CONSTRUCTION OF A VARIABLE FREQUENCY AC MAGNETIC FIELD DEVICE AND MAGNETIC HYPERTHERMIA APPLICATIONS ON FERRITE NANOPARTICLES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

SEÇKİN ÖZTÜRK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
MICRO AND NANOTECHNOLOGY

July 2023

Approval of the thesis:

CONSTRUCTION OF A VARIABLE FREQUENCY AC MAGNETIC FIELD DEVICE AND MAGNETIC HYPERTHERMIA APPLICATIONS ON FERRITE NANOPARTICLES

submitted by **SEÇKİN ÖZTÜRK** in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Micro and Nanotechnology**, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Deniz Üner
Head of the Department, **Micro and Nanotechnology**

Prof. Dr. Mürvet Volkan
Supervisor, **Chemistry, METU**

Assoc. Prof. Dr. Telem Şimşek
Co-Supervisor, **Nanotechnology and Nanomedicine, Hacettepe University**

Examining Committee Members:

Prof. Dr. Mehmet Parlak
Physics, METU

Prof. Dr. Mürvet Volkan
Chemistry, METU

Prof. Dr. Necati Özkan
Polymer Science and Tehnology, METU

Prof. Dr. İsmail Cengiz Koçum
Biomedical Eng., Başkent Uni.

Prof. Dr. Murat Kaya
Chemical Eng., Atılım Uni.

27/07/2023

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Seçkin Öztürk

Signature:

ABSTRACT

CONSTRUCTION OF A VARIABLE FREQUENCY AC MAGNETIC FIELD DEVICE AND MAGNETIC HYPERTHERMIA APPLICATIONS ON FERRITE NANOPARTICLES

Öztürk, Seçkin
Doctor of Philosophy, Micro and Nanotechnology
Supervisor: Prof. Dr. Mürvet Volkan
Co-Supervisor: Assoc. Prof. Telem Şimşek

July 2023, 155 pages

Magnetic hyperthermia is a promising therapeutic method for cancer treatment. This method is based on the intratumoral deposition of biocompatible magnetic nanoparticles followed by exposure to a high-frequency electromagnetic field. Research in this field continues to intensify.

Within the scope of this thesis, a variable frequency magnetic hyperthermia system has been developed for research in magnetic fluid hyperthermia. The developed system can apply an alternative magnetic field in the frequency range of 20 to 80 kHz with amplitudes up to 2 kA/m. With a temperature measurement accuracy of 0.1°C and software developed during operation, the SAR (Specific Absorption Rate) values can be measured with high accuracy even under low heating conditions. Thus, an experimental setup has been obtained that allows the testing of the Linear Response Theory under low magnetic fields.

The magnetic nanoparticles required for testing the developed system were also produced within the scope of this study and characterized using chemical, crystallographic, and magnetic methods. The characterization results indicated a wide distribution of magnetic properties for the obtained materials. The produced and characterized particles were subjected to experiments using the developed hyperthermia system. The results obtained from the experiments were compared with physical models to confirm the functionality of the system.

In hyperthermia studies, the multitude of parameters that can affect heating and their direct or indirect interdependencies make it challenging to draw systematic conclusions from experimental studies. To overcome this challenge, it is necessary to increase the diversity of experimental studies.

Although there are numerous theoretical studies on the Linear Response Theory under low magnetic fields, experimental studies are rarely encountered. It is believed that the hyperthermia system developed during the study will contribute diversity of experiments in this research area. With its low magnetic field intensity and precise temperature measurement capabilities, the system could provide insights into challenging aspects that are often difficult to explain.

Keywords: Magnetic Hyperthermia, Magnetic Nanoparticle, Alternating Magnetic Field Generator, Low-Field Regime, Linear Response Theory

ÖZ

DEĞİŞKEN FREKANSLI AC MANYETİK ALAN CİHAZININ İNŞASI VE FERRİT NANOPARÇACIKLARDA MANYETİK HİPERTERMİ UYGULAMALARI

Öztürk, Seçkin
Doktora, Mikro ve Nanoteknoloji
Tez Danışmanı: Prof. Dr. Mürvet Volkan
Ortak Tez Danışmanı: Doç. Dr. Telem Şimşek

Temmuz 2023, 155 sayfa

Manyetik hipertermi, kanser tedavisi için umut vaat eden bir terapötik yöntemdir. Bu yöntem, biyolojik olarak uyumlu manyetik nanoparçacıkların tümör içine yerleştirilmesine ve sonrasında yüksek frekanslı elektromanyetik alana maruz bırakılmasına dayanmaktadır. Bu alandaki çalışmalar yoğunlaşarak devam etmektedir.

Bu tez kapsamında manyetik akışkan hipertermi araştırmalarında kullanılabilecek değişken frekanslı manyetik hipertemi sistemi geliştirilmiştir. Geliştirilen sistem 20 – 80 kHz frekans aralığında ve 2 kA/m’ye kadar şiddette alternatif manyetik alan uygulayabilmektedir. 0,1 ºC’lik sıcaklık ölçüm hassasiyeti ve çalışma sırasında geliştirilen yazılımlar SAR değerlerinin düşük ısınma şartlarında da yüksek doğruluk ile ölçülmesini sağlamıştır. Bu sayede düşük manyetik alan altında Lineer Tepki Teorisi’nin test edilmesine imkan sağlayacak bir deneysel düzenek elde edilmiştir.

Geliştirilen sistemin test edilmesi için ihtiyaç duyulan manyetik nanoparçacıklar da bu çalışma kapsamında üretilmiş ve kimyasal, kristalografik ve manyetik yöntemlerle karakterize edilmiştir. Karakterizasyon sonuçları elde edilen malzemelerin manyetik özelliklerinin geniş bir skalaya dağıldığını göstermiştir. Üretilen ve karakterize edilen parçacıklar geliştirilen hipertermi cihazı kullanılarak deneylere tabi tutulmuştur. Deneylerden elde edilen sonuçlar fiziksel modeller ile karşılaştırılarak sistemin doğruluğu teyit edilmiştir.

Hipertermi çalışmalarında ısınmaya etki edebilecek parametrelerin çokluğu ve bunların doğrudan ya da dolaylı olarak birbirlerine bağlı olmaları deneysel çalışmalardan sistematik sonuçlar çıkarmayı zorlaştırmaktadır. Bu zorluğun aşılabilmesi için deneysel çalışmaların çeşitliliğinin arttırılması gerekmektedir.

Özellikle düşük manyetik alan altında Lineer Tepki Teorisi üzerine birçok teorik çalışma bulunmasına rağmen deneysel çalışmalara nadiren rastlanmaktadır. Çalışma kapsamında geliştirilen hipertermi sisteminin bu alandaki çalışmalara çeşitlilik katacağı ve sahip olduğu düşük manyetik alan şiddeti ve hassas sıcaklık ölçme yetenekleri ile açıklamada güçlük çekilen noktaların aydınlatılmasına katkı sağlayabileceği düşünülmektedir.

Anahtar Kelimeler: Manyetik Hipertermi, Manyetik Nanoparçacık, Alternatif Manyetik Alan Jenaratörü, Düşük Alan Rejimi, Lineer Tepki Teorisi

…to my lovely Familty

Eda, Demir, Doruk

ACKNOWLEDGEMENTS

I would like to express my sincere regards and gratitude to my supervisor Prof.Dr. Mürvet Volkan for her support, patience, inspiration and excellent guidance during my Ph.D. study.

I would also like to acknowledge my Co-Supervisor, Assoc.Prof.Dr. Telem Şimşek for her invaluable help, supportive advices and her comments during this project.

I am thankful to members of my thesis committee, Prof.Dr. Mehmet Parlak, Prof.Dr. Necati Özkan, Prof.Dr. İsmail Cengiz Koçum, and Prof.Dr. Murat Kaya for their insightful comments and encouragement.

I would like to express my special thanks to my former committee members Prof.Dr. Çiğdem Erçelebi and Prof.Dr. Macit Özenbaş, who provided unwavering support and contributions from the very beginning of both my thesis journey and professional career.

I also want to extend my gratitude to Prof.Dr. Raşit Turan, who supported me under all circumstances, and to GÜNAM, which allowed me to use the thermal camera under his supervision.

I am also deeply indebted to Prof.Dr. Burcu Akata Kurç for her friendship, provided infrastructure support, and tolerance.

I would like to express my gratitude to Dr. İbrahim Çam, who guided me and helped me progress when I was stuck.

I would like to thank Dr. İlker Yıldız, Dr. Tuğba Endoğan Tanır, Dr. Yeliz Akpınar, Pelin Paşabeyoğlu and Mervenur Keleş for their help during the analyses.

This endeavor would not have been possible without enjoyable friendship and support of Sedat Canlı, Hikmet Eren, and Dr. Salih Kaan Kirdeciler for during my thesis study.

I also thank to my mother, my father and my brother for their support and trust throughout my educational life.

I also thank the Tuna family, who have been my second family, for all the help and support.

I would like to express my heartfelt thanks to my dear son Demir, who witnessed and always motivated me throughout a significant portion of my thesis work, and to my dear son Doruk, whose smiling face takes away my fatigue and cheers me up. I am truly grateful for their presence and endless love.

Last but not least, I would like give my deepest gratitude to my dear companion Dr. Eda Tuna Öztürk, who has been by my side since the first day of my work, providing me strength with her presence and love, contributing to the completion of my study through numerous sacrifices, and adding meaning to my life.

TABLE OF CONTENTS

ABSTRACT v

ÖZ vii

ACKNOWLEDGEMENTS x

TABLE OF CONTENTS xii

LIST OF TABLES xvi

LIST OF FIGURES xvii

LIST OF ABBREVIATIONS xx

CHAPTERS

1 INTRODUCTION AND GENARAL THEORY 1

1.1 Hyperthermia and Cancer Treatment 1

1.1.1 How is Hyperthermia Used to Treat Cancer? 2

1.1.2 Magnetic Nanoparticles and Cancer Treatment 3

1.2 Theory of Induction Heating of Magnetic Fluids 3

1.2.1 Eddy Currents 4

1.2.2 Hysteresis Losses 5

1.2.2.1 Magnetic Hysteresis 6

1.2.2.2 Diamagnetism 8

1.2.2.3 Antiferromagnetism 9

1.2.2.4 Paramagnetism 9

1.2.2.5 Ferromagnetism 10

1.2.2.6 Ferrimagnetism 11

1.2.2.7 Superparamagnetism 11

1.3 AMF Hyperthermia of Magnetic Nano Fluids 13

1.3.1 Relaxation Mechanism for MNPs 13

1.3.1.1 Brownian Relaxation Time 14

1.3.1.2 Néel Relaxation Time 15

1.3.2 Superparamagnetic Hysteresis Losses 15

1.3.3 Linear Response Theory 17

1.3.3.1 High Field Regime 18

1.3.3.2 Low Field Regime 19

1.3.4 Effective Magnetic Field Amplitude 19

1.3.5 Anisotropy Parameter 20

1.3.6 Energy Transfer Efficiency 21

2 MATERIALS AND METHODS 23

2.1 Synthesis and Characterization of MNP’s 23

2.1.1 Synthesis of Ferrite Nanoparticles 23

2.1.2 Hydrothermal Growth 25

2.2 Characterization of the Nanoparticles 26

2.2.1 XRD Measurements 26

2.2.2 TEM Images 28

2.2.3 EDX Measurements 29

2.2.4 XPS Measurements 30

2.2.5 Vibrating Sample Magnetometer Measurements 31

2.3 Hyperthermia Setup 33

2.3.1 Alternating Magnetic Field Generator 33

2.3.2 Design of Electronic Circuit 34

2.3.2.1 Tank Circuits 35

2.3.2.2 Half-Bridge Inverter 36

2.3.2.3 Full-Bridge Inverter 37

2.3.3 Temperature Measurement 39

2.3.3.1 Alcohol Thermometer 40

2.3.3.2 Thermal Camera 41

3 EXPERIMENTAL RESULTS 43

3.1 Chemical, Crystallographic and Morphologic Characterization Results ... 43

3.1.1 EDX Results 43

3.1.2 XPS Results 45

3.1.3 XRD Results 47

3.1.4 TEM Results 50

3.1.5 Comparison of Chemical, Crystallographic and Morphologic Characterization Results 53

3.2 Magnetic Characterization Results 54

3.2.1 VSM Results 54

3.2.2 ZFC- FC Measurement Results 57

3.3 Implementation of Custom Hyperthermia System 60

3.3.1 Computer-Controlled Frequency Generator 60

3.3.2 Coil Driver 63

3.3.3 Measurement Cell 65

3.3.4 Video to Heating Curve Software 66

3.4 Validation of the Custom Hyperthermia System 68

3.4.1 Waveform and Magnetic Field Calculations 68

3.4.2 Repeatability Tests of Hyperthermia Experiments 71

3.4.3 Validation with Simulation 74

3.5 Validation of Developed Heating Curve Fitting Software 75
3.6 Results of Magnetic Hyperthermia Experiments 77
3.6.1 Results of Custom HyperthermiaSsystem 77
3.6.2 Results of Commercial System 81
3.6.3 Discussions on Magnetic Heating Behaviors 82
3.6.4 Evaluation of the Experimental Results 83
4 CONCLUSIONS 87
REFERENCES 89

APPENDICES

A: Code of the Curve Fit Software 97
B: Video to Heating Curve Converter Software 114
C: Code of the Computer Side Software for Function Generator 130
D: Code of the Microcontroler Side Software for Function Generator 140
CURRICULUM VITAE 154

LIST OF TABLES

TABLES

Table 2.1 Synthesis solutions of co-precipitation 25

Table 2.2 Synthesis solutions of hydrothermal growth 25

Table 3.1 Table of EDX quantification results 45

Table 3.2 Obtained average crystallite sizes for samples 48

Table 3.3 Normalized peak heights obtained from the peak deconvolution applied to XRD data 50

Table 3.4 Measured particle diameters from TEM images 52

Table 3.5 Summary of obtained characterization results and calculated values 59

Table 3.6 Calculated magnetic field strengths for different frequencies 70

Table 3.7 ΔT values at 400 s for consecutive measurements M1, M2, M3 and M4 72

Table 3.8 ΔT values at 400 s for measurements from consecutive weeks W1, W2, W3 and W4 73

Table 3.9 Experimental simulation results 74

Table 3.10 SAR values obtained with the lab made system at different frequencies 78

Table 3.11 Table of Magnetic saturation and SAR values of Si0, Si7 and Si10 samples obtained from the custom hypertermia system @ 25,5kHz and 1,7kA/m . 80

Table 3.12 SAR values obtained with the commercial system in oil and water 82

Table 3.13 Summary of the results for Si0, Si7 and Si10 samples 84

Table 3.14 The calculated σ and h_0 values from the magnetic measurements 85

LIST OF FIGURES

FIGURES

Figure 1.1 Illustration of Eddy currents 4

Figure 1.2 Typical hysteresis loop of a ferromagnetic material. M vs. H: M_r is the remnant magnetization at H=0; H_c is the coercivity, (the reverse field that reduces M to zero); M_s is the saturation magnetization [37]... 7

Figure 1.3 Graph of magnetization vs applied magnetic field for ferromagnetic, paramagnetic and superparamagnetic materials... 10

Figure 1.4 Schematic representation of the magnetic moment distributions of the ferromagnetic and superparamagnetic materials under an external magnetic field and zero magnetic field 12

Figure 1.5 Schematic representation of Brownian relaxation and Néel relaxation.. 14

Figure 1.6 Representation of hysteric behaviours of superparamagnetic materials. 16

Figure 1.7 SAR and Ω plots obtained from simulations under various magnetic fields for different σ values [54] 21

Figure 2.1 Diagram of the reactor used for co-precipitation.................................. 24

Figure 2.2 Rigaku MiniFlex.. 26

Figure 2.3 Rigaku Ultima IV 27

Figure 2.4 FEI Technai G2 Spirit Biotwin.. 28

Figure 2.5 FEI Quanta400F SEM and EDAX EDX 30

Figure 2.6 PHI Versaprobe 5000 XPS 31

Figure 2.7 Quantum Design PPMS.. 33

Figure 2.8 Schematic representation of induced magnetic field in a current flowing coil.. 34

Figure 2.9 Simple parallel resonant circuit (tank circuit) 35

Figure 2.10 Circuit diagram of the induction heater 36

Figure 2.11 Circuit diagram of a Half-Bridge inverter 37

Figure 2.12 Schematic view of Full-Bridge inverter 38

Figure 2.13 Circuit diagram of a Full-Bridge inverter 39

Figure 2.14 Measurement cell with alcohol thermometer 40

Figure 2.15 Measurement cell with thermal camera 41

Figure 3.1 Graph of EDX spectrums 44

Figure 3.2 XPS data for Si0, Si7 and Si10 samples at a) Fe 2p, b) Fe 3p c) O 1s and d) Na 1s peaks 46

Figure 3.3 Plot of the XRD data obtained from synthesized samples and magnetite, reference 47

Figure 3.4 Graphs of high resolution XRD data deconvoluted peaks and regenerated peak of a) Si_0 b) Si_7 c) Si_10samples 49

Figure 3.5 TEM images of samples a) Si0, b) Si1, c) Si2, d) Si4, e) Si5, f) Si7, g) Si8, h) Si10 51

Figure 3.6 TEM image and the correspondinng size distribution histograms for the samples Si10 52

Figure 3.7 VSM measurements @ 5 K a) full range of magnetic field b) magnetic field from -3000 Oe to 3000 Oe 55

Figure 3.8 VSM measurements @ 300 K a) full range of magnetic field b) magnetic field from -3000 Oe to 3000 Oe 56

Figure 3.9 Results of a) ZFC – FC measurements for Si0 Si7 and Si10 samples and the numerical derivatives of the data 58

Figure 3.10 Schematic representation of experimental setup 60

Figure 3.11 Schematic representation of frequency generator 61

Figure 3.12 Schematic representation of a PWM pulse 61

Figure 3.13 Full Bridge inverter topology 62

Figure 3.14 Schematic representation of “dead time” for two PWM pulses 63

Figure 3.15 Circuit diagram of the coil driver 64

Figure 3.16 PCB layout of the coil driver 65

Figure 3.17 Drawing of the measurement cell 66

Figure 3.18 Flow chart of the video to heating curve software 67

Figure 3.19 The potential drop across a resistor with a known resistance of 1 Ω, connected in series with the coil, was measured using an oscilloscope................... 69

Figure 3.20 Current profiles for 22 kHz and 77 kHz 69

Figure 3.21 Short term repeatability test results 71

Figure 3.22 Long term repeatability test results... 73

Figure 3.23 a) Heating curve of a sample (blue dots) and fitted line with parameters Pin= 0,786, LP = 0,032. (yellow dots) b) Cooling curve of a sample (blue dots) and fitted line with parameters Pin= 0, LP = 0,032. (yellow dots)................................ 77

Figure 3.24 Graph of SAR values obtained with the lab made system at different frequencies 78

Figure 3.25 Graph of SAR values obtained with the lab made system at different periods.. 79

Figure 3.26. Graphs of measured heating curves utilising the custom hypertermia system @ 25,5kHz and 1,7kA/m 80

Figure 3.27 Graphs of measured heating curves in a) oil and in b) water utilising the commercial system at 300kHz and 26kA/m 82

LIST OF ABBREVIATIONS

ABBREVIATIONS	
SAR	Specific Absorption Rate
AMF	Alternating Magnetic Field
MNP	Magnetic Nanoparticle
XRD	X-ray Diffraction
TEM	Transmission Electron Microscopy
VSM	Vibrating Sample Magnetometer
SEM	Scanning Electron Microscopy
EDX	Energy Dispersive X-ray Spectroscopy
ZFC-FC	Zero Field Cooling -Field Cooling
XPS	X-ray Photoelectron Spectroscopy

CHAPTER 1

INTRODUCTION AND GENARAL THEORY

Due to the increasing number of diagnosed cancer cases each year, research on methods that can effectively treat cancer has gained momentum. Alongside various alternative approaches, studies can be found on diagnostic and therapeutic methods that utilize nanomaterials. Among these nanomaterials, magnetic nanoparticles have garnered significant interest due to their targetability, which makes them applicable in both cancer diagnosis and treatment [1-4]. The ability of magnetic nanoparticles to generate heat under an alternative magnetic field makes them a promising candidate for one of the alternative cancer treatment method “hyperthermia” [5-7].

Magnetic fluid hyperthermia is a promising technique for cancer treatment which utilizes magnetic nanoparticles (MNPs) that release heat when placed in an alternating magnetic field. Researchers are actively working to improve the properties of MNPs, which would allow for the treatment of smaller tumors and reduce the amount of material needed to treat a given size of tumor [8-13].

In this introduction section, the scientific bases underlying magnetic fluid hyperthermia will be presented. Specifically, an overview of hyperthermia and cancer therapy, magnetic material properties, their interaction with alternative magnetic fields, and the theoretical relationship with heat generation will be summarized.

1.1 Hyperthermia and Cancer Treatment

Hyperthermia (also called thermal therapy or thermotherapy) is a type of cancer treatment in which body tissue is exposed to high temperatures (up to 45°C) [14]. Research has shown that high temperatures can damage and kill cancer cells, usually with minimal injury to normal tissues [9, 15-17]. By killing cancer cells and damaging proteins and structures within cells, hyperthermia may shrink tumors. Even if it is conceptually proven to be good candidate for cancer treatment hyperthermia is still under study in clinical trials (research studies with people) and is not widely available.

1.1.1 How is Hyperthermia Used to Treat Cancer?

Hyperthermia is usually used in combination with other cancer treatments such as radiation therapy and chemotherapy [14, 16, 18-23]. It can make some cancer cells more sensitive to radiation or damage cancer cells that radiation cannot. In the conducted studies, it has been observed that the effectiveness of radiotherapy can increase by more than 50% when combined with hyperthermia [24]. Additionally, it has been shown to enhance the effects of certain anticancer drugs [25].

There have been many clinical trials that have studied the combination of hyperthermia with radiation therapy and/or chemotherapy for various types of cancer, including sarcoma, melanoma, head and neck cancer, brain cancer, lung cancer, esophageal cancer, breast cancer, bladder cancer, rectal cancer, liver cancer, appendix cancer, cervical cancer, and mesothelioma [26, 27]. Many of these studies have shown a significant reduction in tumor size when hyperthermia is used in combination with other treatments. However, not all of these studies have shown increased survival in patients receiving the combined treatments.

The effectiveness of hyperthermia treatment is determined by the temperature achieved during the treatment, the length of treatment, and the characteristics of the cancer cells and the surrounding tissue. To make sure that the desired temperature is reached and not exceeded, the temperature of the tumor and surrounding tissue is monitored throughout the treatment.

1.1.2 Magnetic Nanoparticles and Cancer Treatment

Magnetic fluid hyperthermia is a cancer treatment method in which nanoparticles made of magnetic materials like metal oxides (ferrites), or conducting material (metals) are injected or targeted into a tumor and then exposed to a high-frequency alternating magnetic field. These nanoparticles absorb the energy of the incident electromagnetic waves through different mechanisms and converts into heat, resulting a raise in the tumor temperature up to 40-46 °C, which can damage the cancer cells [28-30]. Another potential benefit of magnetic nanoparticles is their ability to combine hyperthermia with chemotherapeutic agents [31]. There have been numerous studies on creating composite structures containing magnetic nanoparticle and anticancer drugs aiming to mutually increase the effectiveness of the individual treatments. For this purpose, nanoparticles are covered with drug containing layers (polymers, lipids) and exposed to alternating magnetic fields. With the help of heat generated by the nanoparticles, the drugs are released or activated on the site of tumors exhibiting a combined effect on cancer cells.

1.2 Theory of Induction Heating of Magnetic Fluids

All altering magnetic fields and, as a result of the laws of physics, altering electric fields transfer energy to the interacting material through specific mechanisms. Describing the interactions between electromagnetic waves and materials constitutes a significant portion of the literature in physics. The energy of the electromagnetic wave and the properties of the material determine the type of interaction. At the atomic and molecular levels, these interactions are generally referred to as photon-material interactions. As a result, changes in spin states, ionization, alterations in molecular bonds, and molecular vibrations can be observed. In conductive materials, energy can be gained through plasmonic interactions by the changing electric field with the electrons at the level of conductivity. Magnetic and conductive materials can acquire energy through two different mechanisms under altering magnetic fields. The mechanisms can be named as “Eddy currents” and “Hysteresis losses”. These two mechanisms will be described in the following text.

1.2.1 Eddy Currents

Eddy currents are circular currents induced in conductors by varying magnetic fields. These currents flow in closed loops on conductor surfaces that are perpendicular to the change in magnetic field vector (Figure 1.1). When eddy currents occur in conductors with finite resistance, heat is generated [32]. Induction furnaces and induction cooktops commonly operate based on this mechanism.

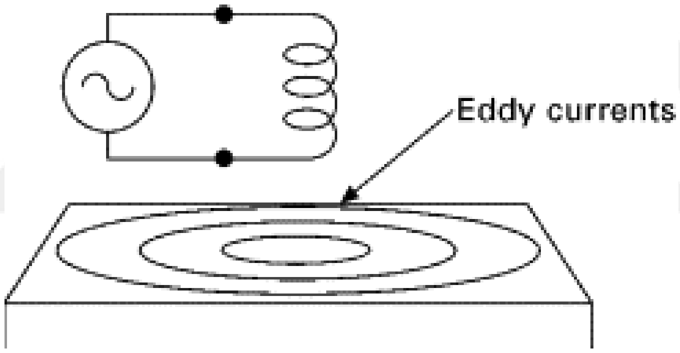


Figure 1.1 Illustration of Eddy currents

In superconductors, where resistance is zero, the generated current does not convert into heat energy. Instead, eddy currents create a magnetic field opposite to the applied magnetic field. This results in a phenomenon called magnetic levitation. Magnetic levitation allows superconducting materials to be lifted or suspended within a magnetic field.

To generate eddy currents, relatively large conducting surfaces are required in the material. In this study, it is believed that the observed heat generation is not related to eddy currents. Since the iron oxide nanoparticles used in the study are small and insulating, the formation of eddy currents is not expected.

1.2.2 Hysteresis Losses

Magnetic materials possess a net magnetic moment due to their atomic and electronic structures. With an externally applied magnetic field, randomly distributed magnetic moments within the material can align in the direction of the applied field, resulting in the accumulation of energy. When a reverse magnetic field is applied after the alignment of magnetic moments, if it can provide enough energy to overcome the energy barrier and reverse the moments, the energy difference is absorbed by the system, causing the moments to reverse. This process releases energy in the form of heat [33].

Magnetic materials generally exhibit resistance to changes in the magnetic field and strive to maintain their magnetic moments. Therefore, there is a recursive function between the applied magnetic moment and the magnetic field. When the graph of this function is plotted under a cyclic magnetic field, a loop can be observed (Figure 1.2). This loop is later referred to as the hysteresis loop. The area of this loop represents the energy released in a single magnetic cycle. The equation for the area of the loop, is stated in Equation 1 [34, 35].

$$A = \int_{-H_{max}}^{+H_{max}} \mu_0 M(H) dH \qquad [1]$$

Where A is the area of the loop *M(H)* is the magnetization and H is the applied magnetic field. A also represents the energy dissipated in one magnetic field cycle. By using A, absorbed energy per unit time or absorbed power can be calculated with the following equation [36].

$$SAR = Af \qquad [2]$$

Where SAR is “specific absorption rate”, A is the area of the hysteresis loop and the f is the frequency of the applied cyclic magnetic field.

M(H) or the resistance of magnetic moment in a material can exhibit different behaviors in different materials. Magnetic hysteresis curves help to define the differentiation of these materials from one to another.

1.2.2.1 Magnetic Hysteresis

When an external magnetic field is applied to a magnetic material, the magnetic moments of the material align with the applied magnetic field. As the strength of the applied magnetic field changes, the amount of aligned magnetic moments also changes. By cyclically varying the magnetic field, a hysteresis loop can be obtained, which represents the changes in the magnetic moment [35]. A hysteresis loop shows the relationship between the induced magnetization (M) and the magnetizing force (H). It is often referred to as the M-H loop (Figure 1.2).

