules.

For The Delta Rule, a basic requirement is the following : The rules consisted in the training data set should be linear [30]. This has greatly limited the use of Delta Rule, Minsky and Papert proposed multi-layer network to solve this problem. However, they quickly realized that the use of linear activation function in Multi-layer neural networks is unable to solve the given problem. Functionally, a multi-layer neural network with linear activation functions is the same as a simple input-output network using linear activation functions. Two main questions are left for the connectionist community : "what kind of activation function should be used ?" and "how to train a multi-layer neural network ?" Blocked by these two questions, research of connectionist's network has been freezed during the 70's.

3.5 Multilayered Neural Network and Back-propagation

After almost ten years of depression in the connectionist's community, a multilayer neural network training algorithm has been proposed independently by Rumelhart [23] and Yann Lecun [33]. The algorithm is known as the Back-propagation (BP). It uses error function of the Delta Rule and make these errors propagates backward through the network to update the weights matrix of the network. As BP is the most essential algorithm used in this thesis, a whole chapter is dedicated to it in order to give a detailed presentation of this algorithm.

3.6 Learning mode

There are two main weights updating mode for neural networks : batch training, and on-line training. For batch training, the weight updates is carried out only when all the individual training data cases are presented to the network and the total error derivative is calculated [34]. On-line mode is also known as sequential learning. It updates the weight value after each training data case submitted to the network.On-line learning is not a true gradient descent, because weights update slightly after each training data case presented to the network instead of updating with total derivative[34]. Normally, Batch learning need more memory capacity while on-line learning requires more weight updates. One advantage of online learning is that for online training, it is more likely to jump out of local minima during the training. Also, when high data redundancy exists training data patterns, the online training mode can be more efficient than batch mode [35]. In this thesis, we have chosen the online model for the network training because of these two advantages.

3.6.1 Activation functions

The most commonly used activations functions for connectionist's network are presented below. They are respectively linear function, step function, sigmoid function and bi-sigmoidal function (see Fig. 3.5- 3.6). The mathematical expression of

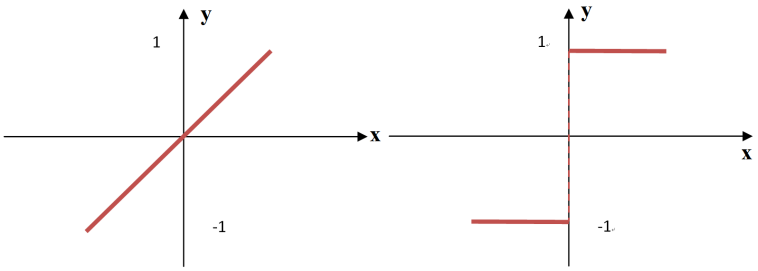


FIGURE 3.5 – Activation function

these activation functions and their derivative are summarized below.
The linear activation functions reads :

$$f(x) = x, \ \forall x. \tag{3.10}$$

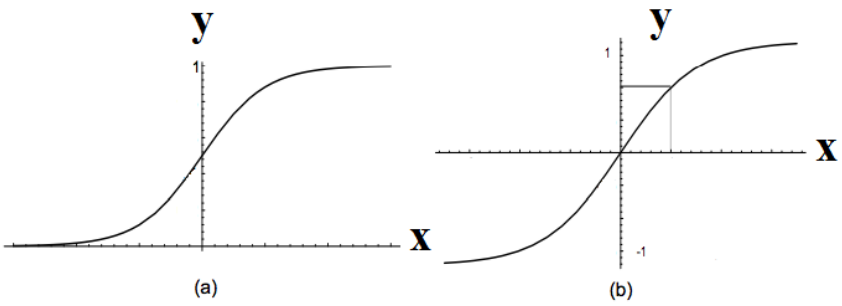


FIGURE 3.6 – Nonlinear activation function

The Binary Step Function with a threshold of θ reads :

$$f(x) = \begin{cases} 1 \; if \; x \geq \theta \\ 0 \; if \; x < \theta \end{cases} \tag{3.11}$$

Binary sigmoid function and its derivative can be written as :

$$f(x) = \frac{1}{1 + \exp(-\sigma x)} \tag{3.12}$$

$$f'(x) = \sigma f(x)[1 - f(x)] \tag{3.13}$$

While the bipolar sigmoid function and its derivation is expressed by :

$$g(x) = 2f(x) - 1 = \frac{1 - \exp(-\sigma x)}{1 + \exp(-\sigma x)} \tag{3.14}$$

$$g'(x) = \frac{\sigma}{2}[1 + g(x)][1 - g(x)] \tag{3.15}$$

Bipolar sigmoid is also closely related to another activation function which is the hyperbolic tangent function :

$$h(x) = tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} = \frac{1 - \exp(-2x)}{1 + \exp(-2x)} \tag{3.16}$$

A derivable non-linear activation function is required for a multi-layer neural network ; the purpose of this is to enable the use of gradient-descent method to training the network. The activation function most commonly used in BP networks is the sigmoid function which is the activation function used in this thesis.

3.7 Initializing the network's weights

It is acknowledged that there is no determined rules to define initial values of weights in a neural network. In practice, it is normally random values that serve as the initial weight values. The random distribution can be used to avoid the local minima [27] during the network training. Usually, small values are recommended for the initial weights, Because high values can cause saturation of the activation function even at the beginning of the network [34]. Thus, in this thesis, we chose to use random small value for initial weights.

3.8 Momentum

Although by increasing the learning rate, the convergence speeds during the network training can be improved. Increasing learning rate can also lead to the instability of the network training based on gradient-descent. The weight values may oscillate erratically on the space of errors. For the back propagation algorithms, the employment of a momentum term is used to speed up the convergence and avoiding instability during the training. The momentum is defined as the product of α [5] by the previous weight update. With momentum, the network training can be accelerated and it also avoided the oscillations of the weight[27]. Further introduction of momentum can be found in the next chapter where a Back-propagation Neural Network (BPNN) is used to show the effects of momentum during the network learning. In this thesis, we used the momentum term in our neural network model, the value of α by default is 0.9.

5. The value of α is defined between0to 1.

4

The Back-Propagation algorithm

4.1 Back-propagation (BP) algorithm

In this chapter, we will focus on the principle of back propagation algorithm (BP).

In 1974, Paul Werbos firstly presented in his work [36] how to make learning algorithm for a network (ANN can be considered as a special network). Unfortunately, the neural network community haven't payed enough attention to his work at that epoch. In the mid 80's, Back-propagation (BP) algorithm was invented by David Rumelhart[23] and Yann LeCun [33] independently. It rapidly gained a wide attention by the whole AI community and finally led the ANN research into a second booming. BP algorithm can be considered as un improvement of Delta rules. It requires that each artificial neurons in the network uses the activation function which must be derivative. BP algorithm has been proved to be very powerful in training of Feed Forward Neural Networks (FFNN).
A FFNN using BP training algorithm is usually called as "Back Propagation Neural Network (BPNN)." In the following, the term "BPNN" will be used for this type of neural network.
Here, we take a three-layered BPNN as an example, a typical network structure is presented here in Fig.4.1 below.
For a given neuron j in the output layer, we have its output error equals to :

$$e_j(n) = d_j(n) - y_j(n) \tag{4.1}$$

where $d_j(n)$ represents the target value of the network output from neuron j while $y_j(n)$ is the real network output on neurone j. So, we have the global error for the whole output layer :

$$E_{(n)} = \frac{1}{2} \sum_{j \in C} {e_j}^2(n) \tag{4.2}$$

where C is the collection of all the neurons in the output layer, the average error

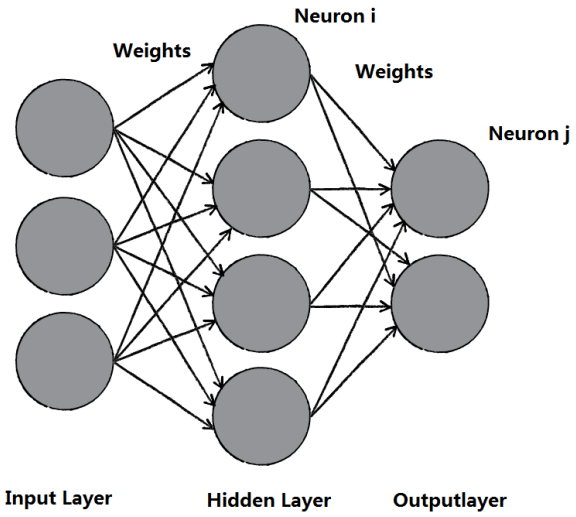


FIGURE 4.1 – Three layered Back-propagation Neural Network (BPNN)

is :

$$E_j(n) = \frac{1}{N} \sum_{n=1}^{N} E_{(n)} \tag{4.3}$$

the input of neuron j can be expressed as :

$$v_j(n) = \sum_{n} w_{ji}(n) y_i(n) \tag{4.4}$$

now, we have the output value from neuron j is :

$$y_j(n) = \sigma(v_j(n)) \tag{4.5}$$

σ here is the activation function.
As the Delta Rules, BP algorithm tries to minimizer the error function E during the training of the network by updating the value in the weights matrix, the partial derivation of E by the weight w_{ij} is :

$$\frac{\partial E(n)}{\partial w_{ij}(n)} = \frac{\partial E(n)}{\partial e_j(n)} \cdot \frac{\partial e_j(n)}{\partial y_j(n)} \cdot \frac{\partial e_j(n)}{\partial y_j(n)} \cdot \frac{\partial v_j(n)}{\partial w_{ji}(n)} \tag{4.6}$$

In this equation above, we have :

$$\frac{\partial E(n)}{\partial e_j(n)} = e_j(n) \tag{4.7}$$

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1 \tag{4.8}$$

$$\frac{\partial y_i(n)}{\partial v_j(n)} = \sigma'(v_j(n)) \tag{4.9}$$

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n) \tag{4.10}$$

Thus, we can rewrite the Eq. (4.6) uder the following form :

$$\frac{\partial E(n)}{\partial w_{ij}(n)} = -e_j(n)\sigma'(v_j(n))y_i(n) \tag{4.11}$$

by consequence, the weights are updated by :

$$\Delta w_{ji}(n) = -\eta \frac{\partial E(n)}{\partial w_{ij}(n)} = \eta e_j(n)\sigma'(v_j(n))y_i(n) \tag{4.12}$$

where η is the learning rate and the local gradient is defined as :

$$\delta_j(n) = \frac{\partial E(n)}{\partial v_j(n)} = \frac{\partial E(n)}{\partial e_j(n)} \cdot \frac{\partial e_j(n)}{\partial y_j(n)} \cdot \frac{\partial y_j(n)}{\partial v_j(n)} = e_j(n)\sigma'(v_j(n)). \tag{4.13}$$

Thus, we have the final form of weights' update (Eq. (4.12)) :

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n) \tag{4.14}$$

As neuron j belongs to the output layer of the network, the local gradient $\delta_j(n)$ can be easily calculated according to Eq. (4.13). However, if neuron j belongs to one of the hidden layer, it can be difficult to obtain its local gradient, because there is no error value $e(n)$ for the hidden layer neurons. A simple model is established in Fig. 4.2, the neuron j is now in the hidden layer while le neurone k is in the output layer. In this case, the local gradient can be put as :

$$\delta_j(n) = -\frac{\partial E(n)}{\partial v_j(n)} = \frac{\partial E(n)}{\partial y_j(n)} \cdot \sigma'(v_j(n)). \tag{4.15}$$

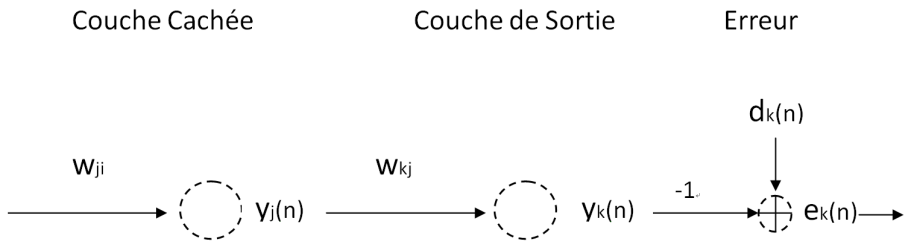


FIGURE 4.2 – Signal path when neuron j is in the hidden layer

The error function now becomes :

$$E_{(n)} = \frac{1}{2} \sum_{k \in C} {e_k}^2(n) \tag{4.16}$$

where C the collections of all output neurons in the network.

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k(n) \frac{\partial e_k(n)}{\partial y_j(n)} = \sum_k e_k(n) \frac{\partial e_k(n)}{\partial v_k(n)} \cdot \frac{\partial v_k(n)}{\partial y_j(n)} \tag{4.17}$$

Here, $e_k(n)$ equals to :

$$e_k(n) = d_k(n) - y_k(n) = d_k(n) - \sigma(v_k(n)) \tag{4.18}$$

So, we have :

$$e_k(n) = d_k(n) - y_k(n) = d_k(n) - \sigma(v_k(n)) \tag{4.19}$$

$$\frac{\partial e_k(n)}{\partial v_k(n)} = -\sigma'(v_k(n)) \tag{4.20}$$

For the neuron K, we have :

$$v_k(n) = \sum_{j=0}^{m} w_{kj}(n) y_j(n) \tag{4.21}$$

Where m is the number of neuron in the hidden layer. So the derivation of $v_k(n)$ by $y_j(n)$ is :

$$\frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n) \tag{4.22}$$

Thus, Eq. (4.17) can be rewritten as :

$$\frac{\partial E(n)}{\partial y_j(n)} = -\sum_k e_k(n)\sigma'(v_k(n))w_{kj}(n) = -\sum_k \delta_k(n) w_{kj}(n) \tag{4.23}$$

and we have :

$$\delta_k(n) = e_k(n)\sigma'(v_k(n)) \tag{4.24}$$

Applying the above equations to Eq. (4.15), we have :

$$\delta_j(n) = \sigma'(v_j(n)) \sum_k \delta_k(n) w_{kj}(n) \tag{4.25}$$

Now we have the local gradient of neuron j in the hidden layer. The mathematical procedure for calculating δ_j is represented by a graphical model in Fig. 4.3 below.
Now, as we already have the local gradient of neuron j in the hidden layer, we can calculate all the updated weight for all the neurons in the hidden layer by reapplying Eq. (4.14). the algorithm presented above is what we called Back Propagation.

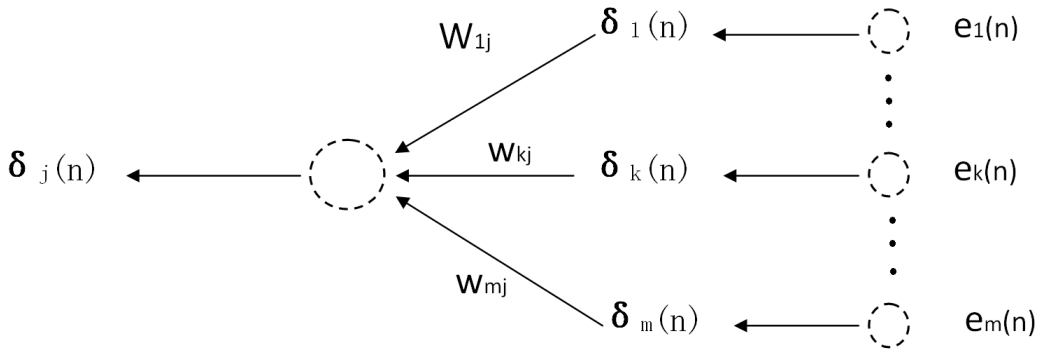


FIGURE 4.3 – Calculation of the local gradient of a neuron in the hidden layer

4.2 Practical aspects of the Back-propagation algorithm and Momentum

The back- propagation algorithm presented in this chapter only requires the weights' updates proportional to the derivation of the error functions (The gradient algorithm). The learning rate simply defines how much the weight changes on each training epoch. Therefore, swing weight is often caused by big learning rate. In practice, the best way is to use a reasonable value of learning rate without causing oscillation. As we explained in Chapter 3, a slight modification of BP algorithm is made that the previous weight update should influence the current weight update by using the Momentum.
With the momentum, once the weight starts to move in a particular direction, they tend to keep going in this direction. If there is a local minima presents, with enough momentum, the weights update can drive through the minima and stay in the right direction. Also, it helps to improve the speed of convergence during the training.
The mathematical expression of momentum is :

$$\Delta w_{ji}(n) = \alpha w_{ji}(n-1) + \eta \delta_j(n) y_i(n) \tag{4.26}$$

where α is a positive value from 0 to 1, et η is the learning rate. Thus, we can rewrite Eq. (4.26) as follows :

$$\Delta w_{ji}(n) - \alpha w_{ji}(n-1) = \eta \delta_j(n) y_i(n) \tag{4.27}$$

$$\Delta w_{ji}(n-1) - \alpha w_{ji}(n-2) = \eta \delta_j(n-1) y_i(n-1) \tag{4.28}$$

Finally we have :

$$\Delta w_{ji}(n) = \eta \sum_{t=0}^{n} \alpha^{n-t} \delta_j(t) y_i(t) \tag{4.29}$$

$$\Delta w_{ji}(n) = -\eta \sum_{t=0}^{n} \alpha^{n-t} \frac{\partial E(t)}{\partial w_{ji}(t)} \tag{4.30}$$

So, The function of momentum can be well explained by Eq. (4.30) :

– if $\frac{\partial E(t)}{\partial w_{ij}(t)}$ has the same sign for all t, $|\Delta w_{ji}|$ grows, and it leads to a faster convergence.

– if $\frac{\partial E(t)}{\partial w_{ij}(t)}$ alternates its sign in every iteration of training, $|\Delta w_{ji}|$ changes in small scale, which can lead to better stability.

The detailed description given in this chapter is necessary and essential to the further application of ANN in modeling with WSN data. In Chapter 5, a new training method for building a more comprehensive ANN model is proposed which is based on the Back-propagation.

4.2.1 Procedure of network learning with Back-propagation

While applying the BP algorithm in Neural Network training, there are basically four phases : first, a case of training data is submitted through the network in a forward direction and finally become the output of the network. Secondly, errors are calculated based on the difference between of network output value and the target value, the weight updates of the output layer neurons are made at this point. Thirdly, the weights update propagate backwards to the preceding layer neurons. In this way, layer by layer, all the weight changes are calculated for the whole network. Finally, these calculated weight updates are implemented throughout the network. The next training iteration begins, and the entire training procedure is repeated.

5

Overview of the WSN technology

5.1 Wireless Sensor Networks (WSN)

December 18,2011, the "New York Times" published an article in which they presented the development of data monitoring and improvements in sensor technology. The conclusion of this article is that the Internet will gradually departing from the scope of the consumer market and transfering to the physical world, which inevitably leads to the new era of "Internet of things".
The Wireless Sensor Networks (WSN) has been marked as one of the most revolutionary technologies ever since it was born. With great long-term economic and scientific potential, ability to transform our lives and bring new challenges in the future. A WSN consisting of autonomous sensor nodes which can provide a rich stream of sensor data representing physical measurements. They are deployed in many applications which rely on sensors to get the necessary information. In this thesis, an overview of wireless sensor networks as well as a systematic introduction of our developed WSN platform is presented.

5.2 A short history

The development of Network and its communication protocols can be traced back to the mid-1970s, since then, network communications has radically changed our lives [37].

- Ethernet
  The Ethernet was developed in the mid 1970s by Xerox, DEC, and Intel, and was standardized in 1979. the official Ethernet standard IEEE 802.3 is released by "Institute of Electrical and Electronics Engineers" (IEEE) in 1983. Data frames using the standard IEEE 802.3 have a variable length between 64 and 1514-byte per packet.

- Client-Server Network
  in the late 1980's C-S networks growth with the popularization of Personal

computers. Software designed for this kind of distributed computing network is separated into two different functionalities : Client-Server or we can call "front end to back end".

- P2P network and computing
  P2P refers to Peer to Peer, it is a kind of architecture that all the units in the network share the same functionality. It uses simply bus topology to transfer files and change data. The mechanism of P2P computing is that it splits the computing tasks into pieces and spreads through the network. The results are then gathered for further processing. P2P computing has enlightened many internet applications since the 90's.

- 802.11 Wireless Local Area Network. As known as the WLAN, IEEE released its standard (802.11 specification) in 1997, The current version is 802.11b which support a transmission rate up to $11Mbit/s$, The WiFi that we use in our daily lives for the PC, laptop, smart-phones is just based on this standard.

- Bluetooth
  Bluetooth is standardized by the specification IEEE 802.15 ; it is defined as Wireless Personal Area Network (WPAN). It is a short range RF technology which provide wireless communication for electronic devices in a nominal range of 10m to 100m. It allows new devices to be hooked up easily to the network. Bluetooth uses $2.4GHz$ band with a transmission rate up to $1Mbit/s$.

With the investments made in the 1970's Defense Advanced Research Projects Agency-USA (DARPA) began the Distributed sensor Network (DSN) program in 1980. Since birth of DSN, Universities like Carnegie Mellon and Lincoln Labs of the Massachusetts Institute of Technology (MIT) have accelerated the research in this field.

Sensor networks have been deployed for monitoring applications such as forest fires detection, air quality qualification, weather stations and structural monitoring. Later, IBM and Bell Labs began to implement sensor networks in heavy industrial applications such as the distribution of energy, waste water processing and factory automation. However, at that time, the sensors were bulky and yet very expensive. They were difficult to meet the requirement of large-scale usage. The high cost of materials had prevented the sensor networks in wider applications.
Although sensor network for industrial and high-volume consumer applications was not possible during that period, both academic and industry have long recognized the potential of these networks

- UCLA, Wireless Integrated Network Sensors (1993)

- University of California, Berkeley, Program of PicoRadio (1999)

- MIT, Multi-field Sensor Adaptive Power Aware program (2000)
- NASA, Sensor Webs(2001)
- ZigBee Alliance (2002)

The aim of these initiatives and trials is to allow wide-range deployment of sensor networks in industrial applications by reducing the fabrication cost, minimizing the energy consumption of sensors, etc.

5.3 Applications of sensor network

The core of any WSN is sensors. Rapid improvements have been made on sensors in the last decade :

- The micro-electromechanical systems (MEMS) - gyroscopes, accelerometers, magnetometers, pressure sensors, sensors pyroelectric effect, acoustic sensors.
- CMOS-based sensors - temperature, humidity, capacitive proximity, the chemical composition.
- LED and photovoltaic sensors - detect ambient light, proximity detection, chemical composition

These sensors with the wireless networks can be used in a wide variety of monitoring applications [38], including the following :

- The environmental condition
- Geographic detection and analyze
- Noise detection
- The presence or absence of certain types of objects
- Levels of mechanical constraint on connected devices
- Speed, direction, acceleration,etc.

The concept of combining micro-sensing devices and wireless communications lead to many new application areas. In this thesis, some existing applications in military, environment, health will be introduced below

5.3.1 Military applications

The wireless sensor networks can be an integral part of the military command, control systems, communications, computers, intelligence, surveillance, reconnaissance and targeting (C4ISRT). Since WSN are usually constructed with self-recovery and auto-routing communication protocols, the destruction of some nodes does not affect a military operation, making sensor networks a better approach for battlefields. Some other the military applications of WSN are monitoring friendly force ; battle field monitoring, recognition of opposing forces ; targeting and detection of nuclear biological and chemical (NBC) attack, etc.

5.3.2 Environmental applications

Some environmental applications of sensor networks include tracking the movements of small animals and insects, and monitoring of environmental conditions, large-scale terrain monitoring and planetary exploration, precision agriculture, biological, and environmental monitoring in the marine environment, soil and atmospheric monitoring, detection of forest fire, meteorological research, geophysical detection and the study of the pollution [39, 40, 41, 42, 43, 44, 45, 46, 47, 48].

5.3.3 Health applications

Some health applications for sensor networks are : interfaces for people with disabilities, integrated patient monitoring, diagnostics, drug delivery in hospitals, monitoring of human physiological data, tracking and monitoring doctors and patients inside the hospital [41, 47, 49, 48, 50].
The physiological data collected by WSN physiological data can be stored for a long period of time [51] and can be used for medical exploration [52]. Sensor networks installed can also monitor and detect the behavior of the elders, for example, a fall or a heart attack [53, 54]. These small sensor nodes can be easily equipped on the patients, giving physicians the possibility to identifier symptoms remotely[55]. For example, A WSN based "Health Smart Home" is designed at the Faculty of Medicine of Grenoble France. [49].

5.3.4 Home applications

Nodes and actuators containing intelligent sensors can be deployed in appliances such as vacuum cleaners, microwave ovens, refrigerators, etc. [56]. These sensor nodes within the domestic devices may interact with each other and with the external network through the Internet. They enable end-users to manage devices locally or remotely.

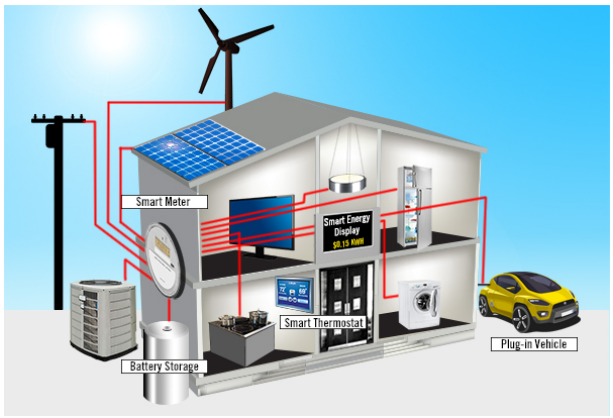


FIGURE 5.1 – "Smart House" a energy efficient solution proposed by MicroChip.

5.3.5 Other commercialized applications

Some other commercial applications of WSN are monitoring fatigue material ; intelligent environment for home and office, inventory management, product quality monitoring, thermal or humidity control in office buildings, automatic production line, interactive toys, smart structures with sensor nodes embedded, the transportation, detection and monitoring of car theft, vehicle tracking and detection, machinery, etc. [39, 41, 43, 57, 58]

5.4 WSN topologies

The most commonly used topologies for WSN will be discussed in this section. They are Peer to Peer (also called point to point), star, tree and mesh.
Peer-to-Peer network enables each node to communicate directly with another node without having to go through a central communication center. Each peer device is capable of operating both as a "client" and "server". They hold the same responsibility in the WSN.
Star networks are connected to a central communication center for information exchanges. For each node in the WSN, all communications have to be routed through the center. This is a basic Client-Server network structure.
Tree networks has a communication center called root node which holds the highest level of the network. The information from the lowest level nodes have to get trough all the higher level nodes to reach its destination. The hierarchy of the network tree can be understood as a hybrid of star and Peer to Peer network topologies.
Mesh network allows data to "hop" from node to node through what we call the routing mechanism. Thus the network can be self-healing and self adaptive. Each node can communicate with the other as data is passed from node to node until it reaches the desired destination. Mesh network is the most used network topology for WSN with a large-scale coverage of geographic area.

An example of these four different topologies is shown in Fig.5.2

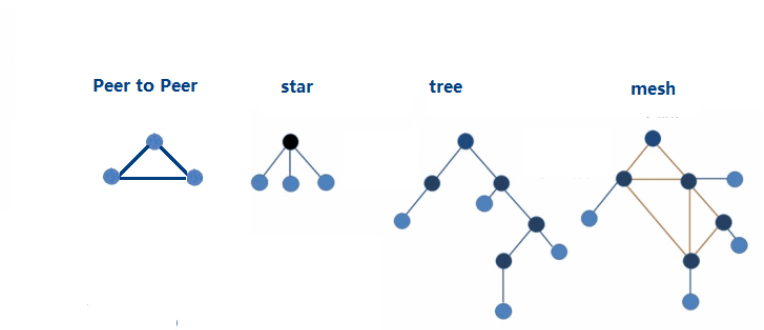


FIGURE 5.2 – Four different topologies of WSN

5.5 The Low power wireless technologies : A comparison

As presented above, Wireless technology can bring great benefits for a vast domains of applications. However, due to the variety of different technologies and specifications, the choice of technology for a specific usage should be carefully discussed. In this section, the advantages and disadvantages of different technologies are presented.

The three most competitive wireless technologies are : Bluetooth, Wi-Fi and ZigBee.

- Bluetooth low energy (BLE) is created by a project in the Nokia Research Center with the name Wibree. In 2007, the technology was taken by the Bluetooth Special Interest Group (SIG) with a new name of "Ultra Low Power Bluetooth or Bluetooth low energy". This technology establishes network with star topology. It is primarily for the usage in mobile phones. Similar to Bluetooth, it is often used betweens the phone and other smart devices.

- ZigBee is a low power wireless technology based on the specification of IEEE 802.15.4. The Zigbee alliance was created in 2002 by a group of 16 companies. It uses mesh networks with low power devices. The aimed applications are monitoring, automation and remote controls.

– "Wi-Fi" stands for "Wireless Fidelity" is a registered trademark of the Wi-Fi Alliance and the brand name for wireless technologies using the IEEE 802.11 specification family. The 802.11g wireless network now supports a transmission rate up to $54Mbps$.

Let's compare the three technologies in Fig.5.3 below.
Thus, we can summarize the functionality and advantages of ZigBee :

Specification	ZigBee IEEE(802.15.4)	Bluetooth (802.15.1)	WI-FI (802.11)
Range	50-400m	10m	50m
Network extension	Auto extend	NO	NO
Battery life	4-6 months/years	Days	Hours
Complexity	Simple	Complex	Very Complex
Transmission Rate	250kbps	1Mbps	1/11Mbps
Operation frequency	868MHz(EU)/2.4GHz	2.4GHz	2.4GHz
Number of Devices/Net	65000	8	50
Network join time	30ms	10s	3s
Equipments cost	Low	Low	High
Connecting Charge	Free	Free	Free/Paying
Security	128 bit AES	64/128bit	SSID
Reliability	High	High	Normal
Use cost	Low	Low	Normal
Installation	Easy	Normal	Difficult

FIGURE 5.3 – Une comparaison entre Zigbee, Bluetooth et WIFI.

– Huge network capacity with large scale network coverage

– Low cost

– High reliability and security in data transmission

– Small and energy efficient terminal devices

Except for Bluetooth, WiFi, there has been some recent wireless technologies like radio frequency for Consumer Electronics (RF4CE), Nike+, IrDA, ANT etc. Before further discussion on these different technologies, a basic concept should be clarified here : all these low-power wireless networks can be divided in to two categories according to their different functionality and features : Local Area Net-

works (LAN) and Personal Area Networks (PAN). ZigBee can cover a large scale of space, thus it is more like a low-power LAN technology, while BLE can cover a shorter distance, so it is proper as a PAN. For example, ZigBee will be used to cover your home and monitor the environmental condition outside the house while BLE will be used to connect your electronic devices inside your house. As in this thesis research, the WSN is used to monitor the whole building space, it is a typical LAN application. Thus, most of the previously mentioned PAN technologies are excluded from our choice.
Fortunately, just before this thesis, the RF4CE Alliance (Panasonic, Philips, Samsung, Sony) has gathered their best *RnD* engineers in a meeting to build a new wireless standard for Remote Control. They evaluated all possible wireless technologies (including low power Wi-Fi and BLE, Wibree) and came to the conclusion that IEEE 802.15.4 best matches the requirements and they agree on the construction of an application profile on top of it. Later, RF4CE Alliance itself is adopted by ZigBee standard (Because they are both constructed on the same IEEE 802.15.4 specification) for reasons of maintenance.

Based on the discussions above, we find the ZigBee adapts the best with our aimed applications. So to summarize, ZigBee and other low-energy technologies are complementary. In general, ZigBee is a LAN technology, it has the ability to essentially support an huge number of nodes in a mesh network structure.

6

Specification and Protocol

6.1 IEEE 802.15.4 and ZigBee

As shown in the previous chapter, ZigBee is considered as the most appropriate technical standard in the development of our WSN. In this chapter, we will give a detailed introduction on the ZigBee technology.
ZigBee is a combination of two parts : the IEEE 802.15.4 specification defines its physical and MAC layers as low rate wireless networks and the standard. ZigBee Alliance has developed general standards for the network layer (NWK) to the application layer (APL) (see Fig. 6.1).

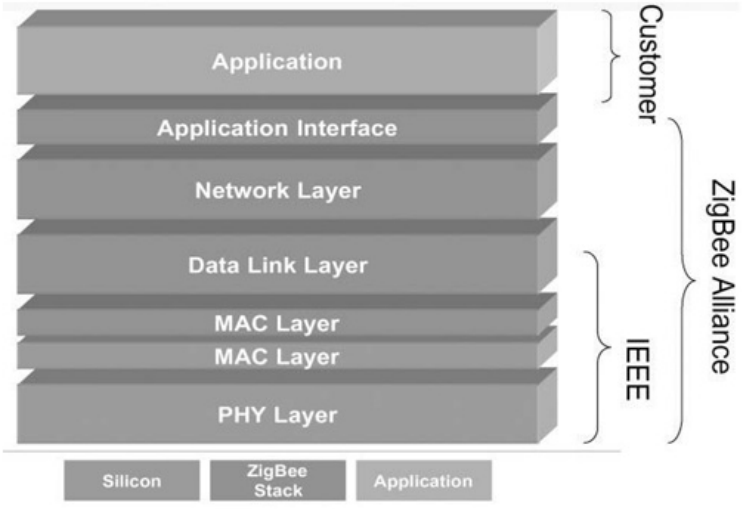


FIGURE 6.1 – The stratification of the ZigBee

6.2 IEEE 802.15.4 specification

IEEE 802.15.4 is a low-power wireless specification for the applications with low data rate [59]. It has a maximum data rate of $250,000\ bits/s$ and a maximum output power of $1mW$. It enables low cost transmitters low complexity and it specifies two layers :

- Physic layer : specifies how messages are transmitted and received on the support of the physical wireless signals.

- Media Access Control (MAC) : Specifies how messages from the physical layer are handled.

The maximum packet size in IEEE 802.15.4 is $133bytes$. The packets are small because IEEE802.15.4 is intended for devices with low data rate. Because the MAC layer adds a header of each packet, the amount of data available for an application protocol in the upper application layer is between 86 and 116 bytes. Protocols of the upper layer.
For the IEEE 802.15.4 compatible devices, the physical layer and part of the low-level MAC processing is implemented in hardware while the high-level logic parts of the MAC layer is integrated in the software.
Although the IEEE 802.15.4 specification defines three types of topologies (star, mesh, and cluster tree), most protocols that run on top of it does not use its topologies. Instead, they build their own network topologies on top of the MAC layer. For this reason, the topology of network will be presented in the part of ZigBee.

6.2.1 Network addresses of IEEE 802.15.4

Each node in an IEEE 802.15.4 network has a 64-bit address which is normally unique and can be used to identify the device. Due to the limited size of the IEEE 802.15.4 packet, the length of the 64-bit address is prohibitive. Therefore, 802.15.4 allows nodes to use a shorter address which is 16 bits long. Short addresses are assigned by the PAN coordinator (highest hierarchy in the network) and are valid only when the network is activated. Nevertheless, it is possible for a device with a short address to communicate with other devices.