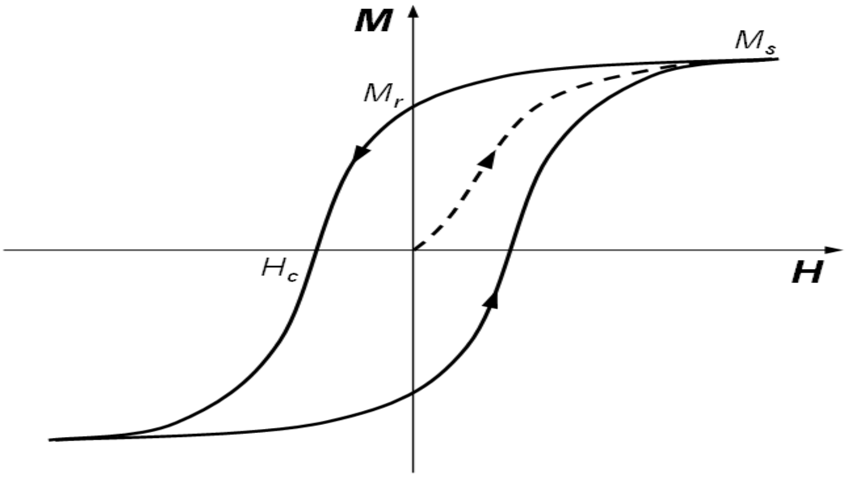


Figure 1.2 Typical hysteresis loop of a ferromagnetic material. M vs. H: M_r is the remnant magnetization at H=0; H_c is the coercivity, (the reverse field that reduces M to zero); M_s is the saturation magnetization [37]

In the following text, terms, which describes the behaviors of magnetic materials is presented.

Magnetic Saturation: When all alienable magnetic moments in the material are aligned with the external magnetic field. The total magnetic moment does not increase even if the applied field strength increases. This condition is known as magnetic saturation [37].

***Remnant* Magnetization:** After reaching magnetic saturation, the remaining magnetic moment in the material is called the remnant magnetization, even when the externally applied magnetic field is removed [37].

Magnetic Coercivety: When an external magnetic field is applied to a magnetic material, the material's magnetic moments align with the applied field. However, when the external field is removed, some materials retain a residual magnetization while others lose their magnetization completely. Magnetic coercivity is the measure of the strength of an opposing magnetic field needed to demagnetize the material and reduce its residual magnetization to zero [37].

Susceptibility: Susceptibility is a measure of how magnetizable a material can become in the presence of a magnetic field and can be used in a general way to describe the various classes of magnetic materials [37].

Magnetic Anisotropy: Magnetic materials can exhibit different magnetic properties depending on the positions of atoms with net magnetic spins within their crystal structure. As a result, magnetization occurs more easily along certain crystal axes, which are referred to as easy axes. Under the influence of an externally applied magnetic field, magnetization occurs along the easy axes, leading to the absorption of magnetic energy. This selective behavior between crystal axes is known as magnetic anisotropy [37].

Under an alternating magnetic field, the magnetization within the material changes direction as the externally applied magnetic field changes direction. The magnetization typically transitions from one easy axis to another, and the energy difference between these axes is converted into heat. The energy difference between the axes is referred to as the anisotropic energy barrier.

The different values of these definitions are associated with different magnetic material classes. Magnetic material behaviors can be classified as diamagnetism, antiferromagnetism, paramagnetism, ferromagnetism, ferrimagnetism and superparamagnetism. These classes are explained in the following text.

1.2.2.2 Diamagnetism

Diamagnetic materials are repelled by the applied magnetic field due to the induced magnetic moments in the opposite direction. Diamagnetism is actually a weak effect based on quantum mechanical principles, specifically the slight changes in orbital energies of paired electrons due to Pauli exclusion principle, which is observed in all materials. In materials where all electrons are paired, the only effect that determines the interaction with the magnetic field is diamagnetism, and thus, these materials are called diamagnetic. In materials with unpaired electrons, stronger mechanisms that interact with the magnetic field come into play, suppressing diamagnetism [38, 39].

1.2.2.3 Antiferromagnetism

Antiferromagnetic materials have atoms with unpaired electrons, resulting in a net magnetic spin. However, in these materials, the spins (magnetic moments) are arranged in an antiparallel manner within the crystal structure, causing the total magnetic moment to be zero. This generally means that the material does not exhibit a net magnetic moment.

However, at temperatures higher than the Neel temperature or under the influence of a high external magnetic field, this ordering can be disrupted, and the material starts to exhibit paramagnetic properties [38, 39].

1.2.2.4 Paramagnetism

Paramagnetism is a characteristic displayed by specific materials that exhibit a weak attraction to an external magnetic field. Unlike ferromagnetic materials, paramagnetic materials do not retain their magnetization after the removal of the external magnetic field. In the absence of an applied magnetic field, the magnetic moments of individual atoms or ions in paramagnetic materials are randomly oriented. However, when an external magnetic field is present, these magnetic moments tend to align themselves in the direction of the field, resulting in a feeble attraction. This alignment is temporary and disappears once the external field is removed. Paramagnetic materials typically possess unpaired electrons, which contribute to their magnetic properties. Examples of paramagnetic materials include oxygen, aluminum, and platinum. Paramagnetism plays a significant role in various scientific and technological applications, such as magnetic resonance imaging (MRI) and magnetic separation techniques (Figure 1.3) [38, 39].

1.2.2.5 Ferromagnetism

Ferromagnetism is a phenomenon exhibited by certain materials in which the atoms or ions within the material have aligned magnetic moments that collectively create a strong and persistent magnetic field. In a ferromagnetic material, the magnetic moments of neighboring atoms or ions interact and align in the same direction, resulting in a net magnetization. This alignment can persist even in the absence of an external magnetic field, making ferromagnetic materials capable of retaining a significant magnetic field and behaving as permanent magnets. Ferromagnetic materials typically have a high magnetic coercivity and exhibit hysteresis in their magnetization behavior [39]. Examples of ferromagnetic materials include iron, nickel, cobalt, and their alloys (Figure 1.3).

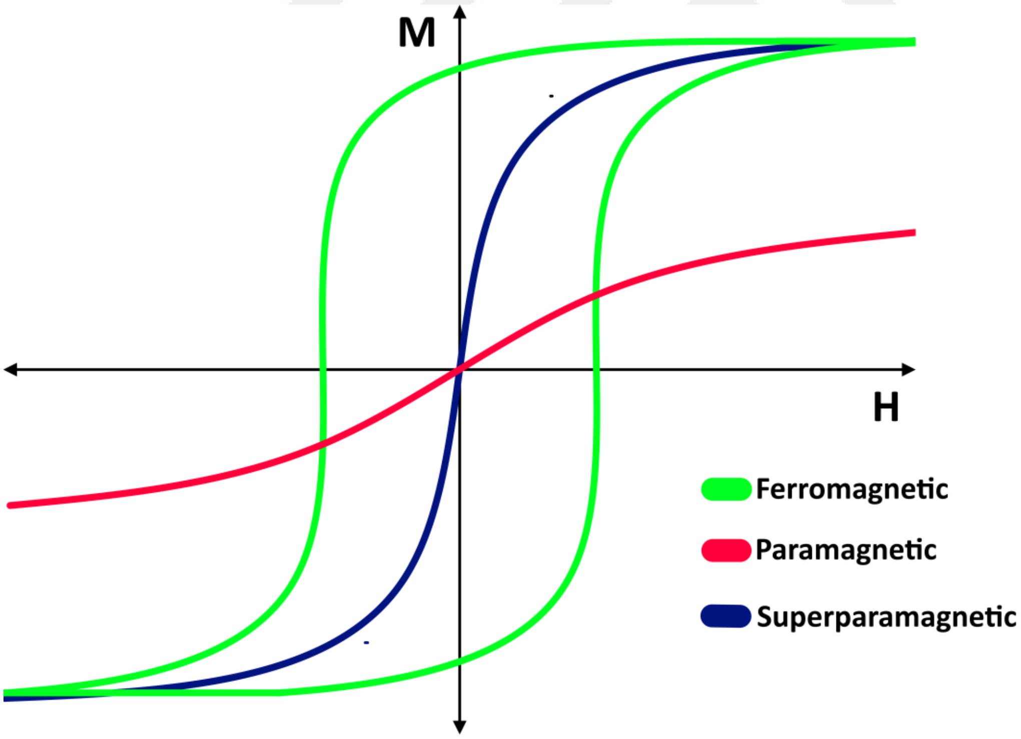


Figure 1.3 Graph of magnetization vs applied magnetic field for ferromagnetic, paramagnetic and superparamagnetic materials

1.2.2.6 Ferrimagnetism

Ferrimagnetism is often considered as a weak form of ferromagnetism, but it is actually a special case of antiferromagnetic. In ferromagnetism, all magnetic spins align parallel to each other, whereas in ferrimagnetism, the magnetic spins align antiparallel to each other like antiferromagnetism, but their magnitudes are not equal. Since the magnitudes of the magnetic spins are not equal, they cannot cancel each other out, and the material exhibits a net magnetic moment even at zero applied magnetic field. Therefore, the magnetic hysteresis loop of ferrimagnetic materials is similar to that of ferromagnetic materials but with a lower magnetic saturation value.

Despite being commonly known as ferromagnetic, both magnetite (Fe_3O_4) and maghemite (γ-Fe_2O_3) have a magnetic structure that involves antiparallel alignment of some of their magnetic spins, while the others remain parallel. Because of their spin structures they are classified as ferrimagnetic materials [39].

1.2.2.7 Superparamagnetism

Superparamagnetism refers to the behavior observed in small magnetic particles or nanoparticles. In superparamagnetic materials, the individual magnetic moments of the particles fluctuate randomly when there is no external magnetic field. This randomness is caused by thermal energy overcoming the energy barriers that would normally keep the moments aligned (Figure 1.4). As a result, superparamagnetic materials do not possess a net magnetic moment in the absence of a magnetic field (Figure 1.3) [40].

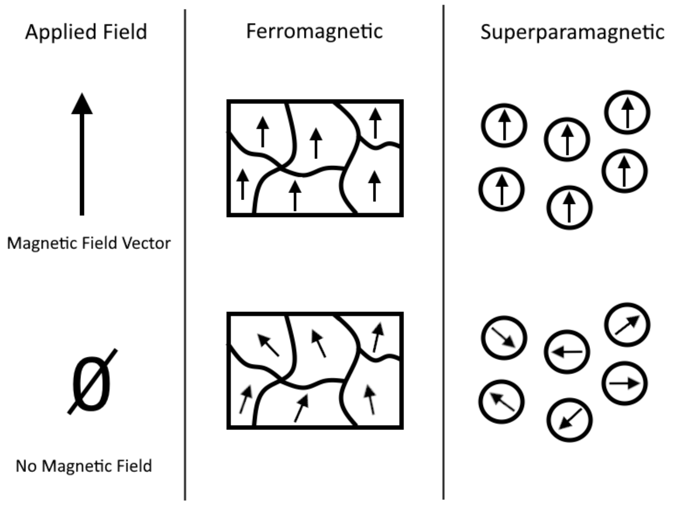


Figure 1.4 Schematic representation of the magnetic moment distributions of the ferromagnetic and superparamagnetic materials under an external magnetic field and zero magnetic field

However, when an external magnetic field is applied, the magnetic moments of the particles align with the field, resulting in a temporary magnetization. Once the external field is removed, the moments return to their random orientations. The degree of magnetization in superparamagnetic materials is highly dependent on temperature, with lower temperatures favoring stronger magnetic alignment.

As evident from the hysteresis curves shown in Figure 1.3, superparamagnetic materials do not exhibit remnant magnetization. Consequently, it can be assumed that they do not possess anisotropic energy barriers and thus would not dissipate heat under a changing magnetic field. However, observations indicate that superparamagnetic particles also experience heating under an alternative magnetic field. The mechanisms behind this phenomenon will be explained in detail in the subsequent sections.

1.3 AMF Hyperthermia of Magnetic Nano Fluids

In hyperthermia applications, magnetic fluids are typically achieved by suspending magnetic nanoparticles in appropriate liquids. Magnetic nanoparticles exhibit superparamagnetic properties due to their small size, and the main factor for heating in these applications is the interaction between superparamagnetic nanoparticles and alternating magnetic fields (AMF). The heating principle of superparamagnetic materials differs from the induction heating principles of other materials [41-45]. In order to further develop these applications, it is crucial to have a thorough understanding of the heating mechanisms of superparamagnetic materials [46].

1.3.1 Relaxation Mechanism for MNPs

Relaxation of magnetic particles can be explained with two different mechanisms, the so-called Néel relaxation and the Brownian relaxation. For both mechanisms magnetization vector of the nanoparticle aligns with the applied external magnetic field (Figure 1.5). For Néel relaxation the magnetization vector changes its orientation in the material without changing the orientation of the particle [47]. On the other hand, for Brownian relaxation, the particle physically rotates to align its magnetic vector [47].

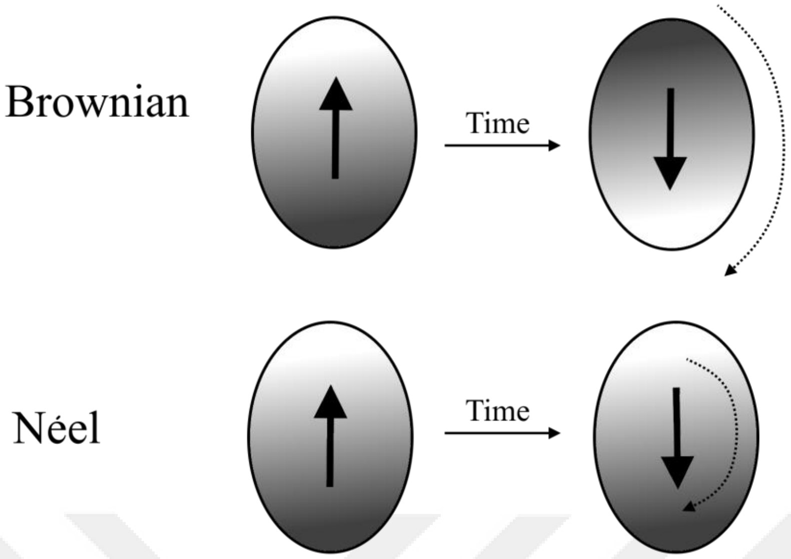


Figure 1.5 Schematic representation of Brownian relaxation and Néel relaxation

For Néel relaxation, time constant mostly depends on the magnetic properties of the material and the magnetic domain size. However Brownian relaxation, time constant can also be affected with the viscosity of the medium and the hydrodynamic volume of the particle. In real physical systems both of the mechanisms occur simultaneously while the faster one dominates the effective relaxation time constant.

1.3.1.1 Brownian Relaxation Time

The Brownian relaxation time defines the time required for the particle to align with the external magnetic field through a rotational motion. Calculation of the Brownian relaxation time τ_B can be found in the following equation 3 in terms of suspension viscosity η, the hydrodynamic volume of the particle V_h and the thermal energy introduced with Boltzmann's constant k_B and local temperature T [46, 48].

$$\tau_B = \frac{3\eta V_h}{k_B T} \quad [3]$$

1.3.1.2 Néel Relaxation Time

In the presence of an external magnetic field, magnetization vector of a magnetic nanoparticle can only stand stable in two different positions called as parallel and antiparallel. These two opposite vector positions are separated by an energy barrier. Each flip from one to the other absorbs or emits certain amounts of energy. Time required for each flip is named as Néel relaxation time τ_N and the formula can be found in equation 4 in terms of anisotropy energy density K_{eff}, volume V and the thermal energy introduced with Boltzmann's constant k_B and local temperature T.

Because of the nanoparticle's magnetic anisotropy, the magnetic moment has usually only two stable orientations antiparallel to each other, separated by an energy barrier. The stable orientations define the magnetic easy axis of the nanoparticle. At finite temperature, there is a finite probability for the magnetization to flip and reverse its direction. The mean time between two flips is called the Néel relaxation time τ_N and is given by the Néel-Arrhenius equation [46].

$$\tau_N = \tau_0 \exp\left(\frac{K_{eff}V}{k_B T}\right) \quad [4]$$

1.3.2 Superparamagnetic Hysteresis Losses

Superparamagnetic materials are known as materials that do not exhibit hysteresis. When the magnetic field is removed, they do not show remnant magnetization due to thermodynamic effects that disrupt the magnetic alignment in the material. However, they can exhibit hysteric behaviour in the presence of a rapidly changing magnetic field that do not exceeds the time scale of thermodynamic effects to disrupt the magnetic alignments. This behaviour can vary depending on the amplitude of the applied alternative magnetic field. In the presence of high magnetic field changes, they exhibit behaviour similar to an unsaturated ferromagnetic hysteresis loop. As the magnetic field strength decreases, this loop evolves from a distorted ellipse towards an ellipse. This behaviour is schematically shown in Figure 1.6 [49].

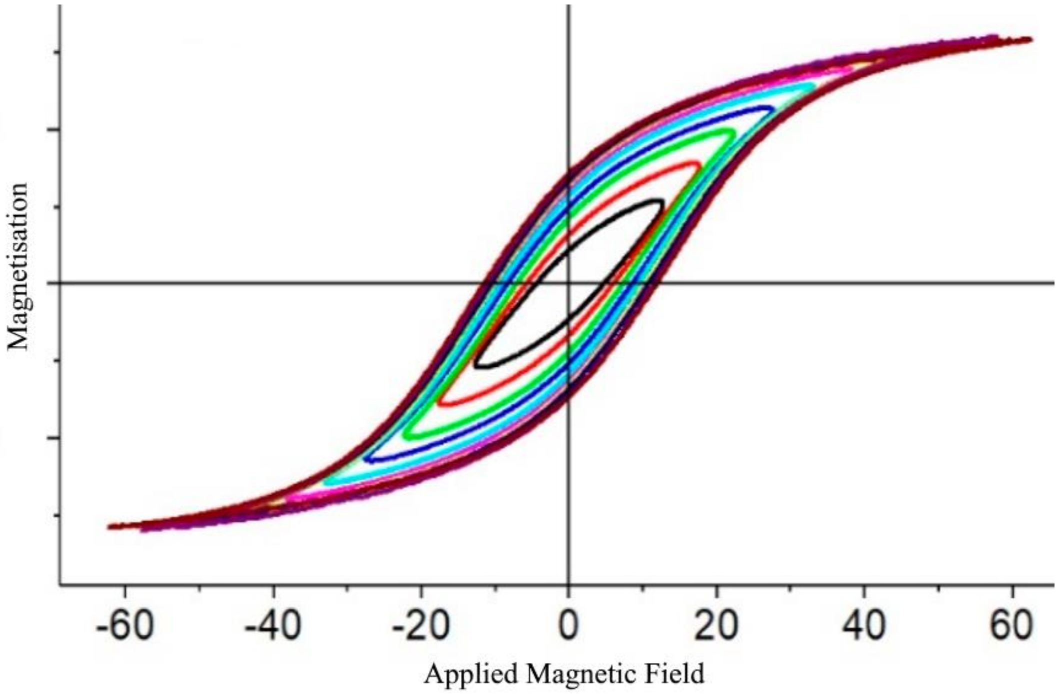


Figure 1.6 Representation of hysteric behaviours of superparamagnetic materials

In the graph, it can be observed that the hysteric loop area is proportional to the magnitude of the applied magnetic field. As the magnitude of the magnetic field increases, the hysteric loop area also increases. This is because higher magnetic fields tend to align the magnetic moments more strongly and bring the material closer to saturation. Therefore, with larger magnetic fields, the hysteric loop area is generally larger. However, this relationship can vary depending on the specific properties of the material and the conditions of the superparamagnetic behaviour.

1.3.3 Linear Response Theory

This approach is based on the assumption that the magnetization in the material has a linear relationship with the applied magnetic field. In this case, the susceptibility (χ_0=M/H) is assumed to be a constant value. Susceptibility consists of two components, one in phase and the other out of phase (Equation 5) [50].

$$\chi_0 = \chi\,' - i\chi\,'' \qquad [5]$$

χ' is the in-phase component and does not cause hysteresis. On the other hand, χ'' is the component that determines the phase difference between the rapidly changing magnetic field and magnetization. Due to this delay-induced hysteresis, heating occurs. Incoherent term of the susceptibility and the absorbed power is given by Equation 6 and 7 [51-53].

$$\chi\,'' = \chi_0 \frac{\omega\tau}{1+(\omega\tau)^2} \qquad [6]$$

Where $\omega = 2\pi f$ and τ is the magnetic relaxation time and

$$P = \mu_0 \pi \chi'' f H_0^2 \qquad [7]$$

Where, P is the absorbed power, μ_0 is the permeability, f is the frequency, and H_0 is the applied magnetic field. As can be understood from the equation, the absorbed power has a quadratic relationship with the applied magnetic field strength. The reason for this relationship is that as the magnetic field strength increases, the number of spins aligned with the magnetic field also increases, leading to an increase in magnetization. However, when all spins are aligned with the magnetic field, further increasing the magnetic field strength will not lead to an increase in magnetization, and the linear relationship between magnetization and magnetic field is disrupted. In this case, it's evident that Linear Response Theory (LRT) becomes invalid when H_0 is greater than H_K. On the other hand, it has been reported in the literature that deviations from LRT become noticeable when H_0 is greater than $0.02H_K$ [54]. Here, H_K represents the anisotropic field (Equation 8).

$$H_K = \frac{2K_{eff}}{\mu_0 M_s} \qquad [8]$$

In this equation, K_{eff} is the anisotropy constant, and M_s is the magnetic saturation value [55].

1.3.3.1 High Field Regime

When the applied magnetic field is large enough to overcome the anisotropic energy barriers of the particles, all particles can align with the applied magnetic field. With each reversal of the magnetic field, an amount of energy equal to the energy barrier is dissipated as heat in the system. In cases where the magnetic field magnitude is sufficiently high, a behaviour similar to the hysteresis loop with the largest loop area is expected seen in Figure 1.6. This regime can be referred to as the high field regime.

In the high field regime, when all other parameters are disregarded, materials with high magnetic saturation are expected to provide larger SAR values [54, 56, 57].

1.3.3.2 Low Field Regime

If the magnitude of the applied magnetic field is not sufficient for all particles to overcome the anisotropic energy barrier and align with the magnetic field, a behaviour resembling elliptical hysteresis loops seen in Figure 1.6 is expected. This condition can be defined as the low field regime.

Under the low field regime conditions, as the magnetic saturation is not approached, the influence of the material's magnetic saturation on the heating behaviour reduces. The contributions of parameters such as particle size, relaxation times, and anisotropy constant become more pronounced [54, 56, 57].

1.3.4 Effective Magnetic Field Amplitude

In the literature, this value is referred to the ratio of the applied magnetic field to the material's magnetic field saturation value and anisotropy constant (Equation 9).

$$h_0 = \frac{H_0 M_S}{2K_{eff}} \tag{9}$$

In this equation, H_0 represents the applied magnetic field, K_{eff} is the effective anisotropy constant, and M_s is the magnetic saturation value. h_0 is an important parameter as it determines the likelihood of the applied magnetic field performing work against the magnetic properties of the material [36].

1.3.5 Anisotropy Parameter

To be able to investigate the heating behaviours of the materials under low field regime, anisotropy parameter, which contains the other probably effecting parameters should be understood. The anisotropy parameter σ is defined with the following equation (Equation 10) [36].

$$\sigma = \frac{K_{eff}V}{k_B T} \tag{10}$$

Here, K_{eff} represents the effective anisotropy constant, V is the volume of the particles, k_B is the Boltzmann constant, and T is the temperature. The effects of the σ parameter are particularly influential, especially in the low field regime, as observed in simulation studies in the literature (Figure 1.7).

Upon examining these graphs, it can be observed that materials with a low σ value (σ = 8) exhibit high SAR and efficiency values at low h_0 values ($h_0 = H_0/H_A$). For these materials, as h_0 increases, the SAR value slowly increases, but the efficiency continuously decreases. Materials with higher σ values, on the other hand, exhibit low SAR values for low h_0 values. However, after a certain threshold h_0 value, the SAR values rapidly increase (for σ =10, $h_0 > 0.1$; for σ =12, $h_0 > 0.2$). Accordingly, it can be observed that the efficiency values also reach their maximum at different h_0 values for different σ values. Therefore, for medical applications, especially under low-field conditions, selecting materials with an optimum σ value rather than maximizing magnetic properties will enhance efficiency.

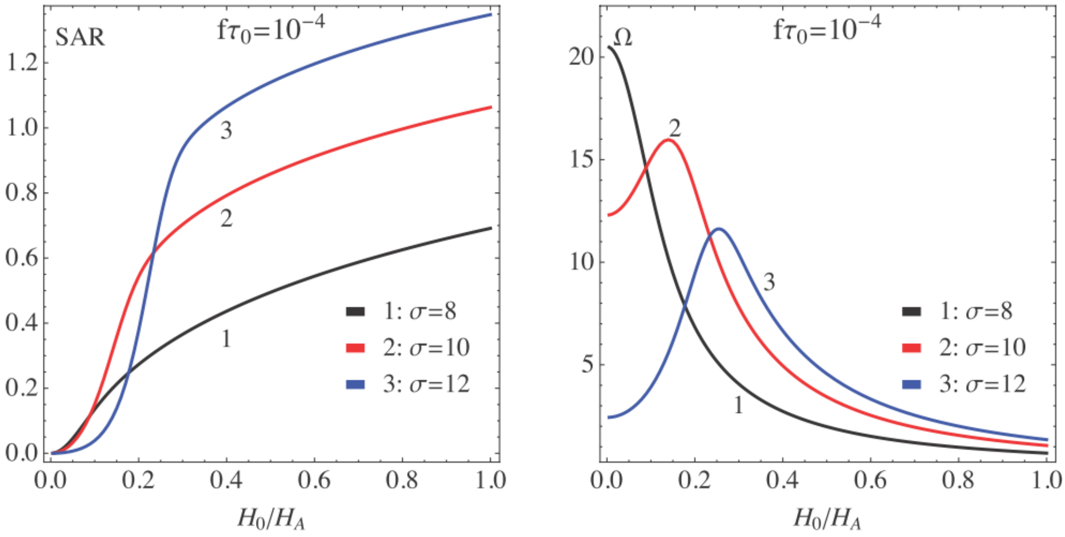


Figure 1.7 SAR and Ω plots obtained from simulations under various magnetic fields for different σ values [54]

1.3.6 Energy Transfer Efficiency

One of the important parameters to understand in hyperthermia studies is the energy transfer efficiency. It is stated that the specific absorption rate (SAR) represents the heat generated by the material through absorption of energy. Equation 11 is provided to calculate SAR as the product of the hysteresis loop area (A) and the frequency (f) of the applied magnetic field (SAR = Af). The value of A can be expressed based on the material properties and the magnitude of the applied magnetic field [34].

$$A_0 = \pi H_0^2 \chi_0 \frac{\omega\tau}{1+(\omega\tau)^2} \qquad [11]$$

When the frequency is assumed to be constant, it can be observed that the SAR value increases quadratically with the H_0 value. This behavior of the A is referred to as the linear regime. On the other hand, the A value reaches a certain limit with the elevated values of H_0 and does not show an increase dependent on the H_0 value beyond that limit.

To measure the energy transfer efficiency, the ratio of the given energy to the SAR value needs to be examined (Equation 12) [36, 58].

$$\Omega = \frac{SAR}{H_0^2} \tag{12}$$

In this equation, H_0^2 represents the energy density given to the particles. As seen from the equation, when the A value behaves as in Equation 11, the Ω value is independent of H_0. This is because both the numerator and the denominator are proportional to H_0^2. On the other hand, when the A value reaches its limit, the SAR value remains constant while the denominator continues to increase with the square of H_0. In this case, increasing the magnetic field amplitude reduces the efficiency.

CHAPTER 2

MATERIALS AND METHODS

This chapter presents the synthesis procedures and parameters of the magnetic nanoparticles used in hyperthermia experiments of this study, as well as the characterization results of the nanoparticles. Various characterization techniques (XRD, XPS, TEM and VSM) are applied to MNP's.