6.2.2 The physical layer of IEEE 802.15.4

The physical layer determines the radio frequency and the radio band to operates, the modulation of the radio, and coding of the RF signal. IEEE802.15.4 operates on three frequency bands. Due to local regulations, the exact frequency is different in different parts of the world. In the United States, IEEE 802.15.4 uses the 902-928 MHz band. In Europe, 802.15.4 uses the 868 to 868.8 MHz band,

2400-2483.5MHz band is globally available.
IEEE 802.15.4 defines 26 different operating channels. Within each operating radio band, there are several defined channels, as shown in Fig.6.2. Channel 0 in Europe is located on the 868 MHz band. Channels 1 to 10 are defined in the United States on the 902-982 MHz band and the spacing between channels is 2MHz. Channels

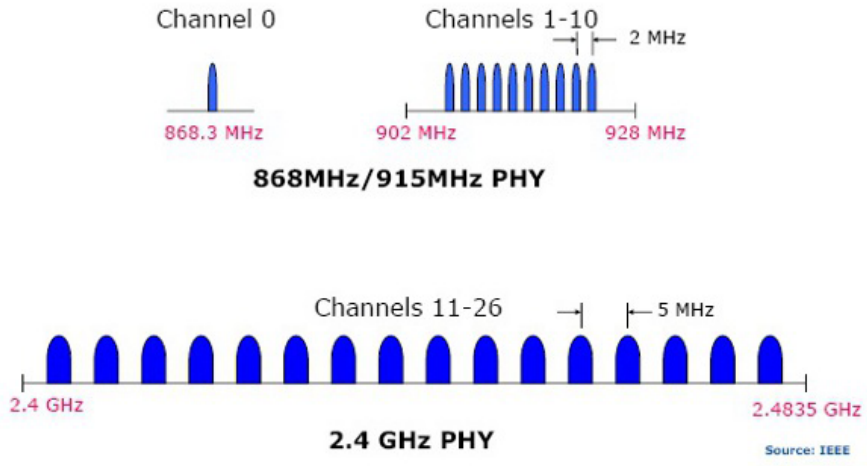


FIGURE 6.2 – Channel assignment of IEEE802.15.4 (provided by IEEE)

11 to 26 are defined on the 2.4 GHz band, and they are available globally. These channels are defined with a channel spacing of 5 MHz (see Fig.6.2).

IEEE 802.15.4 uses two types of modulations depending on different channels. Channels 0-10 use binary phase shift keying (BPSK), while the channels 11-26 use quadrature phase shift keying (QPSK). For all channels, the direct sequence spread modulation Spectrum (DSSS) is applied (see Tab. 6.1).
The physical layer also provides functions to measure the RF energy for a given

TABLE 6.1 – Different bands of IEEE 802.15.4 IEEE 802.15.4

	2450MHz	915MHz	868MHz
Data rate	250kbps	40kbps	20kbps
No. of channels	16	10	1
Modulation	O-QPSK	BPSK	BPSK
Pseudo noise sequences	32	15	15
Bit per symbol	4	1	1

radio channel. It is used by the MAC layer to determine whether another node is transmitting in this channel or not.

6.2.3 The MAC layer of IEEE 802.15.4

The purpose of the MAC layer is to control access through the radio. Since the medium of the radio is shared by all the node in the network, the MAC layer provides method for nodes to find out when the medium is available and when it is ready to send RF signals.
The IEEE 802.15.4 MAC protocol supports both Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA) and Time division Multiple Access (TDMA) simultaneously. It is depending on the requirements of applications, Therefore, IEEE 802.15.4 leaves the flexibility to the applications layers to decide which of these two technologies should be used.
The MAC layer performs the validation of incoming frames by making a Cyclic Redundancy Check of 16 bits (CRC) of the entire frame. The CRC is used to check for transmission errors in the frame and is calculated by the receiver, if it does not match the CRC in the footer, the receiver removes the frame.

6.2.4 The frame format of IEEE 802.15.4

IEEE 802.15.4 specify a universal packet format so that all nodes know how to build and analyze packets from other nodes. The formats consists of three parts (see Fig. 6.3) : a header, data, and a footer. The header contains control data, such as addresses, sequence numbers and flags. Data contains a part protocols from the physical layer and from the MAC layer. The footer usually contains checksums or cryptographic signatures. Such data can often be calculated while the packet is transmitted so, the footer is sent only after the rest of the package has been sent.

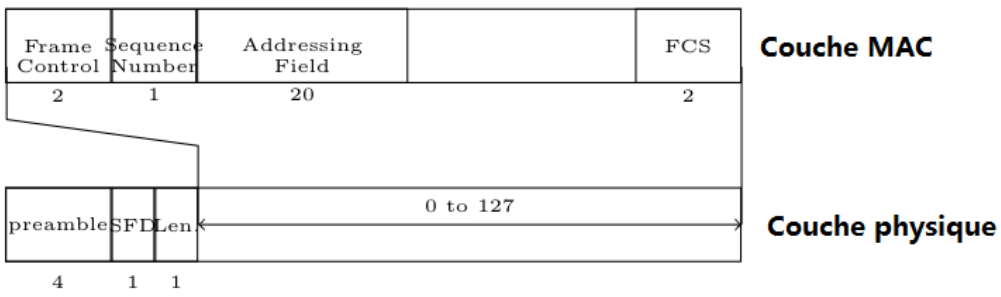


FIGURE 6.3 – Frame format of IEEE802.15.4 (source : IEEE)

6.2.5 Functionality of units in IEEE 802.15.4

The IEEE 802.15.4 MAC layer defines two types of nodes, the first is Reduced Function Devices (RFD), the second is the Full Function Devices (FFD), FFD is equipped with all the functions of the MAC layer, it can be used as coordinator of the network. The coordinator can broadcast tag signals to synchronize and process all the joint request from terminal devices. RFD usually act as terminal devices and they are usually integrated with sensors/actuators.

6.3 ZigBee

There are five versions of the ZigBee specificatin : ZigBee, ZigBee 2004, ZigBee 2006 Pro, ZigBee 2007 and ZigBee 2007 pro. As ZigBee ZigBee 2004 and 2006 are considered obsolete and are not used in new products. ZigBee 2007 is currently the most widely used version, and is often called simply "ZigBee." ZigBee 2007 adds a number of features that were not present in Zigbee 2006 as support packet fragmentation and the ability to dynamically switch channels. ZigBee Pro increased the number of devices in each network from 31,101 to a maximum 65,540 and adds a number of network functions such as multi-cast and source routing.

6.3.1 Types of devices and the network topology of ZigBee

ZigBee specifies three different type of device : coordinator, router and the terminal. The three devices have different roles in a ZigBee network. A ZigBee network can only have one Coordinator. It coordinates the actions of the network as a is responsible for starting the network. Data packets can be freely exchanged through routers to any units in the network. Correspondingly, the router and coordinator are defined as FFD in the IEEE 802.15.4 while the end devices are defined as RFD. They can only be connected to a router or coordinator and can not communicate directly with each other.
In the ZigBee protocol, define three types of topologies : a star network, a network tree and mesh(see Fig.6.4).

6.3.2 The stack ZigBee

ZigBee specification can be divided into five layers, as shown in Fig.6.5 : The physical layer and the MAC layer, as explained in the previous section, are the two bottom layers specified by IEEE 802.15.4. The upper layers are : the network layer (NWK) of the layer application support layer (APS), and the application layer frame (AF). In addition to five layers, There is a cross-layer entity called the ZigBee device object (ZDO) presented in the architecture. Of these layers, the NWK, APS, ZDO and AF four layers belong to the ZigBee specification, Normally, all costumer applications are designed and integrated in these four layers.

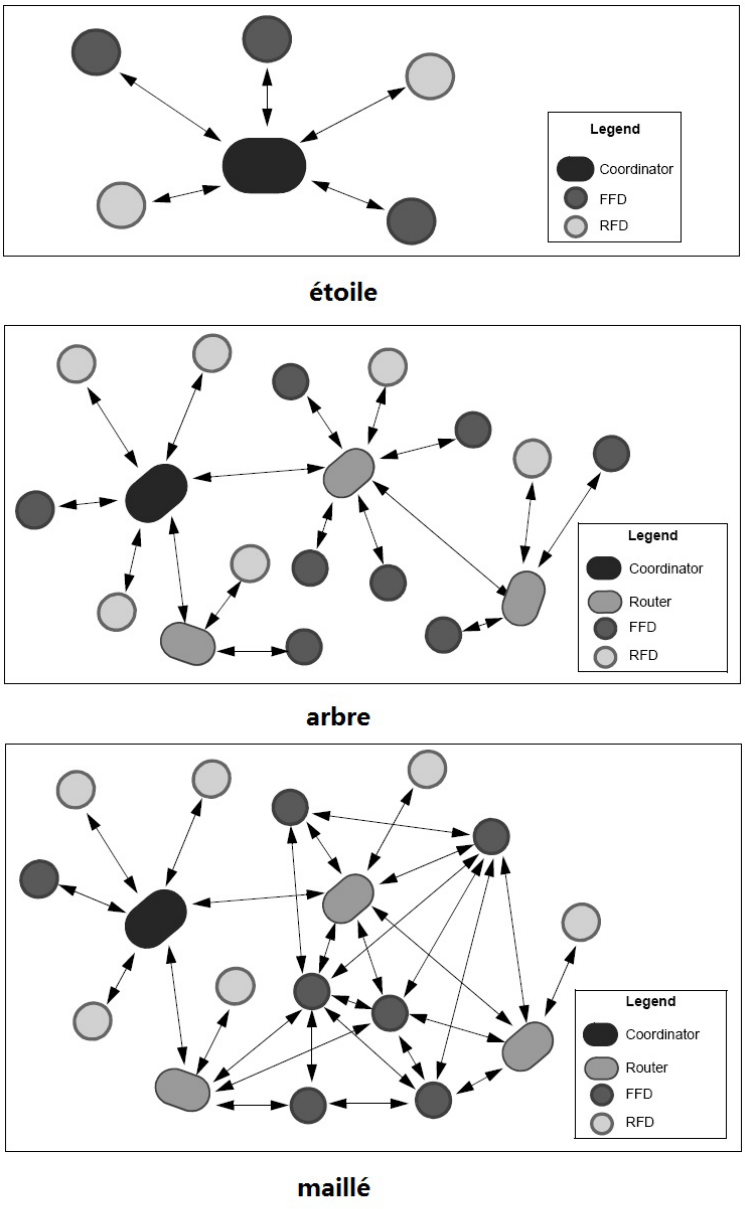


FIGURE 6.4 – Three different topologies ZigBee

The stratification of the ZigBee stack is very similar to IP(Ethernet) structure :

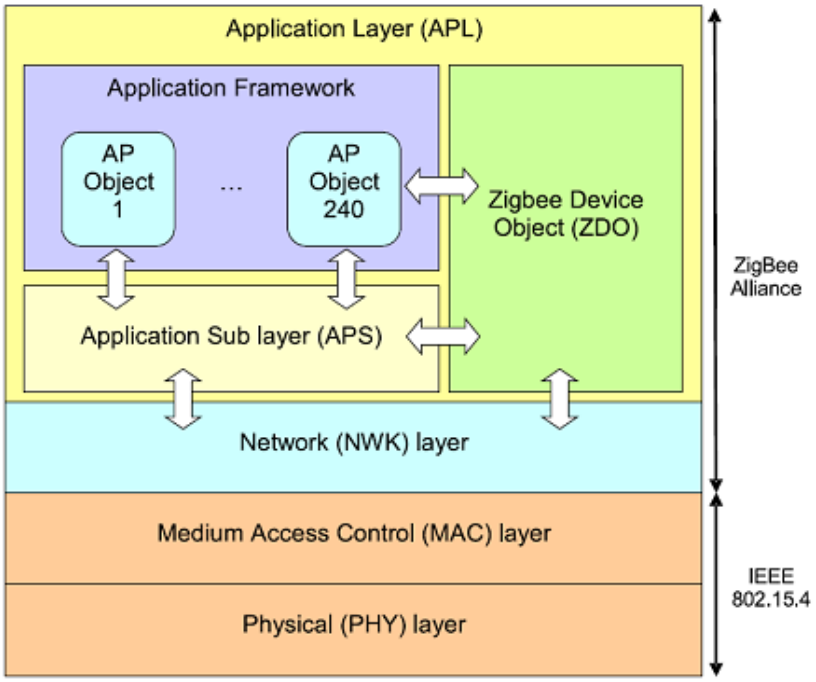


FIGURE 6.5 – Layers in ZigBee stack

each layer in the ZigBee has a specific purpose. The one major difference between the stratification in IP and in ZigBee is that the layer in ZigBee can not be changed. IP architecture is designed to allow multiple types of MAC and PHY layers. The same protocol can be used even if the operating radio band changes. However, the ZigBee specification is based on IEEE 802.15.4 MAC and PHY standard. The upper layer protocols will not be compatible if lower layers are changed.

Network layer (NWK) of ZigBee

The NWK layer is responsible for addressing and routing in the network. It is the equivalent of the IP layer in the IP architecture. The ZigBee network layer provides two ways of communication : broadcast and unicast, multicast is supported in ZigBee 2007 specification. Broadcast send the packet to each node in the network as an expensive operation. Unicast ,on the other hand, only sent packets to the target node.
The ZigBee stack has two possibilities of making the unicast routing : network routing and source routing. In network routing mode, the best routing for the packet to get through the network is handled by the network. By analyzing the link quality indications (LQI) in the network, it can determine the routing with the best

RF signal quality. In source routing, the sender must explicitly state the routing of the message to reach its destination. Source routing is useful for large networks. However, source routing is only available in the ZigBee Pro and it is limited to only five hops to reach the destination.

APS layer de ZigBee

The layer APS is equivalent to the transport layer in the IP architecture. It is a thin layer that acts as an intermediary between the NWK layer and the AF layer and it provides data transfer services between these two layers. APS layer also provides services for the establishment, and maintenance of security of the network.

AF layer de ZigBee

The AF layer of ZigBee runs on top of the APS layer. AF supports multiple tasks and applications. Some of these applications are defined by the ZigBee specification, while others are implemented independently by customers. In ZigBee, an application is called a profile.

The profiles of ZigBee are identified by an integer between 0 and 240. When the AF layer processes a packet, the assigned profile number of the packet is identified. If a packet arrives with a unregistered profile number, the packet is dropped. If the application has been registered, the packet is passed to the application layer. Profile identifiers are allocated and managed by the ZigBee Alliance (see Fig. 6.5).

6.3.3 Configuration of a network ZigBee

The installation process of ZigBee network involves all layers of the ZigBee stack. This process establishes a physical communication link between the nodes of the network, distributes address, discovers and gathers information of the nodes in the network, provides binding service.
The network installation process begins at the PHY layer. The coordinator starts by scanning the 16 IEEE 802.15.4 defined physical radio channel to find an available one.
Since IEEE 802.15.4 operates on the unlicensed 2.4 GHz band, there are several sources of interference such as Wi-Fi and microwave ovens etc. The channel scan samples per channel for 0.5s. Thus, the process takes only eight seconds and gives an overview of the channel activity. When the scan is complete, the coordinator selects the channel with the least network activity. This channel is retained by the lifetime of the network.
After selecting the channel to the PHY layer, the MAC layer creates a new PAN ID for the network. The PAN ID is a 16-bit integer randomly selected by the coordinator. When the physical channel and PAN ID were selected, network formation

is considered complete.
Once the coordinator has formed the network, routers and end devices begin to join. The node will request the join to the network by sending a beacon messages. If a router or coordinator hears a beacon from a node that is not part of a network , it responds by sending a beacon message back. The node collects all the responses it receives and decides which network and router, it should try to associate. If network security is enabled, after node selected a network node and a parent, it authenticates with the parent and the join in of the network is done.

7

Development of a practical WSN system

7.1 WSN system development

In the previous chapter, we gave a basic introduction to WSN and ZigBee. In this chapter, we will focus on the design and implementation of a practical WSN system. To design a hardware system of Zigbee based WSN, we have two main technical choices :

- Transceiver-only : Transceiver Modules only include the RF transceiver, They has no microprocessor and independent memory associated. The protocol stack is provided by an external integrated circuit. There are two main advantages : most transceiver Modules are low cost and it may allow the designer to choose any microprocessor they want. The disadvantages are : potentially, more work is needed to integrate the micro-controller unit (MCU) and system reliability is relatively lower than that of a ready-to-use integrated MCU solutions.

- Transceivers integrated MCU : It is a ready to use module with both MCU and transceiver plus antenna on a single printed circuit board. The ZigBee stack are stored in the flash memory of the MCU, Users and program the stack to add their own applications

In this thesis research, we developed three different WSN platforms. We made a detailed comparison between the three platforms which leads to our final choice. The first platform that we have developed is Transceiver only solution. The system is based on MRF24J40MA transceiver modules with Microchip PIC18LF4520 microcontroller, the other two platforms are Transceivers integrated MCU solutions. One is cored with MC13224 Freescale wireless microcontroller and the other is cored with Texas Instruments CC2530 microcontroller.

7.1.1 MRF24J40MA with PIC18LF4520 microcontroller

MRF24J40MA according to Micochip is an IEEE 802.15.4 compliable radio transceiver module. The MRF24J40MA has an integrated PCB antenna, matching circuitry, and supports ZigBee, MiWi and MiWi P2P protocols (MiWi and MiWi P2P are two simplified protocols based on IEEE 802.15.4 with light upper layer stack). The module MRF24J40MA connects to the PIC microcontroller via a SPI interface Fig. 7.1.
The reason for choosing MRF24J40MA at an early stage, is that this is a product with Microchip software support in Europe, the Microchip microcontroller is most commonly used by students specializing in EE, and we hope that it might be easier for students to be able to participate in this work later, after the thesis. MRF24J40MA is a single transceiver module, Wireless communication protocols are stacked in the flash memory of the PIC microcontroller. Our system design is shown in Fig. 7.2.

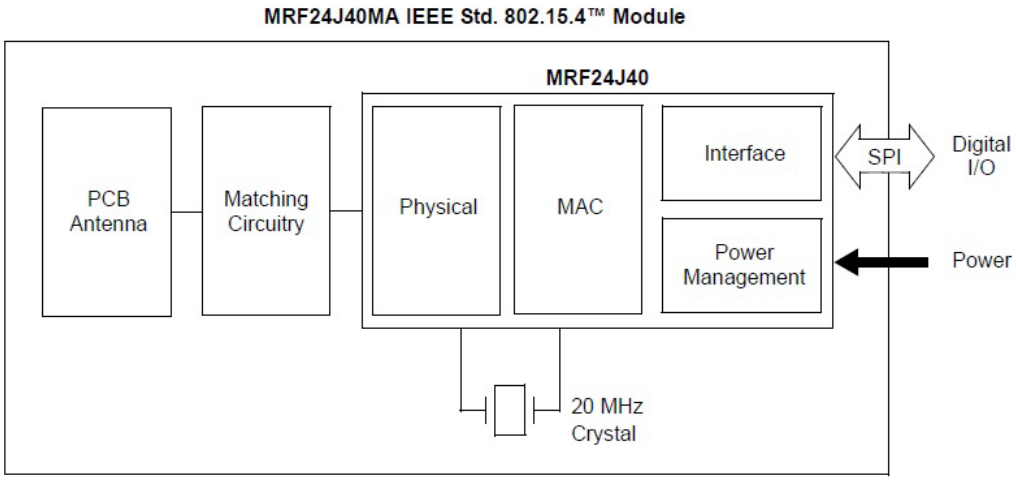


FIGURE 7.1 – Hardware diagram MRF24J40MA

In our developed WSN system, the MRF24J40MA module was assembled with a PIC18LF4520 microcontroller on a PICDEM Z board. These two products are both developed by Microchip. The PICDEM Z board provide a platform which is easier to connect the MRF24J40MA module to the microcontroller. It also provides tools to facilitate the development of the project like push buttons and sensors (thermometer, two knobs and a light sensor). Although PIC18LF4620 is the microcontroller that advised by Microchip to combine the MRF24J40MA Module. Based on the conception of low cost design, we had used the PIC18LF4520 microcontroller. The main difference between the PIC "4520" and "4620" is in the storage capacity of the flash memory. The PIC18LF4620 is more expensive for its bigger memory. In our case, they make no difference as the memory on both PIC is enough for the

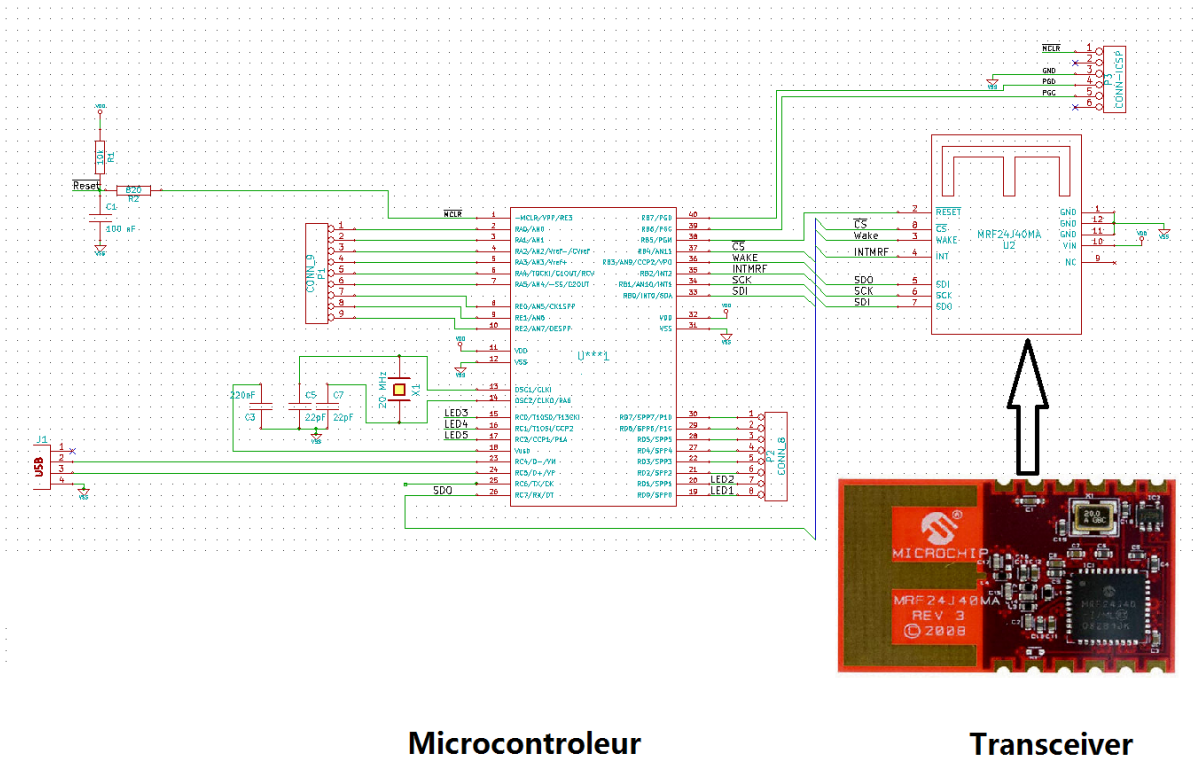


FIGURE 7.2 – WSN system design with MRF24J40MA

light wireless protocols.

The MiWi protocol

The MiWi protocol is a protocol used for the WSN system based on MRF24J40MA Transceiver module. MiWi is a simplified Zigbee protocol : the size of the protocol established in memory is smaller, thus, it is not a complete ZigBee specification. Since we have already presented the WSN configuration in the previous chapter on the IEEE 802.15.4 standard.
Indeed, the MiWi protocol provides three complementary ways to transfer information. A first possible delivery using a connection called " Socket". It is possible in a MiWi network to create a virtual connection between two nodes without the need to know the information for the other node. One node sends a connection request Socket to the Coordinator. If the coordinator receives another request in a lapse of time of about two seconds, then it establishes a connection between two nodes, which can then send messages even if initially they do not know the addresses of each other. The data transfers use the Short Address. This connection can be used to exchange the Long Address, for example. A node can establish a single socket connection at a time. It can also stop the connection by resetting the variables, or attempting to create a new connection.
It is theoretically possible to send messages using Network Address (Short Address). But we do not use it mainly because that the Short Address is changeable and it can not be used to identify different modules. The third method is based on the Long Address.
Each node knows its neighbors. If the message is not for one of his neighbors, the message is then sent to a parent node, which also verifies if the message is addressed to one of his neighbors. If this is not the case, the same operation continues until it reaches to the PAN Coordinator where all the nodes in the network will be checked for addresses.

The packets are composed of at least three bytes. Indeed, before the data payload, we place a header with additional information which allows users to specify what is sent(see Tab. 7.1) :

TABLE 7.1 – The data format defined in Wimi

Octet1	Octet2	Octet3	Octet4 to the end
Type	control	additional	data

During the networking experiments, we added three end-devices in the network. It turns out that our system is not stable enough. This problem was very recurrent and started upon connection : the node could not always find a network, even they

are very close to the coordinator. We calculated the success rate which is about 60%. Presumably, there is a conflict on the 2.4G channels with WiFi, but even without WiFi, it is not 100% certain to join the network, compared to the full size Zigbee protocol The size of the MiWi stack occupies 70% less space of memories. The stability problem can be related to the incomplete of the given wireless protocol. So this solution is finally dropped.

7.1.2 Platform based on Freescale MC13224

MC13224 is the third generation of Freescale ZigBee microcontroller. It incorporates a full range, low power, 2.4 GHz radio frequency transceiver, 32-bit ARM7-based MCU, hardware acceleration for both the IEEE 802.15.4 MAC and AES security and a full set of peripherals[60]. The diagram of MC13224 is shown below in Fig. 7.3 :
The features of MC13224 are :

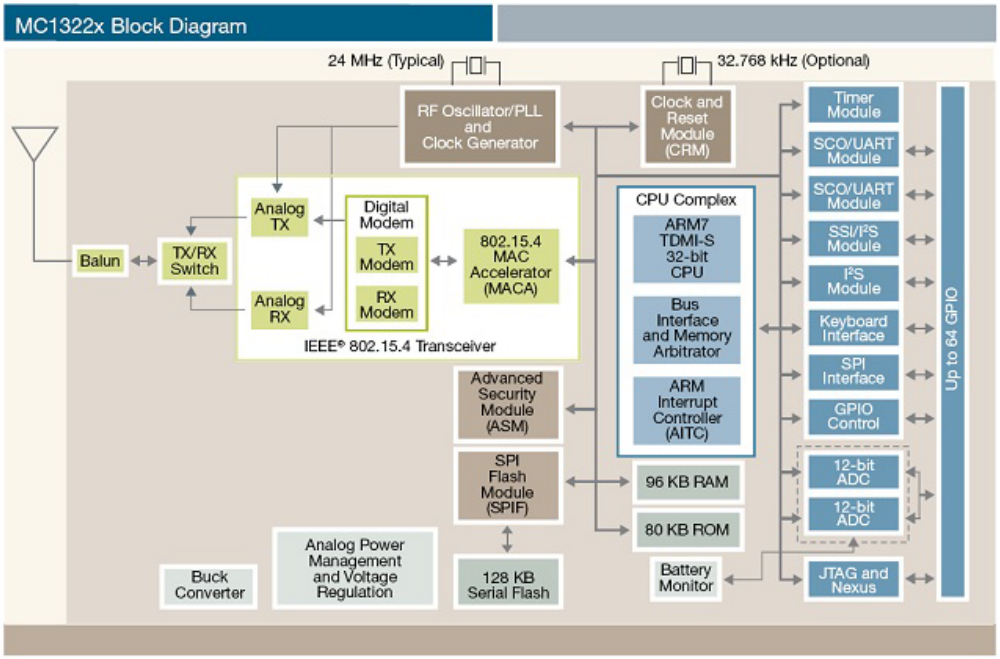


FIGURE 7.3 – Diagram of MC13224 (Source : Freescale)

– Low power : 21mA typical current consumption in RX mode with actif MCU and 29mA typical power consumption in TX mode with active MCU.

– Memory : 128 KB serial flash, 96 KB RAM (device operates from RAM) 80K ROM containing boot code, all device drivers and IEEE 802.15.4 compliant protocols.

– MAC accelerator (sequencer and a DMA interface).

– 128-bit AES hardware encryption/decryption with the random number generator.

– No external RF components required.

In our WSN system, the coordinator is powered by independent power sources. It can communicate with a PC or server via a USB interface. With integrated ZigBee (MC13224) module, wireless communication can be achieved through out the network. The coordinator has played a role of a manager and an expert, not only gathers data from nodes, but also is responsible for the management and maintenance of the system. The MC13224 ZigBee module core is presented in Fig. 7.4 et the node is presented in Fig. 7.5

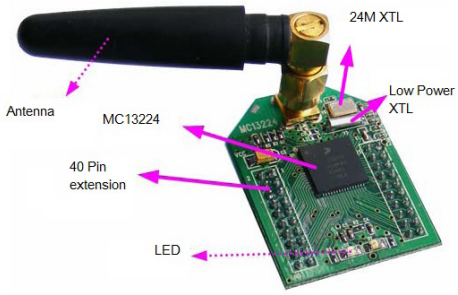


FIGURE 7.4 – The MC13224 core

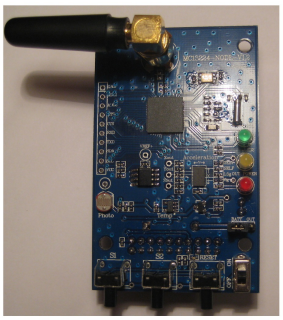


FIGURE 7.5 – The node with MC13224

BeeStack of Freescale

BeeStack is the embedded software provided by Freescale to its ZigBee microcontroller. It is based on the ZigBee stratification presented in the previous chapter. The BeeStack includes the following components :

- ZigBee Device Objects (ZDO) and ZigBee Device Profile (ZDP).
- Application Support sub-layer (APS).
- Application Framework (AF) layer.
- Network layer (NWK).
- Security Services Provider(SSP).
- IEEE 802.15.4-compliant PHY and MAC layers

The structure of the BeeStack is shown in Fig.7.6 The MC13224 cored WSN system

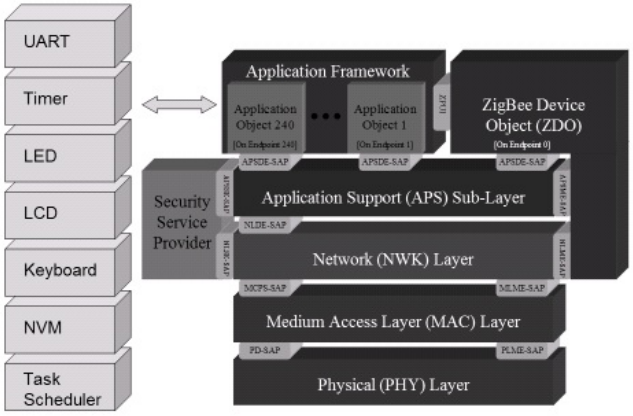


FIGURE 7.6 – La structure de BeeStack

boots itself as shown in Fig. 7.7. Application starts in $BeeAppInit()$. The functions of all application tasks are called during initialization. These commands include, for example, the hardware initialization and set up , initialize the table, and update notification function in power. The $BeeStackInit()$ can be found in the file BeeStackInit.c, its initialization is done by : $BeeAppInitempty(void)$; Our applications are programmed in *BeeApp.c*, where the acquisition is done by operating the sensor with its analog digital converter (ADC) function. When the application tasks are assigned, the system will enter a closed loop. Tasks are then processed according to their order in the task scheduler. An example of application code can

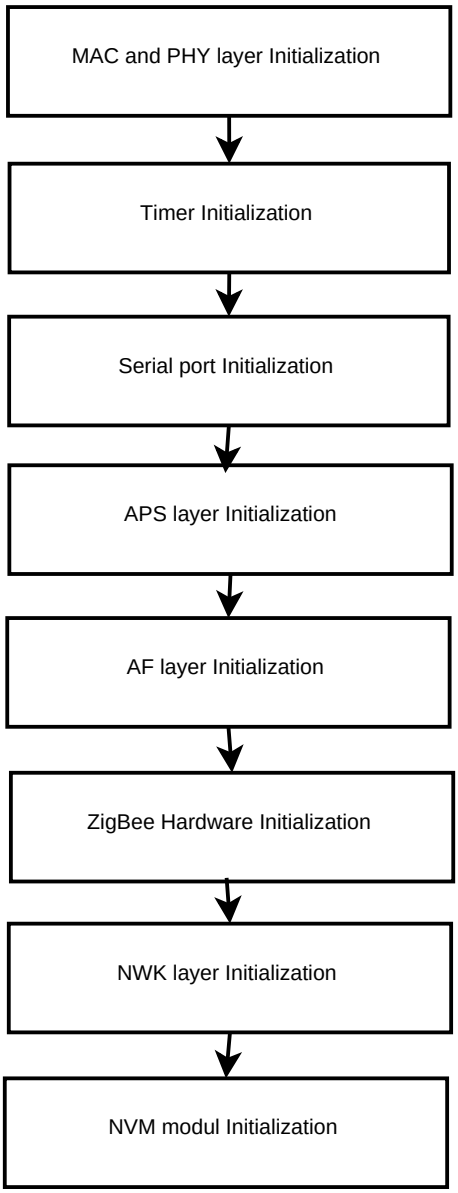


FIGURE 7.7 – initialization of a user-application in BeeStack

be found in the appendix of this thesis.
The MC13224 WSN solution was tested be experiments. Its stability is much better than MRF24J40MA based PIC system. However, there are two reasons that we gave up this solution : First, the MC13224 has only 80kb Read Only Memory (ROM) which will limit the future implementation of large application libraries. Secondly, the supports from manufacturers (Freescale) is very limited, when we run into problems of development, it is very difficult to get support.

7.1.3 WSN solution with cc2530 microcontroller from TI

Except the two main drawback MC13224 as we discussed in the previous section, a comparison between the MC13224 and CC2530 two most powerful ZigBee microcontroller is presented in Fig. 7.8.

Wireless SOC Solutions:	MC13224	CC2530
MCU	Single Chip ARM7 Core, 32bit	Single Chip 8051 Core, 8bit
RF front	IEEE802.15.4	IEEE802.15.4
Protocol	BeeStack ZigBee Pro (free)	Z-Stack ZigBee Pro (free)
Memory	128k	256k
Battery life	1 Year	1 year
Price per unit	4 dollars	3 dollars
IDE	IAR EWARM	IAR EW8051

FIGURE 7.8 – La comparaison entre MC13224 et CC2530

We find that though CC2530 is an 8-bit microcontroller with a improved 8051 core, it has a 256K flash memory and its price is lower. Moreover, what is not shown in this figure is that CC2530 has be vastly used by researchers and industries. This gives us the opportunity to get better support from the developer community, communication between the developers are very easy and direct on the forum of Texas Instruments.

The CC2530 from TI (Texas Instruments) is a true SoC (System on Chip) solution designed for IEEE 802.15.4 and ZigBee Smart Energy applications. Thanks

to its large capacity flash memory up to 256K, the CC2530 is ideal for WSN systems. In addition the CC2530 includes a transmitter fully integrated high performance RF receiver, a 8051 microcontroller, 8K RAM etc. The main features of the CC2530 are therefore :

– Excellent RF signal link quality (102 dBm).

– 49 dB adjacent channel rejection (Best in the same class).

– Four flexible power modes for reduced power consumption

– Powerful five-channel (Direct Memory Access)DMA

The diagram microcontroller CC2530 is shown in Fig. 7.9while the its PCB design is shown in Fig. 7.10.