Addition to this, design principles and details of the hyperthermia setup can be found in this chapter. Details of electronic design, temperature readings and developed software are presented in the following text.

2.1 Synthesis and Characterization of MNP's

It is known that various alternative materials can be used in studies focusing on magnetic hyperthermia. Among these materials, magnetite (Fe_3O_4) and maghemite (γ-Fe_2O_3) is known for its biocompatibility, good magnetic properties, and ease of production. For these reasons, ferrite materials have been selected as the main subject of this study. To be able to reveal the effect of magnetic properties on magnetic heating, no modifications (coatings) are applied to the produced magnetite nanoparticles.

2.1.1 Synthesis of Ferrite Nanoparticles

In this study, ferrite nanoparticles are synthesized by co-precipitation method with three different reactant concentrations. Some of the obtained particles are used as synthesized while some are exposed to different hydrothermal conditions to vary their properties.

Ultra-fine particles of ironoxide are prepared by co-precipitating aqueous solutions of $FeCl_2$ and $FeCl_3$ mixtures. The iron chloride solutions are prepared by adding $FeCl_2$ and $FeCl_3$ powders in to a volumetric flask prior to water addition. Various volumes of DI water (Table 2.1) are added to volumetric flask to achieve different reactant concentrations and the solutions are kept at 80^{o}C for 20 min. NaOH solution (2g NaOH is dissolved in 25 ml of distilled water) is added to this mixture within 5 minutes (2 ml for every 20 – 25 sec) under constant stirring. Magnetite is formed by conversion of metal salts into hydroxides, which take place immediately, and transformation of hydroxides into ferrites. The solution is maintained at 80^{o}C for 40min (Figure 2.1). The Fe_3O_4 particles are washed 3 times with distilled water utilizing magnetic separation techniques. The magnetic ironoxide nanoparticle containing mud is dried at 80 ^{o}C for 24 h under atmospheric conditions [59-64].

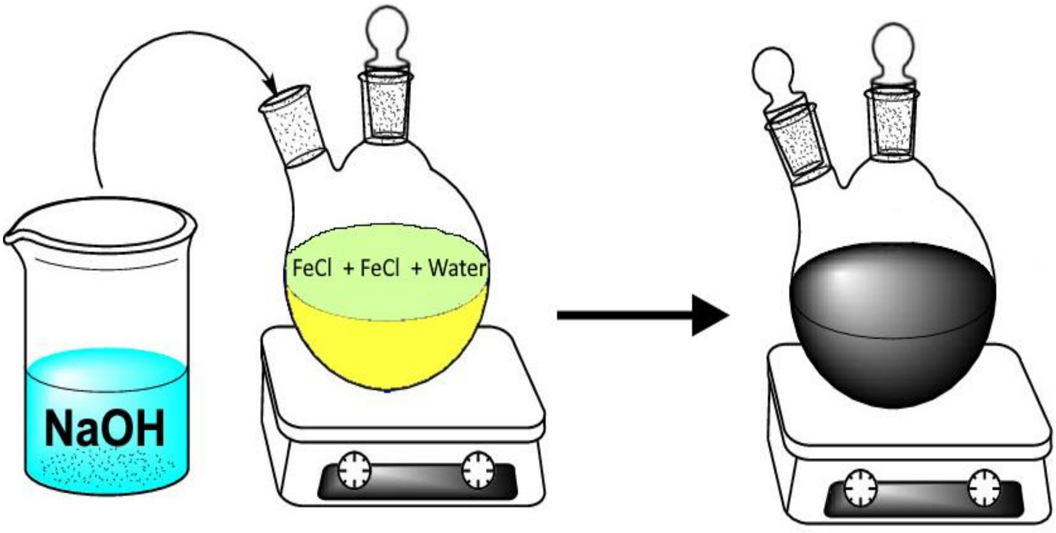


Figure 2.1 Diagram of the reactor used for co-precipitation

Table 2.1 *Synthesis solutions of co-precipitation*

Sample name	$FeCl_2$	$FeCl_3$	NaCl	Water
Si0	2.0 g	1.6 g	-	200 ml
Si7	2.0 g	1.6 g	-	400 ml
Si8	1.0 g	0.8 g	-	400 ml
Si10	4.0 g	3.2 g	2.0 g	400 ml

2.1.2 Hydrothermal Growth

To be able to utilize the Oswald ripening mechanism, synthesized particles are subjected to various hydrothermal conditions depending on the parameters like time and temperature. 400 mg of synthesized and dried particle powder is added into 10 ml autoclaves and re-dispersed in 8ml of DI water. To be able to recreate the synthesis solution conditions, 1,6 g of NaCl is added to the suspensions prior to the addition of HCl solutions. The mixtures are autoclaved under various hydrothermal conditions (Table 2.2).

Table 2.2 *Synthesis solutions of hydrothermal growth*

Sample name	Amount of Si_0	NaCl	1 M HCl solution	Temperature (^{0}C)	Time (h)
Si_0	As Synthesized				
Si_1	400 mg	1.6 g	200 µl	85	24
Si_2	400 mg	1.6 g		85	24
Si_4	400 mg	1.6 g	200 µl	135	24
Si_5	400 mg	1.6 g	200 µl	150	48
Si_7	As Synthesized Si_7				
Si_8	As Synthesized Si_8				
Si_10	As Synthesized Si_10				

2.2 Characterization of the Nanoparticles

In this section, the characterization techniques applied to the fabricated materials are presented. The equipments used for these techniques are described along with the parameters at which they were operated, and the specific information that was sought to be obtained from these techniques.

2.2.1 XRD Measurements

X-Ray diffraction is a technique used for the crystallographic analysis of materials. It relies on illuminating the material with a monochromatic X-ray source. The X-rays interacting with the material are scattered according to Bragg's law from its regular structures (atomic planes). Regularly scattered X-rays interfere with each other, creating a diffraction pattern. A detector measures the intensity of X-rays produced after the diffraction. During an XRD measurement, the detector and the sample have continuously changing angles relative to each other to capture the graph of X-rays scattered at different angles. This graph is called an X-ray diffractogram. By examining the X-ray diffractogram, detailed information about the crystal structure of the material can be obtained, as the distances between the regular atomic planes within the material determine the diffraction pattern.

Figure 2.2 Rigaku MiniFlex

In this study, general XRD measurements were performed using the Rigaku MiniFlex (Figure 2.2) instrument equipped with a copper (Cu-Kα) X-ray source. The scanning speed of 1º/min was used. Diffractograms were generated within the 2θ angle range of 20º to 90º. These measurements aimed to investigate the presence of non-magnetic iron oxide phases. Additionally, the peak widths in the diffractograms were analyzed to obtain information about particle size. The obtained results are presented in the following chapter.

It is known that the separation of magnetite and maghemite phases is not possible in general measurements. To enable the separation of these phases, the Rigaku Ultima IV (Figure 2.3) instrument with a copper (Cu-Kα) X-ray source was used. A monochromator was employed in the measurements to enhance the resolution, and the scanning was conducted at a speed of 1º/15 min. The obtained results are presented in the following chapter.

Figure 2.3 Rigaku Ultima IV

2.2.2 TEM Images

TEM (Transmission Electron Microscopy) is a technique used for the morphological and crystallographic analysis of materials. It is based on the examination of electrons that are transmitted through the material after interacting with it. The transmitted electrons are selectively detected and used to generate images. The image obtained from electrons that pass through the material without diffraction is called a "bright field image."

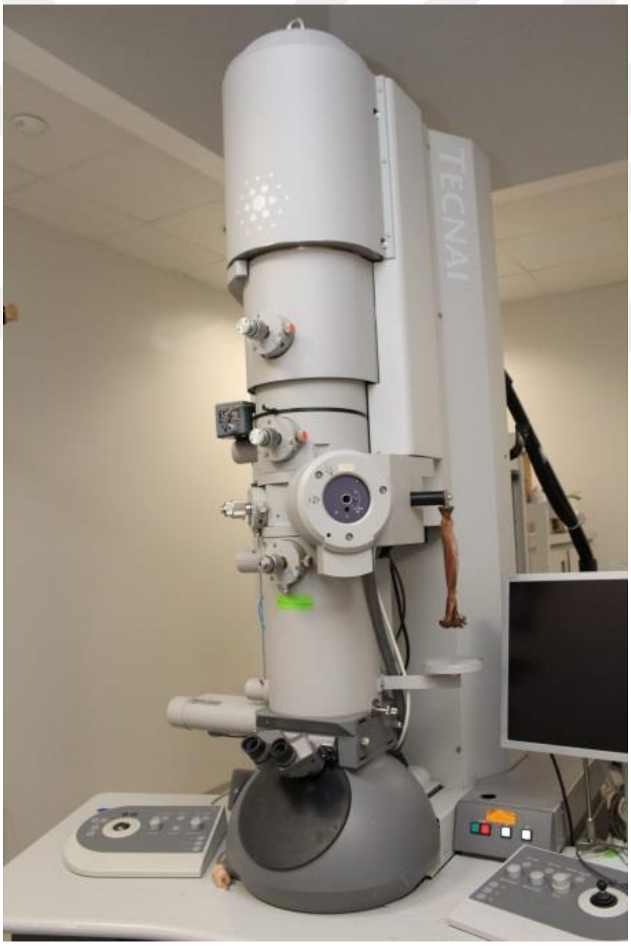


Figure 2.4 FEI Technai G2 Spirit Biotwin

In this study, the synthesized nanoparticles were examined using a FEI Technai G2 Spirit Biotwin TEM (Transmission Electron Microscope) operating at 120 kV (Figure 2.4). The bright field imaging technique was employed during the observation. The nanoparticles were suspended in ethanol and then droplets of the suspension were placed onto carbon-coated copper grids. The samples were left to dry at room conditions. The dried samples were subsequently examined under the microscope. The obtained results will be presented in the following chapter.

2.2.3 EDX Measurements

EDX (Energy-Dispersive X-ray Spectroscopy) is a characterization technique used for examining the elemental composition of materials. It is based on the measurement of the energies of characteristic X-rays emitted by the materials when exposed to high-energy electrons. The technique is applied by a system attached to electron microscopes such as SEM (Scanning Electron Microscope) and TEM (Transmission Electron Microscope).

Exactly, electrons can interact with the atoms of a material in various ways. Some of these interactions with the atoms result in the emission of X-rays at specific energies. The energy of the X-rays is defined by the element of the interacting atom. These types of X-rays are called characteristic X-rays. By measuring the energy and intensity of the emitted X-rays, information about the elemental composition of the material can be obtained (Figure 2.5).

In this study, the synthesized materials were characterized using an EDAX brand EDX detector on a FEI Quanta400F SEM instrument. The measurements were performed using electron energy of 30 kV. Aluminum stubs and conductive carbon tape were used to prepare the samples for measurement. The dry powder was dispersed onto the carbon tape to make it suitable for measurement. The obtained results are presented in the next chapter.

Figure 2.5 FEI Quanta400F SEM and EDAX EDX

2.2.4 XPS Measurements

XPS (X-ray Photoelectron Spectroscopy) is a characterization technique used for examining the elemental composition of materials and their chemical states. This technique is based on the measurement of the kinetic energies of photoelectrons emitted when X-ray photons interact with the material. The kinetic energy of the emitted photoelectrons is defined by the energy of the X-ray photon used and the binding energy of the electron to the atom. Since the energy of the X-ray used is known, the binding energies can be determined by measuring the kinetic energies of the photoelectrons. Each element has its own known binding energies, although these values may vary slightly depending on the chemical state (oxidation state) of the element. These variations are referred as chemical shifts.

Figure 2.6 PHI Versaprobe 5000 XPS

In this study, the samples were examined using the PHI Versaprobe 5000 XPS (Figure 2.6) instrument. Survey scans were conducted in the range of 100 eV to 1200 eV. Regional scans were performed in the Fe2p, Fe3p, and O1s regions to understand the chemical shifts. The dry powder was dispersed onto double-sided copper tape to make it suitable for measurement. The results obtained from these measurements will be presented in the next chapter.

2.2.5 Vibrating Sample Magnetometer Measurements

VSM (Vibrating Sample Magnetometer) technique is used to characterize the magnetic properties of materials such as saturation magnetization, remnant magnetization, and coercivity. This technique is based on measuring the induced electric field generated by magnetizing and vibrating the material under an externally applied magnetic field [65, 66]. The applied magnetic field aligns the magnetic moments within the material, leading to magnetization. The vibration of magnetized material, in turn, causes a change in magnetic flux in the vicinity of the material which leads to a voltage generation within the pick-up coil of the magnetometer. The time-dependent induced voltage is given as follows:

$$V_{coil} = \frac{d\Phi}{dt} = \frac{d\Phi}{dz}\frac{dz}{dt} \qquad [13]$$

Here Φ denotes for the magnetic flux enclosed by the pickup coil, z is the vertical position of the sample with respect to the coil, and t is time. For a sinusoidally oscillating sample position, the voltage is given as:

$$V_{coil} = 2\pi f C m A \sin(2\pi \mathrm{ft}) \qquad [14]$$

where C is a coupling constant, m is the DC magnetic moment of the sample, A is the amplitude of oscillation, and f is the frequency of oscillation. Thus, magnetic moment of a sample can be measured by measuring the voltage response from the detection coil. By using a gradiometer pickup coil configuration, a relatively large oscillation amplitude (1–3 mm peak) and a frequency of 40 Hz, the system is able to resolve magnetization changes of less than 10^{-6} emu at a data rate of 1 Hz.

In this study, VSM measurements were performed using the Quantum Design Physical Property Measurement System (PPMS) (Figure 2.7) instrument located in the SNTG laboratory at Hacettepe University. The magnetic field dependent magnetization measurements (M-H) were conducted at temperatures of both 5K and 300K within the applied field range of ±3 T. The temperature dependent magnetization measurements (M-T) were performed within 10 – 300 K in the presence of 500 Oe applied field. The Zero Field Cooling - Field Cooling (ZFC-FC) protocol is used in the measurements [ref vsm book paper]. The magnetic characterizations will be presented in the next chapter.

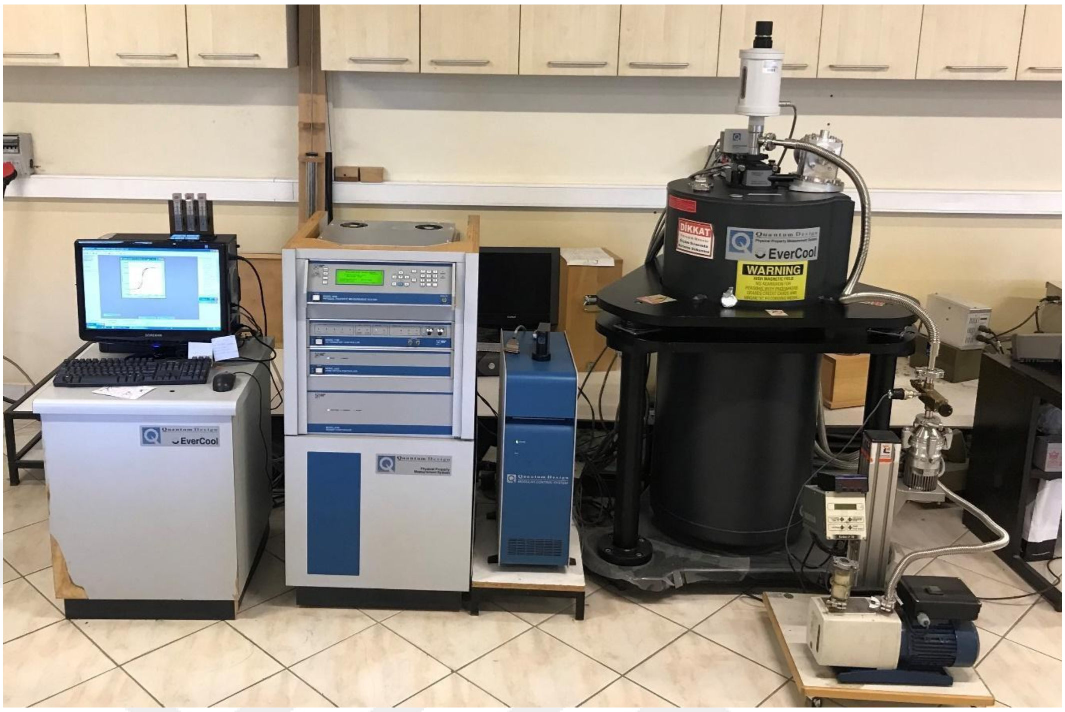


Figure 2.7 Quantum Design PPMS

2.3 Hyperthermia Setup

In this study, the aim was to measure the heating behavior of magnetic nanoparticles under alternative magnetic fields with different amplitudes and frequencies. For this purpose, a hyperthermia system was tried to be set up in the laboratory.

The hyperthermia system consists of two main components. The first component is the alternative magnetic field generator, which provides heat to the material. The second component is the system that measures the temperature change resulting from the heat.

2.3.1 Alternating Magnetic Field Generator

The alternative magnetic field generator is the energy source in hyperthermia systems that transfers energy to magnetic nanoparticles and enables their heating. It consists of an alternating current source and a coil. When a current is passed through the coil in a specific direction, a magnetic field of a certain magnitude, proportional to the magnitude of the current, is generated inside the coil. When the current changes direction, the generated magnetic field also changes direction (Figure 2.8) with a certain delay.

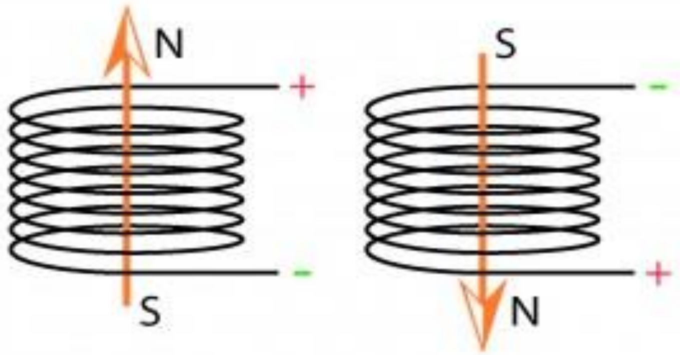


Figure 2.8 Schematic representation of induced magnetic field in a current flowing coil

It is possible to generate a continuously changing magnetic field using a coil connected to an alternating current source. Structures that possess this function are called alternative magnetic field generators or induction heaters.

In the laboratory, three different alternative magnetic field generators were developed using different electronic approaches. However, only one of these systems was able to be used for systematic measurements. The details of the developed systems are provided below.

2.3.2 Design of Electronic Circuit

Alternating current sources are extremely crucial to generate alternating magnetic fields with a desired frequency and amplitude. Alternating current sources are electronic circuits that can convert direct current into an alternating current with a defined frequency. There are three types of designs can be found as alternating current sources in literature.

2.3.2.1 Tank Circuits

Most of the alternating magnetic field generators are based on naturally oscillating tank circuits. Tank circuits are simply made of parallel connected inductors and capacitors. A simple schematic of a tank circuit can be seen in the Figure 2.9.

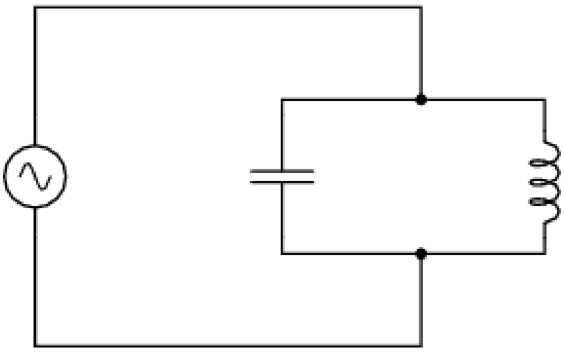


Figure 2.9 Simple parallel resonant circuit (tank circuit)

In the current study slightly modified tank circuit is used as an alternating magnetic field generator for preliminary studies. Schematic representation of the circuit can be seen in the Figure 2.10. The driving part of the circuit is added to the circuit to increase the amplitude of the magnetic field oscillations. The MOSFETs and the diodes are chosen to allow high currents to flow through them and the resistors are capable to deal with high powers.

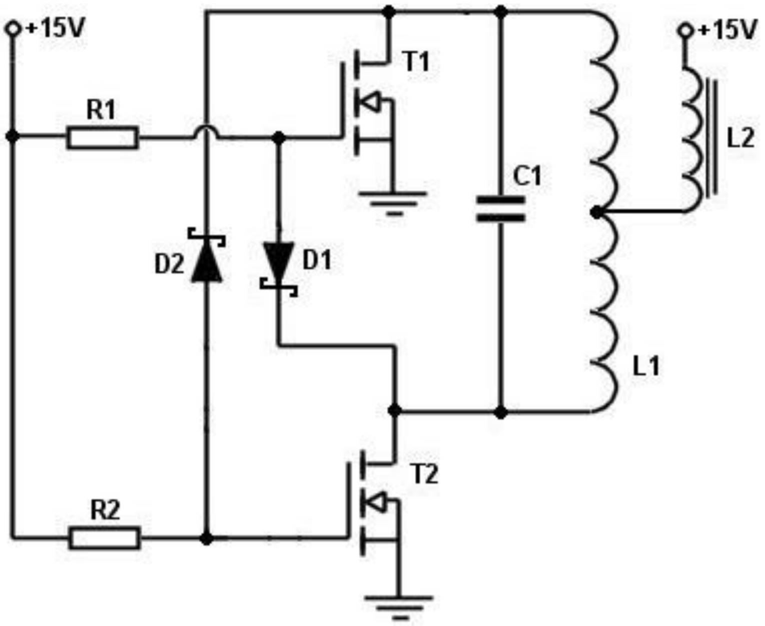


Figure 2.10 Circuit diagram of the induction heater

In the first attempts it was succeeded to generate alternating magnetic fields with a frequency of 130 kHz with high magnetic field amlitudes (20 kA/m). On the other hand, due to the nature of electronic design, both the frequency and the magnetic field intensity were not variable. Further trials are continued with magnetic nanoparticles. Approximately 10ºC of temperature increase is observed when the generated alternating magnetic field is applied to extremely high concentrations of MNP suspensions. These results are found to be not conclusive due to the high concentration of the suspension.

2.3.2.2 Half-Bridge Inverter

Half bridge inverters are electronic devices that can alter the current direction through a load by utilizing 2 switches and 2 capacitors as shown in Figure 2.11. This kind of topology is mostly used for electric motor drivers to control the speed of the electric motor. The circuits designed with this topology require an external function generator. The frequency of the function generator controls the frequency of the altering current. Different from the full bridge inverters, the capacitor values are limiting factors for the working frequencies which decrease the flexibility on the alternating magnetic field frequency. On the other hand, vulnerability to timing of signals for switching can be reduced with this design.

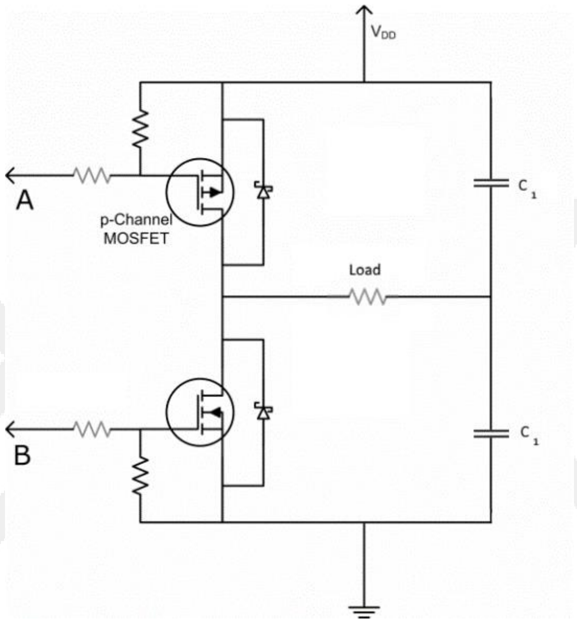


Figure 2.11 Circuit diagram of a Half-Bridge inverter

In this study, a half bridge inverter is implemented and used for the first variable frequency experiments. Utilizing the implemented circuit working at different frequencies, different heating regimes are recorded and will be shown in the next section.

2.3.2.3 Full-Bridge Inverter

Full bridge inverters or H-Bridge inverters are electronic devices that can alter the current direction through a load by utilizing 4 switches as shown in Figure 2.12. This kind of topology is mostly used for electric motor drivers to control the speed of the electric motor. The circuits designed with this topology require an external function generator. The frequency of the function generator controls the frequency of the altering current which brings a total flexibility on the alternating magnetic field frequency. On the other hand, timing of signals for switching becomes extremely important. If any timing mismatch occurs during switching, leak current can pass directly on switches, namely MOSFET's, causing overheat and irreversible damage (Figure 2.13).

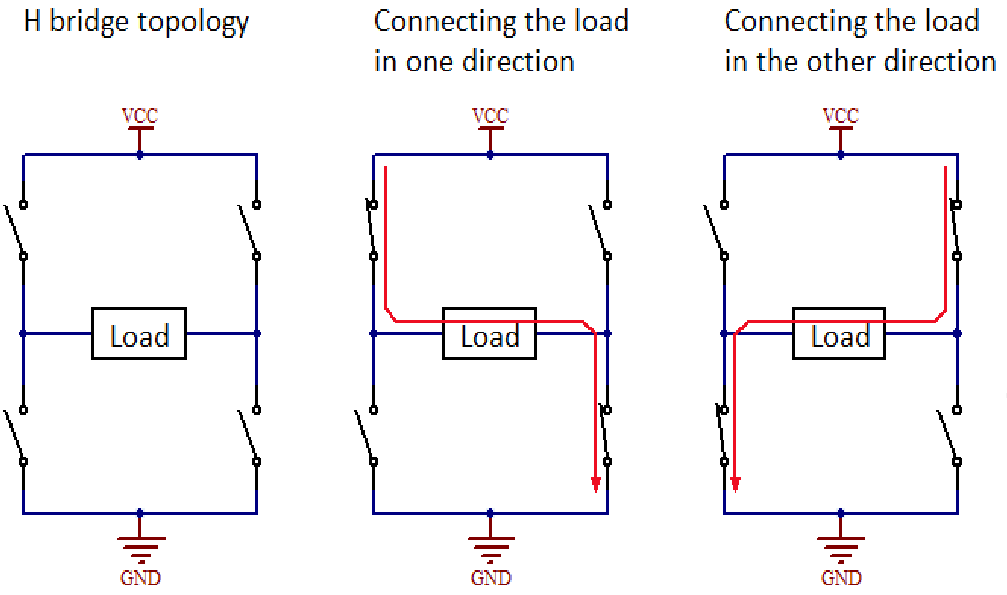

H bridge topology
Connecting the load in one direction
Connecting the load in the other direction
VCC
VCC
VCC
Load
Load
Load
GND
GND
GND

Figure 2.12 Schematic view of Full-Bridge inverter

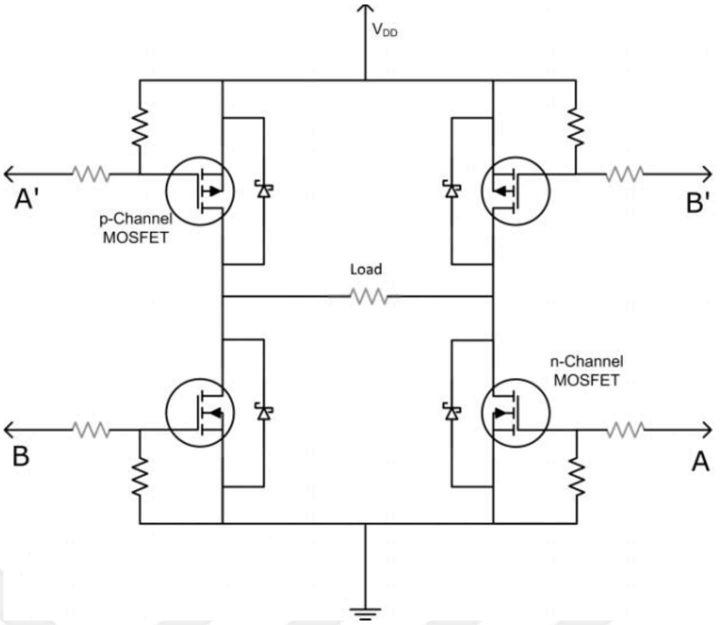


Figure 2.13 Circuit diagram of a Full-Bridge inverter

In this study, a full bridge inverter is implemented and used for most of the measurements. The results will be presented in the next section.

2.3.3 Temperature Measurement

Temperature changes during the experiments are measured with two different systems. Alcohol thermometer measurements and thermal camera measurements will be described in this section.

Other temperature measurement techniques (mercury thermometer, thermocouples) are not preferred due to their conductive parts and inability to measure accurately under intensive RF conditions.

2.3.3.1 Alcohol Thermometer

An alcohol thermometer is immersed directly into the sample suspensions in the measurement cell as shown in the Figure 2.14. Temperature readings are recorded against time and heating curves of the experiment is obtained with this data.

Although this technique seems simple and robust, it is difficult to make comparable and reproducible measurements due to its low precision and difficulty in integrating with automated reading systems.

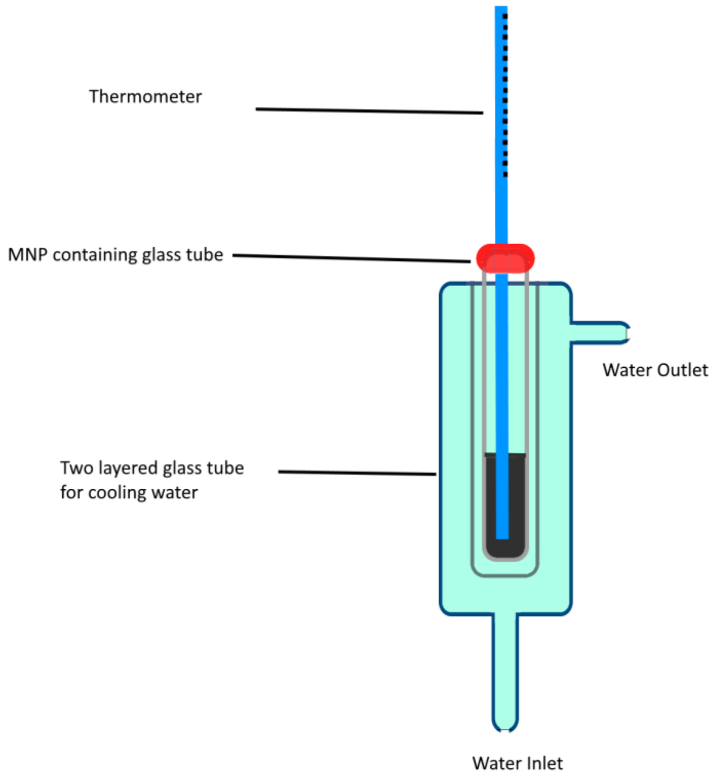


Figure 2.14 Measurement cell with alcohol thermometer

2.3.3.2 Thermal Camera

For thermal camera measurements a Testo 885 model thermal camera was utilized. The camera was positioned in line and facing to the sample cell as shown in Figure 2.15. A thermal video is recorded during the experiments by a computer software (IRSoft) provided by Testo. After the experiments, video recordings were analyzed and heating curve data was extracted by a computer software developed and implemented for this study.

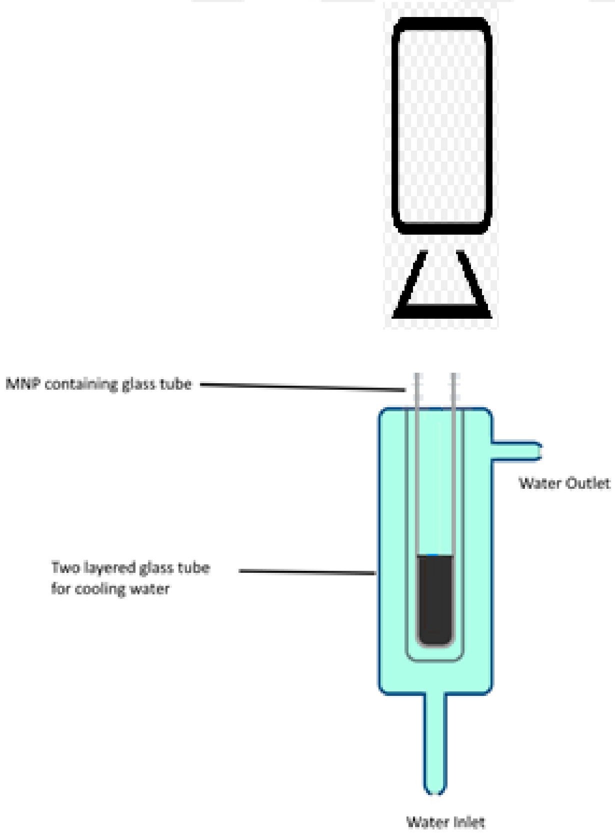


Figure 2.15 Measurement cell with thermal camera

CHAPTER 3

EXPERIMENTAL RESULTS

In this chapter, the results obtained from the applied characterization techniques have been presented to understand the material properties of the synthesized particles. The similarities and differences found in the results have been analyzed. Subsequently, the results obtained from magnetic hyperthermia experiments conducted under various conditions on the materials have been presented and examined.

3.1 Chemical, Crystallographic and Morphologic Characterization Results

Various characterization techniques have been applied in order to understand if the synthesized materials are the intended materials and to determine the levels of impurities that could affect other characterization results. The results of these characterization techniques have been presented in this chapter.

3.1.1 EDX Results

The synthesized samples were first subjected to EDX measurements. EDX analyses were conducted @30 keV electron energy to ensure the detection of impurities. The obtained EDX spectra are presented in Figure 3.1. Upon examination of the spectra, the expected peaks of Oxygen and Iron are clearly observed. The low-intensity peaks observed apart from the Iron and Oxygen peaks correspond to Sodium, Silicon, and Chlorine, respectively.

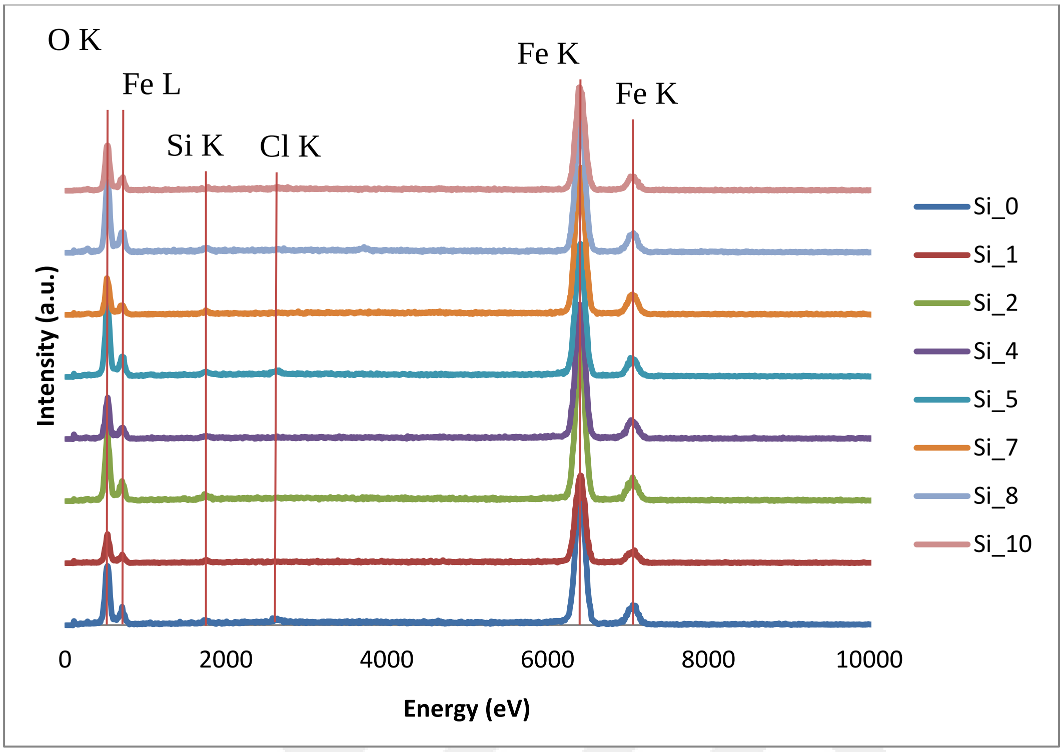


Figure 3.1 Graph of EDX spectrums

The quantification results obtained from these spectra are provided in Table 3.1. According to these results, Iron and Oxygen exhibit values close to the expected stoichiometry. On the other hand, the presence of Sodium and Chlorine in the spectrum is known to be attributed to a secondary product (NaCl) formed during the reaction and not completely removed during the washing process. The observed Silicon is believed to originate from the glassware used during the synthesis. Due to the low levels of these impurities, they are not expected to have a significant impact on other measurement results. Based on the EDX results, it can be concluded that the synthesized materials consist of iron oxide-based (magnetite, maghemite, hematite) particles.

Table 3.1 *Table of EDX quantification results*

	O (at%)	**Fe** (at%)	**Si** (at%)	**Cl** (at%)
Si_0	56.04	42.11	1.09	0.76
Si_1	54.73	43.13	1.61	0.53
Si_2	55.42	42.21	1.99	0.38
Si_4	53.89	44.14	1.64	0.33
Si_5	57.73	39.84	1.54	0.89
Si_7	55.46	42.62	1.62	0.30
Si_8	57.18	41.03	1.44	0.35
Si_10	56.79	42.20	0.76	0.25

3.1.2 XPS Results

XPS (X-ray Photoelectron Spectroscopy) measurements were performed on three selected samples to gain a deeper insight into their structures and determine whether they are magnetite or maghemite. Based on the VSM (Vibrating Sample Magnetometer) (results are presented in the following text) measurement results, the samples with the highest, lowest, and average magnetic saturation values were chosen.

The results obtained from XPS measurements on these samples have been used for two different purposes.

Firstly, XPS data was examined in terms of the formation of by-products during the synthesis process. Although the by-products (NaCl) formed during the synthesis process are not expected to have a direct impact on the magnetic properties of the material, it is known that they can affect the normalization of magnetic measurements based on weight[67-69]. According to the XPS and EDX results, the amount of impurities caused by the by-products was found to be negligible and not expected to significantly affect the other measurement results (Figure 3.1). Based on this finding, it can be concluded that the washing process applied to the materials after synthesis is effective enough.

Secondly, XPS data was used to investigate the quantities of iron ions with different valences (Figure 3.2). Although there were minor differences in the obtained data, a conclusive determination of whether the samples were pure magnetite or pure maghemite could not be made.

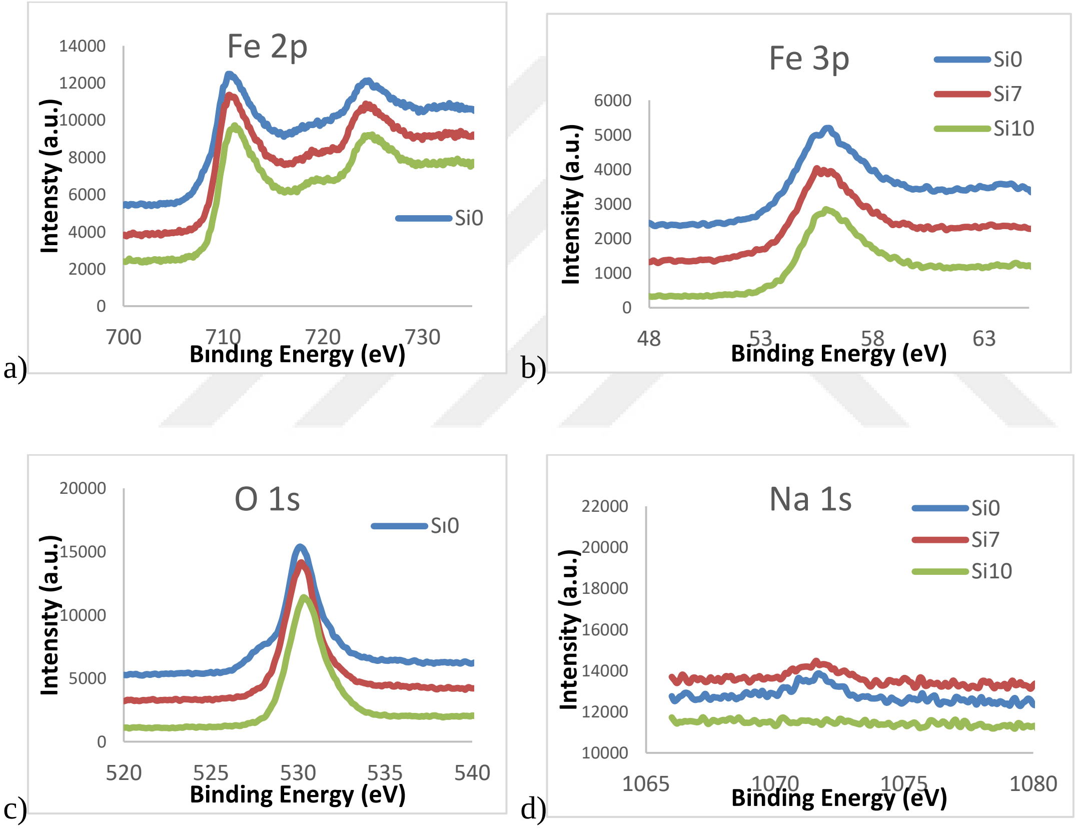


Figure 3.2 XPS data for Si0, Si7 and Si10 samples at a) Fe 2p, b) Fe 3p c) O 1s and d) Na 1s peaks

3.1.3 XRD Results

XRD measurements were also conducted on the synthesized materials to determine their crystal structures and identify the type of iron oxide they belong to. The XRD pattern associated with these measurements is shown in Figure 3.3. Upon examining these graphs, it was determined that the particles obtained from the syntheses are in complete agreement with magnetite and maghemite. The XRD pattern in Figure 3.3 also includes the reference information for magnetite (card JCPDS # 89-4319). The absence of any peaks that could be associated with hematite suggests that hematite did not form during the synthesis or formed in small quantities that it could not be detected.

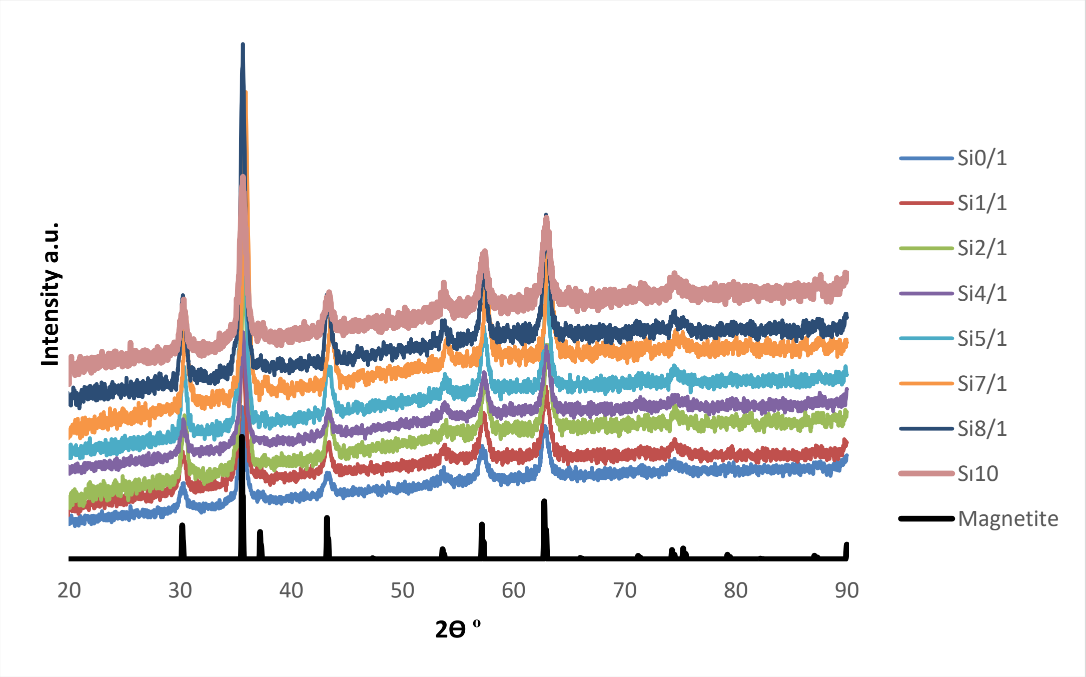


Figure 3.3 Plot of the XRD data obtained from synthesized samples and magnetite, reference

Based on the obtained XRD results, it has been concluded that these synthesized materials are either magnetite (Fe_3O_4) or maghemite (γ-Fe_2O_3). Due to the similarity in their crystal structures, it was not possible to definitively determine whether they are magnetite or maghemite using the XRD method (Figure 3.3). Crystallite sizes were calculated from the obtained XRD data using the Halder-Wagner method, and it was observed that they have different crystallite sizes (Table 3.2).

Table 3.2 *Obtained average crystallite sizes for samples*

Sample Name	**Halder Wagner Size (nm)**
Si_0	11.7 ± 1.3
Si_1	10.3 ± 0.8
Si_2	11.5 ± 0.7
Si_4	12.8 ± 1.4
Si_5	11.6 ± 0.7
Si_7	13.3 ± 1.5
Si_8	13.3 ± 1.5
Si_10	11.2 ± 0.9

High resolution XRD measurements have also been applied to some of the samples to be able to identify whether they are magnetite or maghemite. The measurements were performed with a step size of 0.01° and a scanning rate of 1°/15 min. Two distinct regions in the samples were examined ($56° < 2\theta < 58.5°$ for the (511) peak and $61° < 2\theta < 64°$ for the (440) peak), and the obtained results were deconvoluted using peak deconvolution methods (Figure 3.4) [70].

XRD data obtained in the range of 56° to 58.5° has been fitted with two Pseudo-Voigt profiles, one for magnetite (at 57.2°) and the other for maghemite (at 57.4°) peaks. The same procedure was applied to the data in the range of 61° to 64°, and the peak centers of 62.8° for magnetite and 63.0° for maghemite were used. These peak positions were selected to be consistent with the peak positions reported in the literature for magnetite and maghemite. The peak heights obtained from the samples are provided in Table 3.3.

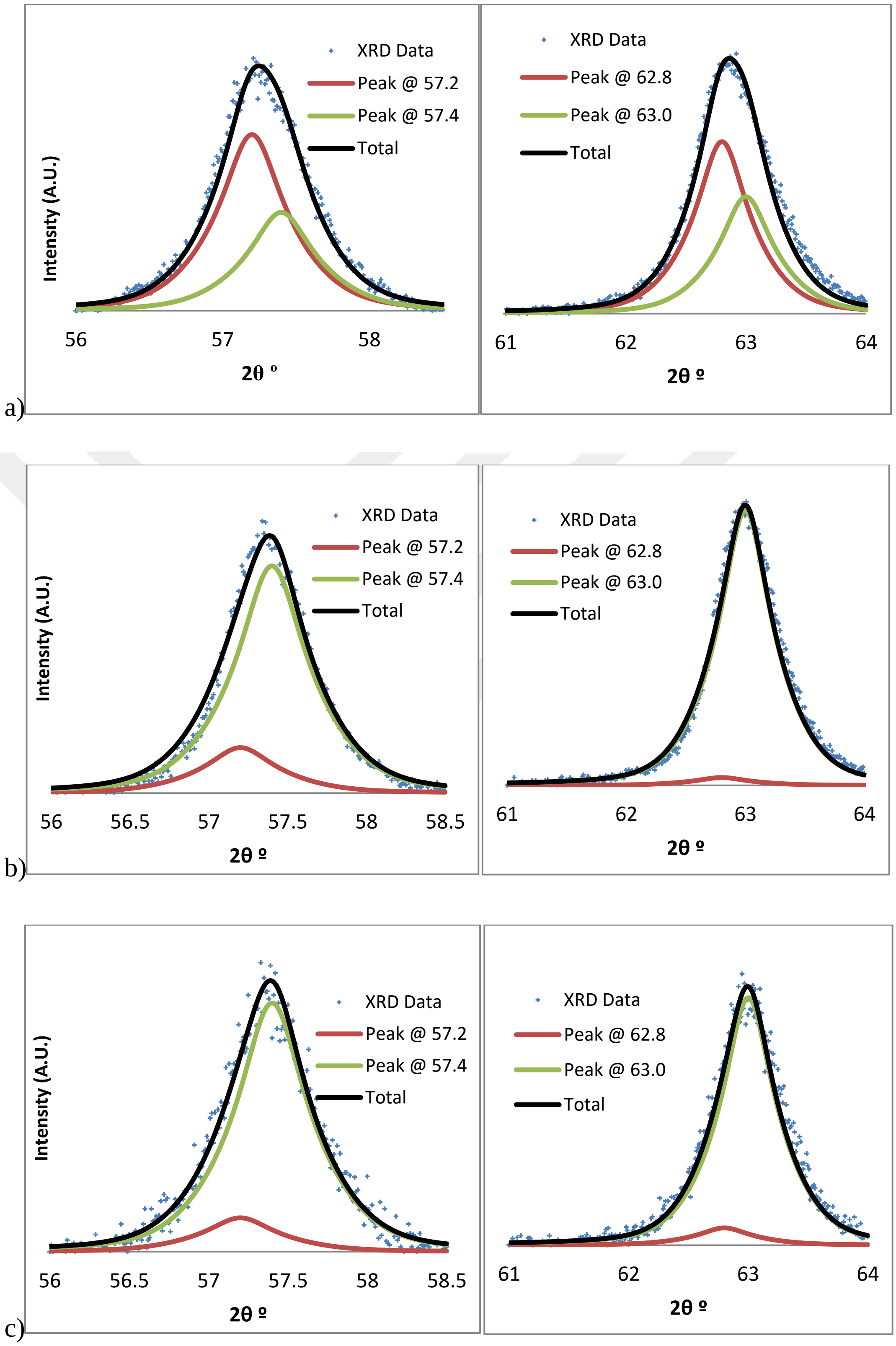


Figure 3.4 Graphs of high resolution XRD data deconvoluted peaks and regenerated peak of a) Si_0 b) Si_7 c) Si_10samples.

Table 3.3 *Normalized peak heights obtained from the peak deconvolution applied to XRD data*

	(511)		**(440)**	
	57.2º Magnetite %	**57.4º Maghemite %**	**62.8º Magnetite %**	**63º Maghemite %**
Si_0	64	36	60	40
Si_7	17	83	3	97
Si_10	12	88	6	94

Upon examining the graphs in Figure 3.4 and the values in Table 3.3, it is evident that there are significant differences among the synthesized materials. When considering both the (511) and (440) peaks, the higher peak intensity associated with magnetite in the Si_0 sample suggest a higher content of magnetite in this sample. On the other hand, ratios of the peak heights for the Si_7 and Si_10 samples indicate that these samples are predominantly composed of maghemite.

On the other hand, a correlation between the obtained information and the results from magnetic measurements of the materials could not be established. Despite the literature reporting that magnetite has higher magnetic properties compared to maghemite, the expected results could not be achieved in this study.

3.1.4 TEM Results

Throughout the study, all obtained samples were examined using TEM. The acquired images were utilized to determine the morphologies and particle size distributions of the samples (Figure 3.5). Upon examination of the images, it was found that a significant proportion of particles exhibited spherical morphologies. Therefore, for the remainder of the study, the particles were considered to be spherical. Additionally, particle size distributions were calculated from the images. The size of 100 – 120 particles from each sample was manually measured, and size histograms were generated (Figure 3.6).

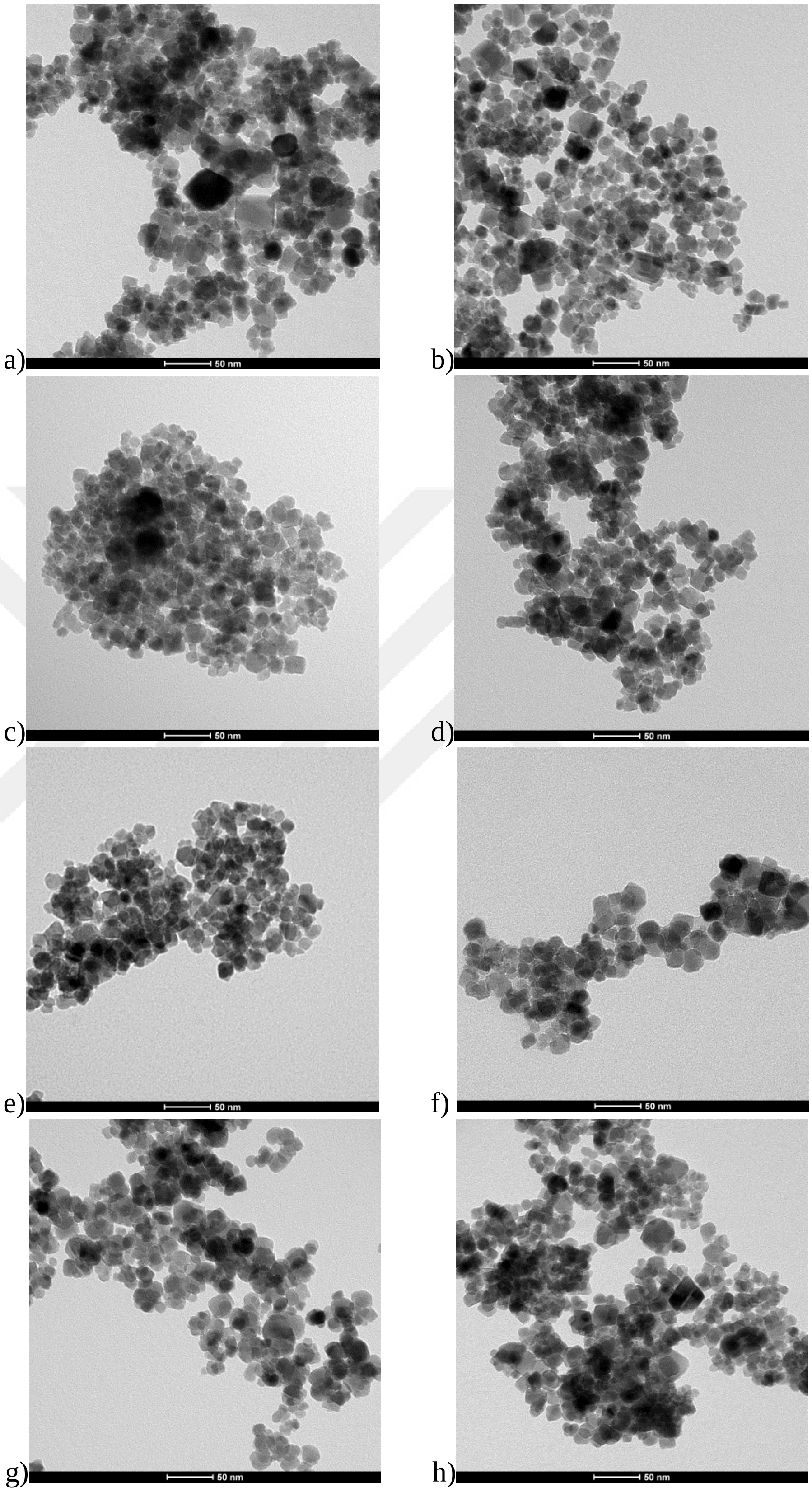


Figure 3.5 TEM images of samples a) Si0, b) Si1, c) Si2, d) Si4, e) Si5, f) Si7, g) Si8, h) Si10

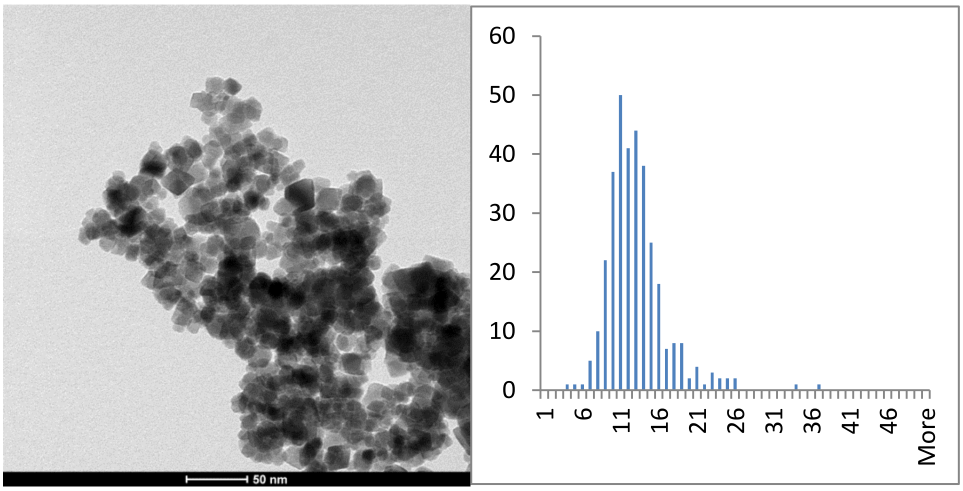


Figure 3.6 TEM image and the correspondinng size distribution histograms for the samples Si10

The obtained histograms were fitted to a log-normal distribution to calculate both the average particle size and the standard deviation. The results are presented in Table 3.4.