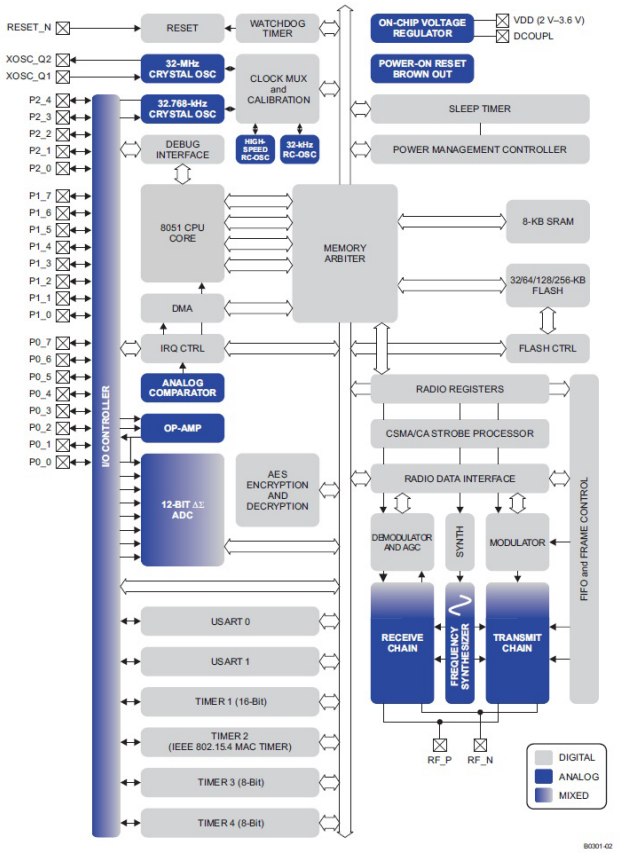


FIGURE 7.9 – Diagram of CC2530 (Source : Texas Instrument)

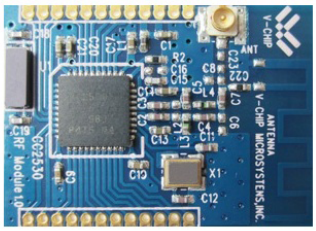


FIGURE 7.10 – PCB module of CC2530

Design a sensor node with CC2530

A traditional design of an node of WSN is presented in Fig. 7.11 wherein the sensors and other components are mounted on a printed circuit board.
We propose to design in this thesis, a miniature node with multi-purpose. Instead

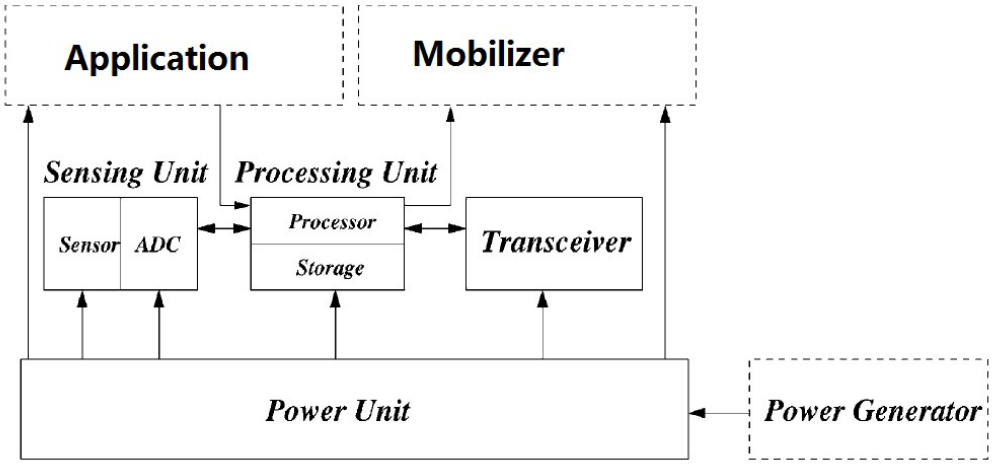


FIGURE 7.11 – Traditional design of a WSN sensor node

of placing all the elements in a printed circuit board, the basic CC2530 circuit and other control circuits in two different PCB are welding together vertically, in this way the area of node can be greatly reduced. Also, instead of designing different nodes with different sensors, the sensors are placed on a separate printed circuit board. The board of the sensor can be connected to the main PCB node through an connector interface. our design of sensor node is presented in Fig. 7.12
The enforcement of the RF coverage and improvement of the network robustness can be realized by adding RF emission power amplification module CC2590 or CC2591 : the default transmit power with CC2530 RF is 1dbm, which can be found in the Z-stack parameters presented below.

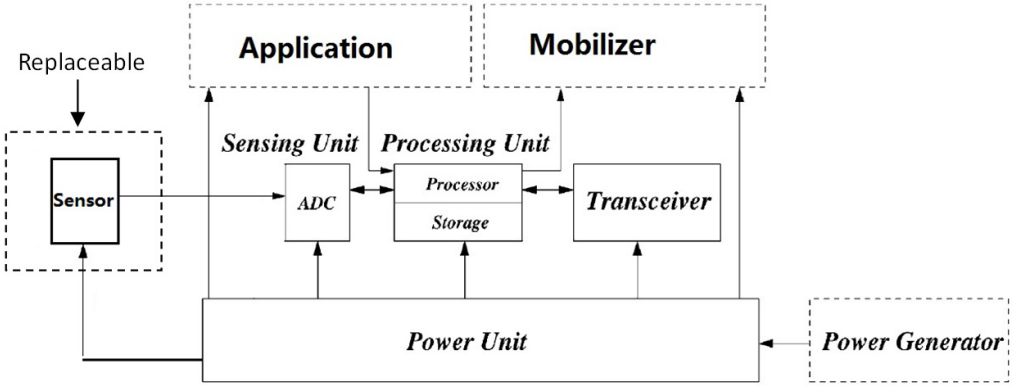


FIGURE 7.12 – New design of a sensor node used in this thesis

$$\#define\ MAC\_RADIO\_TX\_POWER\_DEFAULT\ 0xD5$$

$$\#define\ MAC\_RADIO\_TX\_POWER\_MAX\_MINUS\_DBM\ 25$$

In this configuration, the power of RF emission can be reinforced up to 4.5 dBm, which is still insufficient in some cases. The CC2591 and cc2590 modules can be added to provide up to 14 dBm and 22 dBm RF transmitting power respectively. The design with CC2591 RF module is shown in Fig. 7.13

This enhanced module is used on our Coordinator (see Fig. 7.14). In the next chapter, we will compare the different quality of the wireless signal using two different transmission modules.

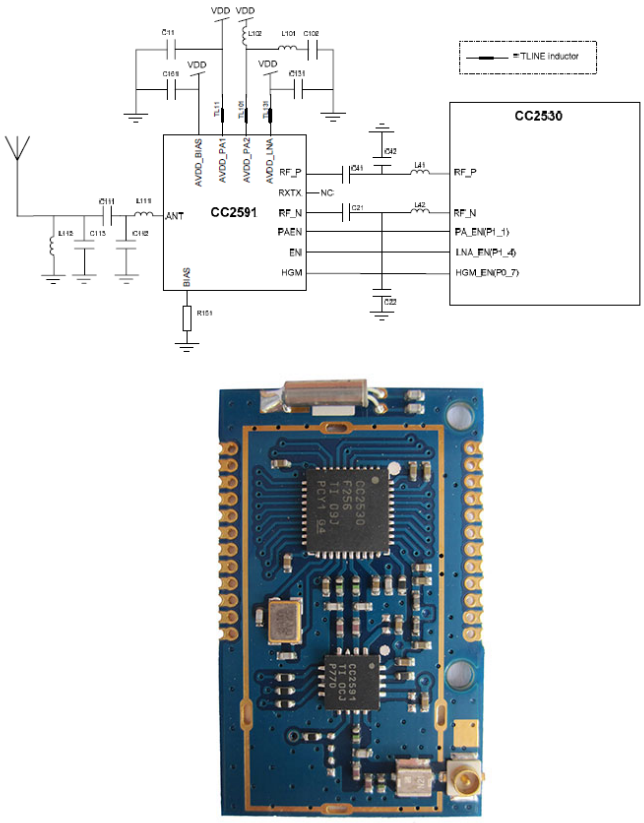


FIGURE 7.13 – The reinforced RF module with CC2530 and CC2591

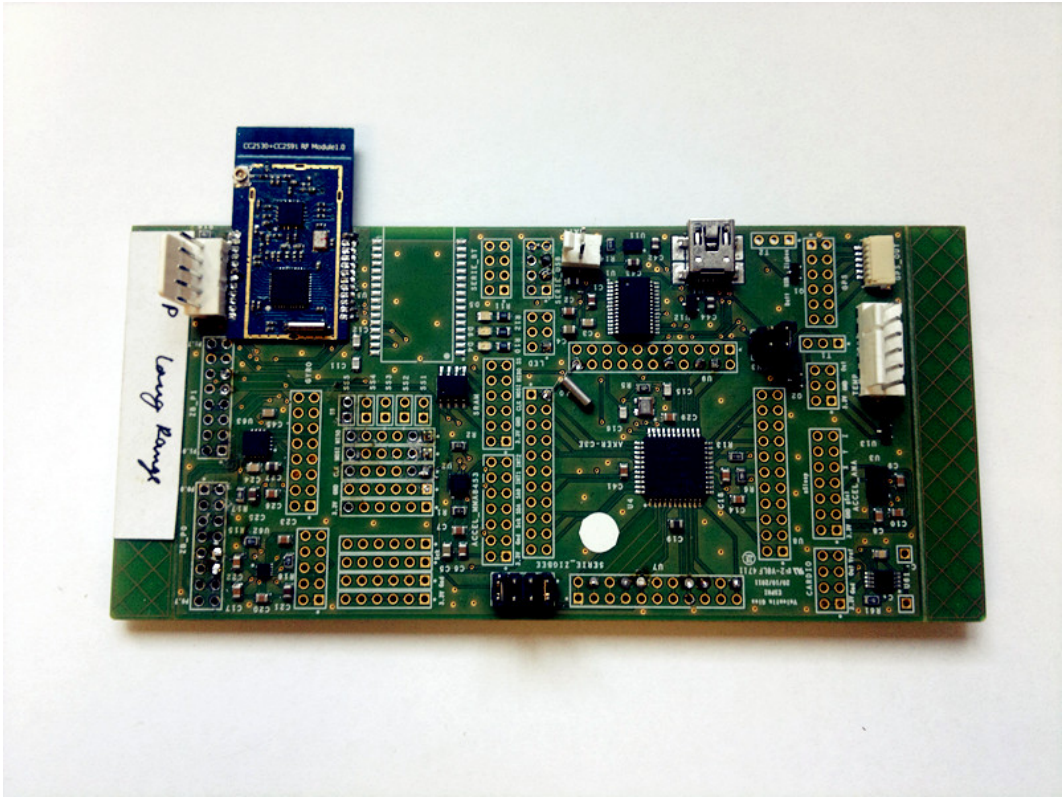


FIGURE 7.14 – Coordinator of WSN

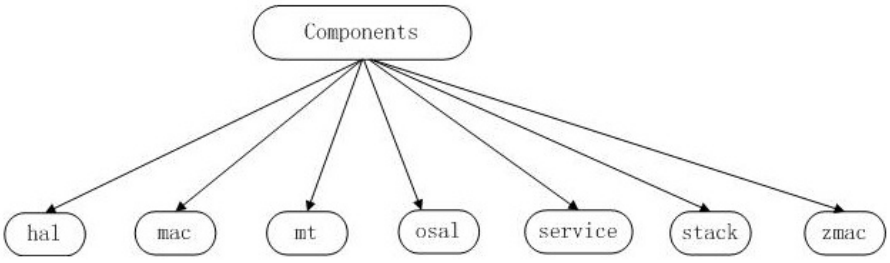


FIGURE 7.15 – le fileset de Z-stack

7.1.4 Z-stack of Texas Instrument

Z-Stack is the ZigBee protocol stack based on IEEE 802.15.4. Z-Stack is compliant with the ZigBee 2007 (ZigBee and ZigBee PRO) specification, supporting both ZigBee and ZigBee PRO feature sets on the SOC such as CC2530, CC2520 CC2520 MSP430 and Stellaris LM3S9B96. As Z-stack is a complete SOC and is highly integrated. A detailed explanation with demonstration of this stack can take over a minimum of 500 pages that will be too long to be presented in this thesis. Here we give a brief introduction of its mechanisms, some of the embedded code can be found in the appendix of this thesis code. Among the files in Z-stack, the most important is the "Component" file, this file can have multiple subfolders, as shown in Fig. 7.15

- File HAL contains the embedded code of the the HAL (Hardware Abstract Layer) layer, its provides the majority of the HAL drivers to control LED, LCD, ADC, keys, clock and UART etc. These services are summarized by a simple API that allows users to use these services without worrying about implementation of these services in terms of hardware configurations.
- File Mac contains the implementation of the IEEE 802.15.4 physical layer, this part is given in the form of library files. That means we do not have access to code in this file, and all changes are impossible in the MAC layer.
- File TM contains the debugging tools and its source files.
- File OSAL (Operating System Abstract Layer) provides the abstraction layer of the operating system and necessary associated files.
- File STACK contains the core components of ZigBee protocol stack, including of AF controls(Application Framework), NWK (network) SAPI (Simple Application Interface), sec (security) ZDO (ZigBee Device Object etc.)

Z-stack can be viewed as a query based event management operating system (see Fig. 7.16).
When the system enters the level of OSAL, it will continue to work in this level with an event handler. The user request will be defined and managed in this layer by the events handler. For example, a button is pushed on the coordinator, this will create an event in the HAL layer that will finally move to the OSAL level, handling mechanism is presented in Fig.7.17. The embedded code and detailed setting concerning the user applications design on the OSAL level are presented in the appendix of this thesis.

7.1.5 The embedded system on CC2530

Since the embedded system is written with the C language, the entry of the whole system is in the function *main*(). The structure of the embedded system is

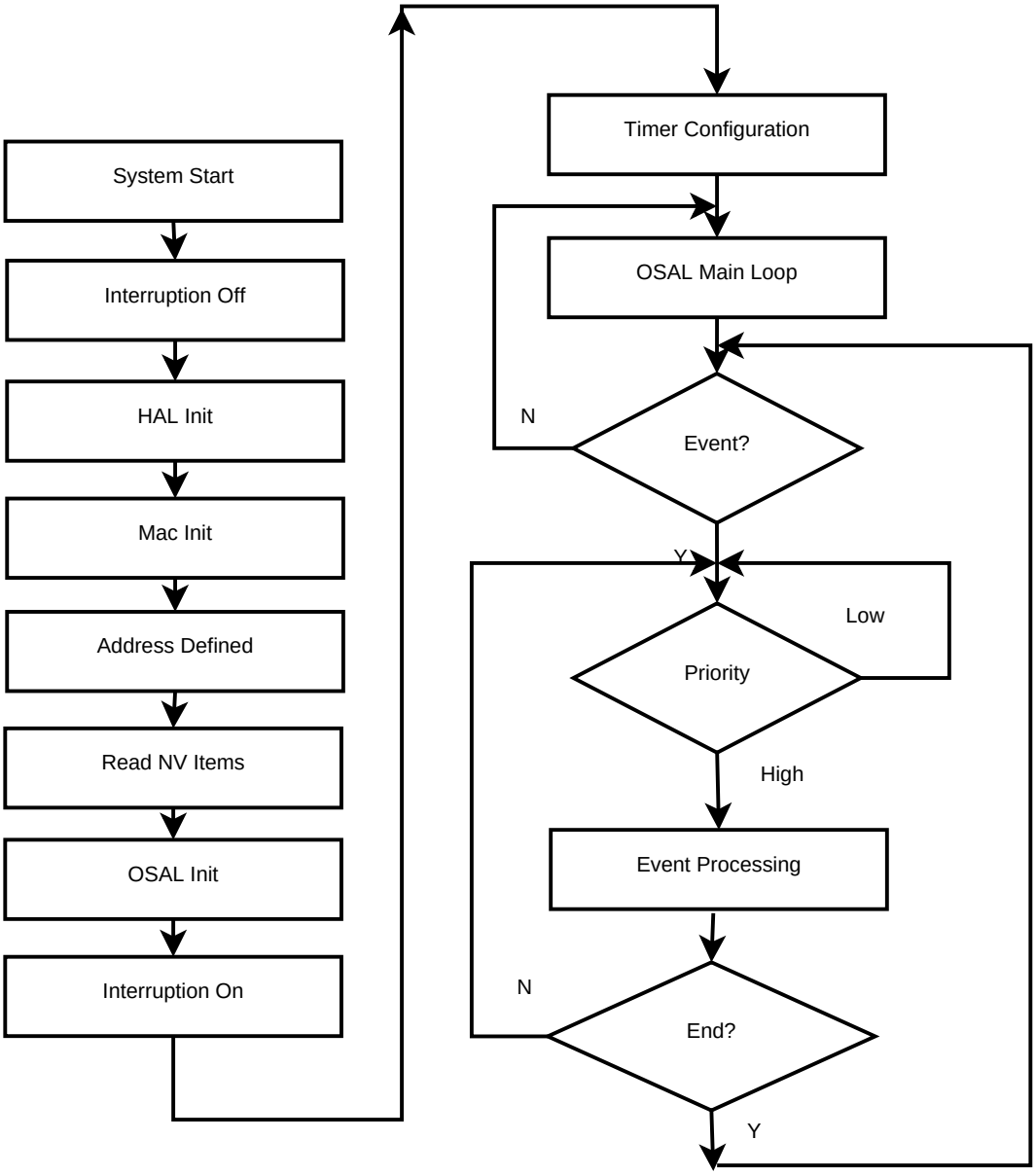


FIGURE 7.16 – The structure of Z-stack

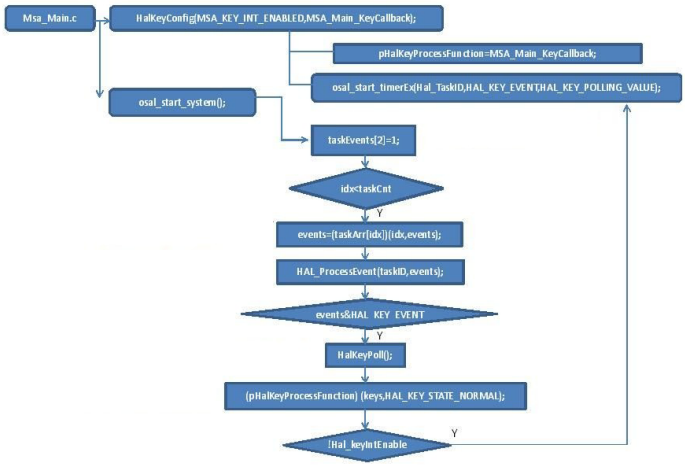


FIGURE 7.17 – The mechanism of the OSAL layer in Z-stack

shown in Fig. 7.16.

```
int main(void)
{
//Éteignez interrompt
osal_int_disable( INTS_ALL );

// Initialisation de HAL

HAL_BOARD_INIT();

//vérifiez que la tension d'alimentation
//est suffisamment élevée pour exécuter
 zmain_vdd_check();

// Initialisation de la mémoire de la pile
zmain_ram_init();

// Initialisation de la carte d'E/S
InitBoard( OB_COLD );

//Initialiser les pilotes HAL
HalDriverInit();

// Initialisation NV système
 osal_nv_init( NULL );

// Déterminer l'adresse étendue
```

```
zmain_ext_addr();

// Initialisation des articles de base NV
zgInit();

// Initialisation de la MAC
ZMacInit();

//Comme l'AF n'est pas une tâche,
//appeler sa routine d'initialisation
#ifndef NONWK
  afInit();
#endif

//Initialisation du système d'exploitation
osal_init_system();

// Autoriser les interruptions
osal_int_enable( INTS_ALL );

//Initialisation finale de la carte
InitBoard( OB_READY);

//Affichage d'informations sur ce dispositif
zmain_dev_info();

// démarrer le système OSAL et aucun retour d'ici
osal_start_system();
```

The *main*() function initialize the entire hardware board of sensor node, coordinator and router. The two most important sub-functions are *osal_init_system*() and *osal_start_system*(). These two functions define initialization and boot of the embedded operating system.

```
osal_init_system()
byte osal_init_system( void )
{
  osal_mem_init();
  osal_qHead = NULL;
#if defined( OSAL_TOTAL_MEM )
  osal_msg_cnt = 0;
#endif
  osalTimerInit();
  osal_pwrmgr_init();

 // Initialize the system tasks.
 osalInitTasks();
```

```
  osal_mem_kick();
  return ( ZSUCCESS );
}
```

This function initializes the main modules used by the operating system, such as memory, battery, etc. The most important sub functions of it is $osalInitTasks()$. $osalInitTasks()$ initializes the different layers Zigbee and all the tasks defined by the user.

```
void osalInitTasks( void )
{
  uint8 taskID = 0;
  tasksEvents =
  (uint16 *)osal_mem_alloc( sizeof( uint16 ) * tasksCnt);
  osal_memset( tasksEvents, 0, (sizeof( uint16 ) * tasksCnt));
  macTaskInit( taskID++ ); //mac_ID = 0
  nwk_init( taskID++ );    //nwk_ID = 1
  Hal_Init( taskID++ );   //Hal_ID = 2
#if defined( MT_TASK )
  MT_TaskInit( taskID++ ); //mt_ID = 3
#endif
  APS_Init( taskID++ ); //APS_ID =4
  ZDApp_Init( taskID++ );//ZDO_ID =5
  WSNApp_Init( taskID ); //WSN_ID = 6
}
const pTaskEventHandlerFn tasksArr[] = {
  macEventLoop,
  nwk_event_loop,
  Hal_ProcessEvent,
#if defined( MT_TASK )
  MT_ProcessEvent,
#endif
  APS_event_loop,
  ZDApp_event_loop,
  WSNApp_ProcessEvent
};
```

The sub function $Osal\_mem\_alloc()$ is a function of memory management. After allocating memory, it will return a pointer that corresponds to a cache space allocated. The return parameter is the cache size. The function $tasksEvents[]$ is addressed to the head of the assigned memory.

The initialization of the task : the system begins with allocation of storage space for each task. Of course, this space by default is all set to 0 (NULL), and then assignments are made for all tasks with different taskID; here the order of the functions in the main loop $tasksEvents[IDX]$ system and $tasksArr[IDX]$ de $taskID$ is unified. In $WSN\_App.c$, we defined a file from a table. Each member

of the array is a function whose order of these functions and the initialization sequence is the same like for the task order in the memory. The difference between $tasksEvents[]$ and $tasksArr[]$ is the array of pointers $tasksEvents[]$ points to the storage space of each task and $tasksArr[]$ finally points to the handling management of different events in each task.
Our own defined event is $WSNApp\_init()$, which is the last in the priority list. Thus, it is on the last position of all events.
The launch of the embedded operation system is presented here : our WSN application WSN_APP will be called periodically by $OSAL\_Timer$ in the main loop of OSAL. Thus, sensor data acquisition is made each time when the WSN_APP is activated. The sensor data will then be processed and sent to the coordinator. The detailed code is presented in the Annexe.

8

Feasibility of our WSN system : Link quality test

8.1 WSN in buildings : signal quality analysis

Wireless sensor networks have enormous potential for energy savings and other building and environmental applications. A major obstacle to their adoption, however, is the uncertainty and the reliability of wireless links. While the rate of packet delivery is difficult to measure in the present applications, the experiments on the wireless signal quality are conducted. The results are verified by analyzing the RSSI and LQI values under different environmental conditions.
Before any application of WSN, its reliability must be be verified. In qualitative terms, reliability means that the desired data is sent to the receiver at the desired time, with little delay, and with a minimum measurement error. Defining reliability itself is a difficult task, because it involves a number of issues and can be affected by many factors. Data transmission success depends on a high (RF) link quality between the transmitter and receiver. Accuracy of the data stream depends on the sensor itself. RF connection can hardly affects the accuracy of data as the data is transmitted in digital format and the corruption of the data stream usually causes problems with data delivery versus modified data values.
Successful data delivery, therefore, becomes the main concern of users of wireless sensor networks in buildings. Ensure that data can be delivered in a particular situation that the signal strength at the receiver is highly related to the ambient noise. Equation (8.1) provides the model of signal receiving by the antenna without interference in the RF medium [61] :

$$P_r(d) = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d^2 L} \tag{8.1}$$

where $Pr(d)$: the receiving power ; Pt : the transmitting power ; Gt : transmitter antenna ; Gr : gain of the receiver antenna ; l : RF signal wave length, d :

distance between the transmitter and receiver ; L : oss factor of the system (associated with the radio equipment). It is indicated in the equation that the received power decreases with the square of the distance between the transmitter and the receiver is directly proportional to the transmitted power. In addition, the received signal strength depends on the wavelength, but this relationship is less critical applications of sensor networks in buildings for more equipment uses the 2.4 GHz band.
This equation is technically valid in free space. However, in reality, the equation has missed some very important factors which is the presence of obstacles and interference from different 2.4GHz wireless signals. Thus, real tests on the WSN system's reliability has to be evaluated by examining the wireless signal link quality and signal strength before its deployment in any real applications.

8.1.1 Packet Error Rate (PER) test

Firstly, we carried out 100 acquisition experiments to evaluate the WSN's Packet Error Rate (PER) [6]. The results are shown in the figures (Fig. 8.1 and Fig. 8.2) below : for short term acquisitions(acquisition for 1 day), the PER of WSN is around $1.41x10^{-4}$ and the maximum PER reached during the long term acquisition tests(acquisition for 20 days) is 0.016. So, considering the limited PER values, the WSN system is reliable at this level.

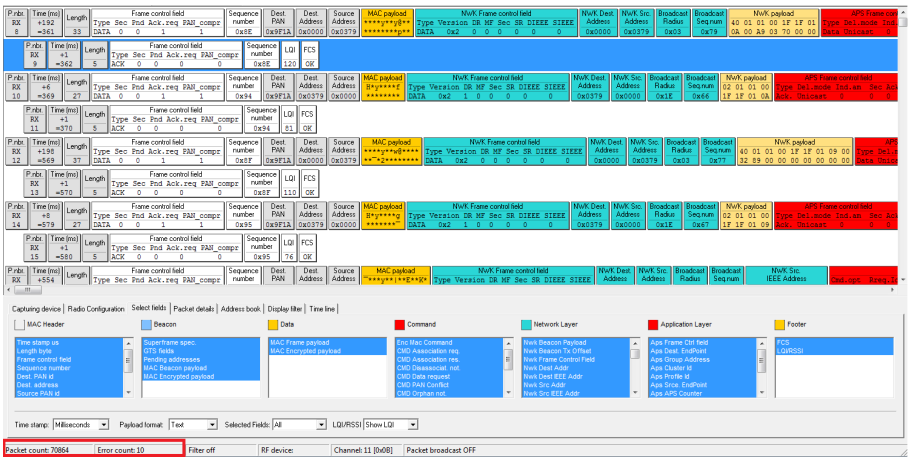


FIGURE 8.1 – PER :Short term acquisition

8.1.2 Signal strength indication and link quality indication

Signal strength indication (RSSI) is a term used to describe the strength of a wireless signal. The units are either those of energy (eg, mW) or, more commonly

6. PER is the number of packet errors divided by the total number of transferred packets during a certain time interval.

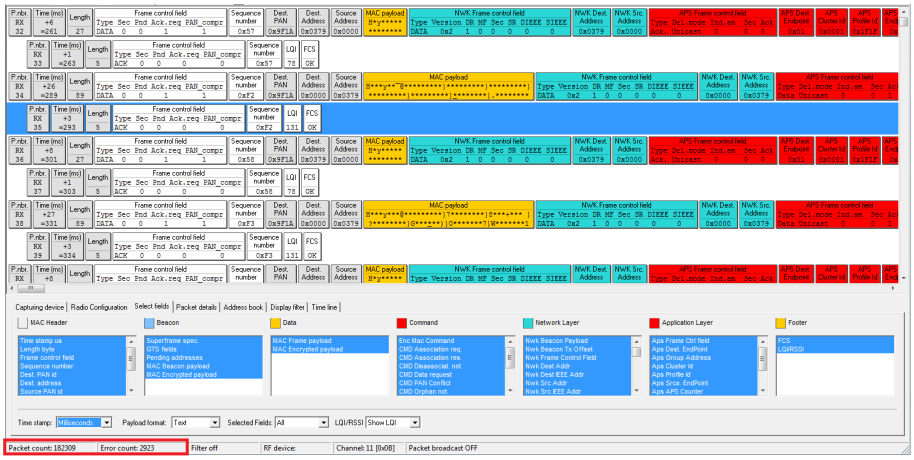


FIGURE 8.2 – PER :Long term acquisition

by $dBm = 10log(Pr/1mW)$.
The IEEE 802.15.4 standard, as introduced in the previous chapters is used in the building industry communications, specifies the use of an indication of link quality (LQI) to evaluate the quality of the communication link between a transmitter and a receiver.

Many current WSN platforms like MicaZ, Telos, and Intel Mote2-use the same radio chip, the CC2420 of TI, in addition to RSSI, CC2420 provides an additional indicator LQI [62].

8.1.3 Experiments and analysis

Our WSN system is based on CC2530 Zigbee microcontroller from TI, which is an upgraded version of CC2420. Thus, In all the CC2530 cored ZigBee networks, a built in received signal strength indication (RSSI) is stored. The RSSI value is an 8-bit 2s-complement signed number on a logarithmic scale with the step of $1dB$. It can be read from a register, it is actually found in the $RSSIL.RSSI\_registre\ VAL$, the status bit $RSSI\_VALID$ should be checked, it indicates that the RSSI value in the resigster is in fact valid, which means that the receiver has been enabled for at least eight symbol periods. to find the actual signal power P at the RF pins with reasonable accuracy, an offset must be added to the RSSI value :

$$P = (RSSI\_VAL + RSSI\_OFFSET)dBm$$

For example, with an offset of -50dB, the RSSI value -10 from the RSSI register means that the RF power is approximately -60dBm. Here, the offset value eventually is a experienced value during the experiment of CC2530, it is around -45dB.

In Zstack the description of the RSSI value is presented below :

$$define\ MAC\_RADIO\_RSSI\_OFFSET$$
$$define\ HAL\_MAC\_RSSI\_OFFSET - 45$$
$$RssiDbm = PROPRIETARY\_FCS\_RSSI\ rxBuf + MAC\_RADIO\_RSSI\_OFFSET$$

The Link quality indication LQI is a measurement of the quality of the received frame as defined by IEEE802.15.4 standard. The LQI value is required by IEEE 802.15.4, limited from 255 to 0, in the case of CC2530, the RF signal does not provide LQI directly. However, it could be calculated by the microcontroller. In Z-stack, the LQI value is presented as below :

$$define\ MAC\_RADIO\_RECEIVER\_SENSITIVITY\_DBM\ -91$$
$$define\ MAC\_RADIO\_RECEIVER\_SATURATION\_DBM\ 10$$
$$define\ MAC\_RADIO\_RECEIVER\_SENSITIVITY\_DBM\ -91$$
$$define\ MAC\_RADIO\_RECEIVER\_SATURATION\_DBM\ 10$$
$$define\ MAC\_SPEC\_ED\_MIN\_DBM\_ABOVE\_RECEIVER\_SENSITIVITY\ 10$$

$$define\ ED\_RF\_POWER\_MIN\_DBM(MAC\_RADIO\_RECEIVER\_SENSITIVITY\_DBM + MAC\_SPEC\_ED\_MIN\_DBM\_ABOVE\_RECEIVER\_SENSITIVITY)$$

$$ED = (MAC\_SPEC\_ED\_MAX * (rssiDbm - ED\_RF\_POWER\_MIN\_DBM)) / (ED\_RF\_POWER\_MAX\_DBM - ED\_RF\_POWER\_MIN\_DBM);$$
$$pRxBuf -> mac.mpduLinkQuality = macRadioComputeLQI(rssiDbm, corr);$$

Thus, the relationship between RSSI and LQI can be expressed by the equation below :

$$LQI = \frac{255(RSSI + 81)}{91} \tag{8.2}$$

There are two methods to get the LQI value from Zstack. The first is to use one of the Mac level function :

$$pRxBuf- > mac.mpduLinkQuality = macRadioComputeLQI(rssiDbm, corr);$$

otherwise, the LQI value could be found directly from the application level :

$$afIncomingMSGPacket\_t * pkt; pkt- > LinkQuality$$

In order to qualify the performance of ZigBee network used in our project, experiments were carried out under different conditions. Since the LQI value is more direct, it is used as an indication of the signal quality of the network. Therefore, software is designed for retrieving the LQI in the network. The first test is performed in the field of sports named Imp. Guy Mocquet in the city of La Garde, 83130, France. The aim is to test the signal quality in the open area without obstacles, unfortunately, there is still some WiFi signal even in this area, because the LQI value is very sensitive, the test result may be somehow affected by the presence of these interference. The result of the test is shown below in Fig. 8.3.
Then, the same test was performed with a 30-degree angle twisted between the

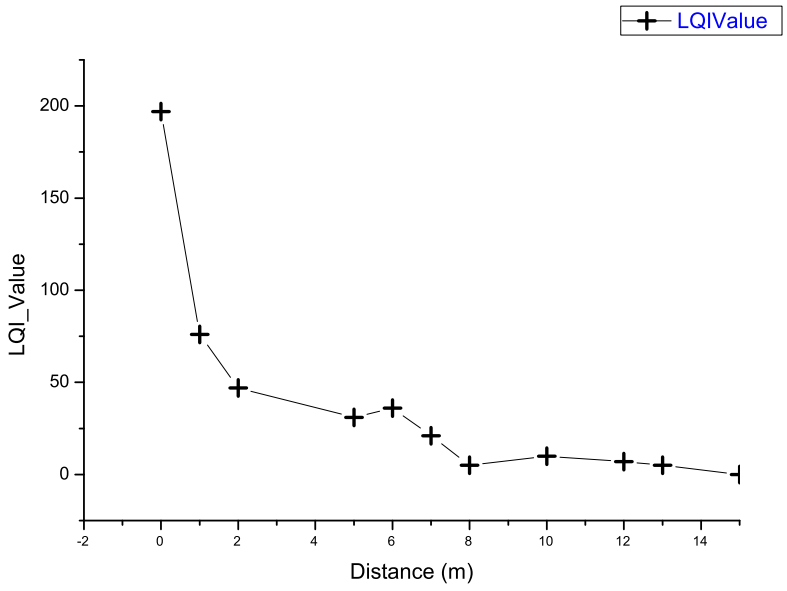


FIGURE 8.3 – LQI : test results in a outdoors sports field

Enddevice and Coordinator. The result is shown in Fig. 8.4 The results shows that the signal quality of Zigbee networks is nearly not affected by small angle presence between the end device and coordinator. Then, experiments have been conducted in the building E of uiversité de Toulon (UTLN). The results are presented below : The Zigbee network is able to penetrate wood, metal or concrete barriers with the

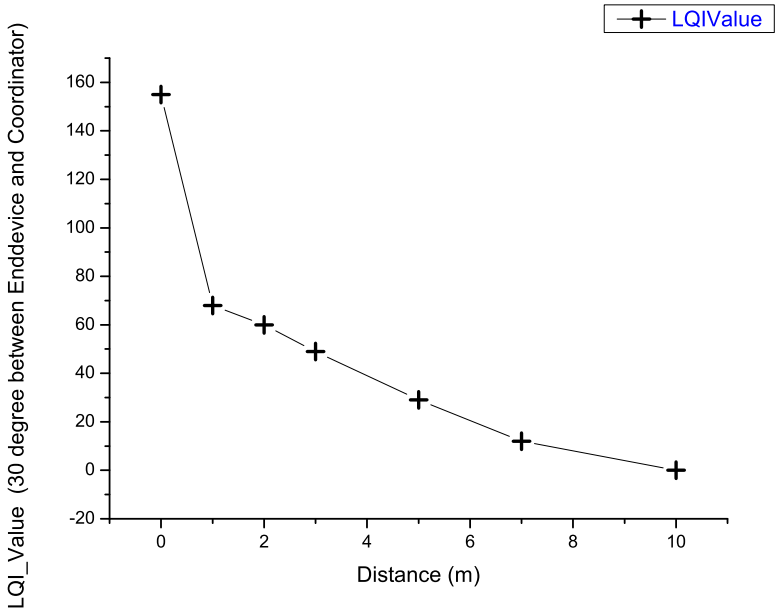


FIGURE 8.4 – LQI : test result with an angle of 30°between the coordinator and Enddevice

LQI value of 65, 31 and 26 respectively. This means that the Zigbee network is applicable in the environment of normal construction. The available coverage range of Zigbee network in the building is tested. The results are presented in Fig. 8.5. The above tests are made with the standard configuration CC2530 RF. To find the limit of our wireless modules, we also conducted experiments with amplified CC2591 RF modules (solution presented in the previous section : CC2530 + CC2591). The test results are shown in Fig. 8.6 below.
As presented in Fig. 8.6 and Fig. 8.7 amplified modules greatly expanded the effective coverage of wireless signal : it can cover up to 250 meters in outdoor environment without signal attenuation. The indoor test shows that the network can cover the whole building with a LQI value from 160 to 48.

The testing result positively marks that although the signal quality decreases when penetrating the wall, it decreases very slowly after with distance. This means the reliability of our WSN system enabled its applications in buildings.

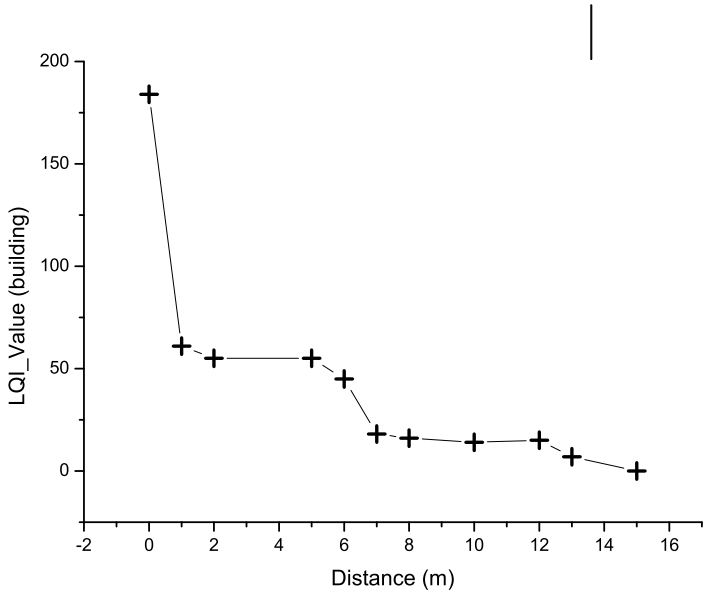


FIGURE 8.5 – LQI through the wall in the building

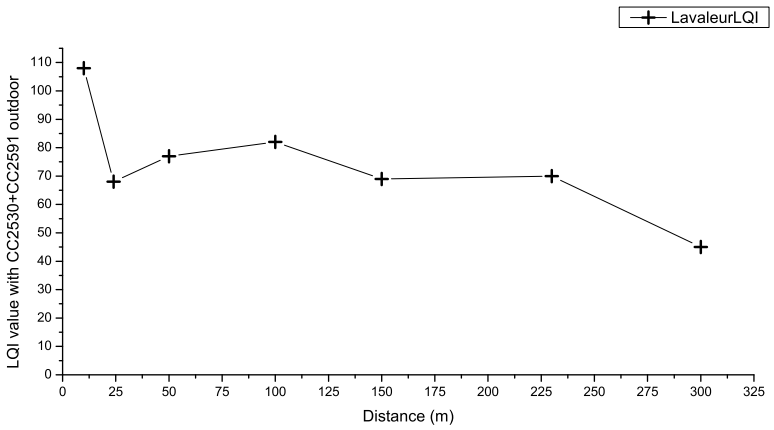


FIGURE 8.6 – LQI : Outdoor test with amplified RF power modules

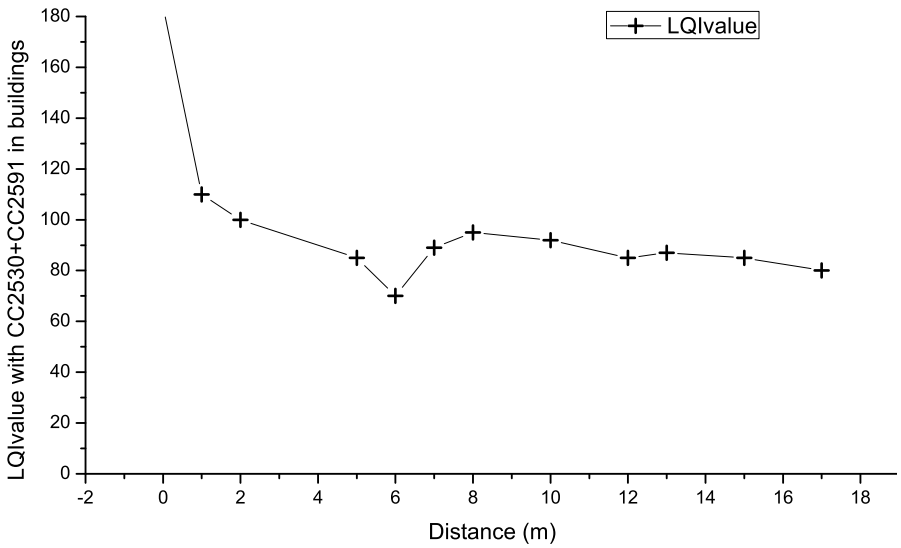


FIGURE 8.7 – LQI : Indoor test with amplified RF power modules

9

Hardware and Software solutions

9.1 Features of the developed Modeling software

In previous sections, we systematically introduced the WSN system. Thus, in this section, we'll just present the technical parameters and features of our WSN developed.
The WSN is based on (TI) CC2530 microcontroller. It fully supports ZigBee 2007 specification and ZigBee and ZigBee PRO feature sets. As discussed in Part 3 of this thesis, the WSN system is a typical mesh network that contains three types of modules : Coordinator, Router, and Enddevice. Different types of sensors are integrated on Enddevices sizes of EndDevice and Coordinator (shown Fig. 9.1) are respectively $5x3$ cm and $6x2.5$ cm.
The performance of our new hardware platform WSN is verified. Its technical pa-

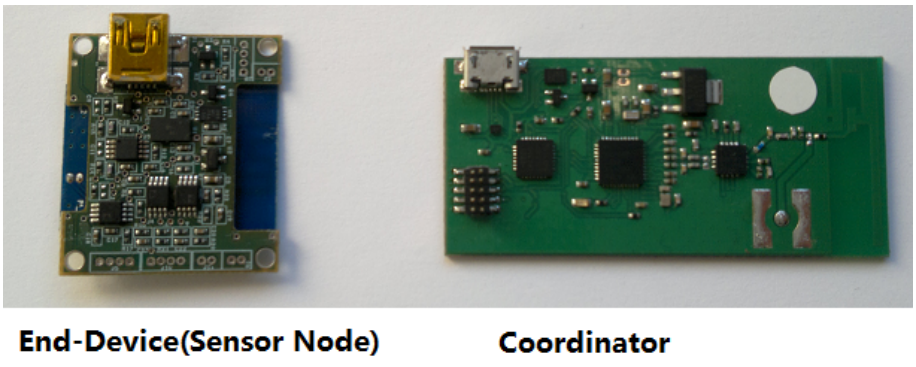


FIGURE 9.1 – Enddevice and Coordinator of the WSN system

rameters are summarized in Tab.9.1. The main features of the WSN system are low cost, low power consumption, high reliability and high network capacity.

The digital thermal sensor we used is Microchip TC1046. According to its producer, it can accurately measure temperature from $-40°C$ to $+125°C$ with a reso-

TABLE 9.1 – Parameters and features of the WSN system

Feature of the WSN system	Values
Cost per unit	8 euros
Network Coverage	20m/110m
Consumption in operational mode	40mA
Consumption in Power saving mode	126μA
Battery Life	>200h
Reliability	High
Packet loss ratio	<0.02
Installation	Easy
Network capacity(Nodes per Network)	25920

lution of $0.0625°C$. Also, during every experiment, sensor calibrations are made : sensors were placed together under an ideal condition which means solar radiation and temperature disturbance have been avoided. The bias for each sensor was obtained by comparing their measurements to a high precision thermometer.

This WSN system was later used in the experiment of thermal modeling of the room. The sensors on the end-devices send the data to the central processing unit (PC or Server) through the coordinator. The central processing unit then calculates the temperature at different points of the building room and records information about the sensor. The temperature and its acquisition time are saved in the SQL database. All these data will be used as a source of training to build the ANN thermal model. All data processing and modeling is done in our developed software called "inferface sensor" which will be presented in the next section.

9.1.1 Graphical user interface Software design under C#

The software is developed under C#. An object oriented programming language created by Microsoft. Because of its efficiency, it has gained a great reputation ever after its birth.
The main purpose of the software is to acquire data from the sensor nodes in the WSN platform, display them either in real time or using the static storage in a database. A dynamic graphing is designed in this software to show the evolution of the measurement for each sensor. An overview of the software "Interface Capteur" is given in Fig. 9.2

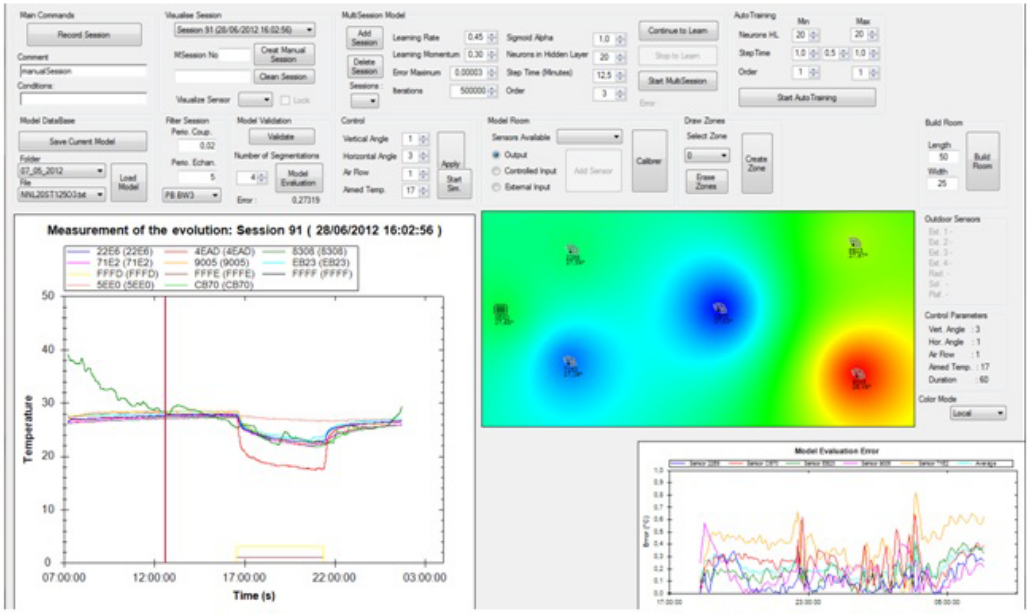


FIGURE 9.2 – The software "Inerface Capteur"

The software is divided into several parts, as shown in Fig. 9.2 : Key commands, experiment-setup, model identification, simulation, signal processing etc. Here we will begin with introduction of the basic functions in this software

Sensor Identification

In this thesis, each sensor node has an unique ID and its measurements are shaped in a standard form, which is a 8-byte string. The 4 first characters are the sensor's identification number and the 4 last ones are the temperature sensor's measurement (in hexadecimal) after an analog-to-digital conversion (ADC) made by the microcontroller CC2530 on the end-device. There are two basic "states" regarding data acquisition : a Boolean variable is used to choose which set of actions (that define the state) will be executed every 5 seconds. This is achieved by using a virtual timer. The WSN works constantly by sending the data to the software. When the software starts, the recording function is not activated. Thus, these data from WSN are used to identify all the sensors present in the experiments. The sensor's measurements are ignored but the sensors'ID are identified and saved into a "Sensor Pool". The Sensor Pool is a list structure inside the code that contains all the sensors available in the experiment. This list contains "Sensor" objects, that in turn contains for example the sensor's actual measurement. Its identification number and some useful functions. The aim of this first step is to know which sensors is working and where it is located inside the room. The second phase begins with the activation of the button "Record Session" (Save Session) and it is the moment

when the software begin to record the measured sensor data.

Acquisition environment

The first thing to do in order to record a session of measurements is to define the abstracted building room environment. In this thesis, the rooms are considered rectangular and one should know the length and width of the room in advance before recording a session. As explained in the previous section, the sensors are now well identified and registered into a structure so called Sensor Pool in the software. This pool is always analyzed in order to fill the drop list of available sensors. Before creating the building room, the software should have already identified the sensors with their ID and the associated information concerning their positions and functions, etc. Thus, these information can be deployed in the dynamic graphing of the room environment.
To create a graphic room, we need to define its width and length in the software, after which the room plan is plotted with the correct dimensions in pixels. Once the room is created, the position of available sensors from the sensor pool can be validated on the room plan.

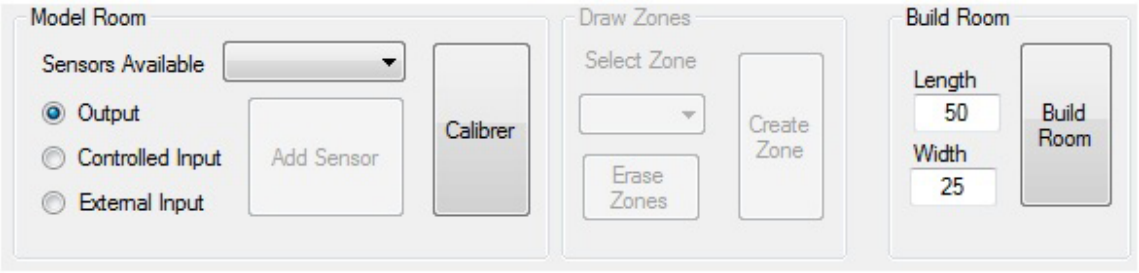


FIGURE 9.3 – Building a dynamic 2D display function for the acquisition

Save a session of acquisition

At this stage, we have all the information needed to start a recording session : the size of the room and all of the sensors' information (its positions, roles as input/output and temperature offsets for calibration).
So, as soon as the Save Session button is clicked, data storage actions begins. The current date and the acquisition time click of the session is registered alongside with the room's length, width and the description of experiment in a data table of session in the database. Every session is identified with a unique number : they are simply chronological. The first session ever recorded is assigned as session number 1. At the same time, each sensor present in the recording session has its identification number stored in the sensors table of the database alone with its ID and

information. Finally, after the recording actions are realized, the actual measurements is stored in the data base in real time. The Boolean variable mentioned in the Sensor Identification session is changed and the state of the program changes to recording mode. Generally, during the first state is responsible of the identification of each sensor used in the acquisition and it is during the saving mode (second state) that the program stores the measurements sent by the sensors into a data table of measurements in the database.

C# programming is based in events that can be triggered by user or hardware actions. In this case, the reception of a sensor data string from the Wireless Sensor Network through the computer's serial port, triggers the concatenation of the received string at the end of a longer buffer string. This way we have a buffer string that grows according to every reception of sensor data.
A virtual clock, already mentioned, is responsible for breaking this long buffer string into processable information at each tick of 200 milliseconds (which is also done during the sensor identification state). This is useful to make the processing and storage synchronous.
The breaking of the buffer is realized using a Regex pattern recognition function, which is able to break any string and find patterns defined by the programmer inside it. The Regex is used to break every 8-character pattern from the buffer string so we can identify the sensor responsible for the reading and the reading itself. The sensor data (the last 4 characters of the 8-character pattern) is in hexadecimal form, this was chosen to increase the precision of the measured sensor data. The program then uses a specific formula provided by the sensor's provider

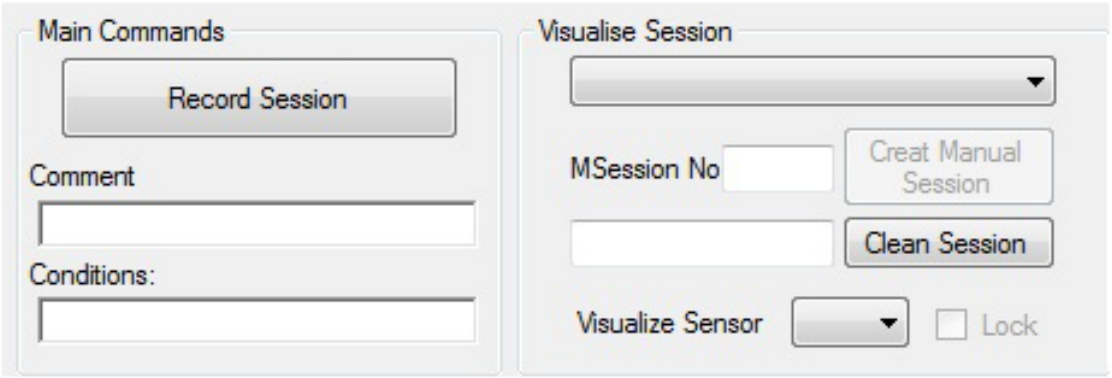


FIGURE 9.4 – Save a session of acquisition

to transform the measured value to real temperature. This can be refereed by the sensors' characteristic voltage/temperature curve. The main reason that we make the software responsible for these calculations instead of the microcontroller is to take the burden from the sensors associated microcontroller and thus reducing the

energy consumption and increasing the battery autonomy of end-devices of WSN.

Finally, after every 8 bytes string is broken and processed, they (already in degrees Celsius) are stored in the Measurement table. Also, for the data table of measurements, each of the table entries has a unique measurement identifier which is the current session number and the time offset at the beginning of this session. Strictly speaking, this time does not correspond exactly to the time at which the measurement was taken by the sensor (because it is rather the time that the data is being processed).
By clicking again in the Record Session button (which is now displaying Stop Recording), all the sensor data is already stored in data tables of the database. These data is thus ready for further processing or neural network model training.

Visualization

There are two modes of displaying data, that are strictly related to the "state" in which the program is running (Recording mode/ Free mode). In either mode, data will be displayed always on three visualization panels on our screen : the graph plotting each sensor's temperature versus time, the "thermal map" of the room (the room vision with every sensor in it and colors representing the temperature of each place) and the temperature of the outdoor sensors.

The graphic presentation in the program was drawn with the help of a C# library called ZedGraph which can be used to draw static or real time plots. As an object oriented library, it is used to the define curve objects like the colors, identifiers, points in the graph and so on. Each curve on the graph represents then a sensor, and plotted on the graph is its temperature evolution regarding time. When display is in real-time, the graph window adjusts itself to fit data accordingly and the curves are updated quickly, at the same clock tick when data was stored in the database.
This graph is located on the left side of the software user interface in Fig. 9.5. The thermal map of the room, though, cannot be updated at the same time when the clock tick used to save data in the database because it would disrupt data processing (mainly the accuracy of the time offsets which are stored along with every measurement). The reason is that the drawing of the room image takes just too long to be completed (more than 200 milliseconds). This thermal map is drawn with C#'s own drawing library, which is very efficient but cannot tackle with the problem that the image contains too many pixels to draw upon. Considering that every pixel will have a different color, we cannot do it in a simpler way, every pixel needs to be redrawn when updating the image.
As mention earlier, besides real-time display, it is useful to observe and analysis the recorded sensor data. This visualization can be realized by selecting the session number of interest in the Visualize Session drop list, seen in Fig.9.5.
When visualizing statically a session, the graph on the left side of the screen plays a

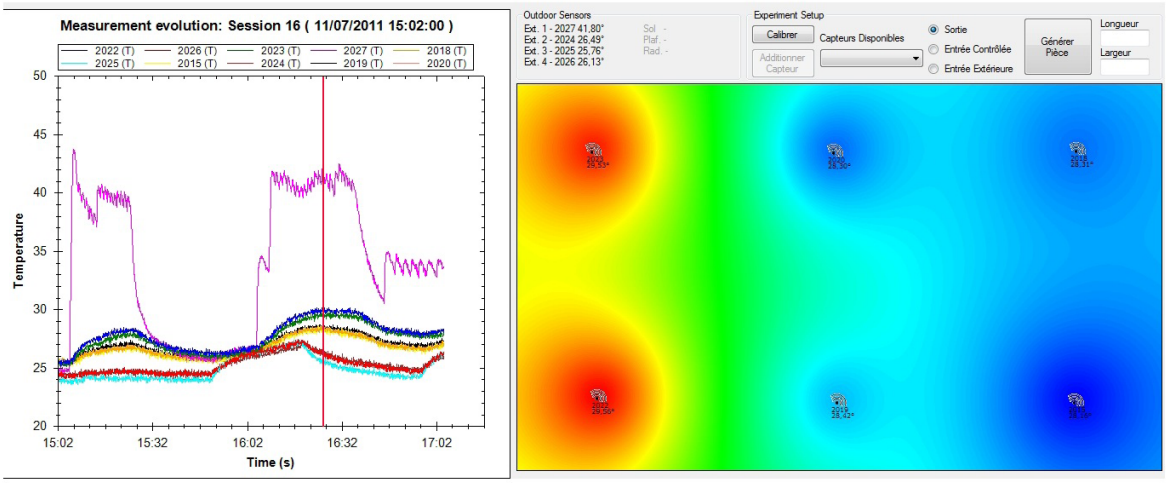


FIGURE 9.5 – Visualization

key role as all measurements will be condensed in curves representing each sensor's measurements.
Initially information will be highly packed, spanning along the recording time of the selected session, but one can quickly zoom in an area of interest by simply dragging the point over it with the left button pressed. When we press the right mouse button on the graph, several options will be displayed, like zooming out or displaying the graph's initial state. Finally and perhaps the most important of all, when we click twice with the left mouse button on a certain spot of the graph, the measurements of that specific time will be represented on the right side of the screen, both on the room thermal map and on the part dedicated to the outdoor sensors. A red line will be drawn on top of the graph at the exact spot the double click was given, so we can identify easily at which time the drawn representation on the right relates to.
In the next section, we present all the higher functions by combining sensor data with WSN ANN modeling.

9.1.2 The library "Aforge" and the implementation of ANN

The reason we chose the ANN modeling approach was already introduced. Also, the great practical value of using ANN as a modeling tool will be summarized in Chapter 10. Here I will just show the implementation of ANN modeling in our software.
An open source library "Aforge" [63] is used in this software to make the modeling function of ANN. The main advantage of using this library is its flexibility, reusability, easy to use and understand. In aforge library, it does not provide a single entity types of neural network to avoid losing the flexibility and clarity. All entities

are defined in different categories for the purpose of reusability. Some libraries have mixed neural networks and learning algorithms together causing a loss of flexibility of algorithms. In the case of Aforge library, the entity is divided into different categories. In this way, it is not only easy to understand but also make it possible of reusing the components and the implementation of small-scale structure of public functions.
The library contains six entities(see Fig. 9.6) :

- Neuron : An abstract base class for all neurons, which encapsulates the common entities such as the weight of a neuron, the output value and the input value. Other classes of neurons inherit the base class to extend with additional properties and specialize transferring functions .

- Layer : represents a set of neurons. It is an abstract based class that encapsulates common functionality to all layers of neurons.

- Network :represents a neural network, which is a collection of layers of neurons. This is an abstract base class that provides common functionality of a generic neural network. In order to implement a specific neural network architecture, it is necessary to inherit the class and extending it to the specific features of a neural network architecture.

- Activation function : interface of the activation function.

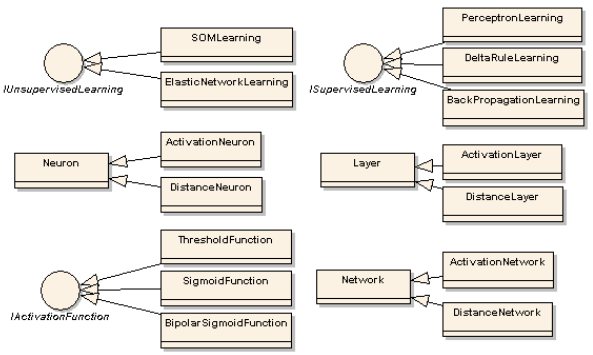
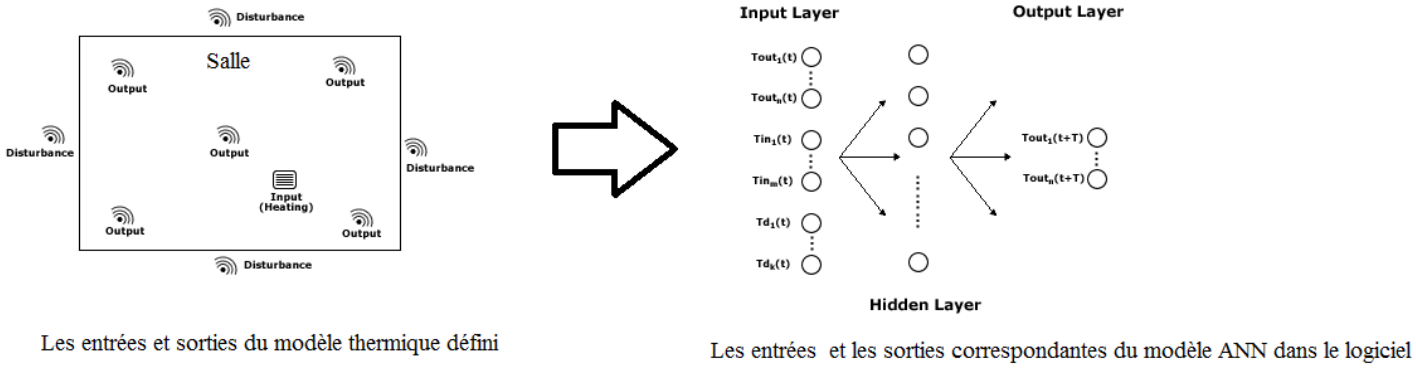


FIGURE 9.6 – Different classes in Aforge

Now, we will give a detailed construction of a thermal model of a room with Aforge example. The problem of black box of a room can be described in the following terms : given a set of "inputs" known controlled (the heating system, for example) and temperature measured "disturbances" (outdoor temperature, solar radiation, etc.). We want to predict from an initial state evolution of the internal temperature. These indoor temperatures are now called the "output" of the system. As a result, the input and output of ANN model can then be defined (see Fig. 9.7).

After the structure of the ANN model is defined, we can begin to train the network with sensor data WSN (see Fig.9.7).



Les entrées et sorties du modèle thermique défini

Les entrées et les sorties correspondantes du modèle ANN dans le logiciel

FIGURE 9.7 – Structuring an ANN thermal model out of the building room

Extraction and segmentation of the sensor data

After the structure of our ANN model is well defined, the next task will be forming the source of training using the raw sensor data WSN. However, if we use directly the raw sensor data as training source, it will lead to a very low training efficiency. Because in the raw sensor data, many repeated measurements can be found, which makes a great data redundancy. Thus, to retrieve the most important section of the raw data from the whole sensor data session can be very important. In our software, we provide direct method to retrieve sensor data needed from the raw data. By operating directly on the SQL database, we can make the segmentations of each session of the sensor data. The results can be found in the Fig. 9.8.

Step-time

The aims of these modeling experiments is to characterize the thermal behavior of building in response to environmental conditions by a model whose function is to recognize the thermal dynamics associated with different parts of the room and be able to reproduce this behavior through the analysis of temperature inside and outside the measured in part and in a discrete time interval. To enable a smooth modeling, two parameters become crucial for the production of input-output pairs. One contribution we introduced in this work is "Step-Time" which defines a time interval, in other words, it defines a sampling period (see Fig . 9.9). Since the frequency of WSN acquisition is high, a single session of acquisitions contains a

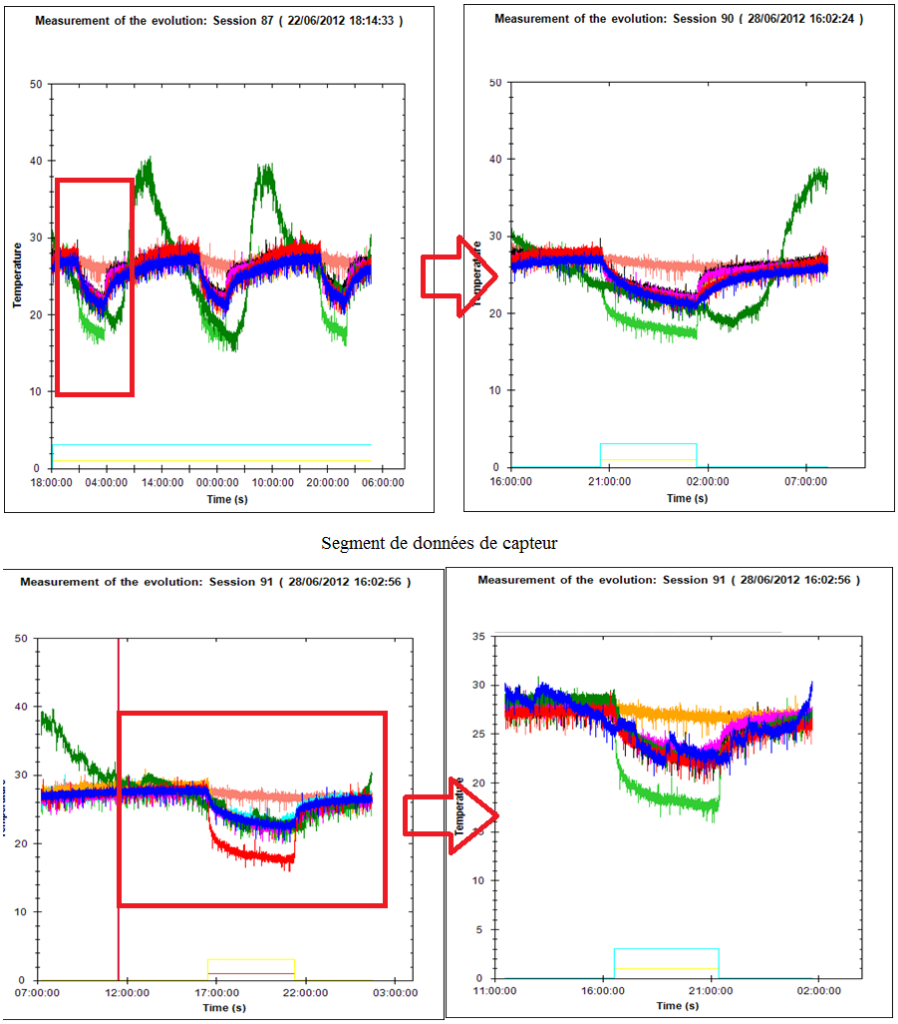


FIGURE 9.8 – Retrieve the useful sensor data segments from raw sensor data

huge amount of sensor raw data in the software database. In order to train ANN model more efficiently, instead of taking all the sensor data from database, the training data matrix for the ANN model is selected sequentially from the raw sensor data according to the Step-Time.

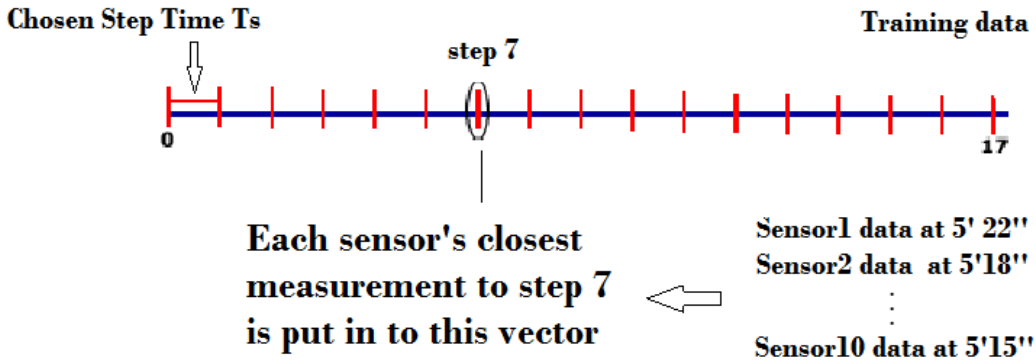


FIGURE 9.9 – The function of StepTime

This parameter could simply affects the ANN modeling results in a way that if the thermodynamics response of the building room is slow, a relatively long Step-Time could be sufficient. Otherwise, if its response is fast, a shorter Step-Time will be necessary to obtain an accurate model.

The order of an ANN model

The other parameter required for the model to represent the dynamics of the temperature distribution is the order of the model. For example, if a model has StepTime 1 minute and order 2 to predict the temperature distribution in the next minute , it uses the distribution of the current temperature and the information on the distribution of the temperature measured one minute before. In the earlier version, the tool for creating input-output pairs was limited to pairs to order 1 and 2 , which may not be sufficient to adequately represent the thermodynamic behavior of the building room. To solve this problem, the implementation of the tool of creating the input-output data pairs was modified to support multiple orders. Even with the possibility of increasing the order of the system indefinitely, it is necessary to understand the implications of this tool. An excessive increase control may include irrelevant information to model the phenomenon, moreover, increased order proportionally increases the number of entries in the neural network which requires a lot of time for training model which in turn may show no improvement in their ability to predict.

Method to determine the best parameter combinations

So far, there is no universally applicable rules to define the best network training parameters. The reasonable solution is to try different combinations of training parameters and evaluate them according to their prediction errors. Thus, to find the best training parameters for our MPCT models, we give two possibilities in this software : 1. Manual Definition, the parameters can be defined by users, 2. Auto evaluation (see Fig. 9.10), the parameters can be selected by a cyclic algorithm. A different range of training parameters are used to train network models. The trained models' prediction errors regarding its training data are then automatically calculated. In the end, the software will select the parameter combination which leads to minimum prediction errors. We present the training errors with a different range of Step-time and network orders. In this training evaluation test, we find the best parameter combination of the Step-time and network order is 13.6 minutes with a second network order.

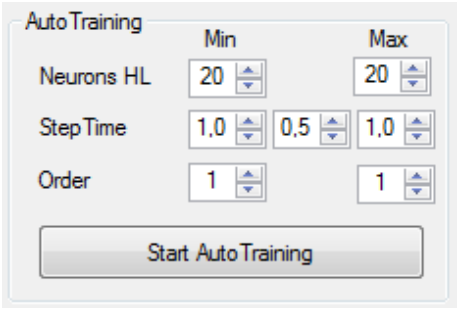


FIGURE 9.10 – Function to determine the best training parameter combination

Model validation

Once the model is complete and the learning procedure has theoretically succeeded in teaching it the dynamics of the room in which we made our measurements, we should validate our model using our software's validation feature in order to confirm that it is consistent. We start by picking a session from the Visualization drop list (after our model was created). Preferably, it should be a different session from the one used to extract the learning set, but we should be sure that the chosen session was also realized inside the same room with the same sensor disposition. It is important to validate the model with another session of measurements because the ANN will be too well adapted to its learning session (Over-fitting problem) and thus we cannot be sure that the general dynamics of the room were absorbed by the network.

The validation process consists, first of all, in setting initial conditions for the output temperatures which equal to the each of the indoor sensors' closest mea-

surement to the first time step on the chosen session period. Next, we gather a "control" data set in a similar way we obtained the learning data set (previous section), with the exception that only the temperature inputs and temperature disturbances are included in this data set.
Next, for each time step, we calculate the network object's output layer values (that is the predicted next-step state of the output temperatures) by calling its computing method. The computing method receives as argument a vector with all the temperatures defined in the network's input layer and returns a vector containing all the temperatures defined in the network's output layer. Thus, in the first call of this function we will give it a vector containing all initial condition of temperatures. In the second call, considering we are dealing with a first-order model (only the current output temperatures are used in the prediction) the output temperatures will receive the values from the output layer obtained in the first call. The input and disturbance temperatures, on the other hand, will always have the real value measured in the current time step.
The algorithm used to validate a model of second order (previous indoor temperatures are included in the input layer of the neural network with current indoor temperatures, disturbances and temperature control) is shown below. Temperatures and disturbance input, located at the end of the input Layer [i][k] vector are not changed in the iterations because they were already pre-defined as actual measured data for each time step. The duration of the step is the same as that specified when creating the model.

```
for(i = 0 ; i < SessionDuration / StepTime ; i++)
{
   if (i == 0) //initial iteration
   outputLayer = NeuralNetwork.Compute(inputLayer[i]);
   else
  {
    for (int k = 0; k < NumberOfIndoorSensors; k++)
      {
        if(order == 2)
       //Sets previous indoor temperatures
        inputLayer[i][k + NumberOfIndoorSensors]=inputLayer[i][k];
        //Sets current indoor temperatures
        inputLayer[i][k] = outputLayer[k];
       }
     outputLayer = NeuralNetwork.Compute(inputLayer[i]);
  }
}
```

9.1.3 Advanced functions

With the experiments, more features and enhancements to the software, in this section, we present improvements we have implemented in this software.