Table 3.4 *Measured particle diameters from TEM images*

	Average Particle Diameter (nm)	**Sigma (nm)**
Si0	12.53	3.5
Si1	11.47	2.8
Si2	12.81	3.2
Si4	13.59	3.3
Si5	12.35	2.7
Si7	16.04	3.8
Si8	15.45	3.1
Si10	11.95	2.3

3.1.5 Comparison of Chemical, Crystallographic and Morphologic Characterization Results

According to the characterization results applied to the samples produced using the same synthesis method with slight parameter variations; it has been observed that they are chemically and crystallographically similar materials.

According to the results obtained from the applied chemical characterization methods, it was concluded that the materials did not contain impurities at levels that would significantly affect other measurements. In the literature, it is mentioned that the distinction between maghemite and magnetite can be made based on the oxidation state of iron [69, 71]. However, the XPS measurements applied to these samples did not reveal a definitive difference. It is speculated that this may be due to the presence of both phases in different proportions within the materials.

The crystallographic investigations showed consistent results with magnetite and maghemite, and no additional phase has been observed. High-resolution XRD measurements have indicated that the peak geometries are more consistent with maghemite, but it has also been determined that magnetite is present [70]. This result is in line with the XPS findings.

When examining the TEM results, it can be observed that the particles mostly exhibit a spherical structure. Additionally, it is evident from the TEM images that each synthesized material has a different particle size distribution. It was observed that the particle sizes obtained from TEM are largely consistent with the crystallite sizes obtained from XRD. Considering these results, the assumption was adopted for the remainder of the study that volumetric calculations would employ a spherical morphology, and the diameter would be taken as equal to the crystallite size obtained from XRD.

3.2 Magnetic Characterization Results

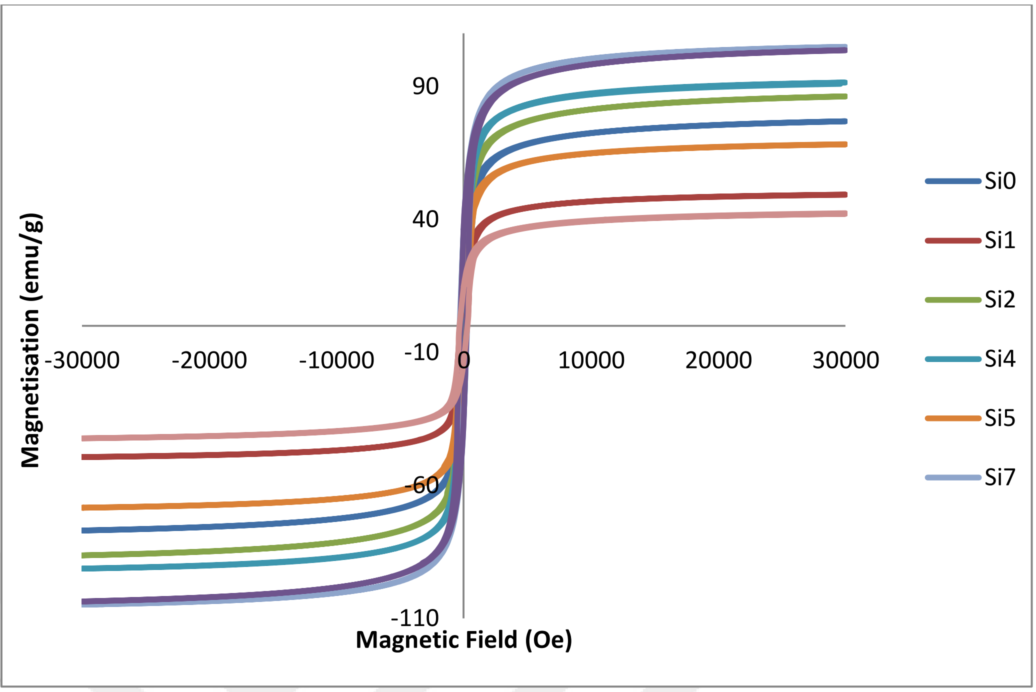
During this study, the magnetic properties of all synthesized materials were measured utilizing different characterization techniques. Results of the measurements are presented in the following text.

3.2.1 VSM Results

The VSM (Vibrating Sample Magnetometer) measurements were performed on the obtained materials to understand their magnetic saturation values and reveal their magnetic behaviors. Despite having similar chemical and crystallographic properties, significant differences in the measured values were observed among these materials.

The VSM measurements of the materials were conducted at both 5 K and 300 K temperatures. As expected from the particle sizes obtained from other measurements, all samples exhibit superparamagnetic properties at room temperature. The measurements conducted at 5 ºK shows the expected ferromagnetic behavior of the materials, as can be seen on the hysteresis curves (Figure 3.7). On the other hand, measurements taken at 300 ºK indicate the transition of the materials into the superparamagnetic regime, where the materials exhibit zero coercivity. This transition is likely due to the increased thermal energy at higher temperatures, which causes the magnetic moments to become more randomized and lose their long-range order. (Figure 3.8).

a)



b)

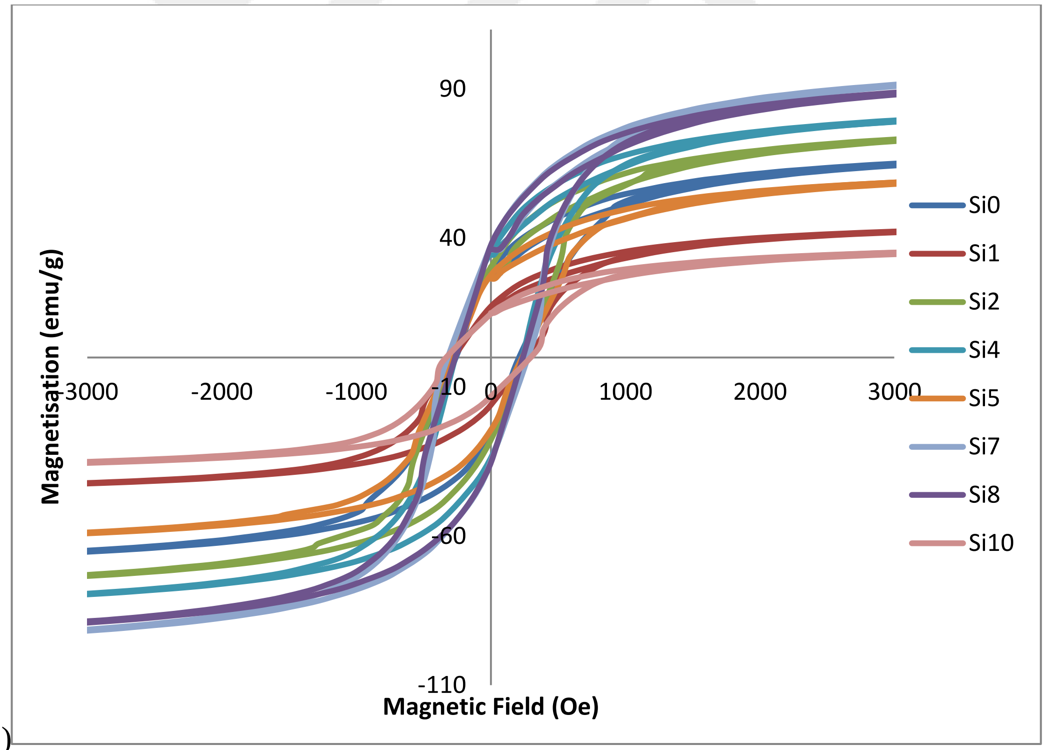
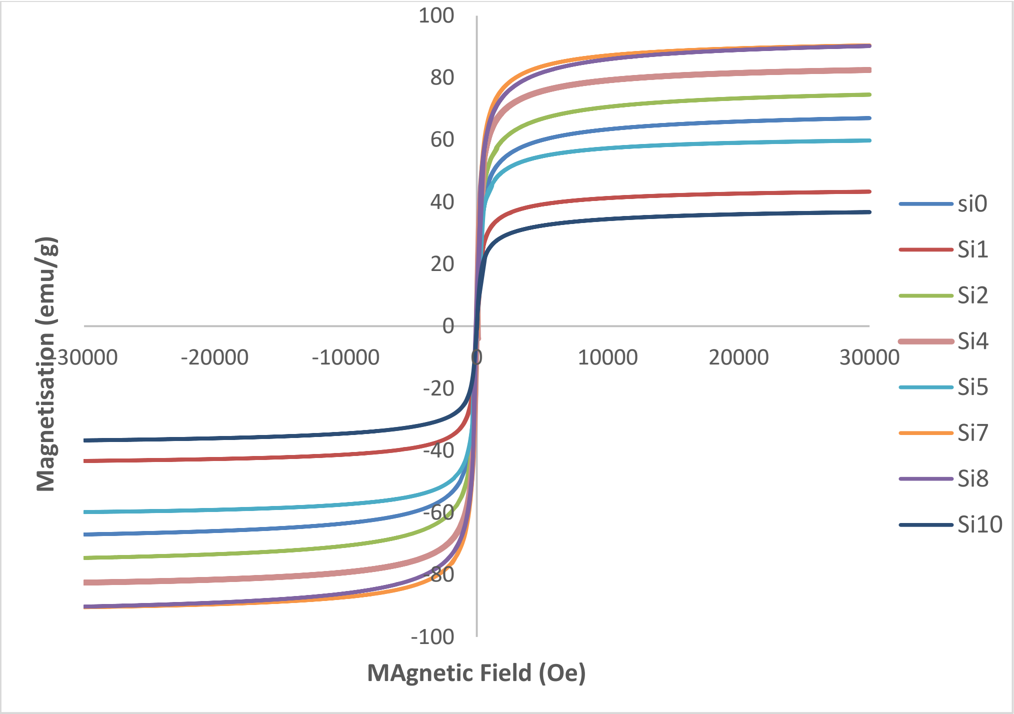


Figure 3.7 VSM measurements @ 5 K a) full range of magnetic field b) magnetic field from -3000 Oe to 3000 Oe


a)

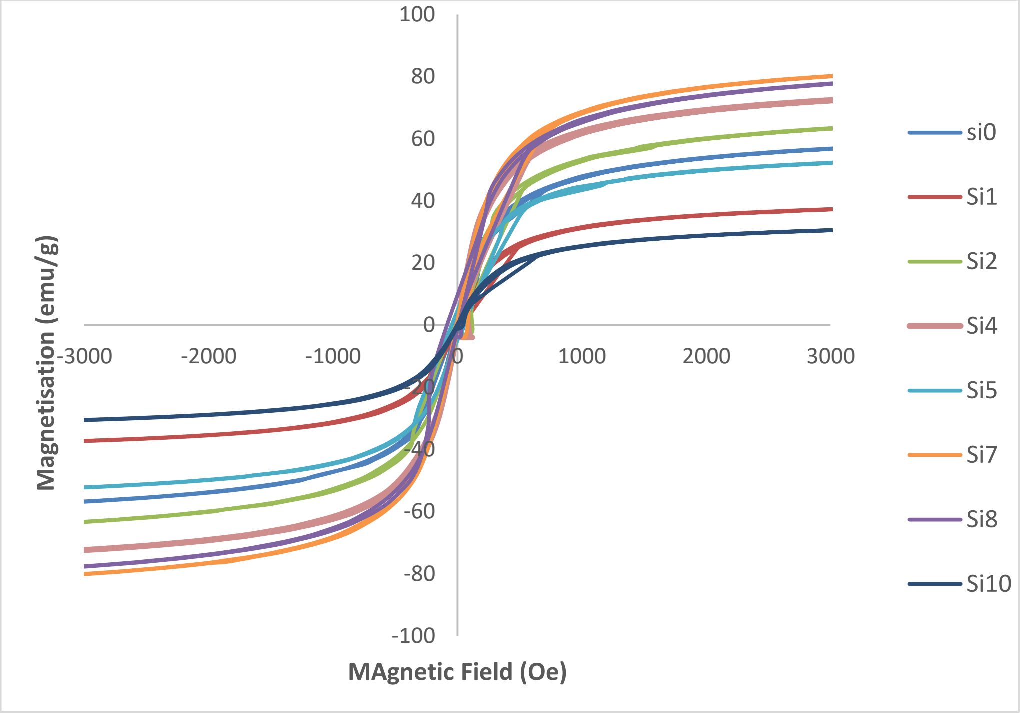

b)

Figure 3.8 VSM measurements @ 300 K a) full range of magnetic field b) magnetic field from -3000 Oe to 3000 Oe

Although the results obtained from other measurements indicate similarities between the materials, the VSM results have shown that these materials are different from each other.

The magnetic saturation values obtained from VSM results are in agreement with the reported saturation values in the literature for magnetite and maghemite. It can be observed that for these materials, the lowest reported saturation value is around 35emu/g, while the highest value is around 90 emu/g. In this study, the synthesized materials completely cover the range of magnetic saturation values reported in the literature when considering their magnetic saturation properties.

3.2.2 ZFC- FC Measurement Results

ZFC-FC (Zero Field Cooled - Field Cooled) measurement is a measurement technique that determines the transition temperature between the ferromagnetic and superparamagnetic regimes. It provides information about the relationship between thermodynamic effects and magnetic anisotropy energy.

By using this technique, it is possible to calculate the temperature at which the thermodynamic fluctuations overcome the magnetic anisotropy energy barrier in the material. This temperature is referred as the blocking temperature. Consequently, the anisotropy energy barrier and anisotropy energy constant can be determined. The anisotropy constant can be calculated using the following, widely accepted, equation.

$$Anisotropic\ Energy\ Barrier = K_{eff}V = 25k_bT_b \qquad [15]$$

Where, V is the volume of the particle k_b Boltzman constant T_b is the blocking temperature and K_{eff} is the anisotropic energy constant. Based on the obtained results (Figure 3.9) and the performed calculations, the anisotropy energy barriers and anisotropy constants of the synthesized materials in the study are presented in Table 3.5.

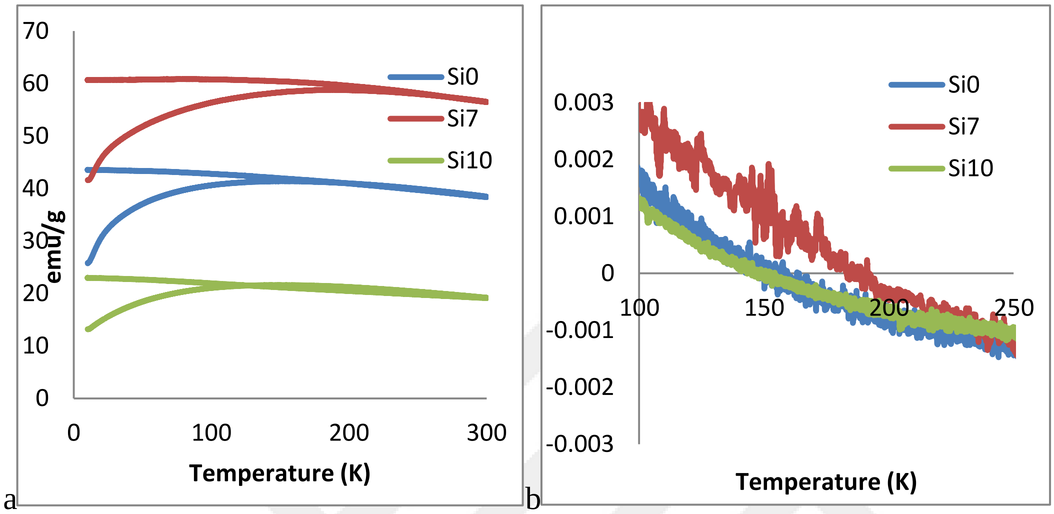


Figure 3.9 Results of a) ZFC – FC measurements for Si0 Si7 and Si10 samples and the numerical derivatives of the data

When examining the graphs in Figure 3.9.a, the peak point in the FC region corresponds to the temperature at which the regime transitions occur. Due to the broadness of the peaks, it can be challenging to determine the exact point from these graphs. Instead, applying a numerical derivative to the data provides a more accurate result. The point at which the slope of the graph changes from positive to negative, in other words the derivative crosses the zero line, can be measured as the blocking temperature.

Table 3.5 *Summary of obtained characterization results and calculated values*

	Saturation Magnetization		Tb (K)	Anisotropic Energy (kJ)	Crystallite size (nm)	Anisotropy constant(kJ/m^3)
	emu/g	emu/cm^3				
Si0	67.0	346	152	5.25E-20	11.7	62.56
Si1	43.3	224	172	5.94E-20	10.3	103.76
Si2	74.6	385	150	5.18E-20	11.5	65.02
Si4	82.5	427	170	5.87E-20	12.8	53.44
Si5	59.8	309	201	6.94E-20	11.6	84.89
si7	90.4	467	189	6.52E-20	13.3	52.96
Si8	90.2	466	164	5.66E-20	13.3	45.95
si10	36.8	190	148	5.11E-20	11.2	69.44

Summary of obtained characterization results and calculated values are displayed in Table 3.5. The magnetic saturation values in this table are obtained from VSM measurements, the T_b values are obtained from ZFC-FC measurements, and the crystallite sizes are obtained from XRD measurements. The K_{eff} value has been calculated using Equation 1. To be able to calculate the volume of the particles they are assumed to be spherical.

The magnetic measurements have shown significant differences among the materials. Some materials exhibit magnetic saturation values that are three times higher than others. Additionally, the blocking temperature values also show significant variations.

In the analyses conducted in this study, there is no conclusive evidence that can definitively explain these differences. However, differences in particle sizes, the presence of amorphous iron oxide/hydroxide forms that cannot be detected by XRD, and defects in the crystal structure are considered as possible reasons for these differences.

3.3 Implementation of Custom Hyperthermia System

The experimental setup used in this study consists of 5 main components. These components are the computer-controlled frequency generator, the coil driver, and the measurement cell. During the study, these components were developed step by step to make them functional and compatible with each other. In this section, the details of these components are described.

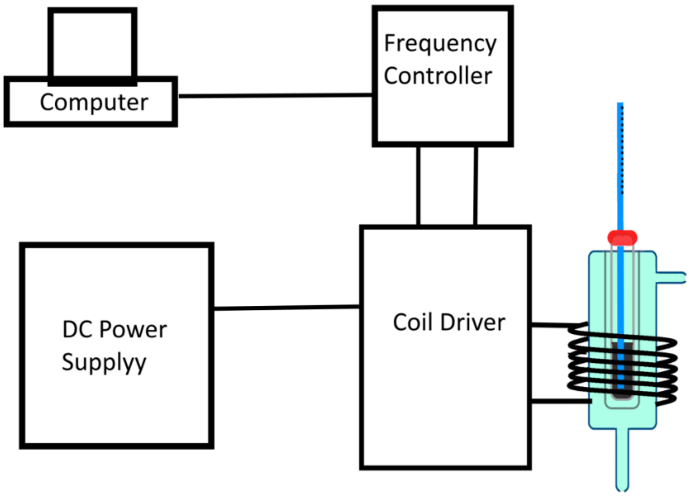


Figure 3.10 Schematic representation of experimental setup

3.3.1 Computer-Controlled Frequency Generator

The computer-controlled frequency generator consists of a computer and a programmable microprocessor card. A software (developed during the study), running on the computer, sends the necessary parameters to the microprocessor via USB to generate the desired frequency. The codes of this software are provided in Appendix C.

In this study, the STM32F401CCU6 microcontroller was used as the programmable processor. To enable the microcontroller to process the commands dictated by the computer and generate PWM (Pulse Width Modulation) signals accordingly, a software was developed and loaded onto the microcontroller during this study (Figure 3.11). The codes developed for this purpose are provided in Appendix D.

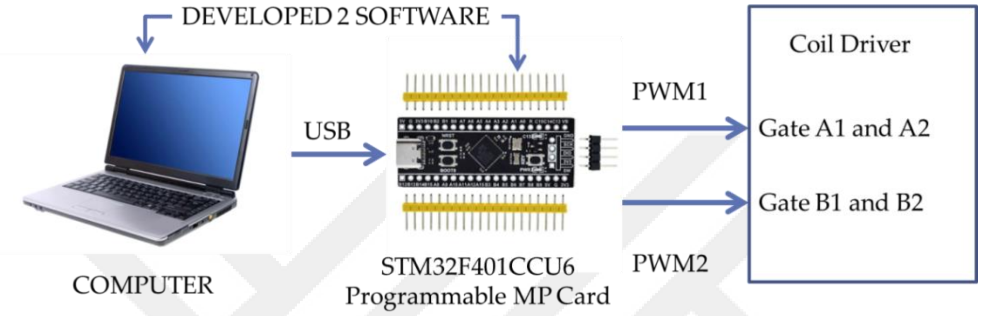


Figure 3.11 Schematic representation of frequency generator

The precise control of PWM signals is crucial for the proper operation of Full Bridge inverters. PWM signals are square wave signals where the duty cycle, which represents the ratio of on-time to the period, can be varied (Figure 3.12). While PWM is commonly used for power control or adjusting voltage levels in many systems, in this study, it is utilized directly to control the frequency.

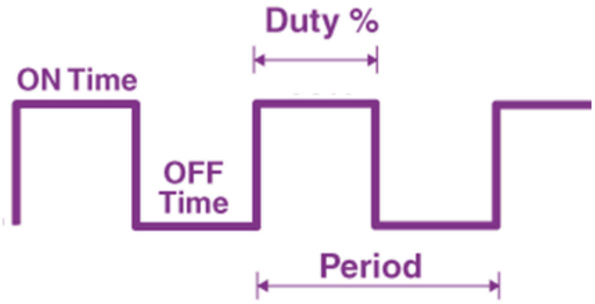


Figure 3.12 Schematic representation of a PWM pulse

In a Full Bridge inverter, the gates conduct current during the On time and block current during the Off time. There are four gates in a Full Bridge inverter. In the first half of a period, two of these gates (A1 and A2 in Figure 3.13) should be open, while the other two gates (B1 and B2 in Figure 3.13) should be closed. In the second half of the period, the situation should be reversed. This way, an alternating current is generated on the load. To control the gates in this manner, two complementary PWM signals are required.

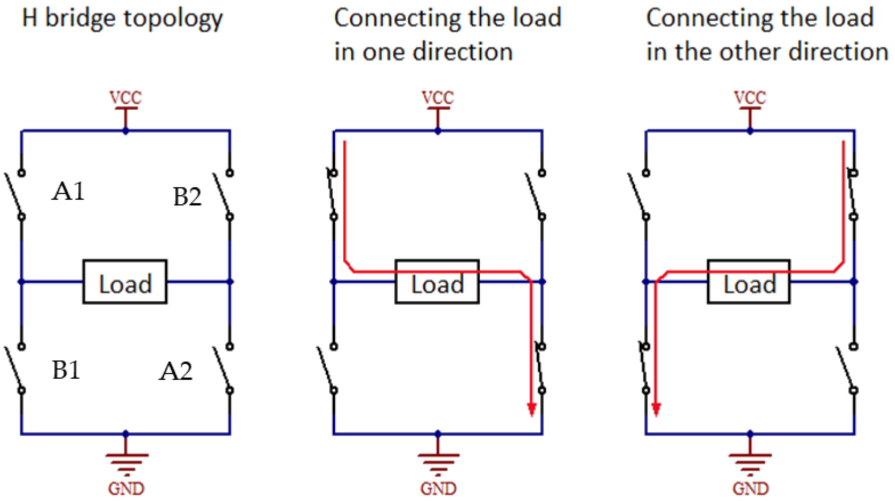

H bridge topology
Connecting the load in one direction
Connecting the load in the other direction
VCC
A1
B2
Load
B1
A2
GND

Figure 3.13 Full Bridge inverter topology

On the other hand, having PWM signals exactly opposite to each other poses another problem. There is a certain time required for the gates to fully open and close (for the MOSFETs used in this study, the turn-on time is 100 ns and the turn-off time is 145 ns). If the PWM signals change exactly at the same time, one gate may start to open before the other fully closes, resulting in current flowing through the gates instead of the load. This can potentially damage the system and should be avoided. In PWM technique, a time interval can be defined between the switching of the two signals where both signals are off, and this time interval is known as dead time (Figure 3.14). In this study, the dead time is set to 5% of the period, ensuring the stable operation of the system.

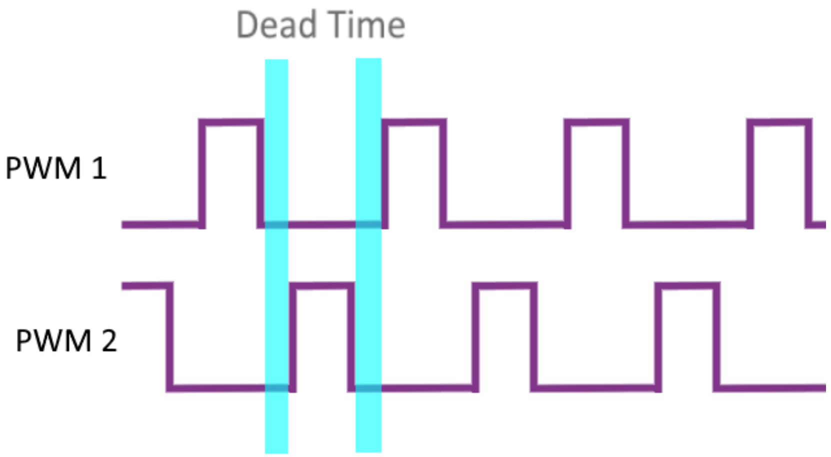


Figure 3.14 Schematic representation of “dead time” for two PWM pulses

3.3.2 Coil Driver

The coil driver is a printed circuit board implemented during the study to generate alternating current at the desired frequency. The full bridge inverter topology was used in the design of the printed circuit board. For the circuit setup, 4 IRF540N MOSFETs were used as gates. These MOSFETs are capable of handling a maximum current of 33A, with a turn-on time of 100 ns and a turn-off time of 145 ns. To drive the MOSFETs, 2 IR2110 MOSFET drivers, each capable of driving 2 MOSFETs, were used. The MOSFET drivers control the MOSFETs according to the signals provided by the frequency generator.

The required DC power for the circuit is supplied by a 12V 60W power supply. The circuit diagram, the other components used, and the printed circuit board layout can be seen in Figure 3.15 and 3.16.