Signal processing of the WSN sensor data

After the data acquisition achieved, it was possible to perceive that the sensor readings are sensitive to a significant amount of noise, causing small fluctuations in temperature curves at high frequency measured by each sensor. In the earlier version, the noisy data were used directly to construct the input-output pairs, resulting in a spread of noise in the model.
As an attempt to make the data for the most representative in relation to the phenomenon measured learning, it was introduced in a software that the user to can choose the type and cutoff frequency and adjust the relevant parameters of the filter used. For example, we have filtered the acquisition of a WSN with a Butterworth filter of order 3 and cutoff frequency at 20Hz. The result is shown in Fig.9.11.

Evaluation of model prediction errors

This function is capable of providing a digital display of the model prediction error regarding a set of measured data. Knowing that the output from the neural network model is discredited, a method was established wherein the dividing of the time discretion by an arbitrary number and by using the linear interpolation methods of least squares, the tool computes the mean squared error between the original model's prediction and real measurements. It can therefore numerically evaluate the performance of the trained models and makes it possible to select the most representative model among all the models trained with different training parameter combinations (see Fig. 9.12).
The main functions of this software was introduced in this chapter. The training method detailed model will be presented in the following chapters.

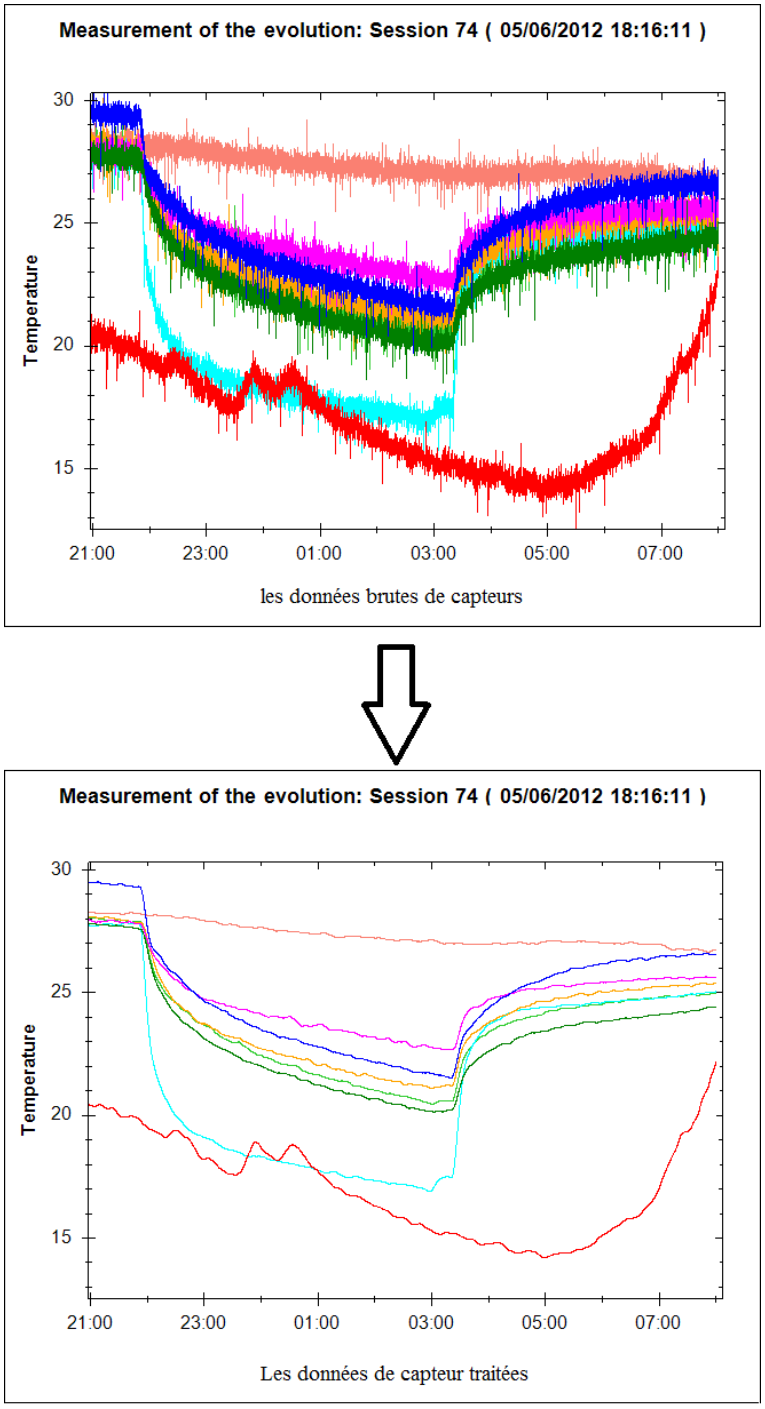


FIGURE 9.11 – The filtered sensor data

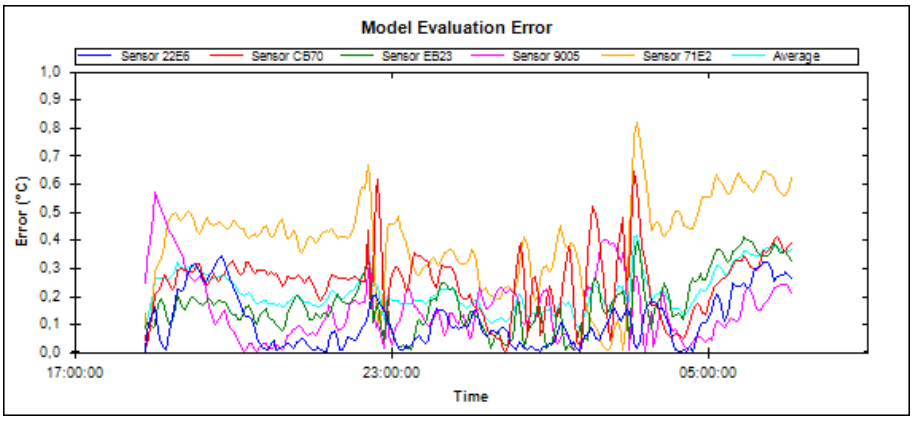


FIGURE 9.12 – Error curves of each sensor in one session (MSE = 0,27°C).

Troisième partie

Combination of WSN and ANN : A new approach of modeling

10

Combination of WSN and ANN

10.1 Combining WSN and ANN

WSN offers a practical solution of distributed sensing, processing, communication and control while ANN's self-adaptivity and nonlinear mapping ability make it more advantageous in modeling nonlinear system or system with unknown dynamics.
We think that the combination of WSN and ANN can be a powerful modeling solution. First, a trainable ANN model built itself from experimental data, thus, sufficient data sources are necessary to obtain an accurate ANN model. The rich sensor data from WSN in return can be used in training the ANN. Also, WSN data based ANN modeling has high practical values : the behavior of certain system is very complex and difficult to analyze, especially when many nonlinear and time-varying effects are present. For example, the dynamic behavior of building in response to environmental factors. In such case, it is nearly impossible to obtain an accurate mathematical model with limited system parameters. Thus, an adaptive ANN thermal model using WSN sensor data can be a reasonable choice. Our proposed modeling solution is presented below in Fig. 10.1

10.2 WSN based ANN thermal modeling

In this section, a concrete example is presented to show the necessity and feasibility of combining WSN and ANN.
As mentioned in the introduction and the first part of this thesis, energy saving in old buildings can provide significant benefits. A common fact of old buildings is this : most of the old buildings are badly isolated. Thus, the building's dynamic thermal behavior in response to environment factors is very complex due to many existing nonlinear and time-varying heat transfer effects. Thus, it is difficult to characterize the indoor heating/cooling effects under different environmental conditions. The lack of understanding on the thermal effects inevitably leads to a squandered usage of energy.

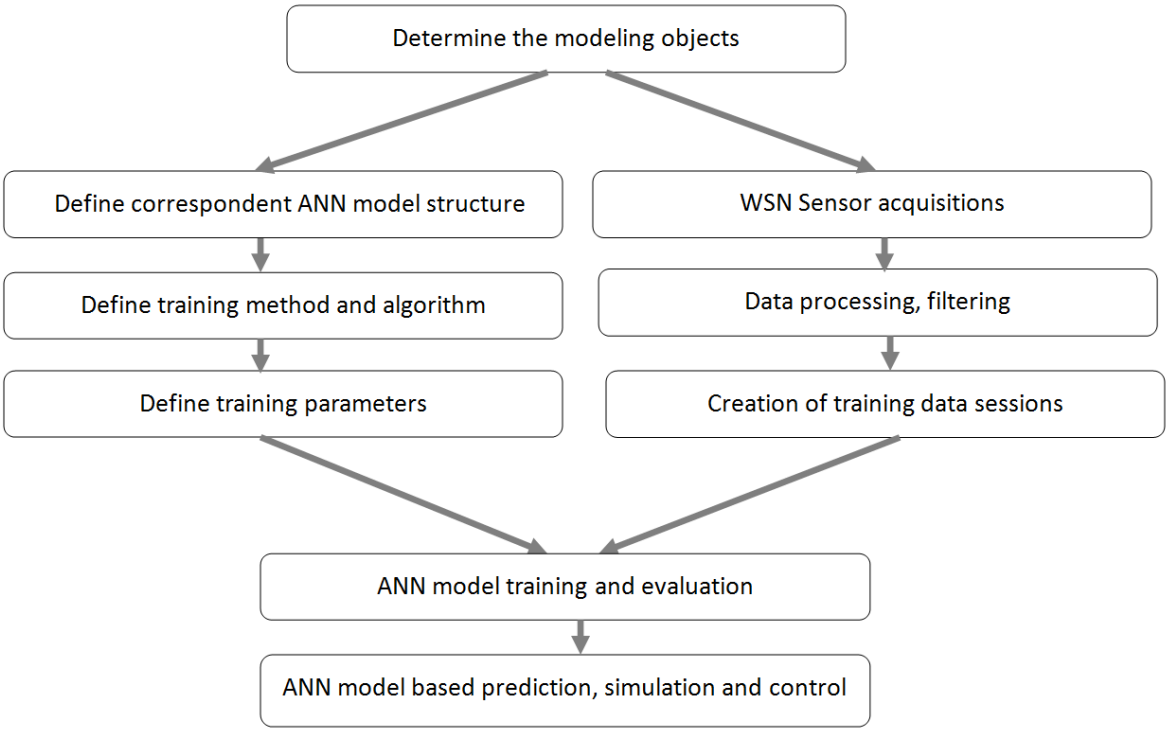


FIGURE 10.1 – Standard procedure of modeling using WSN and ANN

The traditional way to optimize the old building's energy efficiency is to renovate the building with new construction material, which is often expensive and time-consuming. One aim of this thesis is to demonstrate that WSN based ANN model can be used in solving practical problem. In this case, by establishing an accurate adaptive ANN thermal model, the indoor thermal response can be characterized. Thus, the existing heating/cooling system's efficiency can be optimized.

Previous works have been made on the performance of the WSN signal through building constructions [64]. It shows WSN is a suitable solution for a more widespread sensing in buildings. Sensors alone with WSN are usually served as acquisition and monitoring tool. They have been increasingly applied in recent research including building monitoring [65], building occupation detections [66], evaluation of thermal performance on insulation materials [67] and energy consumptions [68]. With the growing interests in modeling and predictions, Artificial Neural Network (ANN) appears frequently in building related research [69]. Previous studies also outlined that ANN model outperformed Auto-Regressive models with eXternal input (ARX models) in predicting the indoor temperature because the ANN models are more sensible to the nonlinearities of the thermal effects in the buildings rooms [18, 19].
Indeed, among most previous works on building thermal modeling, a limited number of sensors have been used, mainly due to the fact that the data acquisition by

cabled sensors is usually expensive and time-consuming. These researches focused on macroscopic modeling of buildings. It considers large rooms as big thermal capacitors with homogeneous temperature. With these models, indoor temperature prediction is hard to realize in certain specific zone of a room, especially in rooms with partitions. ANN used for thermal mapping of a cold storage is presented in [70]. However, this model does not involve outdoor conditions, and it can not provide in-time predictions of indoor temperatures.

So far, WSN and ANN together have hardly been used in indoor building thermal modeling. However, we think there are two main advantages of applying WSN and ANN in building room thermal modeling. First, the nature of WSN and ANN make them a practical combination : on one hand, WSN could be easily and rapidly implemented, providing a huge quantity of sensor data. These data sources in return could be essential for the ANN to identify a fine grained thermal model. Second, they have high practical values : mathematical thermal modeling approaches [8] are usually used in general simulations. They are hard to be applied in some practical applications. It is mainly because these models are based on elements such as room thermal capacitances/resistance, airflow rate, heat transfer coefficient, heat gain coefficient, etc. These parameters are difficult to measure precisely in old buildings. Also, as we mentioned above, the dynamic behavior of building room is very complex, it is nearly impossible to obtain an accurate mathematical model with limited number of system parameters. The WSN system, on the contrary, is highly transplantable as it could be quickly equipped in any buildings to gather real-time thermal data. Additionally, with its self-adaptive learning and mapping ability, ANN can directly simulate the relations between building thermal affecting factors (heating source, solar radiation, outdoor temperature) and indoor temperatures. Based on the two reasons above, we believe that the combination of WSN and ANN can be a valuable solution for indoor thermal modeling.
Admittedly, some existing techniques can be found for the same purpose of energy efficiency. Instead of using WSN and ANN combined solution, some may interest in the usage of smart meters or other simpler modeling techniques. Smart meters can measure real time or near real time energy consumptions. However, it can not provide detailed indoor climate prediction for users. Uncertain and nonlinear thermal effects have long been recognized in buildings. Furthermore, previous studies outlined that ANN model outperformed Auto-Regressive models (ARX models) in predicting the indoor temperature because the ANN models are more sensible to the nonlinearities of the thermal effects in buildings [18, 19]. Different techniques have their own advantages and shortcomings, here, we have to point out again that the WSN and ANN based modeling solution is totally proposed from a consumer perspective : WSN as a very cheap, low-power, easy-to-use, miniature electronic devices which can be installed quickly in most old buildings while ANN's universal approximating ability makes it possible in generating adaptive thermal models under different environmental and indoor conditions. This means that the same system can be easily deployed anywhere without modifications of hardware or software. With very low expenses, users can conveniently characterize the ther-

mal response of their living space, maximize the comfort and optimize the energy consumptions. We think the practicality and wide-applicability of the proposed solution make it distinctive from other techniques.

The room thermal model established in this work is a typical Multi-Inputs Multi-Outputs (MIMO) system : we consider the indoor activated heating or cooling system as a main control input, the measured outdoor temperature and solar radiations as perturbation inputs. The temperatures on each point of the building room are henceforth defined as the outputs.
To fit this MIMO system, a three-layered Back-Propagation Neural Network (BPNN) is chosen as the ANN model structure. The BPNN proposed by Rumelhart *et al.* [23] is one of the most commonly used neural networks for its simplicity and efficiency [71]. Hecht-Nielsen demonstrated that a three-layered BPNN is capable of approximating any continuous mapping [72]. Previous research has also confirmed its performance in engineering applications [2]. Correspondent to the MIMO system, the BPNN contains three layers : an input layer, a hidden layer and an output layer. This determines a first order BPNN structure. In order to increase the model's accuracy, we also proposed higher order models : if we define the standard interval time between every acquisition as T_i, we can form a second order model by involving the previous output temperature $t - T_i$ in the input layer (see Fig. 10.2), or even a third order model by including both t-T_i and $t - 2T_i$ in the input layer, etc.

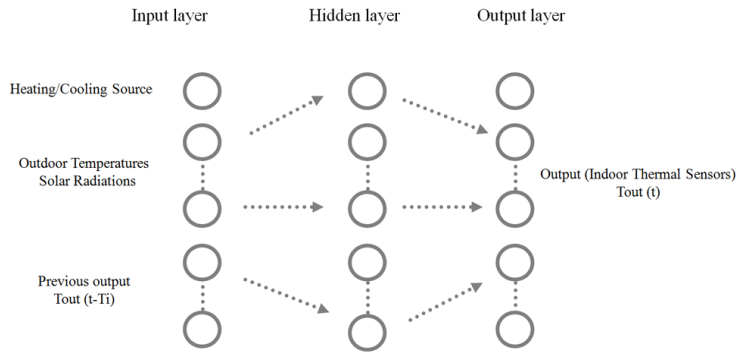


FIGURE 10.2 – A second order ANN model.

The inputs of the model affect the outputs with different weights. During the training phase, the neurons will regularly correct their weights by calculating the error between current ANN model's output and the expected output. After sufficient learning iterations, the error can be considered negligible. At this point, we consider the model is ready and well taught. In this work, we build our network model with a default training iterations of 500000 times and the default error threshold is set to 0.00003. Detailed training parameter configuration can be found in Fig. 10.4 This training algorithm is presented below. As the mathematical representation of

the complete model has been already presented in Part 2 of this thesis, we take a simplified single neuron model (Fig. 10.3) as an example to explain the model's training procedure. If we consider that the two main inputs are Indoor heating/-cooling control sources C_s, the ambient temperature Ta, the only output is the indoor temperature T_{out}. Then, the activation function of the neuron is $f(x)$ and it is responsible for normalizing the incoming values from the previous layer.

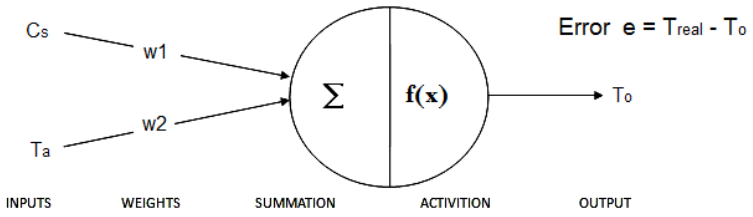


FIGURE 10.3 – A single neuron ANN model

A sigmoid function is chosen as activation function $f(x)$ (see Eq. (3.12)) in the soft. A three-layered BPNN using sigmoid activation function is a universal approximator [73, 74].
Furthermore, its nonlinearity and the computational simplicity of its derivative make it the reasonable choice in modeling non-linear indoor temperatures variations.
If we define :

$$u = C_s w_1 + T_a w_2 \tag{10.1}$$

then, we have the output of the single neuron equals to :

$$T_{out} = f(C_s w_1 + T_a w_2) = f(u) \tag{10.2}$$

After the inputs have reached the output layer through all the neurons in the network, these outputs will be compared with the measured output values. The difference is defined as the error signal e of the output layer neurons in Fig. 10.3. The error signal propagates backwards to the input layers. After the error signal e of every neuron in the model is computed, the weight of each neuron adjusts itself through an optimization gradient descent method, where η is the learning rate. In this way, the error decreases :

$$w_1^{'} = w_1 + \eta e \frac{df(u)}{du} C_s \tag{10.3}$$

$$w_2^{'} = w_2 + \eta e \frac{df(u)}{du} T_a \tag{10.4}$$

As for the number of neurons in the hidden layers, we estimated it with the equation below :

$$Num.\ of\ Hidden\ Neurons = \frac{1}{2}(N_i + N_o) + \sqrt{N_{(tp)}} \tag{10.5}$$

where N_i is the total number of network inputs, N_o is the number of outputs, $N_{(tp)}$ is the total number of training data patterns. This formula has been used in several engineering problems for modeling and prediction with good results [75].
So far, there is no determined rules to define the best network training parameters, the reasonable method is still "trial and error". Thus, we give two possibilities in this software : 1. Manual Definition, the parameters can be defined by users, their default values are presented on the software in Fig. 10.4.
2. Auto Evaluation, the parameters can be selected by a cyclic trial algorithm : dif-

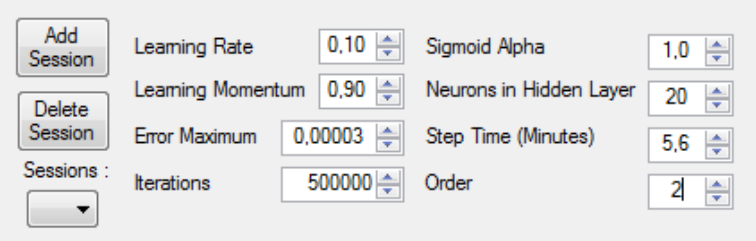


FIGURE 10.4 – The training parameters by default

ferent sets of training parameters are used to train network models. The software will select the parameter combination which leads to the minimum prediction errors regarding the training sets. Detailed training method can be found in author's published work on WSN based ANN model network training[76]

10.2.1 Modeling experiments

Experiments were divided into two phases : First, to evaluate ANN the modeling performance with WSN, experiments have been done on a prototype. Latter, we carried out experiments in real building rooms.

Thermal modeling experiments on prototype

The prototype is a cubical corrugated box. To simulate a room inside a building, we have covered our prototype with same boxes above and four around, leaving only one side towards thermal source. The prototype is placed in a room where the indoor temperature is considered stationary and no circulating air flow presented

during experiments. It simulates several separated thermal resistance zones just like a building. Admittedly, the thermal characteristic of cubical corrugated box differs from the construction materials. The heat transfer coefficient, also known as the U-value,[7] of the cubical corrugated box (board thickness 4.23mm) is $2.21W/Km^2$ while the U-value of the building wall (Aggregate Concrete, wall thickness 0.3m), according to our calculation, is about $5.83W/Km^2$. However, they share similar heat transfer principle [77, 78]. The experiment is described below : Three sensors (in blue in Fig. 10.5.) are located outside the prototype as input disturbance, measuring ambient temperatures. One sensor (in red in Fig. 10.5.) facing the thermal source is considered as control input. Six thermal sensors (in pink in Fig.10.5.) are put horizontally inside the prototype to collect inner temperature variations, giving the output value of the system (see Fig. 10.5).

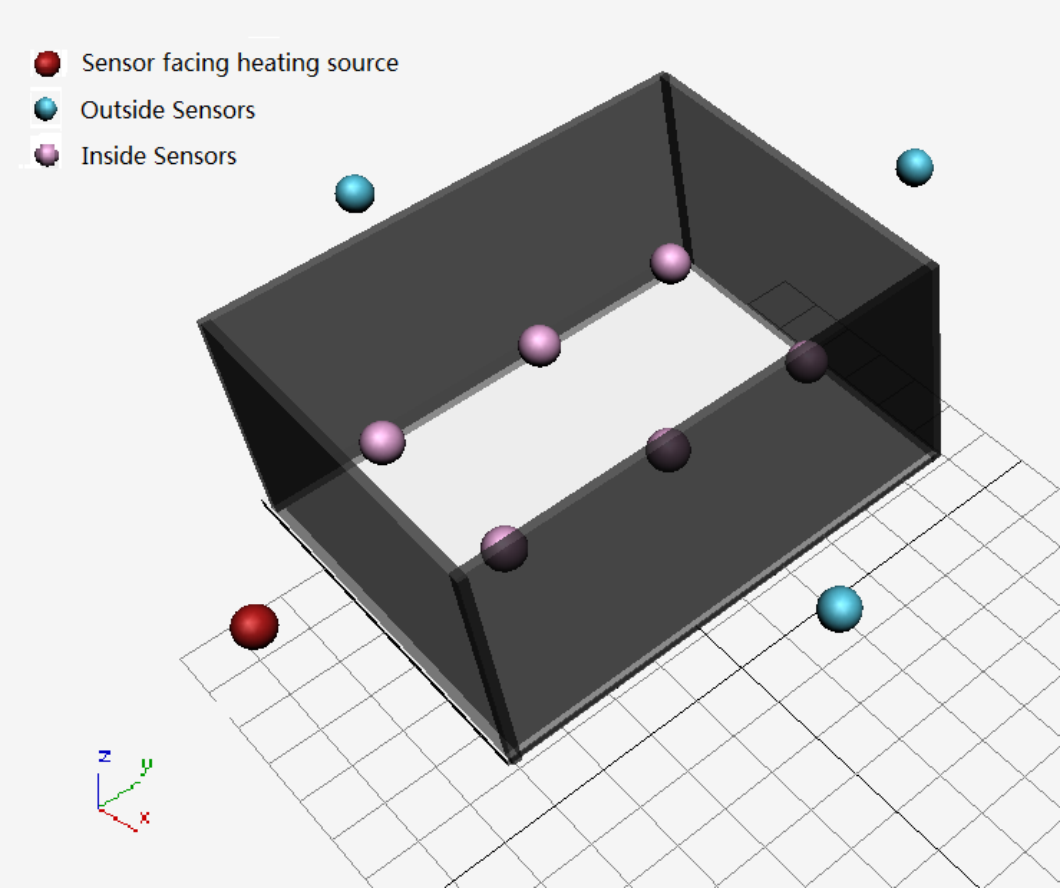


FIGURE 10.5 – A 3D model of the prototype

7. The U value is an energy efficiency indicator. It refers to the heat transfer coefficient (thermal transmittance) of a structure.

Modeling experiments in building room

Later, experiments have then been carried out in real building rooms E106 (Width 5.5m, Length 7.2m, Height 3.15m) located in the building E of IUT in the campus of Université de Toulon (LAT 43°123′ N, LONG 6°11′ E), city of La Garde, south France. The building was built in the year of 1968 and has not yet been renovated. It matches the definition of old buildings : no modern HAVC system equipped inside and it is badly isolated. One infrared sensor is installed outside the building room to measure solar radiations. One thermal sensor is placed outside the building while the other is placed in the corridor. These two sensors collect the ambient temperatures outside the room. The operation status of heating/cooling source (Toshiba RAS-167SKV-E3) is considered as the main input. Six thermal sensors are placed horizontally at the height of 1.10m in the room where the most human activities take place [8]. Since our WSN acquisition system is easily movable and adaptable, the experiments are mainly carried out in this room (See Fig. 10.6). The ANN models are automatically trained by every acquisition carried out and stored in the database.
The results of the modeling in prototype and building room both positively supported our previous assumptions that the combination of WSN and ANN can be an effective and workable solution modeling. These modeling results will be presented in Chapter of Results and Discussion.

8. According to air-conditioning industries in France, the room temperature is usually evaluated at the height of 1.10m.

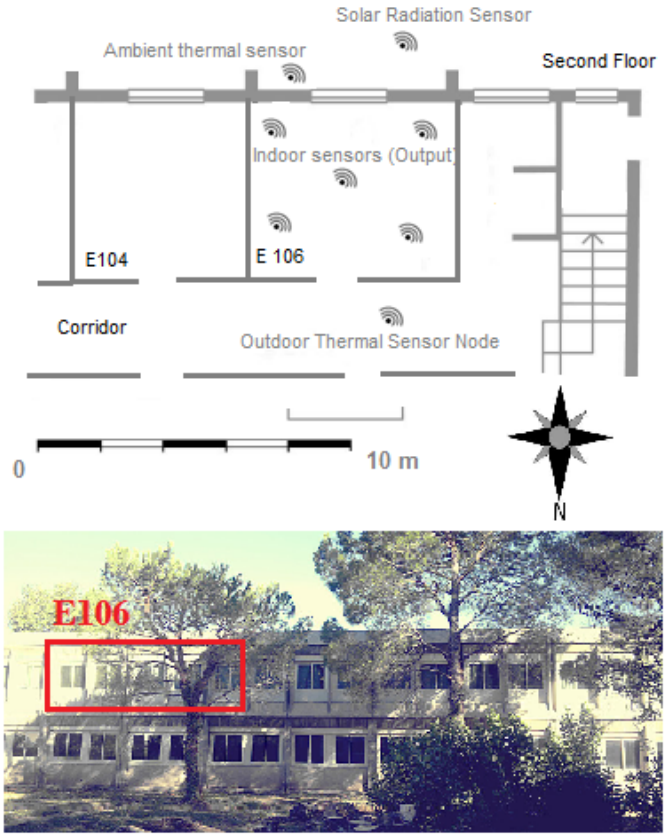


FIGURE 10.6 – Experiments in the room of E106 in University of Toulon

11

Multi-Pattern Cross Training

11.1 Principle and application of Multi-Pattern Cross-Training

As described in the previous chapter, we have conducted experiments with WSN based ANN thermal modeling and the results are very positive. However, during the phase of further testing established models, we found a practical problem : on one hand, long term WSN data acquisition causes high data redundancy which leads to low training efficiency and high computational cost in modeling. On the other hand, short acquisition with limited duration is usually obtained under certain specific conditions. Because environmental conditions change greatly, short acquisition as a training source only contains limited information. The result is that the trained ANN model has poor generalization performance. The model shows low prediction error against its training data, but relatively higher predictions error against other test data measured under different conditions. So, the question we raised here is "Is it possible to use a single ANN model to obtain a more complete understanding on the system's behavior ?"
Indeed, there are existing well-known methods developed for training neural networks [79, 80]. however, they are not aimed to build a more comprehensive ANN model. Perturbation method and sensitivity analyzing [81] can be used to improve the training efficiency by reducing the redundancy as well as the input layer dimension. This method is very useful when the some of the network's inputs are correlated, which is not the case for us here.

The term "Cross Training" was originally defined as a training method for athletes to improve their competitive performance in a certain sport by systematically training in a variety of different sports [82]. Later, Gökhan H.Bakır has introduced this concept into his research on SVMs training [83].
Here, we plant this concept into a new ANN model training method "Multi-Pattern Cross Training"(MPCT) [76]. Using the internal properties of neural networks, this training method is capable of merging knowledge from different training data patterns into a single network model. Thus, by exploring the generality in the system

behaviors, it can adequately be used to describe a more complete phenomena. Our ANN modeling methodology is presented in Fig. 11.1 while the MPCT method is highlighted in the bold black frame in the right part of Fig. 11.1

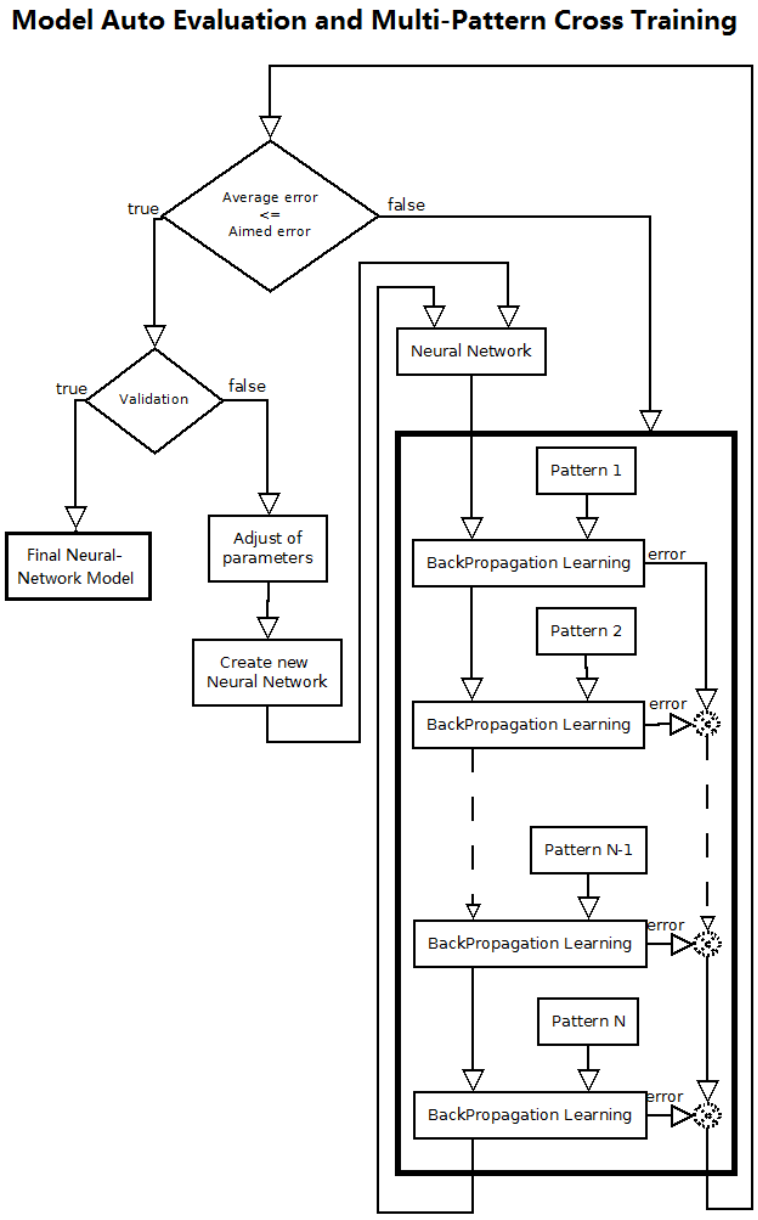


FIGURE 11.1 – Multi-Pattern Cross Training Method.

The mechanism of MPCT method is presented below : The training source consists of WSN sensor data which is called a session of acquisition (pattern) $S_{(1)}$ [9]. Acquisitions under different initial conditions have been made, so we have different sessions of acquisitions "$S_{(p)}$". The error function E is defined as the sum of squares on the differences between the WSN sensor measured data (measured output) and the network output y_j. If we use on-line learning mode, the error obtained from the data set n of pattern $S_{(1)}$ is :

$$E_{S_{(1)}}(n) = \frac{1}{2}\sum_{j=1}^{j=N_L}(S_{(1)}(j) - y_j)^2 \tag{11.1}$$

j is one output layer neuron, N_L is the number of neurons in the output layer and n is the sequence number of data set in a session of acquisition. Then,the weight update value $\Delta\omega_{ij}(n)$ as :

$$\Delta\omega_{ij(S_{(1)})}(n) = -\eta\frac{\partial E_{S_{(1)}}(n)}{\partial\omega_{ij}(n)} \tag{11.2}$$

η is the learning rate which represents the step size on this gradient direction. At the end of this training epoch, instead of recycling the weight update with pattern $S_{(1)}$, we lead the training procedure to the training data sequence 1 in pattern $S_{(2)}$. Here, we define the last data pattern sequence of session 1 is N_1. We also introduce the momentum term to incorporate the past weight updates into the present weight update. Thus, the new weight update at the beginning of the second training epoch can be put as :

$$\Delta\omega_{ij(S_{(2)})}(n) = -(1-\alpha)\eta\frac{\partial E_{S_{(2)}}(1)}{\partial\omega_{ij}(n)} + \alpha\Delta\omega_{ij(S_{(1)})}(N_1) \tag{11.3}$$

where α is the momentum parameter which determines the amount of influence from the previous weight update. n refers to the last data pattern sequence in session 1.
So, if we define the two main variable in the equation below (Eq. (11.4) and Eq. (11.5)) as the session number p, and the sequence number n while the total session number is P and the last sequence count for session p is N_p, we can thus translate the MPCT method into computational language below :
If we define a nested loop with one outer loop and one inner loop :
The Outer loop is : for(p=1 ; p<=P ; p++).
The Inner loop is : for(n=1 ; n<=N_p ; n++).
The general expression of MPCT method can be put as :

$$\begin{aligned} &Outer\ loop\ (Inner\ loop\ (\Delta\omega_{ij(S_{(p)})}(n) = \\ &-(1-\alpha)\eta\frac{\partial E_{S_{(p)}}(n)}{\partial\omega_{ij}(n)} + \alpha\Delta\omega_{ij(S_{(p)})}(n-1))) \end{aligned} \tag{11.4}$$

9. A session of acquisition consists of sets of time-series sensor data collected during a certain period

Specially, when one training epoch is completed and the next epoch begins with the next data session, the weight update is equal to :

$$\Delta\omega_{ij(S_{(p)})}(n) = -(1-\alpha)\eta\frac{\partial E_{S_{(p)}}(n)}{\partial\omega_{ij}(n)} + \alpha\Delta\omega_{ij(S_{(p-1)})}(N_{(p-1)}) \tag{11.5}$$

The training procedure continues until the network error is less than the target error which is considered sufficiently small.
To achieve expected modeling results with MPCT method, some preparations are required. Initially, each training data session should be verified so that they contains well-defined information of system behaviors. At the same time, the neural network should be capable of recognizing these patterns' behaviors. Secondly, it is essential to ensure that the neural network could discern between different sessions of acquisition. Therefore, it is necessary to have at least one input containing a particular value for each session. This value allows the network to distinguish between the difference in patterns. Thereby preventing the system from converge into one intermediary solution. One special point that has to be made here : in order to make the ANN model more practical, the selecting of data session should respect the phenomena's basic statistical distribution. For example, in our work of indoor thermal modeling, two inputs are based on the outdoor condition. As our experiments are carried out in the south of France, the presence of measured environmental data session should obey with local Mediterranean climate statistical characteristics. For example, data sessions obtained under extreme weather conditions should be avoided in the training data source.

11.1.1 Optimum network training parameters with MPCT

As shown in Fig. 11.1, the training of ANN model with MPCT requires different sessions of acquisitions as a source of training. To find the best training parameters, we provide a solution called automatic evaluation : the parameters can be selected by a cyclic algorithm which is also presented in Fig. 11.1 : different sets of training parameters are used to form network models, the model's prediction errors on its training data is then automatically calculated by the software. The combination of parameters which leads to a minimum prediction error will be chosen by the software.

The results of the MPCT training is shown in Chapter 12 of this thesis.

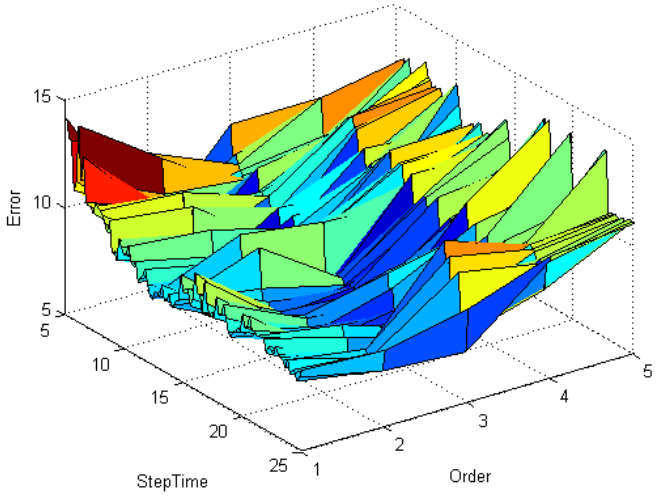


FIGURE 11.2 – Auto-evaluation of ANN training parameters

12

Model predictive control

12.1 Model Predictive Control (MPC)

While the linear model predictive control (LMPC) has been popular since the 70's of last century, a growing attention constantly motivated the control practitioners in the field of nonlinear model predictive control.
A major shortcoming of linear MPC is that linear dynamic models are used in the description of the the control object's (plant) behavior. The Linear MPC can be inadequate in front of plants with high nonlinear dynamics or large operating region with limited nonlinearities. Increasing demands from industries and productions have triggered the development of nonlinear MPC in which a more accurate nonlinear dynamic model is used for the control process.

12.1.1 Principal theory of MPC

The basic idea of Model Predictive Control (MPC) can be understand as a discrete-time state control method. It is based on choosing the best control strategy over the whole control horizon. The selected actual control is applied to the plant and then repeated until new state information is available. We can define a MPC system here : the control at a consecutive sampling instant k is u_k, the system's outputs (or states) are x_k. Then, the output at sampling instant $k+1$ is :

$$x_{k+1} = f(x_k, u_k, w_k). \tag{12.1}$$

where, $x \in \mathbb{R}^n$ is the system's output (or state), the control input is $u \in \mathbb{R}^m$(for multivariable control system, $m > 1$) and the disturbance is $w \in \mathbb{R}^r$. Especially, a system without disturbance can be expressed as below :

$$x_{k+1} = f(x_k, u_k, 0) \triangleq f(x_k, u_k). \tag{12.2}$$

The control law can be described here : For every sampling instant, The difference between the model predicted value of outputs (or states) over the prediction horizon N and the referenced states' trajectory over the control horizon N_u are calculated for minimization. Only the first step of chosen control is applied to the process :

$$u_k = u(k|k) \tag{12.3}$$

if we define the change of control value as Δu, we have :

$$u_k = \Delta u(k|k) + u(k-1). \tag{12.4}$$

The procedure is repeated at the next sampling instant $k+1$ with the update of the process output (or states). The MPC's optimization problem can be considered as a finite horizon optimal control problem, if we define the Control Performance Function is J(x), we have its expression :

$$J(k) = \sum_{i}^{N} \|x(k+i|k) - \hat{x}(x+i|k)\|^2_{A_i} + \sum_{i=0}^{N_u-1} \|\Delta u(k+i|k)\|^2_{B_i}. \tag{12.5}$$

The first part of Eq. (12.5) represents the cost of prediction errors. The second part represents the changes of control values. So the optimal performance index function reads :

$$\begin{aligned} V_k &= \min_{u(k;N)} J(k). \\ &s.t. \\ x_{i+1|k} &= f(x_{i|k}, u_{i|k}), x_{0|k} = x_k, \\ x_{i|k} &\in X, u_{i|k} \in U, \\ \forall i &= 0, 1, \cdots, N-1. \end{aligned} \tag{12.6}$$

Based on the principle of MPC, a basic neural network based MPC (NNMPC) structure is presented in Fig.12.1. It mainly consists of a neural network plant model and an optimizer, The model will predict the process's outputs (or states) over the prediction horizon N and enable the optimizer to find a best combination of controls for the process.

Neural network based MPC (NNMPC)

One of the main drawbacks of nonlinear MPC is the computational cost of nonlinear optimization process. The model of neural network can be made offline, which can effectively reduce the computational cost online. In addition, the formation of a neural network model is unrelated to the length of the control horizon.
As a universal approximation, neural networks are widely used in both control and modeling for its non-linear approximating ability.

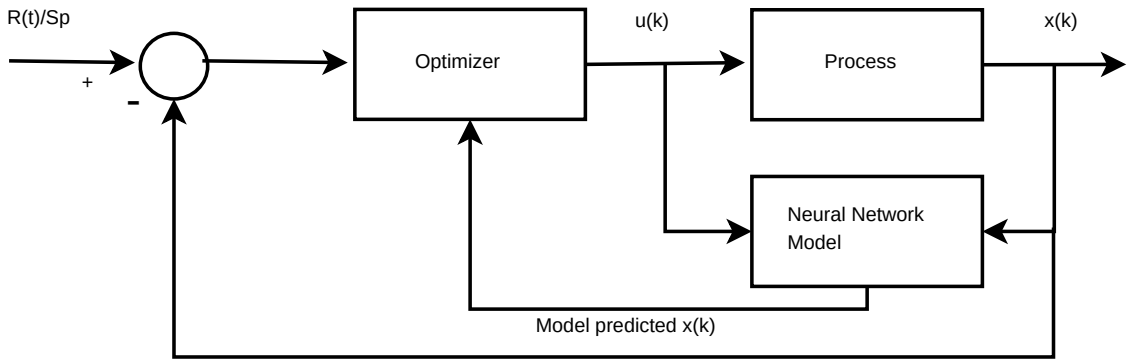


FIGURE 12.1 – Control schema of Neural Network based MPC

The basic idea of NNMPC is : one or more models of neural networks models are used as the predictor. The multi-step prediction horizon control can be achieved, and then the optimizer will choose the optimal control (See Fig.12.1).
If we define the Neural network model used for NNMPC is a three-layer back-propagation(BP) network, then, Eq. (12.5) can be rewritten to the following form :

$$J(k) = \sum_{i}^{N} \|x(k) - x_{nn}(k)\|_{A_i}^2 + \sum_{i=0}^{N_u-1} \|\Delta u(k+i|k)\|_{B_i}^2. \tag{12.7}$$

We can consider that the output of the neural network model is :

$$x_{nn}(k) = f_{nn}(U(k-1), Y(k-1)), \tag{12.8}$$

where f_{nn} is the transfer function of the neural network model replacing the model in Eq. (12.5). The relationship of the correspondent input/output can then be put as :

$$x_{nn}(k) = \sum_{i=1}^{N} W_i^o \sigma_i (W_i^u U(k-1) + W_i^y x(k-1) + b_i) + b, \tag{12.9}$$

where σ_i is the activation function of the neuron i in the hidden layer ; W_i^u represents the weight vector (row vector) for the neuron i from the input stored in $U(k-1)$; W_i^y represents the weight vector (row vector) for the neuron i from the input stored in $x(k-1)$; b_i represents the bias for the neuron i in the hidden layer ; W_i^o represents the weight of the output layer corresponding to neuron i in the hidden layer ; b represents the bias for the output layer.
The principle of NNMPC is introduced in this section, while in the following part of the thesis, we will give a detailed analysis of some shortcomings of NNMPC and we will demonstrate how to use the MPCT method to improve the efficiency of NNMPC.

13

Model predictive control using MPTC trained ANN models

13.1 ANN based model predictive control

One main objective of this thesis is to realize the optimal control using WSN based ANN models. Since the ANN models we have developed are thermal models of building rooms, the goal is to find the best strategy for control of the internal heating/cooling source of buildings that the optimized heating/cooling control can lead to a significant energy saving of buildings.

When a control system is designed, some initial steps must be taken to clarify the operation of the system after a selection of the best control strategy to work correctly. The first step is to define the control variables, i.e., parameters that can modify the process in order to obtain the desired results. Knowing that the aim is to control the temperature inside a room using information on the current temperature and the control variables of the air conditioner. It is obvious that the only modifiable factors are the parameters of the air conditioner, because there is no way to produce an immediate control of the temperature at a specific point of the room. Thus, the control parameters were determined by the types of the parameters most often found in air conditioning systems which are the desired temperature, air flow and the vertical angle of the heated air (see Fig. 13.1).

The control using neural networks is a research topic that has been developing for many years [84, 85].So there is a wide range of methods and applications with different characteristics. Among those that one that seemed most appropriate for inclusion were the opposite Adaptive Control Model and Model Predictive Control, these two methods have their characteristics, benefits and short-comings.
The control structure and the training of a neural network method of Adaptive Inverse Control is presented in Fig. 13.2 below. The principle of this method is to use a neural network to inversely predict the neural network's inputs using the given outputs. Thus, the opposite behavior of the system can be characterized. In this way, it could determine the optimized the control parameters to achieve the

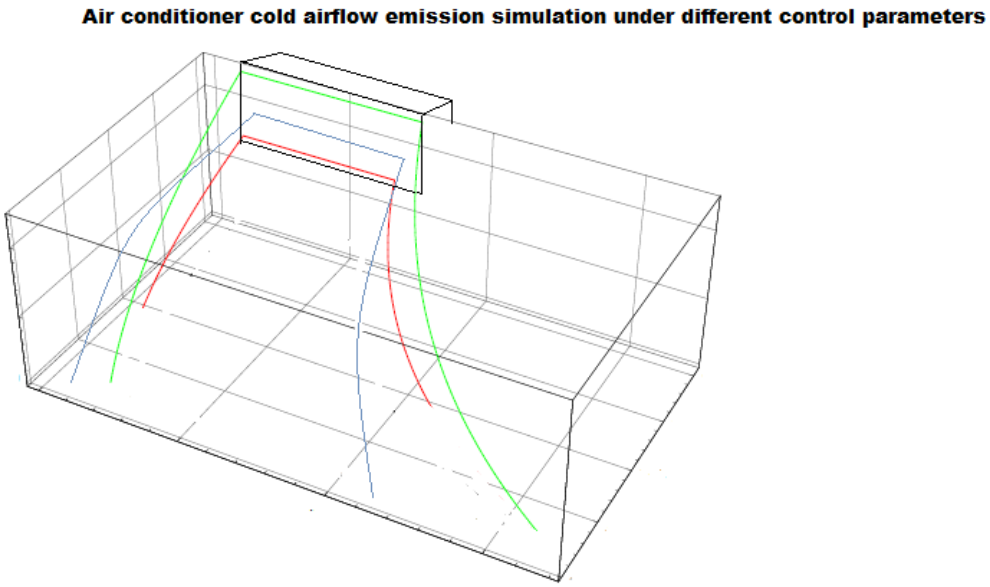


FIGURE 13.1 – Different air emissions under different control parameters

desired control result. To use this method, the implemented model must be reversible, also, the range of revised input-output must be small, so that the system can stay around the point of stability. The advantage of this strategy is that the control strategy is chosen entirely by the neural networks. For cons, the inverse model must be trained online during the ordering process which can lead to high computational cost. In addition, the strategy shows stability problems when used in an area outside the driven area.
The method we have finally chosen is the neural network based model predictive control (NNMPC) (presented in Chapter 9 of this thesis). The great advantage of this method is that the control stability is ensured.

13.2 MPCT method based Model predictive control

Model Predictive Control (MPC) is one of few advanced control techniques which has achieved great success in a variety of industrial applications[86, 87]. The advantages of MPC are dead time compensation, system constraints handling and efficiency in multi-variable process control. Usually, when the control process is nonlinear, a nonlinear dynamic model is applied to handle a larger operating range. Neural Network as a universal approximator is widely used in Model Predictive Control for its nonlinear mapping ability [88, 89, 90, 91].
Multiple models have been combined in adaptive control system for a main purpose of obtaining a more precise control, They use different models to characterize

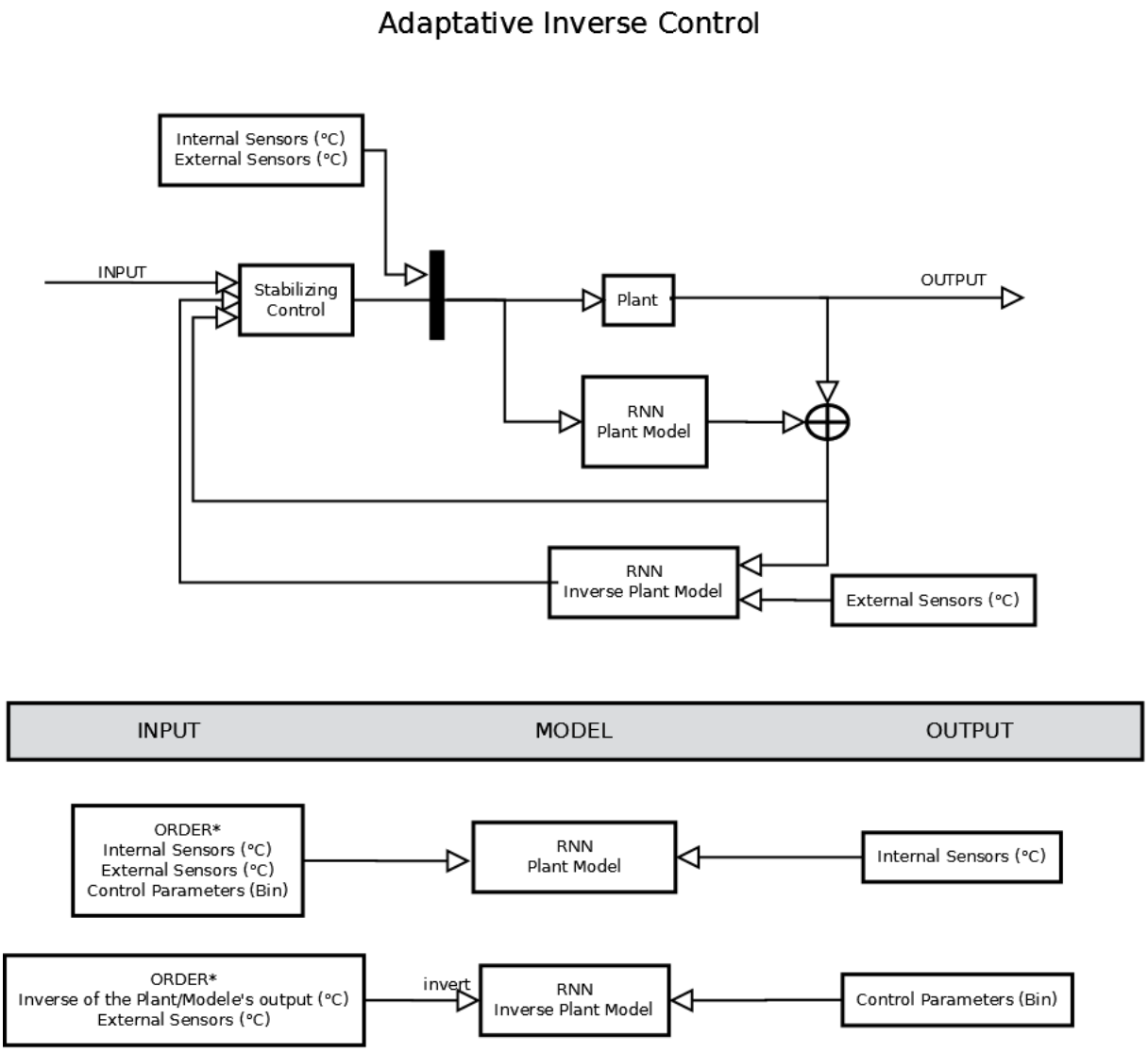


FIGURE 13.2 – Schema of Adaptive Inverse Control

different operating status under variable conditions on the whole control horizon [92, 93].Z. Ahmad, H.Jazayeri-Rad and A. Rabiee suggest that single NN model is not able to extract all relevant information from data set, thus, by deploying multiple neural network models in nonlinear MPC, better control performance can be achieved [94, 95].
However, there are no detailed computational methodologies or software implementation introduced in these previous works. Thus, the control efficiency can not be determined.
As a matter of fact, a main drawback of MPC is that it is usually used in process with slow dynamics where the sample time is in seconds or minutes. The integration of multi models will inevitably increases the MPC control system's computation load. Even the models are trained off-line, the MPC's online computation time will be longer than those with single model. This can limit the application of MPC in fast process. We consider that if a single neural network model can adequately understand and describe a system's general behavior, it will be a more efficient solution for MPC.
In order to compare the different control results of a multi-model based MPC, we also implemented a multiple neural network model based MPC structure in our control software interface (see Fig. 13.3.It uses a model selector to chose the best fitted ANN model to make the prediction. However, to chose the best fitted model, all existing models must be tested on every control step. This is the main extra computational load compared to MPCT trained single model based control scheme (see Fig. 13.4).
The test results will be presented in the next chapter of Results and discussion.

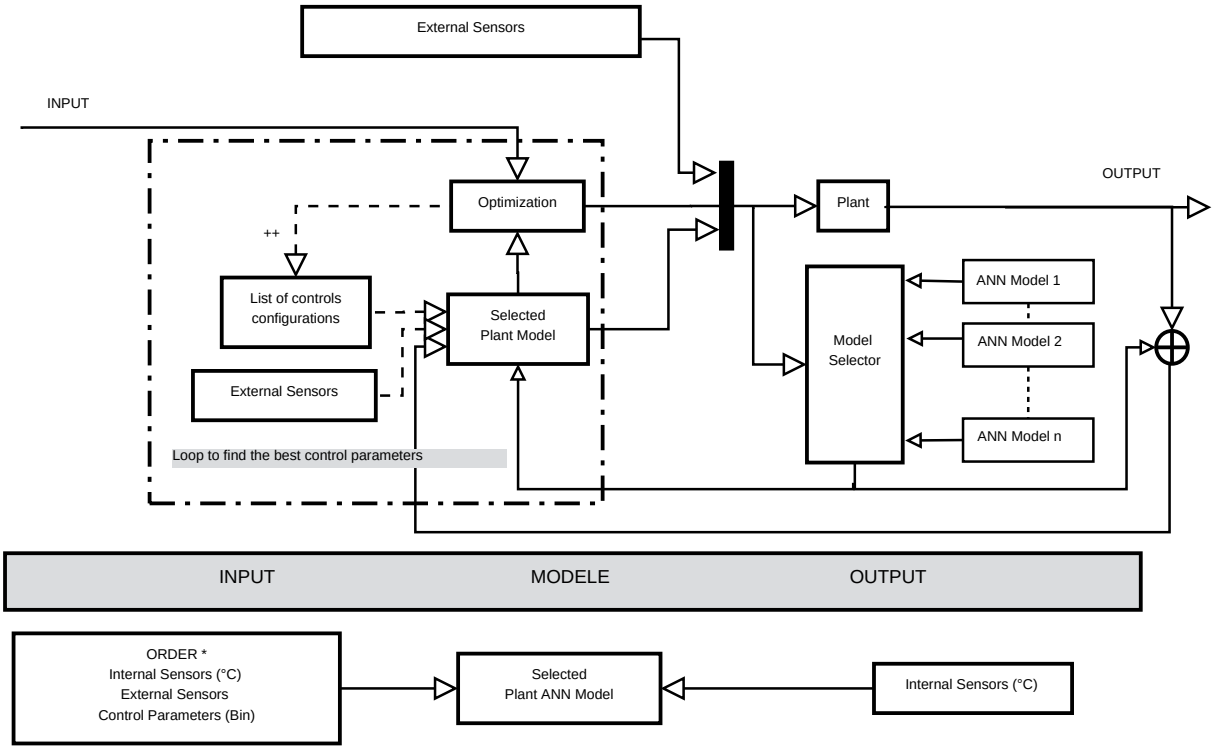


FIGURE 13.3 – Multiple ANN Model based MPC

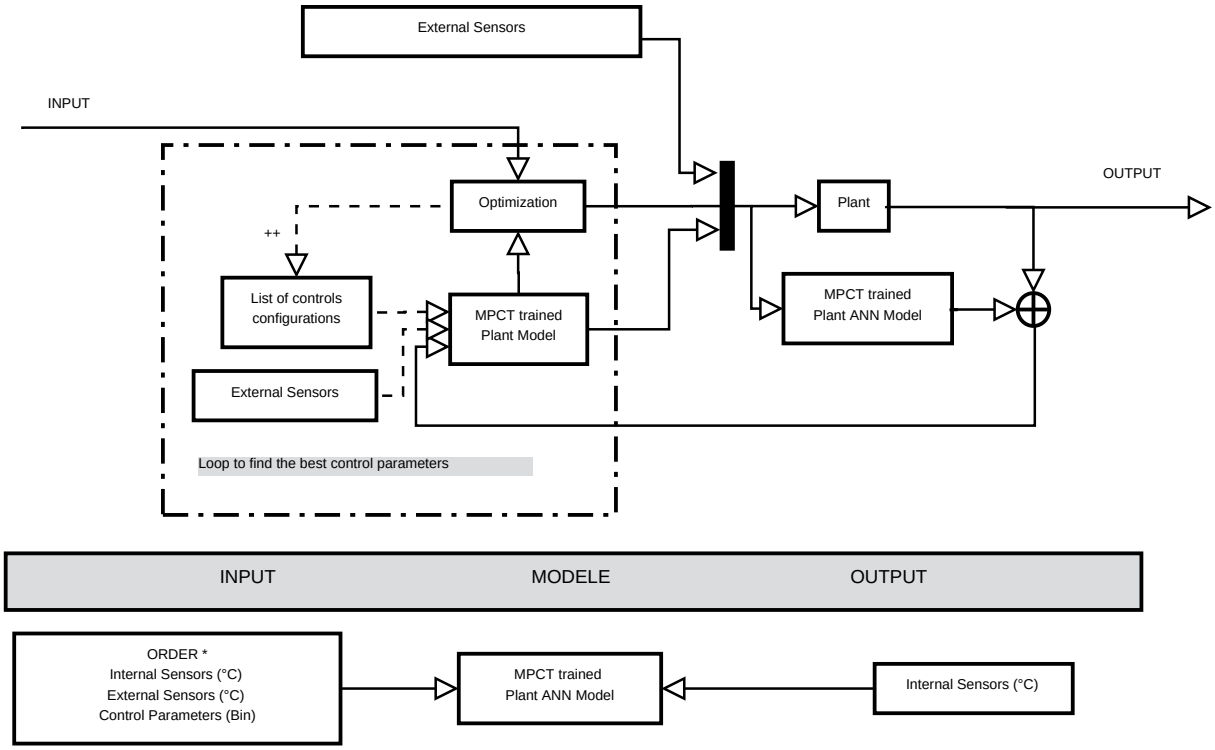


FIGURE 13.4 – MPCT trained model based MPC

Quatrième partie

Results and Perspectives

14

Results and Perspectives

14.1 Modeling results on prototype and buildings

In the first section, the general ANN thermal modeling results is presented. The test data pattern we used is obtained under similar conditions. The modeling results on prototype are presented in Fig. 14.1 We compare the model response regarding its training data in Fig. 14.1a , and regarding another measured test data in Fig. 14.1b. The originally measured outputs (multiple sensors) are colored, the model responses are in black.

In order to evaluate the modeling results, an Average Mean Squared Error (AMSE) of the ANN model are presented in Tab. 14.1.
This value is calculated as below (see Eq. (14.1)-(14.2)). If we consider the model

TABLE 14.1 – Evaluation of modeling performance on prototype :

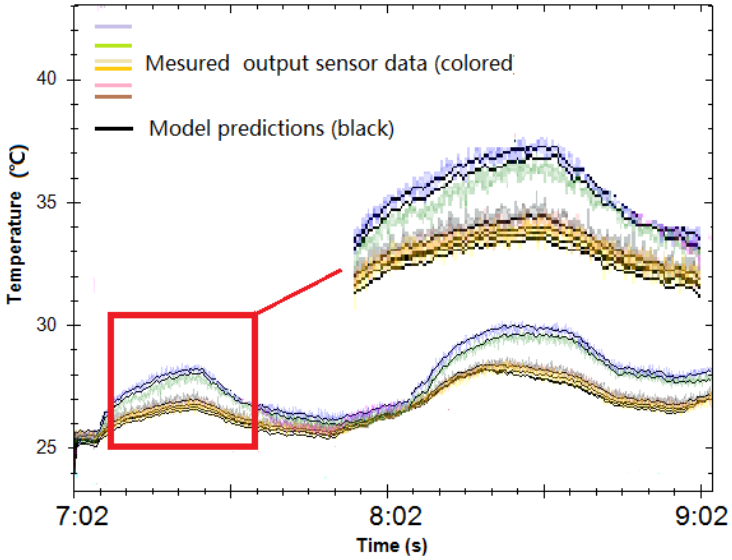
Performance Criteria	AMSE
Par rapport à données d'entraînement	0.11°C
Par rapport à données d'essais	0.17°C

predictions output is T' and the measured temperature is T, the number of measures is n ; we have the Mean Squared Error (MSE) for one model output (one sensor) is :

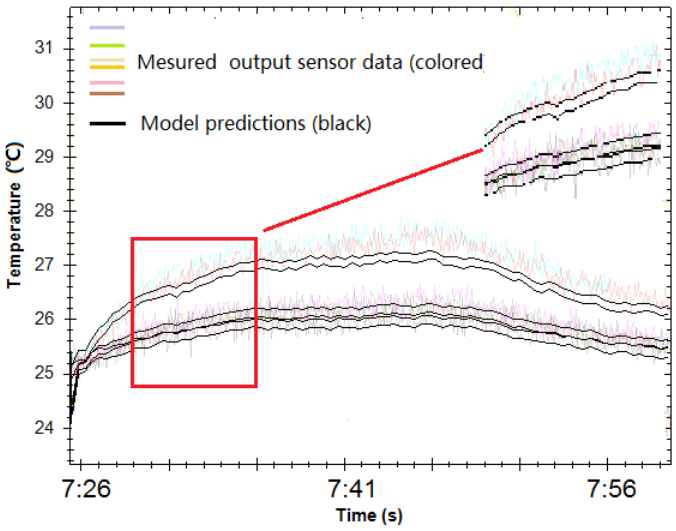
$$MSE = \frac{1}{n}\sum_{i=2}^{n+1}\Big(T^{'}(i) - T(i)\Big)^2 \tag{14.1}$$

the AMSE is the average MSE value of all the k outputs (k sensors) :

$$AMSE = \frac{1}{k}\sum_{i=1}^{k} MSE(k) \tag{14.2}$$



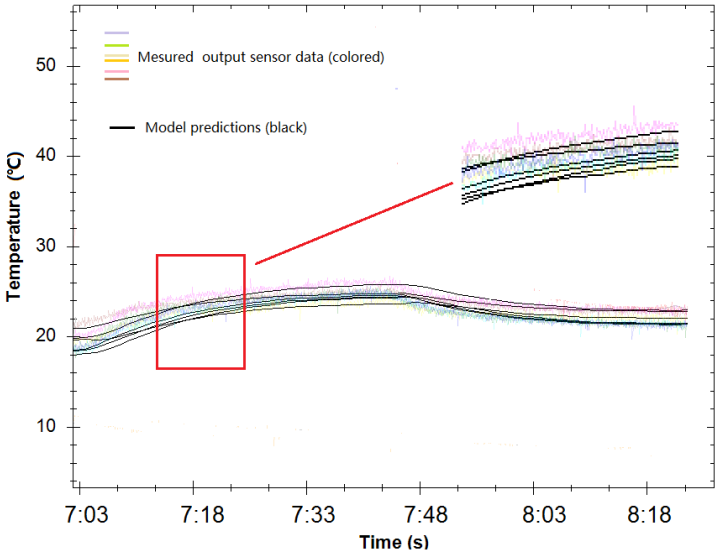
a. Model response regarding training data



b. Model response regarding test data

FIGURE 14.1 – Modeling Results on prototype

The results of real indoor thermal modeling results is presented in Fig. 14.2 , the model predictions are in black while measured data are lightly colored. The model prediction errors are presented in Tab. 14.2



Model response regarding test data

FIGURE 14.2 – Modeling results in buildings

TABLE 14.2 – AMSE of ANN thermal model based on real building experiments

Performance Criteria	AMSE
Par rapport à données d'entraînement	0.20 °C
Par rapport à données d'essais	0.27 °C

It is noticed that the AMSE of the building room thermal model is greater than the AMSE on prototype modeling which is due to the so called *similarity theory* [10].

In order to verify all models' performance, we calculated the AMSE of all the models created individually from 20 different sessions of daily acquisitions, the average prediction errors regarding its own training session is around 0.17 °C . Regarding the test data [11], the model's prediction error is also considered appropriate with a limited range from 0.20 °C to 0.29°C.

10. Similarity theory is the theoretical basis of model experiments. It is usually used in Fluid Mechanics to determine the requirement for experiments on prototype. In our case, the different model prediction errors is due to the discrimination of both initial conditions and boundary conditions on prototype and real buildings.

11. The test data is obtained under similar indoor/outdoor conditions

The modeling results on prototype and building room both positively supported our previous assumption. Based on the WSN thermal sensor data, ANN's self-adaptive learning and mapping ability makes it possible to provide accurate framework for thermal modeling of a building room. The raw thermal data and model predictions match the previous research on zonal model of buildings. It shows that the temperature distribution inside the building room is not homogeneous. Different parts of room react differently to the heating source which is mainly due to the thermal dynamics and air flow convention. This result indicates that the WSN and ANN are capable of capturing the thermal characteristics of each part of the room.
In order to characterize the thermal response of the room, a linear model derived from the previously established ANN thermal model is considered valuable. By simulating the step response of the trained ANN thermal models, the first order model approximation on each part of the room can be realized. Based on these simulations, the EITTC (Effective indoor thermal time constant, see Eq. (2.6)) can then be evaluated.

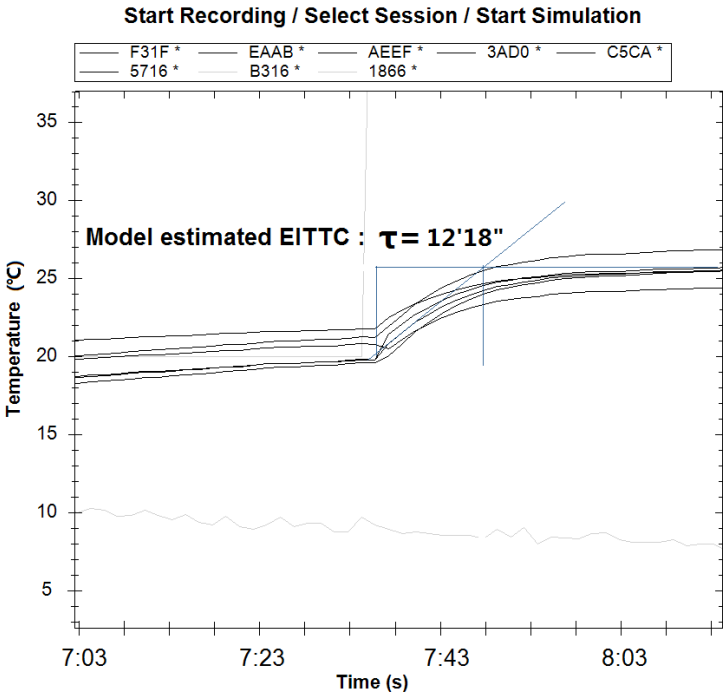
Two examples of the ANN thermal model's step responses are presented in Fig. 14.3. We find that the models' predictions on EITTC vary under different indoor/outdoor conditions

To verify these ANN model's predictions on indoor thermal time constant, we compared the ANN models' predicted time constants with real measured indoor temperature's characteristic time [12]. This comparison indicates the consistency between model's prediction and real indoor thermal response. The results are presented below in Tab. 14.3. Statistic computation has been made. The correlation rate between the model predicted EITTC and the measured room characteristic time is about 0.9526 (with a p-value p <0.041) and the average error is 0.85min with a standard deviation of "0.1978".

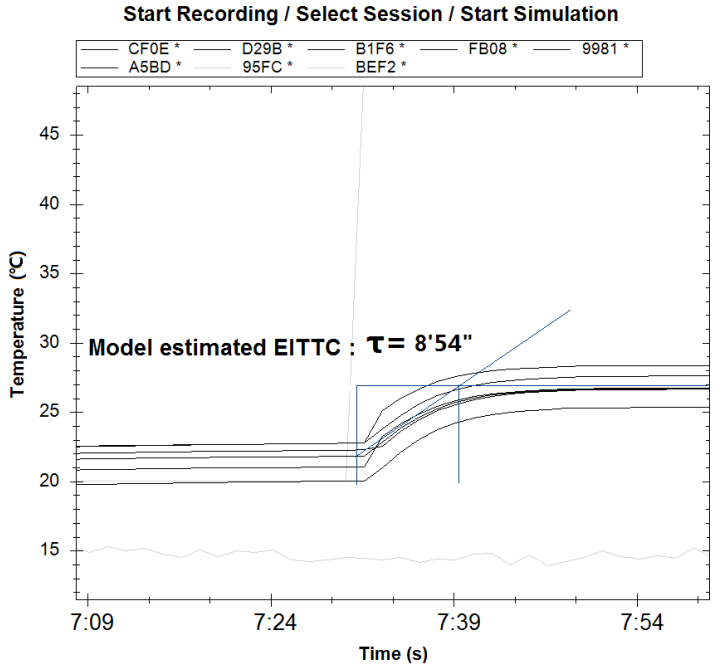
In Tab. 14.3, Measured CT refers to the measured room Characteristic Time. We noticed that the predicted time constants are shorter than the measured room characteristic time. The main reason is that these time constants are simulated from the step response of the ANN model. In reality, indoor temperature changes resulting from air-conditioning always present certain delay. We traced the average time constant of every ANN thermal models that created by daily acquisitions and we compared them with the outdoor temperature during the spring and winter of year 2012. The results are presented in Fig. 14.4

The models' predictions show that the EITTC of this building room varies from 8.5 minutes to 13.4 minutes. This result can be discussed from different perspectives.