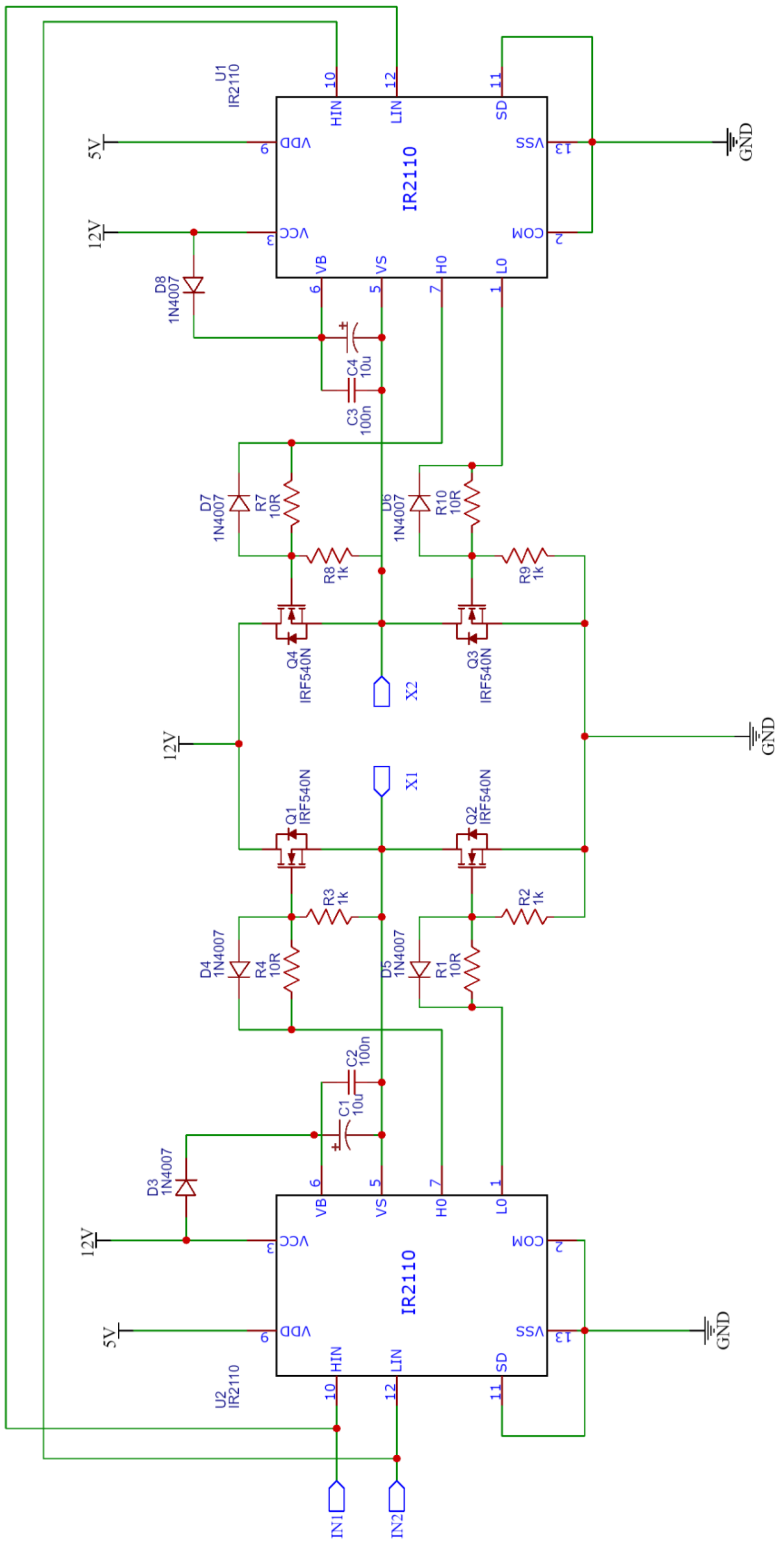


Figure 3.15 Circuit diagram of the coil driver

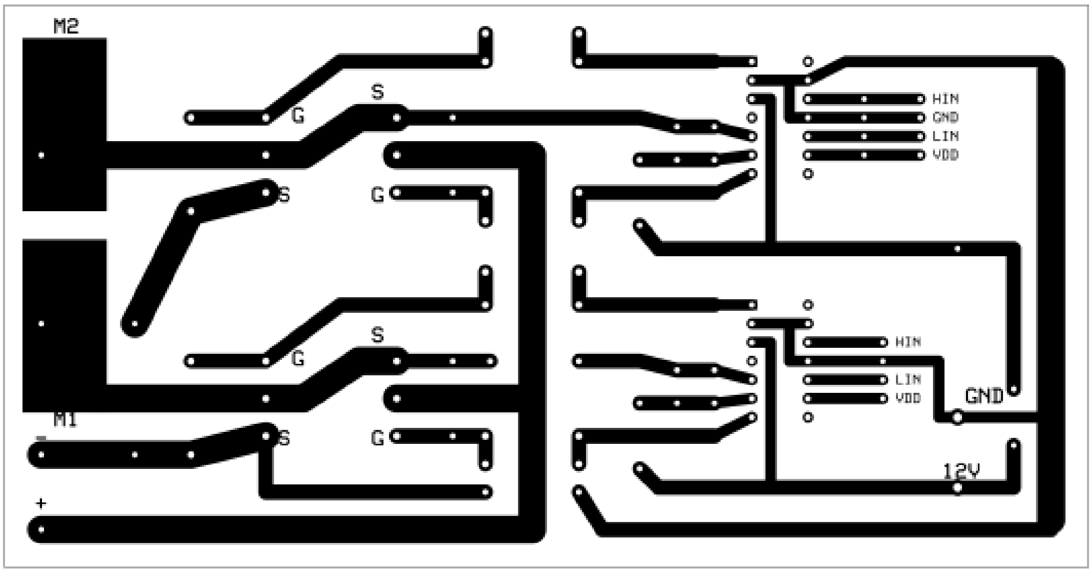


Figure 3.16 PCB layout of the coil driver

3.3.3 Measurement Cell

The Measurement cell is a structure composed of the coil where the magnetic field is generated, a thermal isolation layer, and the sample cavity. The coil driver delivers current through the coil, generating alternating magnetic field at the desired frequency. The insulation layer prevents the heat generated on the coil from being transferred to the sample, thus preventing nonspecific heating of the sample. The sample cavity is where the samples to be measured are placed. The thermal camera, which is the last component of the system, measures temperature changes. It captures videos throughout the experiment, showing the temperature changes over time. The schematic of the system components can be seen in Figure 3.17.

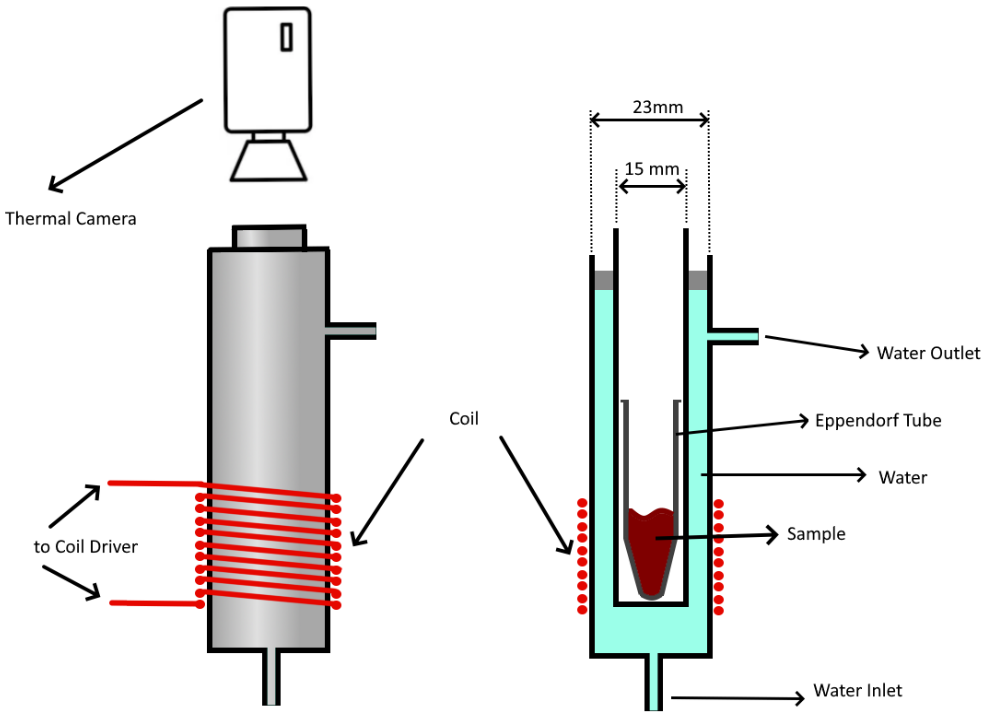


Figure 3.17 Drawing of the measurement cell

3.3.4 Video to Heating Curve Software

The thermal camera used for measuring the temperature values in the custom hyperthermia system is capable of recording videos with a resolution of 240x400 pixels per frame and a frame rate of 30 frames per second. In the recorded videos, the temperature is indicated by a color code, and the temperature value can be determined by comparing it with the color scale within the frame. The temperature value can be obtained with a precision of 0.1ºC. With these features, the recorded videos enable the precise creation of heating curves.

The high data volume that enables high precision also poses a disadvantage for the videos. Manually extracting useful data from a large amount of data and creating heating curves is not feasible. Therefore, it was necessary to develop a software that automatically converts the video into heating curves.

The implemented software opens the video files recorded by the thermal camera and examines each frame individually. Before starting the conversion process, the user selects a pixel to determine the sample position within the frame and sets the maximum and minimum temperature values from the color scale as input. After these settings, the conversion process is initiated. The software takes the average of the color codes in a 5x5 pixel area around the selected pixel. This process improves the signal-to-noise ratio that can be caused by a single pixel. The obtained average color code is compared with the color scale within the frame to obtain the temperature value. The obtained temperature value, along with the time of the examined frame within the video, is recorded. Then, the software proceeds to the next frame after a certain time interval (1 second for this study) and repeats the same process. This way, temperature information is obtained at regular time intervals throughout the entire video. The flowchart of this software can be seen in Figure 3.18.

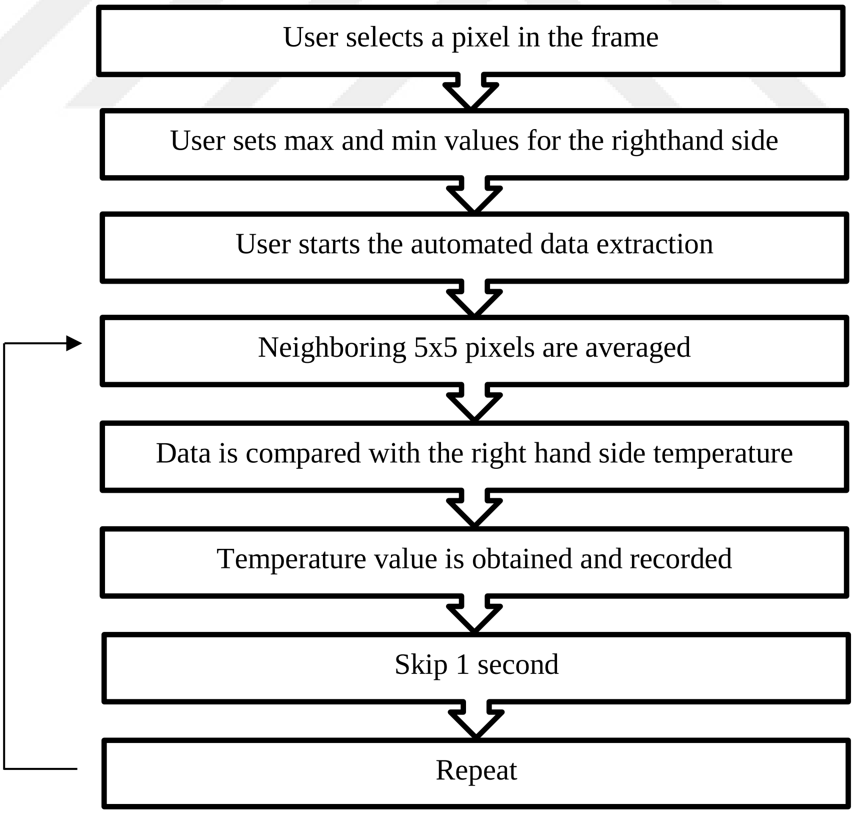

User selects a pixel in the frame
User sets max and min values for the righthand side
User starts the automated data extraction
Neighboring 5x5 pixels are averaged
Data is compared with the right hand side temperature
Temperature value is obtained and recorded
Skip 1 second
Repeat

Figure 3.18 Flow chart of the video to heating curve software

3.4 Validation of the Custom Hyperthermia System

In this section, the accuracy and repeatability of the results produced by the hyperthermia system designed and implemented for this study were tested using various methods in the laboratory.

3.4.1 Waveform and Magnetic Field Calculations

Before being used in hyperthermia experiments, the custom hyperthermia system was tested to determine whether it could generate the desired waveforms at the required frequencies. To do this, the outputs of the microcontroller-based frequency generator were monitored using an oscilloscope. It was observed that square waves at the expected frequencies were obtained, corresponding to the commands sent by the computer. This confirmed that the software on the computer, the embedded software on the microcontroller, and the communication between them were functioning correctly.

Secondly, the accuracy of the coil current profile in the coil driver circuit was tested. For this purpose, the potential drop across a resistor with a known resistance of 1 Ω, connected in series with the coil, was measured using an oscilloscope. This measurement helped assess the actual current flowing through the coil and verify the accuracy of the coil driver circuit (Figure 3.19).

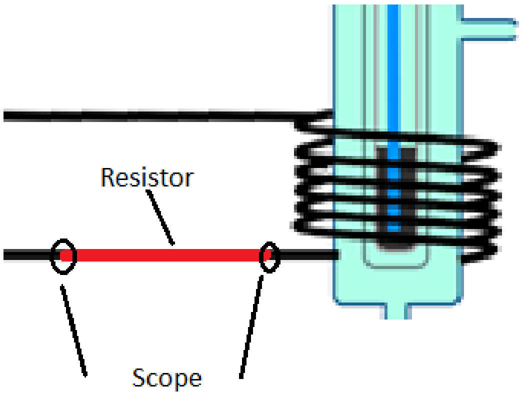


Figure 3.19 The potential drop across a resistor with a known resistance of 1 Ω, connected in series with the coil, was measured using an oscilloscope

The obtained potential difference profile was multiplied by the resistance value to obtain the current profile flowing through the coil (Equation 16). It was concluded that the observed current profile is suitable for use in this study and a successful alternative current generator has been developed.

$$I = V_R \, . \, R \qquad [16]$$

Calculated current profiles can be seen in the Figure 3.20 for 22 kHz and 77 kHz.

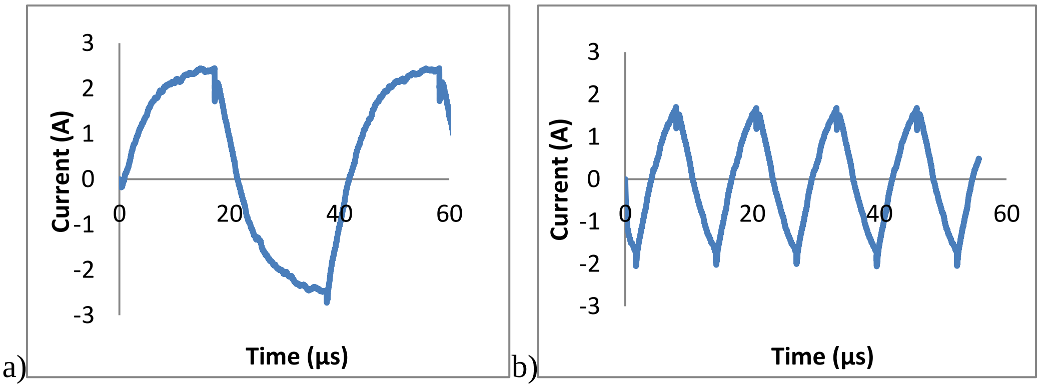


Figure 3.20 Current profiles for 22 kHz and 77 kHz

In such systems, it is not possible to maintain constant magnetic field amplitude. However, utilizing the obtained current profiles, it is possible to calculate the magnetic field amplitude at different frequencies.

The magnetic field strength, *B*, inside the center of a solenoid is found using the equation.

$$\mathrm{B} = \frac{\mu_0 N I}{L} \qquad [17]$$

Where *I* is the current of the solenoid, *N*, is the number of turns the solenoid has, *L* is the length of the solenoid, and μ_0 is the permeability of free space, 1,26 $\times 10^{-6}$ Tm/A. Calculated magnetic field strengths are presented in the Table 3.6.

Table 3.6 *Calculated magnetic field strengths for different frequencies*

Period (6.5 ns)	Frequency (kHz)	RMS Current (A)	Magnetic Field Strength (G)
2000	77	0.95	11.9
3000	51	1.2	15.1
4000	38.5	1.46	18.3
4500	34	1.5	18.8
5000	30	1.54	19.3
5500	28	1.63	20.5
6000	25.5	1.7	21.4
6500	23.5	1.64	20.6
7000	22	1.6	20.1

3.4.2 Repeatability Tests of Hyperthermia Experiments

Short-term and long-term repeatability tests were conducted to assess the reliability and repeatability of the results obtained from the custom hyperthermia system. For the short-term repeatability test, a selected reference sample (Si_0) was subjected to heating measurements at a specific frequency. After the measurement, the sample was left to cool without being removed from the system. Once the sample returned to a stable state, the heating measurement was repeated. In this way, four consecutive measurements were performed, and the results were compared. The results obtained from these tests have been presented in Table 3.7 and Figure 3.21.

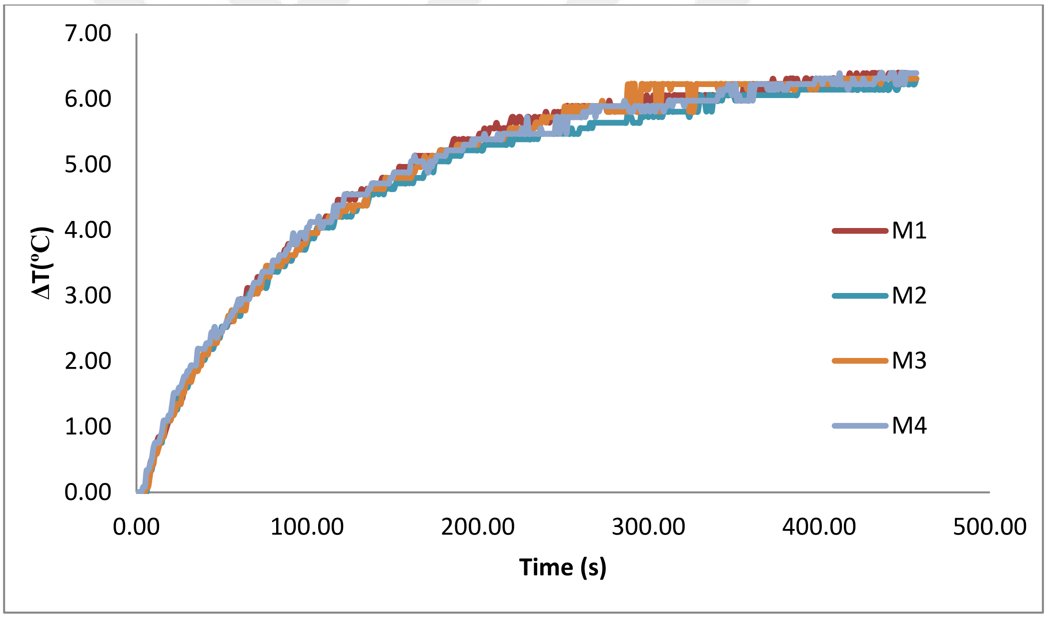


Figure 3.21 Short term repeatability test results

The results indicate that the heating values obtained from consecutive measurements deviate from the mean by a maximum of 0.09ºC. This deviation value is lower than the measurement accuracy of the thermal camera, which is

0.1ºC. Therefore, it can be concluded that these measurements are equivalent and the short-term stability of the system is very high.

Table 3.7 *ΔT values at 400 s for consecutive measurements M1, M2, M3 and M4*

	ΔT ºC @ 400s	Difference from the average
M1	6.32	0.06
M2	6.17	0.09
M3	6.27	0.01
M4	6.28	0.02
Average	6.26	

**The values are obtained through the '25 pixel averaging' mentioned in the video to heating curve section. Although it is known that these values do not comply with significant figure rules, they are presented in this way. Otherwise, if all the values were the same, it would not be possible to compare them.*

To conduct the long-term repeatability tests, a reference sample was measured at a specific frequency and then removed from the system. After a one-week interval, the same sample was repositioned in the system, and the measurement at the same frequency was repeated. Four measurements were performed and compared. The results obtained from these tests are presented in Figure 3.22 and Table 3.8.

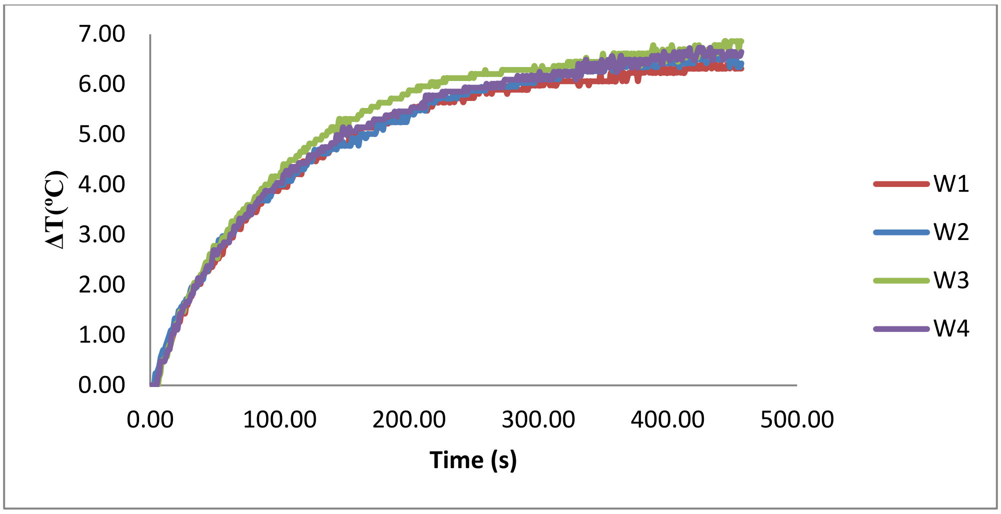


Figure 3.22 Long term repeatability test results

The results obtained from the long-term stability tests indicate that while they may not be as good as the short-term results, the system remains stable over extended periods. It was observed that the maximum deviation in measurements over a one-month period was 0.19ºC. This value is deemed sufficient for the objectives of this study.

Table 3.8 *ΔT values at 400 s for measurements from consecutive weeks W1, W2, W3 and W4*

	ΔT ºC @ 400sec	Difference from the average
W1	6.32	0.19
W2	6.48	0.03
W3	6.66	0.15
W4	6.59	0.08
Average	6.51	

**The values are obtained through the '25 pixel averaging' mentioned in the video to heating curve section. Although it is known that these values do not comply with significant figure rules, they are presented in this way. Otherwise, if all the values were the same, it would not be possible to compare them.*

3.4.3 Validation with Simulation

In order to test whether the developed hyperthermia system is in line with theoretical expectations, simulation software developed by a group of researchers at Darmstadt University was utilized. This simulation software is capable of simulating the Specific Absorption Rate (SAR) that a given magnetic fluid can exhibit under a given alternative magnetic field, based on the provided characteristics of the fluid.

To compare the simulation and experimental results, a well-designed experiment was conducted with suitable parameters. For this purpose, a sample (Si_7) with magnetic characterizations was suspended in oil and subjected to hyperthermia experiments in the custom hyperthermia system. Then, to test the accuracy of the parameters and the simulation, an oily suspension of the same material was prepared and subjected to hyperthermia measurements in a commercial hyperthermia system with known magnetic field characteristics. The experimental SAR values were obtained from these measurements. The parameters and characteristics of the magnetic fluid in these systems were input into the simulation to obtain the expected SAR values. The results are provided in Table 3.9.

Table 3.9 *Experimental simulation results*

	Experiments			Simulation parameters						
	Total MNP mass	Mass of oil	**SAR**	Viscosity	Diameter	Field	Frequency	K_{eff}	Ms	**Simulated SAR**
	(g)	(g)	**W/g**	Ns/m^2	nm	G	(kHz)			**W/g**
Commercial	0.005	0.9	**99**	0.025	16	326	300	53	467	**99.6**
Custom system	0.3	0.5	**0.30**	0.025	16	12	77	53	467	**0.134**

Upon examining these results, it can be observed that the SAR value obtained from the commercial system is consistent with the SAR value obtained from the simulation with the parameters of this system. This indicates that the simulation was correctly executed and the appropriate parameters were provided.

There is a small difference observed between the SAR values obtained from the custom system and the simulated SAR values based on the parameters of this system. This difference may be attributed to the inadequate characterization of the magnetic field properties of the custom system, especially the field magnitude. Despite this small difference, these results indicate that the custom system largely conforms to the theoretical expectations.

3.5 Validation of Developed Heating Curve Fitting Software

Measurement of SAR values is of great importance in hyperthermia experiments. SAR value allows for comparisons between different hyperthermia experiments. SAR value is not directly measurable and is often calculated from temperature change graphs over time. For this calculation, a time interval near the beginning of the experiment is selected. The difference between the temperature at the end of this interval and the temperature at the beginning is divided by the time interval to measure the rate of temperature change. This measurement is based on the assumption that the temperature change is constant and linear within this interval. This assumption is valid only for systems where there is no heat loss. On the other hand, at the beginning of the experiment, when the temperature difference between the ambient temperature and the sample temperature is low, the assumption of no heat loss can be considered acceptable.

The temperature change in these systems can be simply formulated as follows:

$$T_{t+\Delta t} = T_t + \Delta t \; \frac{P_{in} - LP(T_t - T_{ambient})}{C} \qquad [18]$$

In this formula, T_t represents the temperature at time t, P_{in} is the power input to the system, LP is the heat loss parameter, $T_{ambient}$ is the ambient temperature, and C is the total heat capacity of the system. With this formula, the temperature $T_{t+\Delta t}$ can be calculated from T_t after a time interval of Δt. It can be observed from this formula that when the difference between T_t and $T_{ambient}$ is close to zero, it corresponds to a linear fit condition. This indicates that the assumption of a constant and linear temperature change is valid in cases where the temperature difference is small.

On the other hand, in cases where Pin is small, even if T_t - $T_{ambient}$ has a small value, it deviates from linearity in temperature change. This complicates power calculations using temperature change in low-power systems. In such situations, it is necessary to accurately account for LP to ensure proper power calculations.

A software has been developed to enable a more objective evaluation of the temperature curves obtained during the study. The developed software utilizes numerical methods to perform curve fitting. By trying different values for the LP and P_{in} parameters in the given formula, it aims to reproduce the heating curves. The parameters that yield the curve with the least deviation from the experimental curve among the reproduced curves are determined as the system's LP and P_{in} values.

This approach and the implemented algorithm were tested by comparing them with various experimental data. Firstly, the heating curve of a specific sample at a specific frequency was obtained. Then, after turning off the alternating magnetic field, the cooling curve of the heated material was also recorded. The heating curve was analyzed using the software to obtain the P_{in} and LP values. It is known that the P_{in} value is zero when the AMF system is turned off. On the other hand, the LP value is expected to be a constant value for the same material and system according to its definition. When the cooling curves were analyzed using the software, it was observed that the case where P_{in} is zero and LP is constant fit well with the experimental results (Figure 3.23).

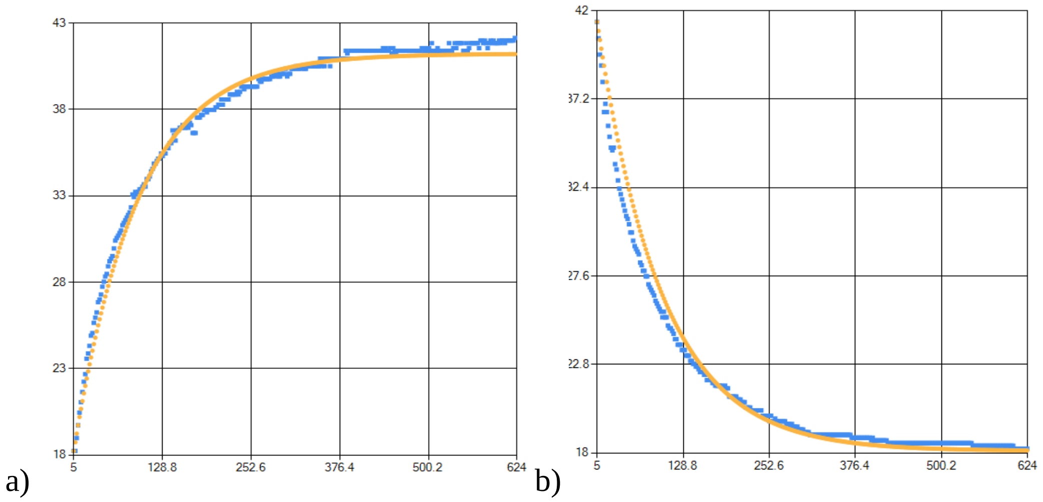


Figure 3.23 a) Heating curve of a sample (blue dots) and fitted line with parameters Pin= 0,786, LP = 0,032. (yellow dots) b) Cooling curve of a sample (blue dots) and fitted line with parameters Pin= 0, LP = 0,032. (yellow dots)

The results obtained suggest that this approach provides more accurate results compared to the linear fit approach for the custom hyperthermia system. Therefore, all the results obtained from the custom hyperthermia system were analyzed using this software in the study.