12. We define the 63 percent of the time needed to heat the room from its initial indoor temperature to a final stable level, in other words, it is the measured indoor time constant

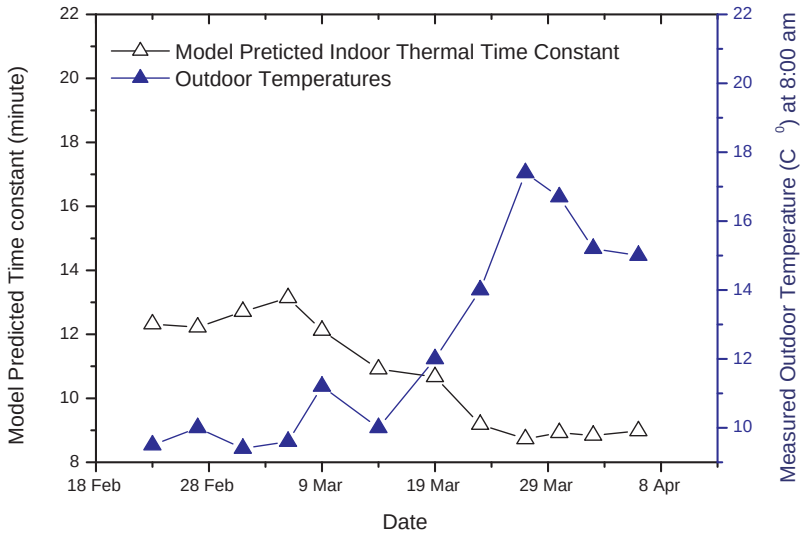


a. Model response the $Feb.23, 2012$

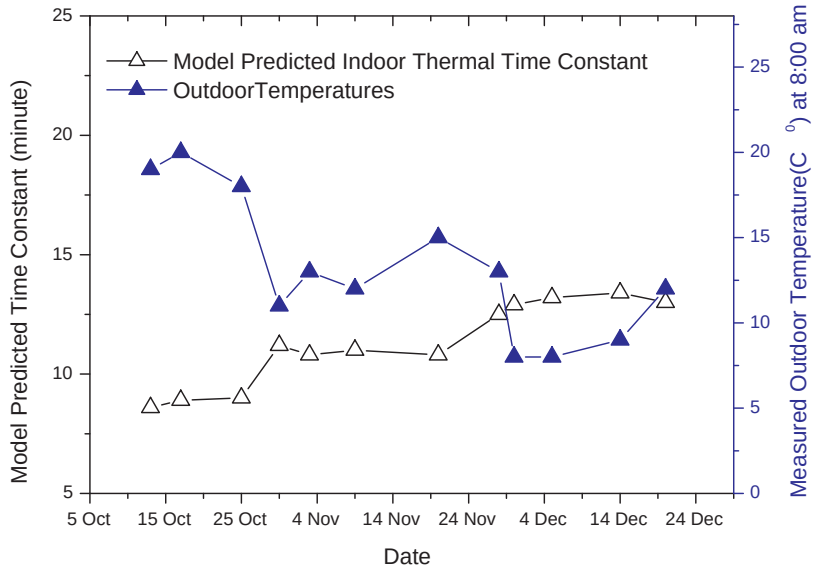


Model response the $April.06, 2012$

FIGURE 14.3 – The predicted thermal time constant on different date of the year 2012



a. Spring 2012



b. Winter 2012

FIGURE 14.4 – Compare the predicted indoor thermal time constant and the outdoor temperatures

TABLE 14.3 – Model predicted room thermal time constants and measured room characteristic time (minute)

Date	Predicted TTC	Measured CT	Date	Predicted TTC	Measured CT
23/2/2012	12.3	12.9	13/10/2012	8.6	9
21/2/2012	12.2	12.7	17/10/2012	8.9	9.5
2/3/2012	12.7	11.5	25/10/2012	9	10
6/3/2012	13.1	14	28/10/2012	12.5	13.4
9/3/2012	12.1	14	30/10/2012	11.2	12
14/3/2012	10.9	11.2	3/11/2012	10.8	11.5
19/3/2012	10.6	11.4	9/11/2012	11	11.8
23/3/2012	9.1	10.9	12/11/2012	12.5	13.4
27/3/2012	8.7	9.6	16/11/2012	12.8	13.6
30/3/2012	8.9	9.8	20/11/2012	10.8	11.8
2/4/2012	8.8	9.5	26/11/2012	8.9	10
6/4/2012	8.9	9.9	28/11/2012	11.2	12.2
9/4/2012	8.5	9.7	30/11/2012	12.9	13.8
10/4/2012	9	10.4	5/12/2012	13.2	14.1
11/4/2012	8.9	10.5	14/12/2012	13.4	14.5
12/4/2012	9	10.7	20/12/2012	13	14

Firstly, we noticed that time constant changes in response of different environmental factors, for example, the outdoor temperature. This fact highlights the well-known existence of nonlinearity in the heat transfer phenomena in buildings. Again, we take previous work [16] as a reference. In their mathematical model of a building room, J. Florez and G.C. Barney made a main simplification that all heat transport phenomena in building are linear. Thus, they calculated the room thermal time constant in Eq. (2.2) According to Florez's mathematical model, the room thermal time constant does not change. On the contrary, our sensor data based model predictions have shown that room time constant slightly varies under different environmental conditions. This difference in return points out the fact that the nonlinearity in heat transport phenomena in buildings are non-negligible and it can be one main cause of the variable indoor thermal time constant. This fact can be also stated in a more detailed mathematical thermal model (see Eq. (2.6)).

Except for the existing non-linearity, other possible causes of variable EITTC is related to the indoor air-conditioning system's output. The heating/cooling system's efficiency can be the second reason for the variation of indoor thermal time constant. In this work, the indoor heating/cooling source, as mentioned in the section of experiment, is Toshiba RAS-167SKV-E3/Toshiba RAS-167SAV-E3. It is a inverter air-conditioning system. As a result, its output heated air contains nonli-

nearities, this can be considered as the second cause of the variable EITTC.
For building habitants, it is difficult to obtain the exact inner control mechanism of their indoor air-conditioning system to make a quantitative analysis. This points out again the advantage of our WSN and ANN combined solution : benefitting from the universal approximating ability of ANN models, habitants can characterize their indoor thermal effects without further computation or parameter estimation on their existing heating/cooling system.
Based on the discussions above, the ANN thermal model is able to describe building room's thermal response adaptively. Further more, it could characterize the existing control source's heating effects in the building room under different outdoor conditions.

14.2 Energy efficient approaches

Energy consumption of residential and commercial facilities takes about 40% in Europe and USA. In China, residential urban energy consumption tripled between 1996 and 2008 [12]. As a matter of fact, the energy consumption varies between different types of buildings. For example, low energy efficient buildings (especially old buildings more than 30 years) consume from 300 to $400kWh/m^2/year$, while modern buildings consume approximately from 150 to $200kWh/m^2/year$ [13]. The improvement of energy efficiency in low energy efficient buildings can bring considerable environmental and financial benefits.

The traditional way to optimize the building's energy efficiency is to renovate the building with new construction material, which is often expensive and time-consuming. One important purpose of this thesis is to demonstrate that by establishing un accurate adaptive indoor thermal model, the indoor thermal response can be characterized. Thus, the existing heating/cooling system's efficiency can be optimized.

As the most energy consuming periods for building room E106 located at this latitude are the early spring and winter, in order to keep the room warm with stable temperature, the air-conditioning system is activated all day long, sometimes even during the night which generate great energy wastes.

To ameliorate this situation, we designed in this work a new indoor heating control strategy called "Adaptive Start/Shut Control" which is based on the ANN thermal model's prediction. The necessity of adaptive Start/Shut control can be illustrated in Fig. 14.5. The accurate Start/Shut control of indoor heating equipment can lead to both indoor comfort and a minimum energy consumptions. However, the control performance depends on the accuracy of the the preheat time estimations. As the WSN based ANN thermal model can provide precise predictions on indoor temperature variations under different environmental conditions, we then

designed an adaptive Start/Shut heating control strategy : firstly, the computer records the actual initial outdoor/indoor conditions from real measured WSN sensor acquisitions. Secondly, the software uses the correspondent trained ANN thermal model to make the calculation of the preheat time (t_p) needed. Thirdly, calculation will be made to find the optimum start time of indoor heating equipment based on the occupancy start time and the model prediction ($t_{in} - t_p$). Finally, the software generates control command and sends it to activate the heating system though the controller. The same processus can be applied for an earlier shut-off of the heating system(see Fig. 14.5) which brings direct energy savings.
This control method has been proved effective by experiments carried out during

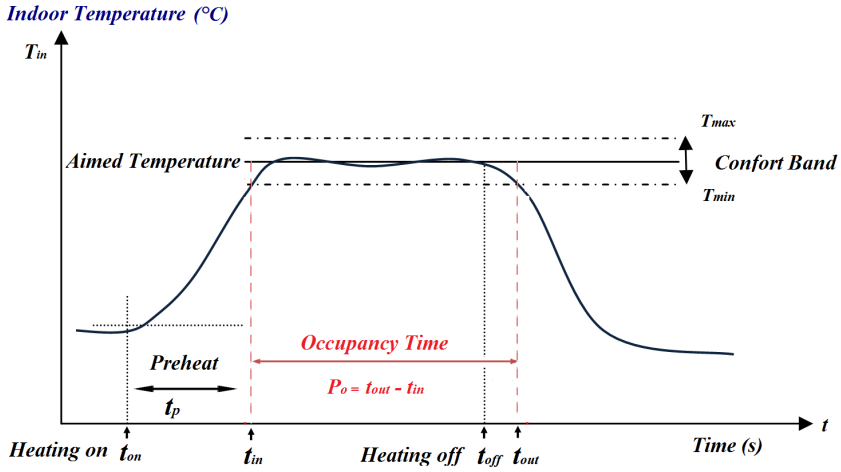
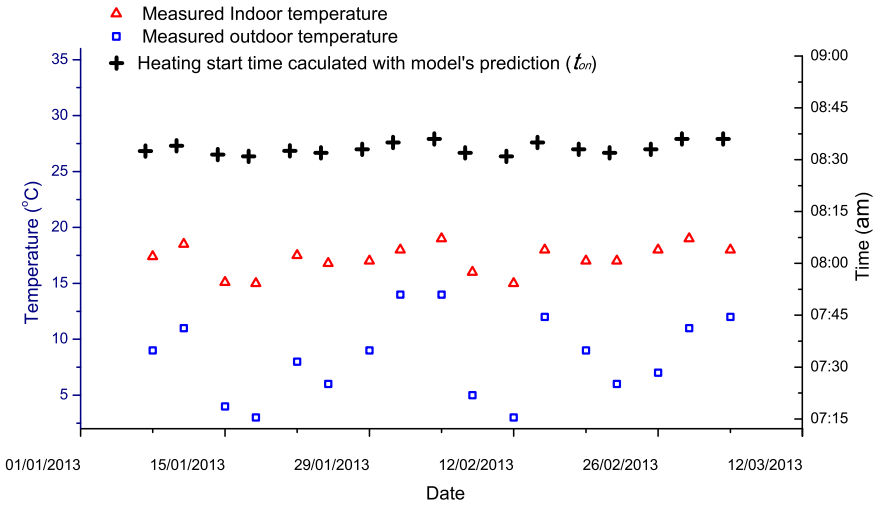


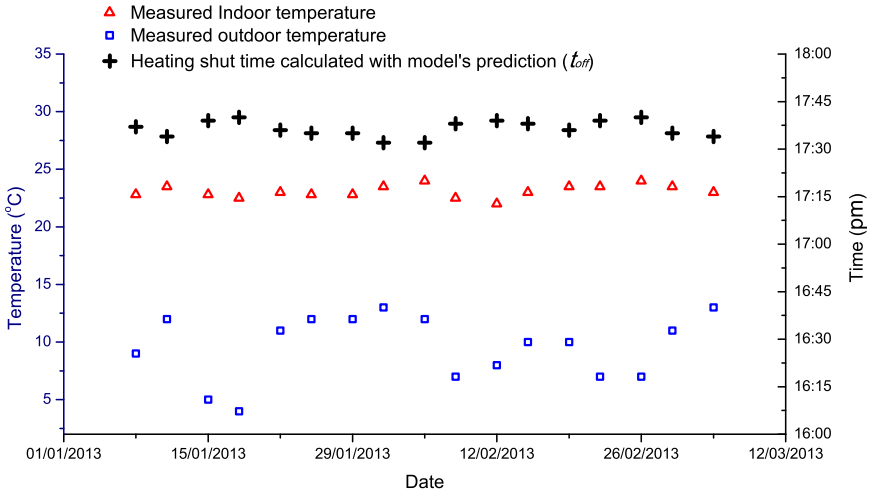
FIGURE 14.5 – Heating Start/Shut operation in buildings

the spring 2013 in the room E1106. Normally, the heating system in this room is activated from 8 :00 am to 6 :00 pm while the occupancy period is from 8 :45 am (t_{in}) to 5 :45 pm (t_{out}). By deploying the proposed control method, the adaptive start and shut off of heating system in controlled precisely based on model's predictions under different indoor/outdoor conditions. Thus, the operating period of the heating system is shortened. The control results are presented in Fig. 14.6.

Comparing to the previous Start/Shut operation of heating system (08 :00 am to 6 :00 pm), the new control method has shortened the whole period up to an average of 56 minutes (33 minutes saved for start control and 23 minutes saved for shut-off control). If we consider the power of the heating system is constant, this control method leads to an average energy saving of 9.3%.



a. Adaptive heating start control result



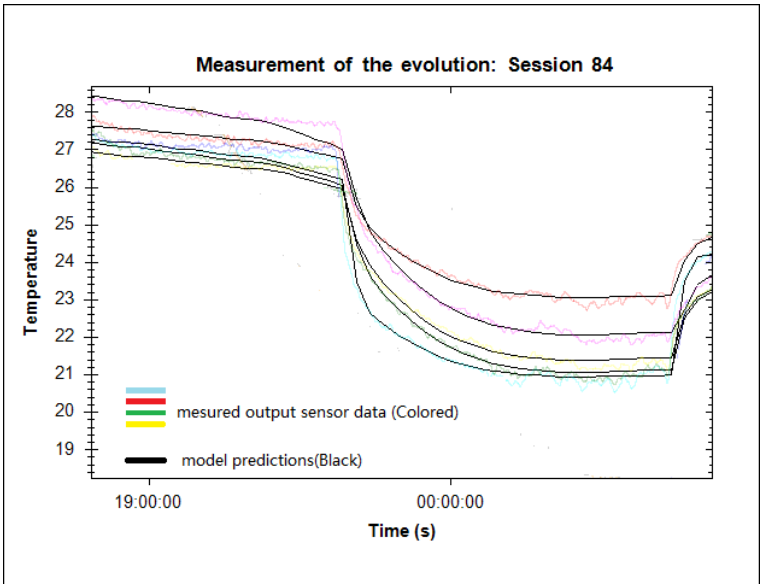
b. Adaptive heating shut control result

FIGURE 14.6 – Adaptive Start/Shut heating control results

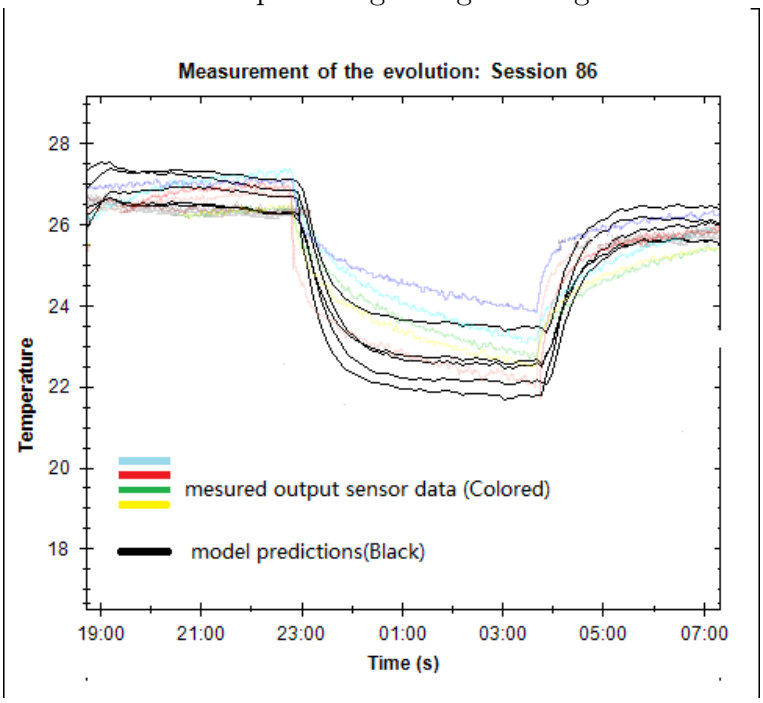
14.3 Modeling results by applying MPCT method

As presented in the previous section, the ANN model characterized the thermal response on different points of a building room ; predicted temperature drops is consistent with the real measurement. The model's prediction error was very small compared to the total indoor temperature variation. The results also suggest that the ASME of the single pattern trained model is acceptable when it is applied to similar test data. However, following experiments found that its performance is not so convincing when facing test data which is measured under different indoor and outdoor conditions.
We compared the model response regarding its training data session 84 in Fig. 14.7a, with that regarding test data session 86 in Fig. 14.7b. The originally measured outputs (multiple sensors) are colored, the model responses are in black.
It is apparent in Fig. 14.7 that a higher model prediction error is presented when applying the model to testing data measured under different conditions. The main reason, as explained in introduction, is that the single session of WSN acquisition as training data contains limited information. So the trained network model cannot cover the whole phenomena.
The modeling results using MPCT method are presented below in Fig. 14.8 Session 84 is one of the three training sessions used for this MPCT model while session 86 is a test data session which is not used in training the model. The originally measured outputs (multiple sensors) are colored, the model responses are in black.
We found that models using Pattern Cross training Method have much lower prediction errors regarding test data. The reason is that with MPCT method, the trained model can merge knowledge from different training sources, understand and cover a wider system behaviors. Thus, its generalization performance outperformed the model trained with single session.
Another particular argument for using the MPCT method should also be discussed here. Based on the back-propagation algorithm, the MPCT method was used in successive training epochs while alternating between the separated data sessions. For each epoch, the neural network tends to adapt the model for the present data session. Thus, by changing the input data session, the convergence direction changes too. In this way, the presence of local minima can probably be avoided.
More experiments allowed us to further compare their different modeling results in Tab.14.4. Trn pattern refers to the training data pattern while Tst Pattern refers to Test data pattern. These experiments confirm again that models trained with the MPCT method have lower prediction error on different test data. However, it indicates that the increase of data sessions used in MPCT training does not guarantee lower prediction errors. As we have mentioned in the introduction of this thesis, the MPCT modeling quality is directly related to the capacity of the network and the structuring of training data sessions. Limited differences or data redundancy in the selected data sessions may cause low efficiency of MPCT training.

More experiments allowed us to further compare their different modeling results in Tab. 14.4. Trn pattern refers to the training data pattern while Tst Pattern

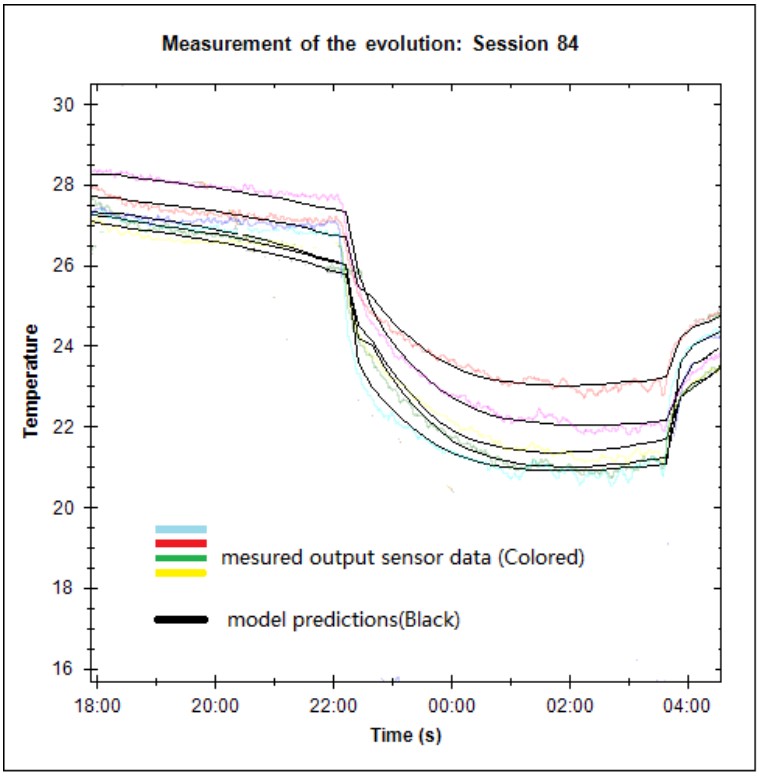


a. model response regarding training data

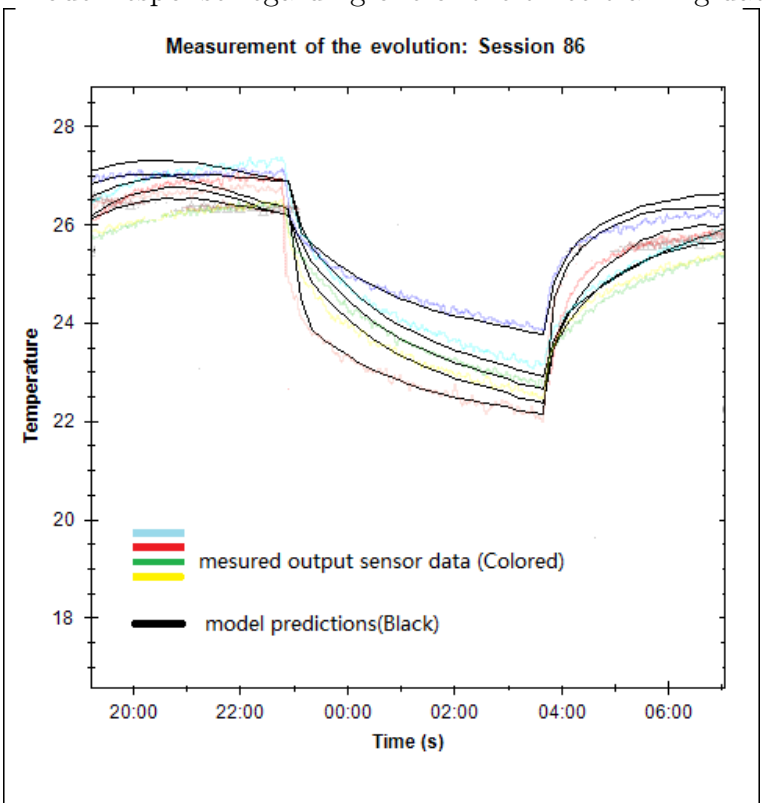


b. model response regarding the test data which measured under different conditions

FIGURE 14.7 – Modeling performance of single pattern trained ANN model



a. Model response regarding one of the three training data



b. model response regarding the test data which measured under different conditions.

FIGURE 14.8 – Modeling performance of ANN model trained with MPCT training method

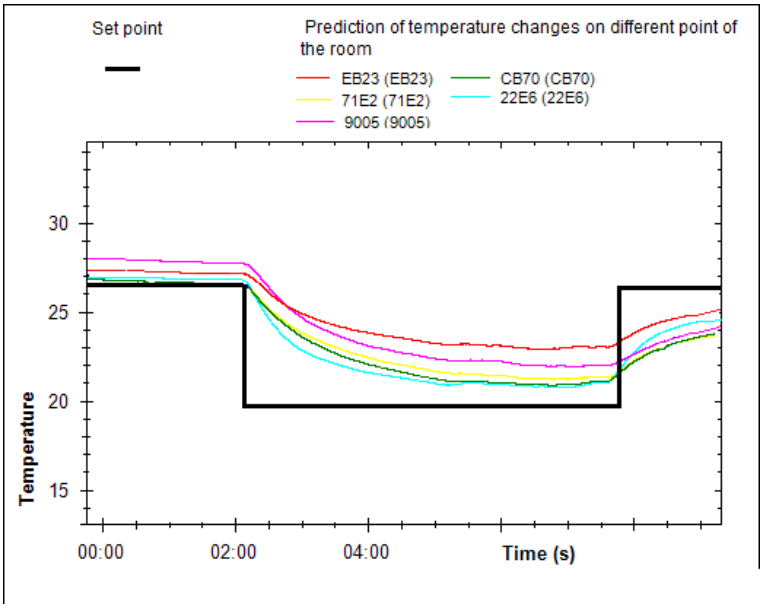
refers to Test data pattern. These experiments confirm again that models trained with the MPCT method have lower prediction error on different test data. However, it indicates that the increase of data sessions used in MPCT training does not guarantee lower prediction errors. As we have mentioned in the introduction of this paper, the MPCT modeling quality is directly related to the capacity of the network and the structuring of training data sessions. Limited differences or data redundancy in the selected data sessions may cause low efficiency of MPCT training.

TABLE 14.4 – Comparing the performance between single pattern trained model and MPCT trained model

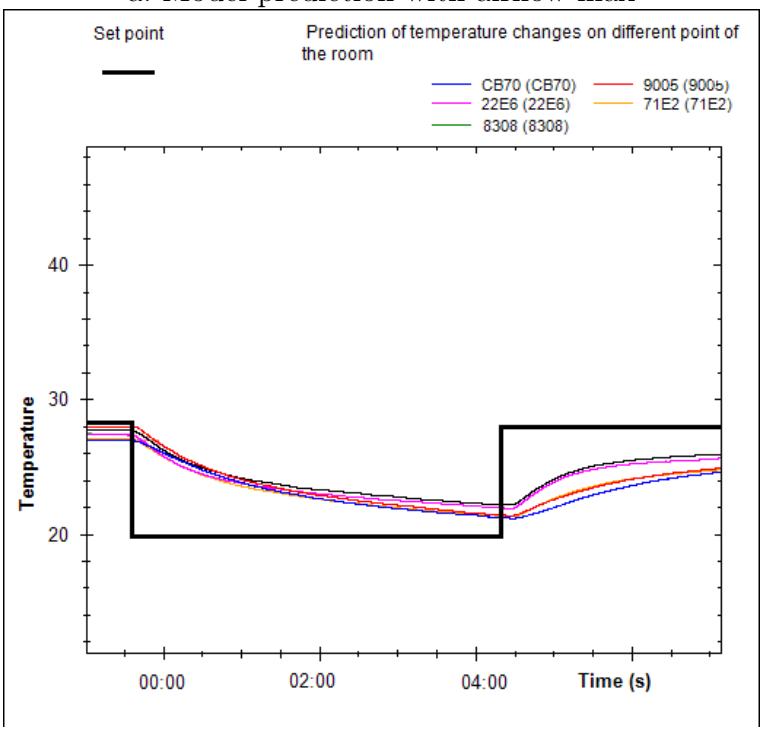
Data	AMSE (Seule session)	AMSE (Multi sessions(MPCT))		
Data Session	Single session	2 sessions	3 sessions	4 sessions
Trn sessions 1	0.1179	0.1292	0.131	0.1194
Trn sessions 2	-	0.1479	0.131	0.1394
Trn sessions 3	-	0.1214	0.1105	0.1240
Trn sessions 4	-	0.1501	0.1571	0.1724
Tst sessions 1	0.2142	0.212	0.1982	0.2038
Tst sessions 2	0.9571	0.2720	0.2589	0.2990
Tst sessions 3	1.4172	0.2435	0.2724	0.2865
Tst sessions 4	2.2414	0.3675	0.2959	0.3745
Tst sessions 5	2.5720	0.2514	0.3720	0.2984

14.4 Model prediction control based on MPCT method

The shortcoming of single pattern trained neural model is that the information it gets from the training data is limited which is the main reason that multi-models are used in MPC. However, the MPCT trained model shows the ability to absorb and distinguish different behavior from multiple training patterns. We present here in Fig. 14.9 the model's first step prediction on the prediction horizon with different emission airflow angle. We can find that the MPCT trained single network model characterized the different system response with different input combinations.
The optimizer will repeat this operation until all the feasible control parameter combinations have been tried. By comparing their performances, the optimizer give the plant the best parameters for the second step control. Fig. 14.10 presents the different model prediction errors. When the air-conditioner's compressor works at its maximum level, we can find that the best parameter combination is emission power max with the airflow angle equals to 45°.



a. Model prediction with airflow max



b. Model prediction with airflow min

FIGURE 14.9 – Different step predictions made from the MPCT trained model on the control horizon

Another advantage of using MPCT trained model in MPC control is its efficiency.

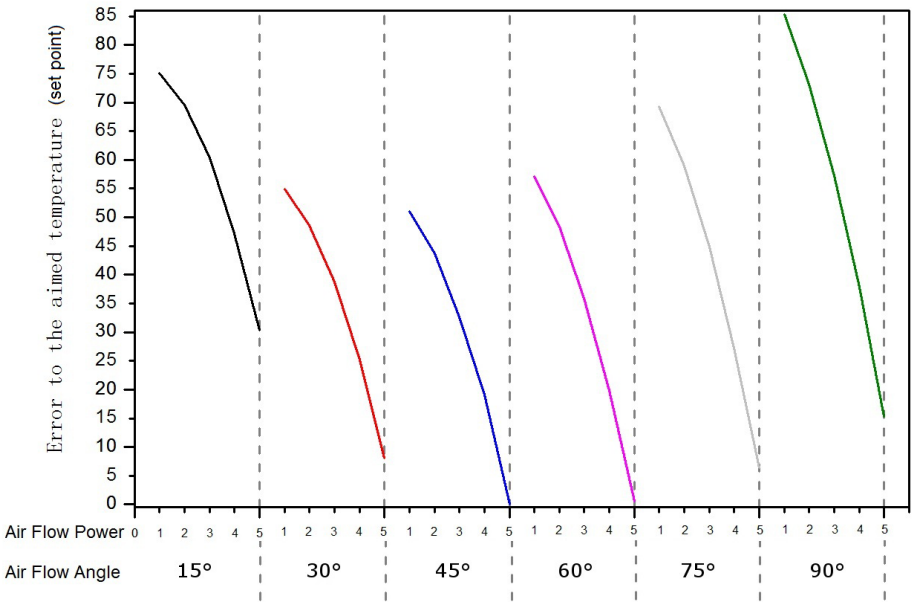


FIGURE 14.10 – The first step prediction comparison.

The online calculation speed is much faster than Multiple model based MPC control : we tested the same control process (The indoor thermal MPC control action composite of 420 controls combinations and a horizon of 48 time steps) with both different control methods on two different computers, With the multiple model structure, the time consumed for a complete iteration is relatively long : On a computer with Duo core 2.1 GHz and a OS of 32bits, it costs a average of 860 millisecond while on a computer with Duo core 2.8 GHz, it costs 750 milliseconds, while the results for a MPCT trained single model on the same control process cost 30 milliseconds and 21 milliseconds separately on these two computers. This means our MPCT model based MPC can be operated at 50Hz for such a complicated control process, about 35 times faster than multiple model based MPC.

15

Conclusion

15.1 Conclusion

This thesis established that the combination of WSN (real-time acquisition) and ANN (system Identification tool) leads to accurate modeling results. The fine grained thermal models of both prototype and faculty building room positively highlighted consistent results. By tracing the characteristics time constant of the linear approximation of the ANN thermal model, we can characterize the existing control source's heating/cooling effects under different outdoor conditions. These results have been confirmed by statistical computations since a strong correlation have been found between the model predicted room time constant and real measured characteristic time of the building room. Optimal control methods have been tested.

We also presented a training method "Multi-Pattern Cross Training"(MPCT). This training method makes it possible to train a single ANN model with multiple training data sessions. Experiments of WSN sensor data based building ANN thermal modeling show that the Cross Training ANN model outperformed model trained with single training data pattern on the model's generalization performance. Model trained with MPCT method shows lower prediction errors. We also demonstrated in this work that MPCT method can be used to optimize the control efficiency of multivariable-MPC by 20 times

Further research and explorations will be made mainly in two aspects : firstly, taking advantage of the high energy efficiency of our WSN system (battery life over 30 days), long term measurement on the purpose of enriching the thermal Model will be necessary. Secondly, based on the indoor ANN thermal models, real-time control with optimize energy consumption could be realized. Yet, the ultimate goal of MPCT is to use minimum training data sessions to obtain a model with the best generalization performance. To realize this, further exploration can be done in the following aspect : to find the optimum combination of neural network capacity and the structuring of training data sessions.

It seems that not only WSN and ANN have a great potential in modeling applications on a variety of domains, but also it appears that WSN and ANN together

can bring huge changement : after 30 years of the ice age in the classical neural network and modern artificial intelligence communities, they have both glowed tremendously in recent years. The research results of machine learning, deep learning, big data have been successfully employed in voice and image processing (Iphone serri) and many internet services. For the first time, we planted intelligence in "machines", Imaging the picture that intelligent machines connected each other with WSN. That is the inevitable future.

Conclusion in French

Cette thèse a établi que la combinaison de WSN (acquisition en temps réel) et ANN (outil d'identification du système) conduit à des résultats de modélisation précis. Les modèles thermiques à grains fins établies de prototypes et une salle du bâtiment de la faculté exposées positive des résultats cohérents. En traçant des caractéristiques constantes de l'approximation linéaire du modèle thermique ANN temps, nous pouvons caractériser chauffage/refroidissement des effets de la source de contrôle existant dans différentes conditions extérieures. Ces résultats ont été confirmés par des calculs statistiques depuis une forte corrélation a été trouvée entre le modèle a prédit constante et en temps réel chambre temps caractéristique mesurée de la salle de musculation . Méthodes de contrôle optimal ont été proposées.
Nous avons également proposé une méthode de formation "Formation de la Croix-Multi- Pattern" (MPCT) . Cette méthode de formation permet de former un seul modèle ANN avec plusieurs sessions de données d'entraînement . Expériences de données de capteurs WSN basées bâtiment ANN spectacle de modélisation thermique que le modèle ANN de formation Croix surperformé modèle formé avec le modèle de données unique de formation sur la performance de généralisation du modèle. Formé avec la méthode MPCT modèle montre les erreurs de prévision inférieurs. Nous avons également démontré dans ce travail que la méthode MPCT peut être utilisé pour optimiser l'efficacité du contrôle des variables multiples MPC de 20 fois
Des recherches et des explorations seront effectués principalement dans deux aspects : d'une part , profitant de la grande efficacité énergétique de notre système de WSN (autonomie de plus de 30 jours), la mesure à long terme sur le but d'enrichir le modèle thermique sera nécessaire. Deuxièmement, sur la base des modèles thermiques ANN intérieures , le contrôle en temps réel la consommation d'énergie d'optimiser pourrait être réalisé. Pourtant, le but ultime de MPCT est d'utiliser des sessions de données minimales de formation pour obtenir un modèle avec la meilleure performance de généralisation. Pour réaliser cela, une exploration plus poussée peut être fait dans l'aspect suivant : Pour trouver la combinaison optimale de la capacité de réseau neuronal et la structuration des sessions de données d'entraînement.