The codes of the main algorithm and related fonctions can be found in the Appendix A.

3.6 Results of Magnetic Hyperthermia Experiments

In this section, the heating behavior of the synthesized and characterized samples under alternative magnetic fields was investigated. The results have been presented within the scope of this section.

3.6.1 Results of Custom HyperthermiaSsystem

The heating behaviors of the synthesized materials under different frequencies of alternating magnetic fields (AMF) were measured with the previously described lab made system. Results are presented in Figure 3.24. The specific absorption rate (SAR) values calculated from these graphs can be seen in Table 3.10.

Table 3.10 *SAR values obtained with the lab made system at different frequencies*

Frequency	**Period**	**SAR values(W/g)**							
kHz	(x6.5 ns)	si0	si1	si2	si4	si5	si7	si8	si10
77	2000	0.0108	0.0207	0.0169	0.0246	0.0292	0.0276	0.0138	0.0476
51	3000	0.0138	0.0292	0.0315	0.0376	0.0507	0.0507	0.0230	0.0822
38.5	4000	0.0192	0.0376	0.0422	0.0492	0.0691	0.0745	0.0307	0.1213
34	4500	0.0238	0.0422	0.0438	0.0538	0.0760	0.0860	0.0323	0.1367
30	5000	0.0269	0.0438	0.0461	0.0515	0.0791	0.0929	0.0323	0.1482
28	5500	0.0207	0.0438	0.0461	0.0545	0.0806	0.0991	0.0338	0.1559
25.5	6000	0.0238	0.0399	0.0461	0.0568	0.0791	0.0991	0.0315	0.1559
23.5	6500	0.0200	0.0407	0.0453	0.0515	0.0760	0.0983	0.0315	0.1536
22	7000	0.0230	0.0384	0.0422	0.0461	0.0730	0.0952	0.0292	0.1482

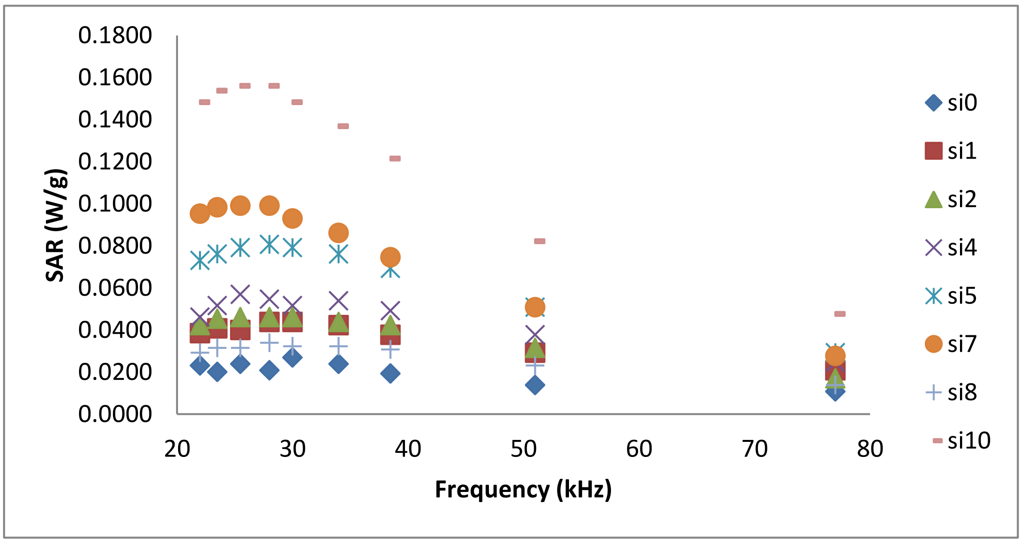


Figure 3.24 Graph of SAR values obtained with the lab made system at different frequencies

The data is presented in relation to frequency, as it is conventionally used in the literature. However, if we want to analyze the data based on relaxation time concept, presenting the data in the period domain would facilitate understanding. While the measurement points appear to be concentrated in a single region when viewed in the frequency domain, this is not the case when viewed in the period domain. Figure 3.25.

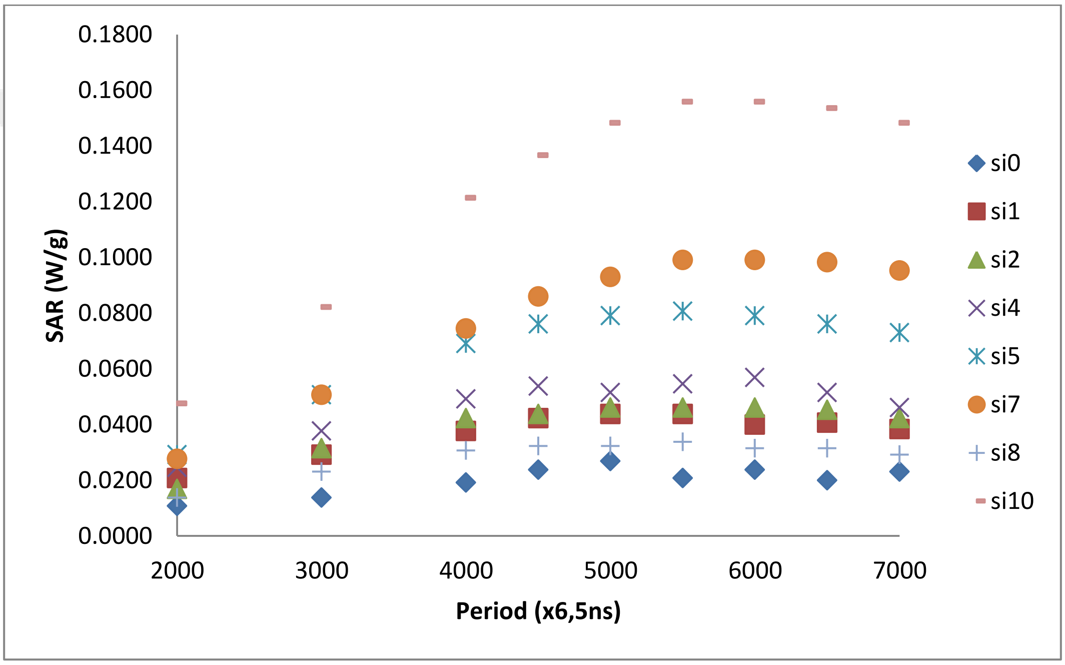


Figure 3.25 Graph of SAR values obtained with the lab made system at different periods

As the data on the graph examined, it can be observed that all samples exhibit higher SAR values at lower frequencies. This can be explained in two ways. The first explanation is that the magnetic field amplitudes generated by the system may exhibit slight variations at different frequencies. It is known that the system operates at higher magnetic field intensities at lower frequencies. The second explanation is that at lower frequencies, the particles have more time to align their magnetic moments with the applied magnetic field. The time required for the alignment of magnetic moments is well-established at lower frequencies.

Generally, samples with higher M_s values are expected to exhibit higher SAR values. The SAR values of three samples with the lowest, highest, and intermediate M_s values were compared by examining the VSM results in Table 3.5. This comparison can be seen in Figure 3.26 and Table 3.11.

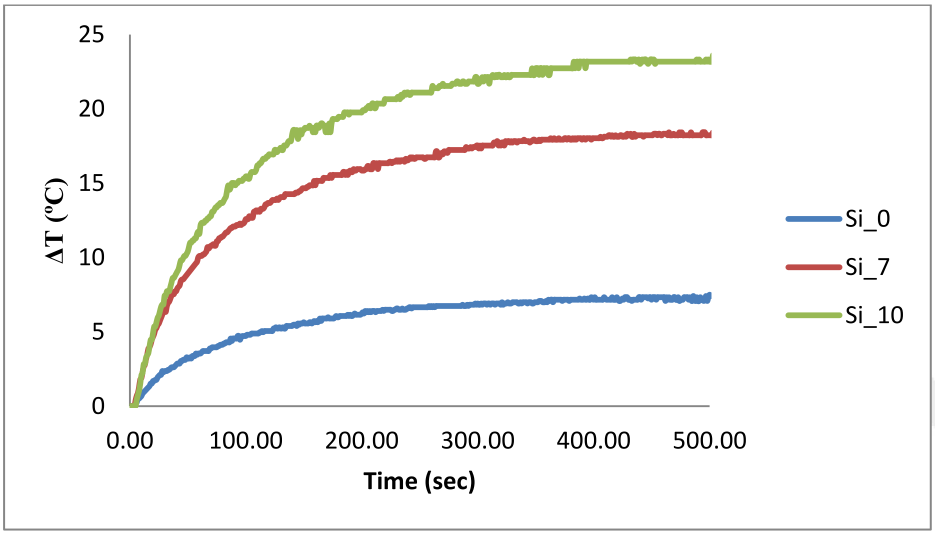


Figure 3.26. Graphs of measured heating curves utilising the custom hypertermia system @ 25,5kHz and 1,7kA/m

Table 3.11 *Table of Magnetic saturation and SAR values of Si0, Si7 and Si10 samples obtained from the custom hypertermia system @ 25,5kHz and 1,7kA/m*

	Magnetic Saturation	SAR
	emu/cm^3	W/g
Si0	346	0.0238
Si7	467	0.0991
Si10	190	0.1559

Even though the Si10 sample has the lowest magnetic saturation value, it exhibits the highest SAR values. On the other hand, the Si7 sample, which has the highest magnetic saturation value, shows SAR values that are 35% lower than those of the Si10 sample. The Si0 sample, despite having a magnetic saturation value that lies between the saturation values of these two samples, demonstrates considerably lower SAR values (Table 3.11).

To investigate the particles' behaviors that do not conform to the general expectations, the same measurements were conducted using the commercial system, which is considered as a reference in validating the custom hyperthermia system.

As stated in the experimental section, the custom hyperthermia measurements were performed with powdered samples. Due to the high heating observed in dry samples at high field amplitudes, it was decided to disperse the samples both in water and oil medium to control the heating behavior.

Based on these results, it is observed that the heating values of the samples do not align with their magnetic properties. The reasons behind these mismatched values will be explained in subsequent sections.

3.6.2 Results of Commercial System

To confirm the results obtained from the custom hypertermia system, the same samples were subjected to additional experiments using a commercial system (EasyHeat LI) in aqueous and oily suspensions. Measurements were conducted using this commercial alternative field generator at a frequency of 300 kHz and a magnetic field strength of 26 kA/m. The results obtained from the samples can be seen in Figure 3.20. The calculated SAR values from these graphs, as well as the values obtained from magnetic characterization measurements, can be found in Table 3.12.

Table 3.12 *SAR values obtained with the commercial system in oil and water*

	SAR values (W/g)	
	Oil	Water
Si0	109.8	120.4
Si7	99.0	118.6
Si10	75.6	95.8

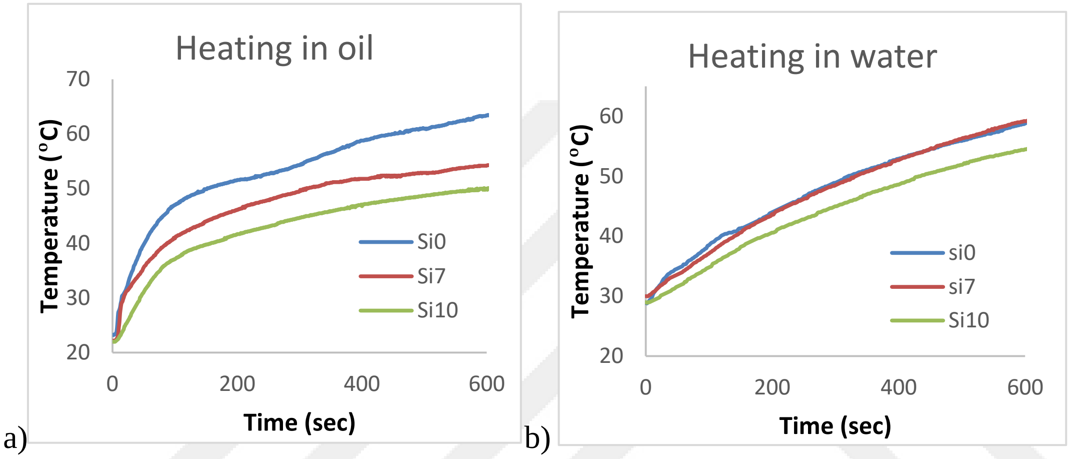


Figure 3.27 Graphs of measured heating curves in a) oil and in b) water utilising the commercial system at 300kHz and 26kA/m

When these results are evaluated, it can be observed that the heating and SAR results obtained in a system having high magnetic field and high frequency are more consistent with the magnetic saturation values.

3.6.3 Discussions on Magnetic Heating Behaviors

When comparing the experiments conducted using two different alternative magnetic field generators, magnetic field strength can be used as the main parameter. The commercial system has a magnetic field strength that is 50 times higher than the custom hyperthermia system. The second important parameter is the magnetic properties of the materials. The magnetic characteristics of the materials used in the experiments should be taken into account when comparing the results. The third parameter that can be considered is the viscosity of the medium. The experiments were conducted in both dry and viscous mediums. The viscosity of the medium can affect the heat transfer and, consequently, the heating amount. By considering these parameters, a comprehensive comparison can be made between the experiments conducted using different magnetic field generators.

Regardless of the mechanism, amplitude and frequency of the applied field, magnetic properties of the material, particle size, and viscosity of the fluid for suspended systems plays a definitive role on the amount of heating and SAR value.

The generation of heat by magnetic nanoparticles under an alternating magnetic field can occur through two mechanisms, as mentioned before. One mechanism is the heat energy generated by the mechanical rotation of the nanoparticles (Brownian relaxation), and the other mechanism is the energy generated by the reversal of the magnetic moment within the particle (Neel relaxation). In solid environments, Brownian motion is not expected, so these systems can be described as pure Neel systems [72, 73]. On the other hand, particles suspended in a liquid can undergo heating through both mechanisms.

Measurements conducted in two different systems have yielded different results. This seemingly contradictory results can be explained by taking the differences in the two measurement systems into account.

3.6.4 Evaluation of the Experimental Results

In measurements conducted in a suspended medium and high magnetic field, obtained results are found to be consistent with the high field regime. When comparing samples Si0 and Si7, similar SAR values were obtained due to the influence of magnetic saturation values and particle sizes. However, in experiments conducted in oil medium, the low viscosity led to a longer Brownian relaxation time. The larger particle size of Si7 was more affected by this effect, resulting in a lower SAR value compared to the Si0 sample. Sample Si10 exhibited the lowest

SAR value under both conditions due to its lower magnetic saturation value (Table 3.13).

Table 3.13 *Summary of the results for Si0, Si7 and Si10 samples*

	Magnetic Saturation emu/cm^3	K_{eff} kJ/m^3	SAR Values W/g		
			Custom system		Commercial system
			Powder	Oil	Water
Si0	346	62.56	0.0238	109.8	120.4
Si7	467	52.96	0.0991	99.0	118.6
Si10	190	69.44	0.1559	75.6	95.8

In contrast, measurements conducted on the same samples in a dry environment and under low magnetic field conditions yielded different results. The required σ and h_0 values for understanding these results are provided in Table 3.14. These values were calculated using the results obtained from the magnetic measurements, and the following equations.

$$h_0 = \frac{H_0 M_S}{2K_{eff}} \qquad [18]$$

$$\sigma = \frac{K_{eff} V}{k_B T} \qquad [19]$$

Table 3.14 *The calculated σ and h_0 values from the magnetic measurements*

	σ	**h_0**	**SAR @25 kHz**
Si0	12.7	0.009	0.024
Si1	14.3	0.003	0.040
Si2	12.5	0.009	0.046
Si4	14.2	0.012	0.057
Si5	16.8	0.006	0.079
Si7	15.8	0.014	0.099
Si8	13.7	0.016	0.032
Si10	12.3	0.004	0.156

All the σ values observed in the Table 3.14 are above the σ_{max} value, indicating that the data should be evaluated under high barrier region approach. In the low barrier region, the efficiency of energy transfer decreases as the magnetic field amplitude increases [73]. This is the opposite of what is expected for the high barrier region. On the other hand, when h_0 values are examined, it can be seen that all values are below 0.02 and should be analyzed using the low-field regime approach. The values of h_0 used in simulations in the literature are not as low as the h_0 values achieved in experiments. Hence it is difficult to make a direct comparison. However, the simulation results indicate consistent outcomes with the experiments, suggesting a correlation. The relations to this conclusion can be found in the definitions part of the Introduction Chapter.

When evaluating the values provided in this table and considering the information related to the high barrier - low filed regime, it can be observed that the obtained results are consistent. In the case of the Si0 sample, it can be seen that it has a relatively low σ value compared to the other samples in the experimental set. Under these conditions, the calculated effective magnetic field is found to be above the average of the experimental set. As a result, a lower level of heating is observed. When evaluating the Si7 sample, it can be observed that it has a relatively high σ value and a high h_0 value, resulting in a high heating value. The highest heating value was obtained from the Si10 sample, which has the lowest σ and h_0 values. This approach also explains the heating values of the other samples.

During the study, it was observed that the experimental results obtained from the implemented hyperthermia system are consistent with the expectations obtained from simulations in the literature. It can be concluded that this system, or similar systems that may be developed, can support low field regime studies.

CHAPTER 4

CONCLUSIONS

In this study, a low field variable frequency hyperthermia system was implemented to be able to evaluate the impact of parameters that can affect the efficiency of hyperthermia for medical purposes. Additionally, different magnetic nanoparticles were synthesized and characterized to be used in the experiments. These nanoparticles were subjected to experiments in the implemented hyperthermia device to establish relationships between their properties and heating behaviors.

During the study, magnetic nanoparticles were synthesized using different synthesis parameters, resulting in different magnetic properties. The obtained experimental set includes materials with magnetic saturation values that cover the entire range reported in the literature for magnetite and maghemite. This wide range of magnetic saturation values aligns successfully with the goals of the study.

The multitude of parameters and the interdependencies among them make it difficult to conduct experimental studies in this field with a single and independent parameter. On the other hand, simulation-based studies relying on mathematical methods have been conducted with a single independent parameter. These simulation experiments have found correlations between the parameters and heating expectations. In mathematical approaches, it has been observed that both the desired conditions for medical applications and the sufficient level of heating for cancer therapy can be achieved.

The system implemented during the study yielded consistent results with the mathematical approach. It was considered an innovative system due to its ability to provide results that cannot be obtained from systems applying high magnetic fields. Also this system is found to be promising for its potential contributions to the low field regime of this research area.

The developed system is capable of generating an alternating magnetic field in the range of 1-2 kA/m and 20-80 kHz frequency. These values have been found to be useful for understanding the low-field regime behavior. Moreover, the long-term stable operation and reliable results obtained from this system have enabled systematic studies to be conducted.

During the study, a curve fitting software was developed to enable more accurate measurement of SAR values, particularly in low-power systems. This software reduces the errors that may arise from the linear fit approach used in low-power systems. Additionally, the consideration of heat loss in the calculations through this program allows for the comparison of results from different materials and systems with different heat loss characteristics.

In the future, there is a potential for further developments and wider application of this system to various materials. The results obtained from this system can be supported and complemented by mathematical approaches, allowing for a deeper understanding and advancement of knowledge in the field.

REFERENCES

1. Azie, O., K. Castillo-Torres, Z. Greenberg, D.P. Arnold, and J. Dobson, Magnetically triggered release of active TGF-B from spin vortex microdiscs. Journal of Magnetism and Magnetic Materials, 2022. 546: p. 168732.

2. Józefczak, A.A. Skumiel, Study of heating effect and acoustic properties of dextran stabilized magnetic fluid. Journal of Magnetism and Magnetic Materials, 2007. 311(1): p. 193-196.

3. Pankhurst, Q., N. Thanh, S. Jones, and J. Dobson, Progress in applications of magnetic nanoparticles in biomedicine. Journal of Physics D: Applied Physics, 2009. 42(22): p. 224001.

4. Shokrollahi, H., A. Khorramdin, and G. Isapour, Magnetic resonance imaging by using nano-magnetic particles. Journal of Magnetism and Magnetic Materials, 2014. 369: p. 176-183.

5. Gavas, S., S. Quazi, and T.M. Karpinski, Nanoparticles for Cancer Therapy: Current Progress and Challenges. Nanoscale Research Letters, 2021. 16(1).

6. Kline, T.L., Y.H. Xu, Y. Jing, and J.P. Wang, Biocompatible high-moment FeCo-Au magnetic nanoparticles for magnetic hyperthermia treatment optimization. Journal of Magnetism and Magnetic Materials, 2009. 321(10): p. 1525-1528.

7. Ribeiro, P.T., J.A. Moreira, J.F. Monterio, and S.M. Laranjeira, Nanomaterials in cancer: Reviewing the combination of hyperthermia and triggered chemotherapy. Journal of Controlled Release, 2022. 347: p. 89-103.

8. Khandhar, A.P., R.M. Ferguson, and K.M. Krishnan, Monodispersed magnetite nanoparticles optimized for magnetic fluid hyperthermia: Implications in biological systems. Journal of applied physics, 2011. 109(7).

9. Li, Z., M. Kawashita, N. Araki, M. Mitsumori, M. Hiraoka, and M. Doi, Magnetite nanoparticles with high heating efficiencies for application in the hyperthermia of cancer. Materials Science and Engineering: C, 2010. 30(7): p. 990-996.

10. Liu, X.L., H.M. Fan, J.B. Yi, Y. Yang, E.S.G. Choo, J.M. Xue, and J. Ding, Optimization of surface coating on Fe_3O_4 nanoparticles for high performance magnetic hyperthermia agents. Journal of Materials Chemistry, 2012. 22(17): p. 8235-8244.

11. Purushotham, S., P. Chang, H. Rumpel, I. Kee, R. Ng, P. Chow, C. Tan, and R. Ramanujan, Thermoresponsive core–shell magnetic nanoparticles for combined modalities of cancer therapy. Nanotechnology, 2009. 20(30): p. 305101.

12. Purushotham, S.R. Ramanujan, Modeling the performance of magnetic nanoparticles in multimodal cancer therapy. Journal of Applied Physics, 2010. 107(11).

13. Thomas, L.A., L. Dekker, M. Kallumadil, P. Southern, M. Wilson, S.P. Nair, Q.A. Pankhurst, and I.P. Parkin, Carboxylic acid-stabilised iron oxide nanoparticles for use in magnetic hyperthermia. Journal of Materials Chemistry, 2009. 19(36): p. 6529-6535.

14. Wust, P., B. Hildebrandt, G. Sreenivasa, B. Rau, J. Gellermann, H. Riess, R. Felix, and P. Schlag, Hyperthermia in combined treatment of cancer. The lancet oncology, 2002. 3(8): p. 487-497.

15. Hergt, R., R. Hiergeist, I. Hilger, W.A. Kaiser, Y. Lapatnikov, S. Margel, and U. Richter, Maghemite nanoparticles with very high AC-losses for application in RF-magnetic hyperthermia. Journal of Magnetism and Magnetic Materials, 2004. 270(3): p. 345-357.

16. Hilger, I., R. Hergt, and W. Kaiser. Use of magnetic nanoparticle heating in the treatment of breast cancer. in IEE Proceedings-Nanobiotechnology. 2005. IET.

17. Kim, D.-H., E.A. Rozhkova, I.V. Ulasov, S.D. Bader, T. Rajh, M.S. Lesniak, and V. Novosad, Biofunctionalized magnetic-vortex microdiscs for targeted cancer-cell destruction. Nature materials, 2010. 9(2): p. 165-171.

18. Attaluri, A., S.K. Kandala, M. Wabler, H. Zhou, C. Cornejo, M. Armour, M. Hedayati, Y. Zhang, T.L. DeWeese, and C. Herman, Magnetic nanoparticle hyperthermia enhances radiation therapy: A study in mouse models of human prostate cancer. International Journal of Hyperthermia, 2015. 31(4): p. 359-374.

19. Duval, K.E., N.A. Vernice, R.J. Wagner, S.N. Fiering, J.D. Petryk, G.J. Lowry, S.S. Tau, J. Yin, G.R. Houde, and A.S. Chaudhry, Immunogenetic effects of low dose (CEM43 30) magnetic nanoparticle hyperthermia and radiation in melanoma cells. International Journal of Hyperthermia, 2019. 36(sup1): p. 37-46.

20. Johannsen, M., B. Thiesen, P. Wust, and A. Jordan, Magnetic nanoparticle hyperthermia for prostate cancer. International Journal of Hyperthermia, 2010. 26(8): p. 790-795.

21. Oei, A.L., P. Korangath, K. Mulka, M. Helenius, J.B. Coulter, J. Stewart, E. Velarde, J. Crezee, B. Simons, and L.J. Stalpers, Enhancing the abscopal effect of radiation and immune checkpoint inhibitor therapies with magnetic nanoparticle hyperthermia in a model of metastatic breast cancer. International Journal of Hyperthermia, 2019. 36(sup1): p. 47-63.

22. Petryk, A.A., A.J. Giustini, R.E. Gottesman, P.A. Kaufman, and P.J. Hoopes, Magnetic nanoparticle hyperthermia enhancement of cisplatin chemotherapy cancer treatment. International Journal of Hyperthermia, 2013. 29(8): p. 845-851.

23. Vorotnikova, E., R. Ivkov, A. Foreman, M. Tries, and S.J. Braunhut, The magnitude and time-dependence of the apoptotic response of normal and malignant cells subjected to ionizing radiation versus hyperthermia. International journal of radiation biology, 2006. 82(8): p. 549-559.

24. Kim, J.H., E.W. Hahn, and S.A. Ahmed, Combination hyperthermia and radiation therapy for malignant melanoma. Cancer, 1982. 50(3): p. 478-482.

25. Takahashi, I., Y. Emi, S. Hasuda, Y. Kakeji, Y. Maehara, and K. Sugimachi, Clinical application of hyperthermia combined with anticancer drugs for the treatment of solid tumors. Surgery, 2002. 131(1): p. S78-S84.

26. Datta, N.R., E. Puric, D. Klingbiel, S. Gomez, and S. Bodis, Hyperthermia and radiation therapy in locoregional recurrent breast cancers: a systematic review and meta-analysis. International Journal of Radiation Oncology Biology Physics, 2016. 94(5): p. 1073-1087.

27. Spirou, S.V., M. Basini, A. Lascialfari, C. Sangregorio, and C. Innocenti, Magnetic hyperthermia and radiation therapy: radiobiological principles and current practice. Nanomaterials, 2018. 8(6): p. 401.

28. Javidi, M., M. Heydari, M.M. Attar, M. Haghpanahi, A. Karimi, M. Navidbakhsh, and S. Amanpour, Cylindrical agar gel with fluid flow subjected to an alternating magnetic field during hyperthermia. International Journal of Hyperthermia, 2015. 31(1): p. 33-39.

29. Javidi, M., M. Heydari, A. Karimi, M. Haghpanahi, M. Navidbakhsh, and A. Razmkon, Evaluation of the effects of injection velocity and different gel concentrations on nanoparticles in hyperthermia therapy. Journal of biomedical physics & engineering, 2014. 4(4): p. 151.