More detailed code is available by requesting to yzhao@univ-tln.fr

1cm

.1 Embedded code for MRF24J40MA and PIC18LF4520

Voici la fonction utilisée générale pour envoyer les messages :

```
void Send_Message_by_long_address
(BYTE report_type, BYTE commande,
 BYTE complement, char* data, char Address[8])
{
char* ptr =data;
// vide le buffer
TxPayLoad();
WriteData(report_type);

WriteData(commande);
WriteData(complement);
while (*ptr != NULL)
{
WriteData(*ptr++);
}
SendReportByLongAddress(Address);
}
```

Message reception

```

if(RxPacket())
{
switch(*pRxData++)
{
case MIWI_USER_REPORT:
switch(*pRxData++)
{
case SET_DIGITAL_OUTPUT:
switch(*pRxData++)
{
case 0x01: // LED n1
switch(*pRxData++)
{
case 0x01:
LED_1=0;
ConsolePutROMString((ROM char*)"LED1 : OFF \r\n");
break;

case 0x02:
LED_1=1;
ConsolePutROMString((ROM char*)"LED1 : ON \r\n");
break;
}
```

```
break;

case 0x02:
switch(*pRxData++) // la donnée : on allume ou on éteint ?
{
case 0x01:
LED_2=0;
ConsolePutROMString((ROM char*)"LED2 : OFF \r\n");
break;
case 0x02:
ConsolePutROMString((ROM char*)"LED2 : ON \r\n");
break;
}
break;
}
break;
}
```

.2 Main functions of Embedded code on MC13224

BeeApp.c :

```
 // Private type definitions

#define   RFDSensorReportTime_c 5000//
#define   RouterSensorReportTime_c 2000//
#define   RunReportTime_c 300//
#define   mAppRxFromUart_c (1 <<  2) /* 0x0004 */



void BeeAppTask(event_t events);
void BeeAppDataIndication(void);
void BeeAppDataConfirm(void);
uint8 RfSendData(uint16 addr, uint8 *buf, uint8 Leng);
uint8 RfSendData(uint16 addr, uint8 *buf, uint8 Leng);
uint8 CheckUartData(uint8 *arr, uint8 n);
static void UartRxCallBack(void);
void UartRxComCallBack(void);
static void UartTxCallBack(unsigned char const *pBuf);
void SensorTimerCallBack  (tmrTimerID_t timerId );
void NetworStartSucc(void);
void RunTimerCallBack  ( tmrTimerID_t timerId );

* Public memory declarations
 /
```

```
tmrTimerID_t  appTimerId; /* for use in reporting */
tmrTimerID_t  RunTimerId; /* for use in reporting */
zbClusterId_t appDataCluster;
zbEndPoint_t appEndPoint;
uint8 Runflag = 0;
uint8 NewNodeDispTime = 0;
uint8_t  NetState = mStateIdle_c;//
uint8 HaveFlag;//


NvDataItemDescription_t const gaNvAppDataSet[] = {
  gAPS_DATA_SET_FOR_NVM,
  {&gZclCommonAttr,
   sizeof(zclCommonAttr_t)},
  {&gAslData, sizeof(ASL_Data_t)},
  {NULL, 0}
};

union f1{
  uint8 RxBuf[32];
  struct UARTCOMBUF
  {
 uint8 Head;
 uint8 HeadCom[3];
 uint8 Laddr[8];
 uint8 Saddr[2];
 uint8 DataBuf[16];
 uint8 CRC;
   uint8 LastByte;
  }RXDATA;
}UartRxBuf;

union e{
  uint8 TxBuf[32];
  struct UARTBUF
  {
 uint8 Head;
 uint8 HeadCom[3];
 uint8 Laddr[8];
 uint8 Saddr[2];
 uint8 DataBuf[16];
 uint8 CRC;
   uint8 LastByte;
  }TXDATA;
}UartTxBuf;
```

```
struct join
{
 uint8 RfdCount;
 uint8 RouterCount;
 uint8 RfdAddr[20][10];
 uint8 RouterAddr[20][10];
}JoinNode;



union h{
  uint8 RxBuf[29];
  struct RFRXBUF
  {
 uint8 HeadCom[3];
 uint8 Laddr[8];
 uint8 Saddr[2];
 uint8 DataBuf[16];
  }RXDATA;
}RfRx;


union j{
  uint8 TxBuf[29];
  struct RFTXBUF
  {
 uint8 HeadCom[3];
 uint8 Laddr[8];
 uint8 Saddr[2];
 uint8 DataBuf[16];
  }TXDATA;
}RfTx;//
//Initialize the application.

void BeeAppInit
  (
  void
  )
{
 uint8_t i;

 InitLcd();

 TurnOnDisp();//


for(i=0; i<gNum_EndPoints_c; ++i)
```

```
{
 (void)AF_RegisterEndPoint(endPointList[i].pEndpointDesc);
}
 appEndPoint = endPointList[0].pEndpointDesc
  ->pSimpleDesc->endPoint;


 Copy2Bytes(appDataCluster, endPointList[0].pEndpointDesc
 ->pSimpleDesc->pAppInClusterList);

 UartX_SetRxCallBack(UartRxCallBack);
 appTimerId = TMR_AllocateTimer();
 RunTimerId = TMR_AllocateTimer();
 ASL_InitUserInterface("HaGenericApp");
}

uint8 RfSendData(uint16 addr, uint8 *buf, uint8 Leng)
{
 afAddrInfo_t  addrInfo;
 zbStatus_t state;

 addrInfo.dstAddrMode = gZbAddrMode16Bit_c;
 addrInfo.dstAddr.aNwkAddr[0] = (uint8)addr;
   addrInfo.dstAddr.aNwkAddr[1] = addr>>8;
   addrInfo.dstEndPoint = 8;
 addrInfo.srcEndPoint = appEndPoint;
 addrInfo.txOptions = gApsTxOptionNone_c;
 addrInfo.radiusCounter = afDefaultRadius_c;

 /* set up cluster */
 Copy2Bytes(addrInfo.aClusterId, appDataCluster);

 /* send the data request */
 state = AF_DataRequest(&addrInfo, Leng, buf, NULL);
 if(state == gZbSuccess_c)
 {
  return 1;
 }
 else
 {
  return 0;
 }
}
  uint8 CheckUartData(uint8 *arr, uint8 n)
uint8 CheckUartData(uint8 *arr, uint8 n)
{
 uint8 sum=0;
```

```
 uint8 i;
 for(i=0; i<n; i++)
 {
  sum += *arr;
  arr++;
 }
 return sum;
}
static void UartRxCallBack(void)
{

 unsigned char byte;
 static char count = 35;

 if(UartX_GetByteFromRxBuffer(&byte))
 {
  if((byte == '&') && (count > 31))
  {
   UartRxBuf.RxBuf[0] = '&';
   count = 1;
  }
  else if((byte == '*') && (count == 31))
  {
  UartRxBuf.RxBuf[count] = byte;
  count++;
  TS_SendEvent(gAppTaskID, mAppRxFromUart_c);
  }
  else if(count < 31)
  {
   UartRxBuf.RxBuf[count] = byte;
   count++;
   }
  else
 {
 count++;
 }
 }
}

void SensorTimerCallBack  (
  tmrTimerID_t timerId
  )
{
 (void)timerId;

 TMR_StopTimer(appTimerId);
```

```
 memset(UartTxBuf.TxBuf,'x',32);

UartTxBuf.TXDATA.Head = '&';
memcpy(UartTxBuf.TXDATA.HeadCom,UartRxBuf.RXDATA.HeadCom,3);//
memcpy(UartTxBuf.TXDATA.Laddr,UartRxBuf.RXDATA.Laddr,8);//
UartTxBuf.TXDATA.DataBuf[0] = 'E';
UartTxBuf.TXDATA.DataBuf[1] = '2';//
UartTxBuf.TXDATA.CRC = CheckUartData(&UartTxBuf.TxBuf[1],29);
UartTxBuf.TXDATA.LastByte = '*';
Uart1_Transmit(UartTxBuf.TxBuf, 32, UartTxCallBack);//
}



//The application task.
void BeeAppTask
  (
  event_t events  /*IN: events for the application task */
  )
{
 /* received one or more data confirms */
 if(events & gAppEvtDataConfirm_c)
 BeeAppDataConfirm();

 /* received one or more data indications */
 if(events & gAppEvtDataIndication_c)
 BeeAppDataIndication();

 /* ZCL specific */
 if(events & gAppEvtAddGroup_c)
 ASL_ZclAddGroupHandler();

 if(events & gAppEvtStoreScene_c)
 ASL_ZclStoreSceneHandler();

 if(events & gAppEvtSyncReq_c)
 ASL_Nlme_Sync_req(FALSE);

 if(events & mAppRxFromUart_c)//
 UartRxComCallBack();

 if(events & gNetworkOver)//
 NetworStartSucc();

}

  //Process incoming ZigBee over-the-air messages
```

```
void BeeAppDataIndication(void)
{
uint8 i, j;
char arr[20];
static char state = 0;
apsdeToAfMessage_t *pMsg;
zbApsdeDataIndication_t *pIndication;

if(state)
{
 LED_SetLed(LED2, gLedOn_c);
 state = 0;
}
else
{
 LED_SetLed(LED2, gLedOff_c);
 state = 1;
}
while(MSG_Pending(&gAppDataIndicationQueue))
{

 pMsg =
  MSG_DeQueue( &gAppDataIndicationQueue );

 pIndication =
  &(pMsg->msgData.dataIndication);
  FLib_MemCpy(RfRx.RxBuf, pIndication->pAsdu, 29);
if((RfRx.RXDATA.HeadCom[0] == 'J')
&& (RfRx.RXDATA.HeadCom[1] == 'O')
&& (RfRx.RXDATA.HeadCom[2] == 'N'))//
{
 if((RfRx.RXDATA.DataBuf[0] == 'R')
  && (RfRx.RXDATA.DataBuf[1] == 'F')
  && (RfRx.RXDATA.DataBuf[2] == 'D'))//
 {

 NewNodeDispTime = 8;
  Print(2, 5, "New RFD Jion...", 1);
for(i=0; i<8; i++)
{
JoinNode.RfdAddr[JoinNode.RfdCount][i] =
  RfRx.RXDATA.Laddr[i];
}
 for(i=0; i<2; i++)
{
JoinNode.RfdAddr[JoinNode.RfdCount][8+i] =
  RfRx.RXDATA.Saddr[1-i];
```

```
  }

for(j=0; j<JoinNode.RfdCount; j++)//
{
HaveFlag = 1;
for(i=0; i<8; i++)
{
if(JoinNode.RfdAddr[JoinNode.RfdCount][i] !=
 JoinNode.RfdAddr[j][i])
{
HaveFlag = 0;
break;//
}
}
if(HaveFlag == 0)continue;
 JoinNode.RfdCount--;//ÊÇ
 JoinNode.RfdAddr[j][8] = RfRx.RXDATA.Saddr[1];
 JoinNode.RfdAddr[j][9] = RfRx.RXDATA.Saddr[0]; //
 break;
}
 JoinNode.RfdCount++;
}
else if((RfRx.RXDATA.DataBuf[0] == 'R')
 && (RfRx.RXDATA.DataBuf[1] == 'O')
 && (RfRx.RXDATA.DataBuf[2] == 'U'))//
{
  NewNodeDispTime = 8;
 Print(2, 5, "New ROU Jion...", 1);
 for(i=0; i<8; i++)
{
JoinNode.RouterAddr[JoinNode.RouterCount][i] =
RfRx.RXDATA.Laddr[i];
}
for(i=0; i<2; i++)
{
JoinNode.RouterAddr[JoinNode.RouterCount][8+i] =
RfRx.RXDATA.Saddr[1-i];
}

for(j=0; j<JoinNode.RouterCount; j++)//
{
HaveFlag = 1;
for(i=0; i<8; i++)
{
 if(JoinNode.RouterAddr[JoinNode.RouterCount][i] !=
  JoinNode.RouterAddr[j][i])
 {
```

```
  HaveFlag = 0;
  break;//
}
      }
      if(HaveFlag == 0)continue;
      JoinNode.RouterCount--;//ÊÇ
      JoinNode.RouterAddr[j][8] = RfRx.RXDATA.Saddr[1];
      JoinNode.RouterAddr[j][9] = RfRx.RXDATA.Saddr[0];
      break;
    }
    JoinNode.RouterCount++;
   }
  }
  else if((RfRx.RXDATA.HeadCom[0] == 'P')
&& (RfRx.RXDATA.HeadCom[1] == 'R')
&& (RfRx.RXDATA.HeadCom[2] == 'E'))//
  {
   memcpy(arr, &RfRx.RXDATA.Laddr[0], 8);
   arr[8] = ':';
   arr[9] = RfRx.RXDATA.DataBuf[0];
   arr[10] = RfRx.RXDATA.DataBuf[1];
  arr[11] = '␣';
  arr[12] = '␣';
  arr[13] = '␣';
   arr[14] = '\0';
   NewNodeDispTime = 4;
   Print(2, 5, (uint8*)arr, 1);
  }
  else
  {
   TMR_StopTimer(appTimerId);

   UartTxBuf.TXDATA.Head = '&';
   memcpy(&UartTxBuf.TxBuf[1],&RfRx.RxBuf[0],29);
   for(i=0; i<8; i++)
   {
    UartTxBuf.TXDATA.Laddr[i] =
RfRx.RXDATA.Laddr[i];
   }
   for(i=0; i<2; i++)
   {
    UartTxBuf.TXDATA.Saddr[i] =
RfRx.RXDATA.Saddr[1-i];
   }
   UartTxBuf.TXDATA.CRC =
  CheckUartData(&UartTxBuf.TxBuf[1],29);
   UartTxBuf.TXDATA.LastByte = '*';
```

```
    Uart1_Transmit(UartTxBuf.TxBuf, 32, UartTxCallBack);
  }

  AF_FreeDataIndicationMsg(pMsg);
 }
}
```

.3 Embedded code on CC2530

```
void ProcessReceivedUartMessage(uint8* rcvBuffer, size_t length)
{
  // Syntaxe des messages reçus de l'UART
  static afIncomingMSGUart_t* msg;
  static uint8 msgRxUARTEnCours = 0;
  static int indice;

  for(int i = 0; i < length; ++i)
  {
  uint8 rcvChar = rcvBuffer[i];
  if (!msgRxUARTEnCours && rcvChar == 0x02)
  {
 // On a un début de chaine
 indice=0;
 msgRxUARTEnCours=1;
 msg =
 (afIncomingMSGUart_t*)osal_msg_allocate
  (sizeof(afIncomingMSGUart_t));
 msg->hdr.event = UART_MSG_RECEIVED_EVT;
 msg->command = 0;
 msg->destShortAddress = 0;
 msg->payloadLength = 0;
  }
  else if (msgRxUARTEnCours)
  {
 switch(indice)
 {
 case 0:
 //Récupère la short address de destination
 msg->destShortAddress = (uint16)(rcvChar)<<8;
 ++indice;
 break;
 case 1:
 //On traite le bit de poids faible de la short address
 msg->destShortAddress += (uint16)(rcvChar);
 ++indice;
 break;
```

```
 case 2:
 //On traite le bit de poids fort de la commande
 msg->command = (uint16)(rcvChar)<<8;
 ++indice;
 break;
 case 3:
 //On traite le bit de poids faible de la commande
 msg->command += (uint16)(rcvChar);
 ++indice;
 break;
 case 4:
 //On traite le bit de poids fort de la longueur
 msg->payloadLength = (uint16)(rcvChar)<<8;
 ++indice;
 break;
 case 5:
 //On traite le bit de poids faible de la longueur
 msg->payloadLength += (uint16)(rcvChar);
 if (msg->payloadLength > 256)
 goto error;
 ++indice;
 break;
 default:
 msg->payload[indice-6] = rcvChar;
 ++indice;
 if (indice > msg->payloadLength+6)
 {
 msgRxUARTEnCours=0;
 if (rcvChar == 0x03)
 {
  //La taille est bonne on traite le message
  osal_msg_send(GenericApp_TaskID, (uint8*)msg);
 }
 else
 {
  // error
 error:
  msgRxUARTEnCours=0;
  osal_msg_deallocate((uint8*)msg);
 }
 }
 }
 }
}

void ProcessReceivedZigbeeMessage(afIncomingMSGPacket_t *MSGpkt)
```

```
{
  uint8* rxPacket = MSGpkt->cmd.Data;
  uint16 msgRxZigbeeTotalLength = MSGpkt->cmd.DataLength;
  uint16 msgRxZigbeeOriginAddress = MSGpkt->srcAddr.addr.
         shortAddr;

  //Analyse et check du message

  //Récupère la commande
  uint16 msgRxZigbeeCommand;
  msgRxZigbeeCommand = (uint16)(rxPacket[2]<<8);
  msgRxZigbeeCommand += (uint16)(rxPacket[3]);

  //Récupère la taille
  size_t msgRxZigbeePayloadLength;
  msgRxZigbeePayloadLength = (uint16)(rxPacket[4]<<8);
  msgRxZigbeePayloadLength += (uint16)(rxPacket[5]);

  if(msgRxZigbeePayloadLength==
   msgRxZigbeeTotalLength-6)
  {
  //Les tailles sont identiques - a priori ok

  if (msgRxZigbeeCommand==POWER_MODE_SAVE_ENERGY)
  {
 osal_pwrmgr_device(PWRMGR_BATTERY);
 osal_pwrmgr_task_state( GenericApp_TaskID, PWRMGR_CONSERVE);
  }
  else if (msgRxZigbeeCommand==POWER_MODE_ALWAYS_ON)
  {
 osal_pwrmgr_device(PWRMGR_ALWAYS_ON);
 osal_pwrmgr_task_state( GenericApp_TaskID, PWRMGR_HOLD);
  }
#ifndef RTR_NWK
  else if (msgRxZigbeeCommand == 0x30)
  {
 SendZigBeeAddress(NWK_PAN_COORD_ADDR, NLME_GetExtAddr());
  }
  else if (msgRxZigbeeCommand == 0x31)
  {
 SendPingResponse();
  }
#endif
 else if (msgRxZigbeeCommand == THERMO_REPORT)
 {
  P1_0=!P1_0;
  SendUartMsg(msgRxZigbeeOriginAddress, msgRxZigbeeCommand,
```

```
 msgRxZigbeePayloadLength,rxPacket+6 );
 }
 else
 {  // No classic envoi is needed here.
 //Sinon, envoi classique : forward du zigbee vers le UART
 //SendUartMsg(msgRxZigbeeOriginAddress, msgRxZigbeeCommand,
  msgRxZigbeePayloadLength,rxPacket+6 );
 //P1_0=!P1_0;
 }
  }
  else
  {
  // Invalid message

  }
}

@fn SendZigbeeMsg
*
* @brief Envoie un message formaté sur le Zigbee
*
* @param none
*
* @return  none
*/
void SendZigbeeMsg(uint16 address, uint16 command,
                   size_t length, uint8* payload)
{

  uint16 msgTxZigbeeOriginShortAddress = NLME_GetShortAddr();
  g_msgTxZigbee[0] = (uint8)(msgTxZigbeeOriginShortAddress >> 8);
  g_msgTxZigbee[1] = (uint8)(msgTxZigbeeOriginShortAddress) ;
  g_msgTxZigbee[2] = (uint8)(command >> 8);
  g_msgTxZigbee[3] = (uint8)(command) ;
  g_msgTxZigbee[4] = (uint8)(length >> 8);
  g_msgTxZigbee[5] = (uint8)(length) ;
  osal_memcpy(g_msgTxZigbee+6, payload, length);

  // Preparation du message
  // Envoi du message AF
  GenericApp_DstAddr.addr.shortAddr = address;
  AF_DataRequest( &GenericApp_DstAddr, &GenericApp_epDesc,
  GENERICAPP_CLUSTERID,
  length+6,
  g_msgTxZigbee,
  &GenericApp_TransID,
  AF_DISCV_ROUTE, AF_DEFAULT_RADIUS );
```

```
}
void ZigbeeTempReport()
{
  // my_short_addr= _NIB.nwkDevAddress;
 fValue=0;
 //ADC Configuration
 // for (i=0; i<10;i++)
 ADCIF = 0;
 //Clears the ADC interrupt flag
 ADCCON3 = 0xB5;
 //0xB5 uses VDD;
 ADCCON1 &= 0x70;
 //Waits for converstion to finish
 while (!(ADCCON1 & 0x80));

 Send_T_MSG[0]= ADCH;
 Send_T_MSG[1]= ADCL;
  P1_0=!P1_0;
 SendZigbeeMsg(NWK_PAN_COORD_ADDR, THERMO_REPORT,2 ,Send_T_MSG);
}
```

Bibliographie

[1] John W Tukey. Exploratory data analysis. *Reading, MA*, 231, 1977.

[2] MY Rafiq, G Bugmann, and DJ Easterbrook. Neural network design for engineering applications. *Computers & Structures*, 79(17) :1541–1552, 2001.

[3] Leonid I Perlovsky. *Neural networks and intellect : using model-based concepts*. Oxford University Press New York, NY, 2001.

[4] Kevin Swingler. *Applying neural networks : a practical guide*. Morgan Kaufmann, 1996.

[5] Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. Wireless sensor network survey. *Computer networks*, 52(12) :2292–2330, 2008.

[6] Jay Warrior. Smart sensor networks of the future. *Sensors-the Journal of Applied Sensing Technology*, 14(3) :40–45, 1997.

[7] Ian Flood and Nabil Kartam. Neural networks in civil engineering. ii : Systems and application. *Journal of Computing in Civil Engineering*, 8(2) :149–162, 1994.

[8] Kartik B. Ariyur Qi Luo. Building thermal network model and application to temperature regulation. *2010 IEEE International Conference on Control Aapplications*, pages 2190 – 2195, 2010.

[9] Cyrille Bertelle, Gâerard HE Duchamp, and Hakima Kadri-Dahmani. *Complex systems and Self-organization modelling*, volume 51. Springer, 2009.

[10] Moulay A Aziz-Alaoui and Cyrille Bertelle. *Emergent properties in natural and artificial dynamical systems*. Springer, 2006.

[11] Michel Cotsaftis. A passage to complex systems. In *Complex Systems and Self-organization Modelling*, pages 3–19. Springer, 2009.

[12] Climate Policy Initiative, Hermann Amecke, Jeff Deason, Andrew Hobbs, Aleksandra Novikova, Yang Xiu, and Zhang Shengyuan. Buildings energy efficiency in china, germany, and the united states. Technical report, Climate Policy Initiative, 2013. CPI Report.

[13] P. Szuppinger and Eva Csobod. Changing energy use in old buildings. Technical report, Regional Environmental Center, Hungary, 2011. Baltic Environmental Forum Latvia.

[14] Milo E Hoffman and Moshe Feldman. Calculation of the thermal response of buildings by the total thermal time constant method. *Building and Environment*, 16(2) :71–85, 1981.

[15] Robert Jennings Heinsohn and John M Cimbala. *Indoor air quality engineering : environmental health and control of indoor pollutants*. CRC Press, 2003.

[16] G.C. Barney J.Florez. Adaptive control of central heating systems : part 1 : optimum start time control. *Applied Mathematical Modelling*, 11 :89–95, 1987.

[17] Jaisinh K Nimbalkar. Use of neural network in wsns : A survey. *Advancement in electronics and computer engineering*, 2012.

[18] A. Mechaqrane and M. Zouak. A comparison of linear and neural network arx models applied to a prediction of the indoor temperature of a building. *Neural Comput and Applic*, 33 :32–37, 2004.

[19] B.Thomas and M.S.Mohseni. Artificial neural network models for indoor temperature prediction : investigations in two buildings. *Neural Comput and Applic*, 16 :81–89, 2007.

[20] Jin Woo Moon, Sung Kwon Jung, Youngchul Kim, and Seung-Hoon Han. Comparative study of artificial intelligence-based building thermal control methods–application of fuzzy, adaptive neuro-fuzzy inference system, and artificial neural network. *Applied Thermal Engineering*, 31(14) :2422–2429, 2011.

[21] Jay L McClelland, David E Rumelhart, and Geoffrey E Hinton. *The appeal of parallel distributed processing*. MIT Press, 1986.

[22] George F Luger. *Artificial intelligence : Structures and strategies for complex problem solving*. Addison-Wesley Longman, 2005.

[23] David E Rumelhart, Geoffrey E Hintont, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088) :533–536, 1986.

[24] James L McClelland and David E Rumelhart. *Explorations in Parallel Distributed Processing-Macintosh Version : A Handbook of Models, Programs, and Exercises*. MIT press, 1989.

[25] Elaine Rich and Kevin Knight. Artificial intelligence. *Computer Science Series. McGraw-Hill*, 8, 1991.

[26] Patrick Henry Winston. Artificial intelligence. *Reading : Addison-Wesley*, 1984.

[27] Stephen I Gallant. *Neural network learning and expert systems*. MIT press, 1993.

[28] John A Richards. *Remote sensing digital image analysis : an introduction*. Springer, 2012.

[29] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of mathematical biology*, 5(4) :115–133, 1943.

[30] Marvin Minsky and Papert Seymour. *Perceptrons*. MIT press, 1969.

[31] Bernard Widrow, Marcian E Hoff, et al. *Adaptive switching circuits*. Defense Technical Information Center, 1960.

[32] Sang-Hoon Oh. Improving the error backpropagation algorithm with a modified error function. *Neural Networks, IEEE Transactions on*, 8(3) :799–803, 1997.

[33] Y Le Cun. Une procedure d'apprentissage pour reseau a seuil assymetrique. *Proceedings of Cognitiva*, 85 :599–604, 1985.

[34] Russell D Reed and Robert J Marks. *Neural smithing : supervised learning in feedforward artificial neural networks*. Mit Press, 1998.

[35] Christopher M Bishop et al. *Neural networks for pattern recognition*. Clarendon press Oxford, 1995.

[36] Paul Werbos. *Beyond regression : New tools for prediction and analysis in the behavioral sciences*. PhD thesis, Harvard University, 1974.

[37] Ramiro Jordan and Chaouki T Abdallah. Wireless communications and networking : an overview. *Antennas and Propagation Magazine, IEEE*, 44(1) :185–193, 2002.

[38] Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. Next century challenges : Scalable coordination in sensor networks. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 263–270. ACM, 1999.

[39] Jon Agre and Loren Clare. An integrated architecture for cooperative sensing networks. *Computer*, 33(5) :106–108, 2000.

[40] Manish Bhardwaj, Timothy Garnett, and Anantha P Chandrakasan. Upper bounds on the lifetime of sensor networks. In *Communications, 2001. ICC 2001. IEEE International Conference on*, volume 3, pages 785–790. IEEE, 2001.

[41] Nirupama Bulusu, Deborah Estrin, Lewis Girod, and John Heidemann. Scalable coordination for wireless sensor networks : self-configuring localization systems. In *International Symposium on Communication Theory and Applications (ISCTA 2001), Ambleside, UK*, 2001.

[42] Alberto Cerpa, Jeremy Elson, Deborah Estrin, Lewis Girod, Michael Hamilton, and Jerry Zhao. Habitat monitoring : Application driver for wireless communications technology. *ACM SIGCOMM Computer Communication Review*, 31(2 supplement) :20–41, 2001.

[43] SeongHwan Cho and Anantha P Chandrakasan. Energy efficient protocols for low duty cycle wireless microsensor networks. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 4, pages 2041–2044. IEEE, 2001.

[44] Wendi Rabiner Heinzelman, Joanna Kulik, and Hari Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 174–185. ACM, 1999.

[45] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion : a scalable and robust communication paradigm for sensor networks.

In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 56–67. ACM, 2000.

[46] Chaiporn Jaikaeo, Chavalit Srisathapornphat, and C-C Shen. Diagnosis of sensor networks. In *Communications, 2001. ICC 2001. IEEE International Conference on*, volume 5, pages 1627–1632. IEEE, 2001.

[47] Joseph M Kahn, Randy H Katz, and Kristofer SJ Pister. Next century challenges : mobile networking for smart dust. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 271–278. ACM, 1999.

[48] Sasa Slijepcevic and Miodrag Potkonjak. Power efficient organization of wireless sensor networks. In *Communications, 2001. ICC 2001. IEEE International Conference on*, volume 2, pages 472–476. IEEE, 2001.

[49] Norbert Noury, Thierry Hervé, Vicent Rialle, Gilles Virone, Eric Mercier, Gilles Morey, Aldo Moro, and Thierry Porcheron. Monitoring behavior in home using a smart fall sensor and position sensors. In *Microtechnologies in Medicine and Biology, 1st Annual International, Conference On. 2000*, pages 607–610. IEEE, 2000.

[50] Brett Warneke, Matt Last, Brian Liebowitz, and Kristofer SJ Pister. Smart dust : Communicating with a cubic-millimeter computer. *Computer*, 34(1) :44–51, 2001.

[51] P Johnson and DC Andrews. Remote continuous physiological monitoring in the home. *Journal of telemedicine and telecare*, 2(2) :107, 1996.

[52] Mitsuhiro Ogawa, Toshiyo Tamura, and Tatsuo Togawa. Fully automated biosignal acquisition in daily routine through 1 month. In *Engineering in Medicine and Biology Society, 1998. Proceedings of the 20th Annual International Conference of the IEEE*, volume 4, pages 1947–1950. IEEE, 1998.

[53] BG Celler, T Hesketh, W Earnshaw, and E Ilsar. An instrumentation system for the remote monitoring of changes in functional health status of the elderly at home. In *Engineering in Medicine and Biology Society, 1994. Engineering Advances : New Opportunities for Biomedical Engineers. Proceedings of the 16th Annual International Conference of the IEEE*, pages 908–909. IEEE, 1994.

[54] G Coyle, L Boydell, and L Brown. Home telecare for the elderly. *Journal of Telemedicine and Telecare*, 1(3) :183, 1995.

[55] Young Han Nam, Zeehun Halm, Young Joon Chee, and Kwang Suk Park. Development of remote diagnosis system integrating digital telemetry for medicine. In *Engineering in Medicine and Biology Society, 1998. Proceedings of the 20th Annual International Conference of the IEEE*, volume 3, pages 1170–1173. IEEE, 1998.

[56] Emile M Petriu, Nicolas D Georganas, Dorina C Petriu, Dimitrios Makrakis, and Voicu Z Groza. Sensor-based information appliances. *Instrumentation & Measurement Magazine, IEEE*, 3(4) :31–35, 2000.

[57] Deborah Estrin and Ramesh Govindan. Embedding. *Communications of the ACM*, 43(5) :39, 2000.

[58] Nissanka B Priyantha, Anit Chakraborty, and Hari Balakrishnan. The cricket location-support system. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 32–43. ACM, 2000.

[59] Jose A Gutierrez, Marco Naeve, Ed Callaway, Monique Bourgeois, Vinay Mitter, and Bob Heile. Ieee 802.15. 4 : a developing standard for low-power low-cost wireless personal area networks. *network, IEEE*, 15(5) :12–19, 2001.

[60] FreeScale. Freescale semiconductor product preview document number : Mc1322x rev, 11 2008.

[61] Theodore S Rappaport. Wireless communications : Principles and practice 2nd ed. prentice hall ptr, upper saddle river, 2002.

[62] Gilman Tolle and David Culler. Design of an application-cooperative management system for wireless sensor networks. In *Wireless Sensor Networks, 2005. Proceeedings of the Second European Workshop on*, pages 121–132. IEEE, 2005.

[63] Andrew Kirillov. Aforge .net framework, 2009.

[64] W.M. Healy W.S. Jang. Wireless sensor network performance metrics for building applications. *Energy and Buildings*, 42 :862–868, 2010.

[65] Malcolm J. Cook Birgit Painter, Neil Brown. Practical application of a sensor overlay system for building monitoring and commissioning. *Energy and Buildings*, 48 :29–39, 2012.

[66] Dale Tiller Robert H. Dodier, Gregor Henze. Building occupancy detection through sensor belief networks. *Energy and Buildings*, 38 :1033–1043, 2006.

[67] Yu-Fang Peng Chih-Yuan Chang, San-Shan Hung. An evaluation of the embedment of a radio frequency integrated circuit with a temperature detector in building envelopes for energy conservation. *Energy and Buildings*, 43 :2900–2907, 2011.

[68] L. Borsani A.Barbato. A wireless sensor network based system for reducing home energy consumption. *Conference on Sensor Mesh and Ad Hoc Communications and Networks (SECON)*, pages 1–3, 2010.

[69] S.A. Kalogirou. Applications of artificial neural networks in energy system a review. *Energy Conversion and Management*, 40 :1073–1087, 1999.

[70] N. Yamani and A. Al-Anbuky. Object-centric thermal mapping : A wireless sensor network perspective. *IEEE Sensors Conference, 2009, Christchurch, New Zealand*, pages 1015–1019, 2009.

[71] Maureen Caudill. The view from now. *AI expert*, pages 24–31, 1992.

[72] Robert Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural Networks, 1989. IJCNN., International Joint Conference on*, pages 593–605. IEEE, 1989.

[73] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5) :359–366, 1989.

[74] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4) :303–314, 1989.

[75] Soteris A Kalogirou and Milorad Bojic. Artificial neural networks for the prediction of the energy consumption of a passive solar building. *Energy*, 25(5) :479–491, 2000.

[76] Yi Zhao, Valentin Gies, Jean Marc Ginoux, and Ademir Felipe Teles. Multi-pattern cross training : An ann model training method using wsn sensor data. In *Neural Networks, 2013. IJCNN., International Joint Conference on*, pages 418–423, Dallas, USA, 2013. IEEE.

[77] F. Lorenz and G. Masy. Methode d'evaluation de l'economie d'energie apportee par l'intermittence de chauffage dans les batiments. Technical report, Faculte des Sciences Appliquees, University de Liege, Belgium, 1982. Report No. GM820130-01.

[78] Bormett and David W. Overall effective thermal resistance of corrugated fiberboard containers. Technical report, DTIC Document, 1981.

[79] Hung-Han Chen, Michael T Manry, and Hema Chandrasekaran. A neural network training algorithm utilizing multiple sets of linear equations. *Neurocomputing*, 25(1) :55–72, 1999.

[80] Robert S Scalero and Nazif Tepedelenlioglu. A fast new algorithm for training feedforward neural networks. *Signal Processing, IEEE Transactions on*, 40(1) :202–210, 1992.

[81] Jacek M Zurada, Aleksander Malinowski, and Shiro Usui. Perturbation method for deleting redundant inputs of perceptron networks. *Neurocomputing*, 14(2) :177–193, 1997.

[82] Hirofumi Tanaka. Effects of cross-training. *Sports Med*, 18(5) :I994, 1985.

[83] Gökhan Bakır, Léon Bottou, and Jason Weston. Breaking svm complexity with cross training. *Advances in neural information processing systems*, 17 :81–88, 2005.

[84] Bernt M Åkesson and Hannu T Toivonen. A neural network model predictive controller. *Journal of Process Control*, 16(9) :937–946, 2006.

[85] Gregory L Plett. Adaptive inverse control of linear and nonlinear systems using dynamic neural networks. *Neural Networks, IEEE Transactions on*, 14(2) :360–376, 2003.

[86] S Joe Qin and Thomas A Badgwell. A survey of industrial model predictive control technology. *Control engineering practice*, 11(7) :733–764, 2003.

[87] Graham Clifford Goodwin, Stefan F Graebe, and Mario E Salgado. *Control system design*, volume 240. Prentice Hall Upper Saddle River, 2001.

[88] Eduardo F Camacho, Carlos Bordons, Eduardo F Camacho, and Carlos Bordons. *Model predictive control*, volume 2. Springer London, 2004.

[89] Andreas Draeger, Sebastian Engell, and Horst Ranke. Model predictive control using neural networks. *Control Systems, IEEE*, 15(5) :61–66, 1995.

[90] Stephen Piche, Bijan Sayyar-Rodsari, Doug Johnson, and Mark Gerules. Nonlinear model predictive control using neural networks. *Control Systems, IEEE*, 20(3) :53–62, 2000.

[91] Chi-Huang Lu and Ching-Chih Tsai. Generalized predictive control using recurrent fuzzy neural networks for industrial processes. *Journal of Process Control*, 17(1) :83–92, 2007.

[92] Kumpati S Narendra and Jeyendran Balakrishnan. Improving transient response of adaptive control systems using multiple models and switching. *Automatic Control, IEEE Transactions on*, 39(9) :1861–1866, 1994.

[93] Lingji Chen and Kumpati S Narendra. Nonlinear adaptive control using neural networks and multiple models. *Automatica*, 37(8) :1245–1255, 2001.

[94] H Jazayeri-Rad. The nonlinear model-predictive control of a chemical plant using multiple neural networks. *Neural Computing & Applications*, 13(1) :2–15, 2004.

[95] Zainal Ahmad, Rabiatul Mat Noor, Jie Zhang, et al. Multiple neural networks modeling techniques in process control : a review. *Asia-Pacific Journal of Chemical Engineering*, 4(4) :403–419, 2009.