30. Heydari, M., M. Javidi, M.M. Attar, A. Karimi, M. Navidbakhsh, M. Haghpanahi, and S. Amanpour, Magnetic fluid hyperthermia in a cylindrical gel contains water flow. Journal of Mechanics in Medicine and Biology, 2015. 15(05): p. 1550088.

31. Estelrich, J., E. Escribano, J. Queralt, and M.A. Busquets, Iron oxide nanoparticles for magnetically-guided and magnetically-responsive drug delivery. International journal of molecular sciences, 2015. 16(4): p. 8070-8101.

32. Kriezis, E., T.D. Tsiboukis, S.M. Panas, and J.A. Tegopoulos, Eddy currents: Theory and applications. Proceedings of the IEEE, 1992. 80(10): p. 1559-1589.

33. Chen, R., M.G. Christiansen, and P. Anikeeva, Maximizing hysteretic losses in magnetic ferrite nanoparticles via model-driven synthesis and materials optimization. ACS nano, 2013. 7(10): p. 8990-9000.

34. Carrey, J., B. Mehdaoui, and M. Respaud, Simple models for dynamic hysteresis loop calculations of magnetic single-domain nanoparticles: Application to magnetic hyperthermia optimization. Journal of applied physics, 2011. 109(8).

35. Miyazaki, T.H. Jin, The physics of ferromagnetism. Vol. 158. 2012: Springer Science & Business Media.

36. Landi, G.A. Bakuzis, On the energy conversion efficiency in magnetic hyperthermia applications: A new perspective to analyze the departure from the linear regime. Journal of Applied Physics, 2012. 111(8).

37. Aharoni, A., Introduction to the Theory of Ferromagnetism. Vol. 109. 2000: Clarendon Press.

38. Chikazumi, S., Physics of ferromagnetism. 1997: Oxford university press.

39. Dennis, C.L.R. Ivkov, Physics of heat generation using magnetic nanoparticles for hyperthermia. International Journal of Hyperthermia, 2013. 29(8): p. 715-729.

40. Gittleman, J., B. Abeles, and S. Bozowski, Superparamagnetism and relaxation effects in granular Ni-SiO_2 and Ni-Al_2O_3 films. Physical review B, 1974. 9(9): p. 3891.

41. Coffey, W., D. Crothers, Y.P. Kalmykov, and S. Titov, Precessional effects in the linear dynamic susceptibility of uniaxial superparamagnets: Dependence of the ac response on the dissipation parameter. Physical Review B, 2001. 64(1): p. 012411.

42. Ebert, H., S. Mankovsky, D. Ködderitzsch, and P.J. Kelly, Ab initio calculation of the Gilbert damping parameter via the linear response formalism. Physical review letters, 2011. 107(6): p. 066603.

43. Mrabti, H.E., S.V. Titov, P.-M. Déjardin, and Y.P. Kalmykov, Nonlinear stationary ac response of the magnetization of uniaxial superparamagnetic nanoparticles. Journal of Applied Physics, 2011. 110(2).

44. Poperechny, I., Y.L. Raikher, and V. Stepanov, Dynamic magnetic hysteresis in single-domain particles with uniaxial anisotropy. Physical Review B, 2010. 82(17): p. 174423.

45. Usov, N., Low frequency hysteresis loops of superparamagnetic nanoparticles with uniaxial anisotropy. Journal of Applied Physics, 2010. 107(12).

46. Pucci, C., A. Degl'Innocenti, M.B. Gümüş, and G. Ciofani, Superparamagnetic iron oxide nanoparticles for magnetic hyperthermia: Recent advancements, molecular effects, and future directions in the omics era. Biomaterials Science, 2022. 10(9): p. 2103-2121.

47. Kötitz, R., W. Weitschies, L. Trahms, and W. Semmler, Investigation of Brownian and Neel relaxation in magnetic fluids. Journal of magnetism and magnetic materials, 1999. 201(1): p. 102-104.

48. Torres, T.E., E. Lima Jr, M.P. Calatayud, B. Sanz, A. Ibarra, R. Fernández-Pacheco, A. Mayoral, C. Marquina, M.R. Ibarra, and G.F. Goya, The relevance of Brownian relaxation as power absorption mechanism in Magnetic Hyperthermia. Scientific reports, 2019. 9(1): p. 3992.

49. Morales, I., R. Costo, N. Mille, G.B. Da Silva, J. Carrey, A. Hernando, and P. De la Presa, High frequency hysteresis losses on γ-Fe_2O_3 and Fe_3O_4: Susceptibility as a magnetic stamp for chain formation. Nanomaterials, 2018. 8(12): p. 970.

50. Rosensweig, R.E., Heating magnetic fluid with alternating magnetic field. Journal of magnetism and magnetic materials, 2002. 252: p. 370-374.

51. Landi, G.T., Influence of the magnetization damping on dynamic hysteresis loops in single domain particles. Journal of Applied Physics, 2012. 111(4).

52. Landi, G.T.A.D.d. Santos, Longitudinal dynamic hysteresis in single-domain particles. Journal of Applied Physics, 2012. 111(7).

53. Andreu, I.E. Natividad, Accuracy of available methods for quantifying the heat power generation of nanoparticles for magnetic hyperthermia. International Journal of Hyperthermia, 2013. 29(8): p. 739-751.

54. Verde, E.L., G.T. Landi, M. Carrião, A.L. Drummond, J.d.A. Gomes, E.D. Vieira, M.H. Sousa, and A.F. Bakuzis, Field dependent transition to the non-linear regime in magnetic hyperthermia experiments: Comparison between maghemite, copper, zinc, nickel and cobalt ferrite nanoparticles of similar sizes. Aip Advances, 2012. 2(3).

55. Eberbeck, D., C.L. Dennis, N.F. Huls, K.L. Krycka, C. Gruttner, and F. Westphal, Multicore magnetic nanoparticles for magnetic particle imaging. IEEE Transactions on Magnetics, 2012. 49(1): p. 269-274.

56. Carrião, M.S., V.R. Aquino, G.T. Landi, E.L. Verde, M.H. Sousa, and A.F. Bakuzis, Giant-spin nonlinear response theory of magnetic nanoparticle hyperthermia: A field dependence study. Journal of Applied Physics, 2017. 121(17).

57. Verde, E.L., G.T. Landi, J.d.A. Gomes, M.H. Sousa, and A.F. Bakuzis, Magnetic hyperthermia investigation of cobalt ferrite nanoparticles: Comparison between experiment, linear response theory, and dynamic hysteresis simulations. Journal of Applied Physics, 2012. 111(12).

58. Rodrigues, H.F., G. Capistrano, and A.F. Bakuzis, In vivo magnetic nanoparticle hyperthermia: A review on preclinical studies, low-field nano-heaters, noninvasive thermometry and computer simulations for treatment planning. International Journal of Hyperthermia, 2020. 37(3): p. 76-99.

59. Ba-Abbad, M.M., A. Benamour, D. Ewis, A.W. Mohammad, and E. Mahmoudi, Synthesis of Fe3O4 nanoparticles with different shapes through a co-precipitation method and their application. JOM, 2022. 74(9): p. 3531-3539.

60. Besenhard, M.O., A.P. LaGrow, A. Hodzic, M. Kriechbaum, L. Panariello, G. Bais, K. Loizou, S. Damilos, M.M. Cruz, and N.T.K. Thanh, Co-precipitation synthesis of stable iron oxide nanoparticles with NaOH: New insights and continuous production via flow chemistry. Chemical Engineering Journal, 2020. 399: p. 125740.

61. Blanco-Andujar, C., D. Ortega, Q.A. Pankhurst, and N.T.K. Thanh, Elucidating the morphological and structural evolution of iron oxide nanoparticles formed by sodium carbonate in aqueous medium. Journal of Materials Chemistry, 2012. 22(25): p. 12498-12506.

62. Daoush, W.M., Co-precipitation and magnetic properties of magnetite nanoparticles for potential biomedical applications. J. Nanomed. Res, 2017. 5(3): p. 00118.

63. Khalil, M.I., Co-precipitation in aqueous solution synthesis of magnetite nanoparticles using iron (III) salts as precursors. Arabian Journal of Chemistry, 2015. 8(2): p. 279-284.

64. Vreeland, E.C., J. Watt, G.B. Schober, B.G. Hance, M.J. Austin, A.D. Price, B.D. Fellows, T.C. Monson, N.S. Hudak, and L. Maldonado-Camargo, Enhanced nanoparticle size control by extending LaMer's mechanism. Chemistry of Materials, 2015. 27(17): p. 6059-6066.

65. Kumar, C.S., Magnetic characterization techniques for nanomaterials. 2017: Springer.

66. Mamiya, H., H. Fukumoto, J.L. Cuya Huaman, K. Suzuki, H. Miyamura, and J. Balachandran, Estimation of magnetic anisotropy of individual magnetite nanoparticles for magnetic hyperthermia. ACS nano, 2020. 14(7): p. 8421-8432.

67. Chowdhury, S.R., E.K. Yanful, and A.R. Pratt, Chemical states in XPS and Raman analysis during removal of Cr (VI) from contaminated water by mixed maghemite–magnetite nanoparticles. Journal of hazardous materials, 2012. 235: p. 246-256.

68. Molodtsova, T., M. Gorshenkov, A. Saliev, V. Vanyushin, I. Goncharov, and N. Smirnova, One-step synthesis of γ-Fe_2O_3/Fe_3O_4 nanocomposite for sensitive electrochemical detection of hydrogen peroxide. Electrochimica Acta, 2021. 370: p. 137723.

69. Yamashita, T.P. Hayes, Analysis of XPS spectra of Fe^{2+} and Fe^{3+} ions in oxide materials. Applied surface science, 2008. 254(8): p. 2441-2449.

70. Kim, W., C.-Y. Suh, S.-W. Cho, K.-M. Roh, H. Kwon, K. Song, and I.-J. Shon, A new method for the identification and quantification of magnetite–maghemite mixture using conventional X-ray diffraction technique. Talanta, 2012. 94: p. 348-352.

71. Gao, Y.S.A. Chambers, Heteroepitaxial growth of α-Fe_2O_3, γ-Fe_2O_3 and Fe_3O_4 thin films by oxygen-plasma-assisted molecular beam epitaxy. Journal of crystal growth, 1997. 174(1-4): p. 446-454.

72. Landi, G.T., Simple models for the heating curve in magnetic hyperthermia experiments. Journal of magnetism and magnetic materials, 2013. 326: p. 14-21.

73. Landi, G.T., Role of dipolar interaction in magnetic hyperthermia. Physical Review B, 2014. 89(1): p. 014403.

APPENDICES

A: Code of the Curve Fit Software

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Runtime.InteropServices;

namespace Test
{
    public partial class Form2 : Form
    {
        object mydata;
        object mydata2;
        string excellpath;
        string[] datafilenames =new string[30];
        int start = 5;
        int end = 400;
        int colnumber = 0;
        double[] measured = new double[10000];
        int max_col = 0;
        double[] D_AmbientTemp = new double[25];
        double[] D_powerin = new double[25];
        double[] D_loss = new double[25];
        double[] D_stddev = new double[25];


        public Form2(string excepath)
        {
            InitializeComponent();
            excellpath = excepath;

            for (int i = 0; i <= 19; i++)

            {
                D_stddev[i] = -1;
            }


            mydata = GetFromExcel("Sheet1", "A1", "GZ8000");
            var list = (object[,])mydata;


            if (list[1, 2] != null)
            {
                radioButton1.Text = list[1, 2].ToString();
                max_col = 2;
            }
            if (list[1, 3] != null)
            {
                radioButton2.Text = list[1, 3].ToString();
                max_col = 3;
```

```
}
if (list[1, 4] != null)
{
    radioButton3.Text = list[1, 4].ToString();
    max_col = 4;
}
if (list[1, 5] != null)
{
    radioButton4.Text = list[1, 5].ToString();
    max_col = 5;
}
if (list[1, 6] != null)
{
    radioButton5.Text = list[1, 6].ToString();
    max_col = 6;
}
if (list[1, 7] != null)
{
    radioButton6.Text = list[1, 7].ToString();
    max_col = 7;
}
if (list[1, 8] != null)
{
    radioButton7.Text = list[1, 8].ToString();
    max_col = 8;
}
if (list[1, 9] != null)
{
    radioButton8.Text = list[1, 9].ToString();
    max_col = 9;
}
if (list[1, 10] != null)
{
    radioButton9.Text = list[1, 10].ToString();
    max_col = 10;
}
if (list[1, 11] != null)
{
    radioButton10.Text = list[1, 11].ToString();
    max_col = 11;
}
if (list[1, 12] != null)
{
    radioButton11.Text = list[1, 12].ToString();
    max_col = 12;
}
if (list[1, 13] != null)
{
    radioButton12.Text = list[1, 13].ToString();
    max_col = 13;
}
if (list[1, 14] != null)
{
    radioButton13.Text = list[1, 14].ToString();
    max_col = 14;
}
if (list[1, 15] != null)
{
```

```
            radioButton14.Text = list[1, 15].ToString();
            max_col = 15;
        }
        if (list[1, 16] != null)
        {
            radioButton15.Text = list[1, 16].ToString();
            max_col = 16;
        }
        if (list[1, 17] != null)
        {
            radioButton16.Text = list[1, 17].ToString();
            max_col = 17;
        }
        if (list[1, 18] != null)
        {
            radioButton17.Text = list[1, 18].ToString();
            max_col = 18;
        }
        if (list[1, 19] != null)
        {
            radioButton18.Text = list[1, 19].ToString();
            max_col = 19;
        }
        if (list[1, 20] != null)
        {
            radioButton19.Text = list[1, 20].ToString();
            max_col = 20;
        }

        for (int i = 2; i <= max_col; i++)
        {
            colnumber = i;
            datafilenames[colnumber] = list[1, colnumber].ToString();
        }

    }

    private int findcol()
    {
        if (radioButton1.Checked) return 2;
        if (radioButton2.Checked) return 3;
        if (radioButton3.Checked) return 4;
        if (radioButton4.Checked) return 5;
        if (radioButton5.Checked) return 6;
        if (radioButton6.Checked) return 7;
        if (radioButton7.Checked) return 8;
        if (radioButton8.Checked) return 9;
        if (radioButton9.Checked) return 10;
        if (radioButton10.Checked) return 11;
        if (radioButton11.Checked) return 12;
        if (radioButton12.Checked) return 13;
        if (radioButton13.Checked) return 14;
        if (radioButton14.Checked) return 15;
        if (radioButton15.Checked) return 16;
        if (radioButton16.Checked) return 17;
        if (radioButton17.Checked) return 18;
        if (radioButton18.Checked) return 19;
        if (radioButton19.Checked) return 20;
```

```
            return 0;

        }
        private object GetFromExcel(string sheetName, string c1, string c2)
        {
            Microsoft.Office.Interop.Excel.Application oXL = null;
            Microsoft.Office.Interop.Excel._Workbook oWB = null;
            Microsoft.Office.Interop.Excel._Worksheet oSheet = null;
            Microsoft.Office.Interop.Excel.Range range = null;
            object value = null;
            try
            {
                oXL = new Microsoft.Office.Interop.Excel.Application();
                oWB = oXL.Workbooks.Open(excellpath + "\\MyExcel.xlsx");
                oSheet = String.IsNullOrEmpty(sheetName) ? 
(Microsoft.Office.Interop.Excel._Worksheet)oWB.ActiveSheet : 
(Microsoft.Office.Interop.Excel._Worksheet)oWB.Worksheets[sheetName];

                range = oSheet.get_Range(c1, c2);

                value = range.Value;

                //MessageBox.Show("Get Done!");

            }
            catch (Exception ex)
            {
               // MessageBox.Show(ex.ToString());
            }
            finally
            {
                if (oWB != null)
                    oWB.Close();
            }
            return value;
        }

        private void smooth(double[] measured)
        {
            for (int i = 3; measured[i] > 0; i++)
            {
                if (measured[i + 1] > 0 && measured[i + 2] > 0)
                {
                    if ((measured[i + 1] > measured[i] && measured[i + 1] > 
measured[i + 2])
                        || (measured[i + 1] < measured[i] && measured[i + 1] 
< measured[i + 2]))

                        measured[i + 1] = (measured[i] + measured[i + 2]) / 
2.0;

                }
            }

        }

        private double calculate_deviation(double powerin, double 
Ambienttemp, double loss, double heatcap, int startl, int endl,bool show)
```

```
        {
            chart1.ChartAreas[0].AxisY.Minimum = 1000;
            chart1.ChartAreas[0].AxisY.Maximum = 0;


            double deviation = 0;
            double difference = 0;

            double[] model = new double[10000];
            model[start] = measured[start];
            chart1.Series["Series1"].Points.Clear();
            chart1.Series["Series2"].Points.Clear();

            if (show)
            for (int i = 3; measured[i] > 0; i++)
            {
                if (chart1.ChartAreas[0].AxisY.Minimum > measured[i])
chart1.ChartAreas[0].AxisY.Minimum = Math.Floor(measured[i]);
                if (chart1.ChartAreas[0].AxisY.Maximum <= measured[i])
chart1.ChartAreas[0].AxisY.Maximum = Math.Floor(measured[i]) + 1;
            }


            deviation = 0;
            double currentdif = 0;
            int dstep = 2;
            double deltaT;
            for (int i = startl; i<=endl ; i += dstep)
            {

                deltaT = (model[i] - Ambienttemp);
              //  model[i + dstep] = model[i] + dstep * ((powerin / heatcap)
- ((model[i] - Ambienttemp) * loss / heatcap));
                model[i + dstep] = model[i] + dstep * (   (powerin - (deltaT
* loss )    ) / heatcap);


                currentdif = measured[i] - model[i];

                difference += currentdif;
                deviation += currentdif * currentdif;
                if (show)
                {
                    chart1.Series["Series2"].Points.AddXY(i, model[i]);
                    chart1.Series["Series1"].Points.AddXY(i, measured[i]);
                }
            }
            if (show)chart1.Refresh();

            textBox1.Text = powerin.ToString();
            textBox2.Text = loss.ToString();
            textBox3.Text = Ambienttemp.ToString();
            textBox4.Text = (deviation / (endl - startl)).ToString();
            groupBox2.Refresh();
            return deviation;
        }
        private void button4_Click(object sender, EventArgs e)
        {

            for(int  i= 2; i<=max_col; i++)
            {
               colnumber = i;

               loadparameter();

                fitcurve();
               saveparameter();

            }

        }

        private void button5_Click(object sender, EventArgs e)
        {
            colnumber = findcol();
            fitcurve2();
            saveparameter();

        }

        private void button1_Click(object sender, EventArgs e)
        {
            colnumber = findcol();
            fitcurve();
            saveparameter();
        }
       private void saveparameter()
       {
           D_powerin[colnumber]     = Convert.ToDouble(textBox1.Text);
           D_loss[colnumber]        = Convert.ToDouble(textBox2.Text);
           D_AmbientTemp[colnumber] = Convert.ToDouble(textBox3.Text);
           D_stddev[colnumber]      = Convert.ToDouble(textBox4.Text);

           UpdateExcel("Sheet2", 1, colnumber , datafilenames[colnumber]);
           UpdateExcel("Sheet2", 2, colnumber , textBox1.Text);
           UpdateExcel("Sheet2", 3, colnumber , textBox2.Text);
           UpdateExcel("Sheet2", 4, colnumber , textBox3.Text);
           UpdateExcel("Sheet2", 5, colnumber , textBox4.Text);


           UpdateExcel("Sheet2", colnumber + 10, 2,
datafilenames[colnumber].Substring(0,4)) ;
           UpdateExcel("Sheet2", colnumber + 10, 3, textBox1.Text);
           UpdateExcel("Sheet2", colnumber + 10, 4, textBox2.Text);
           UpdateExcel("Sheet2", colnumber + 10, 5, textBox3.Text);
           UpdateExcel("Sheet2", colnumber + 10, 6, textBox4.Text);

       }

       private void UpdateExcel(string sheetName, int row, int col, string
data)
       {
           Microsoft.Office.Interop.Excel.Application oXL = null;
```

```
            Microsoft.Office.Interop.Excel._Workbook oWB = null;
            Microsoft.Office.Interop.Excel._Worksheet oSheet = null;

            try
            {
                oXL = new Microsoft.Office.Interop.Excel.Application();
                oWB = oXL.Workbooks.Open(excellpath + "\\MyExcel.xlsx");
                oSheet = String.IsNullOrEmpty(sheetName) ?
(Microsoft.Office.Interop.Excel._Worksheet)oWB.ActiveSheet :
(Microsoft.Office.Interop.Excel._Worksheet)oWB.Worksheets[sheetName];

                oSheet.Cells[row, col] = data;

                oWB.Save();

                //MessageBox.Show("Done!");
            }
            catch (Exception ex)
            {
               // MessageBox.Show(ex.ToString());
            }
            finally
            {
                if (oWB != null)
                    oWB.Close();
            }
        }

        private void loadparameter()
        {

            var list = (object[,])mydata;
            mydata2 = GetFromExcel("Sheet2", "A1", "GZ8000");
            var list2 = (object[,])mydata2;



            for (int i = 3; list[i, colnumber] != null; i++)
             {
                 measured[i] = Convert.ToDouble(list[i, colnumber]);
                 measured[i + 1] = -1;
                 end = i;
            }


            D_powerin[colnumber] = Convert.ToDouble(list2[2, colnumber]);
            D_loss[colnumber] = Convert.ToDouble(list2[3, colnumber]);
            D_AmbientTemp[colnumber] = Convert.ToDouble(list2[4, colnumber]);
            D_stddev[colnumber] = Convert.ToDouble(list2[5, colnumber]);

            if (checkBox3.Checked) D_powerin[colnumber]     =
Convert.ToDouble(textBox1.Text);
            if (checkBox1.Checked)  D_loss[colnumber]       =
Convert.ToDouble(textBox2.Text);
            if (checkBox2.Checked) D_AmbientTemp[colnumber] =
Convert.ToDouble(textBox3.Text);
```

```
            if (!checkBox3.Checked) textBox1.Text =
D_powerin[colnumber].ToString();
            if (!checkBox1.Checked) textBox2.Text =
D_loss[colnumber].ToString();
            if (!checkBox2.Checked) textBox3.Text =
D_AmbientTemp[colnumber].ToString();
            textBox4.Text = D_stddev[colnumber].ToString();


            groupBox2.Refresh();
            if(D_stddev[colnumber] >0)
            calculate_deviation(D_powerin[colnumber],
D_AmbientTemp[colnumber], D_loss[colnumber], 3, start, end, true);


        }

        private void fitcurve()
        {


            double powerin_min = 0.00;
            double powerin_max = 1.00;
            double powerin_best = 0;
            double powerin = 0.004;

            double loss_min = 0;
            double loss_max = 0.10;
            double loss_best = 0;
            double loss = 0.001;
            double heatcap = 3;
            double deviation;
            double min_dev = 1000000000;
            double Ambienttemp = measured[start];

            if (checkBox2.Checked)
            {
                Ambienttemp = Convert.ToDouble(textBox3.Text);
            }

            if (checkBox1.Checked)
            {
                loss = Convert.ToDouble(textBox2.Text);
                loss_min = loss;
                loss_max = loss;
                loss_best = loss;
            }


            if (checkBox3.Checked)
            {
                powerin = Convert.ToDouble(textBox1.Text);
                powerin_min = powerin;
                powerin_max = powerin;
                powerin_best = powerin;
            }
            // smooth(measured);

            int count = 0;
            for (int i = 3; measured[i] > 0; i++)
            {
                count++;
                if (chart1.ChartAreas[0].AxisY.Minimum > measured[i])
chart1.ChartAreas[0].AxisY.Minimum = Math.Floor(measured[i]);
                if (chart1.ChartAreas[0].AxisY.Maximum <= measured[i])
chart1.ChartAreas[0].AxisY.Maximum = Math.Floor(measured[i]) + 1;
            }
            end = count;

            chart1.ChartAreas[0].AxisX.Minimum = start;
            chart1.ChartAreas[0].AxisX.Maximum = start + count;

            chart1.Refresh();

            double powerinterval = powerin_max - powerin_min;
            double lossinterval = loss_max - loss_min;
            double bestAmbienttemp = 0;
          //  Ambienttemp += 5.0;
            double XAmbienttemp = Ambienttemp;
            // for (int red = 1; red <= 1; red++)
            min_dev = 10000000000;

            for (; XAmbienttemp - Ambienttemp < 1.0; Ambienttemp -= 0.2)
            {

/////////////////////////////////////////////////////////////////////////////
/////////////////////////
                for (int rep = 1; rep <= 3; rep++)
                {
                    double powerstep = (powerin_max - powerin_min) / 10.0;
                    double lossstep = (loss_max - loss_min) / 10.0;
                    for (loss = loss_min; loss <= loss_max; loss += lossstep)
                        for (powerin = powerin_min; powerin <= powerin_max;
powerin += powerstep)
                        {
                            deviation = calculate_deviation(powerin,
Ambienttemp, loss, heatcap, start, end, false);

                            if (deviation < min_dev)
                            {
                                min_dev = deviation;
                                powerin_best = powerin;
                                loss_best = loss;
                                bestAmbienttemp = Ambienttemp;
                                calculate_deviation(powerin_best,
bestAmbienttemp, loss_best, heatcap, start, end, true);

                            }

                            if (checkBox1.Checked) loss_max = -10000;
                            if (checkBox3.Checked) powerin++;
                            if (checkBox2.Checked) XAmbienttemp = 1000000;
                    }

                    loss_min = loss_best - (1 * lossstep);
                    loss_max = loss_best + (1 * lossstep);
                    powerin_min = powerin_best - (1 * powerstep);
                    powerin_max = powerin_best + (1 * powerstep);
                }


///////////////////////////////////////////////////////////////////////////
                double narrow = 1;

                loss_min = loss_best - (lossinterval / narrow);
                loss_max = loss_best + (lossinterval / narrow);
                lossinterval = lossinterval / narrow;

                powerin_min = powerin_best - (powerinterval / narrow);
                powerin_max = powerin_best + (powerinterval / narrow);
                powerinterval = powerinterval / narrow;


            }

            calculate_deviation(powerin_best, bestAmbienttemp, loss_best,
heatcap, start, end, true);

        }
        private void fitcurve2()
        {


            double powerin_min = Convert.ToDouble(textBox1.Text) - 0.1;
            double powerin_max = Convert.ToDouble(textBox1.Text)+0.1;
            double powerin_best = Convert.ToDouble(textBox1.Text);
            double powerin = 0.004;

            double loss_min = Convert.ToDouble(textBox2.Text) - 0.01;
            double loss_max = Convert.ToDouble(textBox2.Text) + 0.01;
            double loss_best = Convert.ToDouble(textBox2.Text);
            double loss = 0.001;
            double heatcap = 3;
            double deviation;
            double min_dev = 1000000000;
            double Ambienttemp = measured[start];

            if (checkBox2.Checked)
            {
                Ambienttemp = Convert.ToDouble(textBox3.Text);
            }

            if (checkBox1.Checked)
            {
                loss = Convert.ToDouble(textBox2.Text);
                loss_min = loss;
                loss_max = loss;
                loss_best = loss;
            }


            if (checkBox3.Checked)
            {
                powerin = Convert.ToDouble(textBox1.Text);
                powerin_min = powerin;
                powerin_max = powerin;
                powerin_best = powerin;
            }



            // smooth(measured);

            int count = 0;
            for (int i = 3; measured[i] > 0; i++)
            {
                count++;
                if (chart1.ChartAreas[0].AxisY.Minimum > measured[i])
chart1.ChartAreas[0].AxisY.Minimum = Math.Floor(measured[i]);
                if (chart1.ChartAreas[0].AxisY.Maximum <= measured[i])
chart1.ChartAreas[0].AxisY.Maximum = Math.Floor(measured[i]) + 1;
            }
            end = count;

            chart1.ChartAreas[0].AxisX.Minimum = start;
            chart1.ChartAreas[0].AxisX.Maximum = start + count;

            chart1.Refresh();

            double powerinterval = powerin_max - powerin_min;
            double lossinterval = loss_max - loss_min;
            double bestAmbienttemp = 0;
            double XAmbienttemp = Ambienttemp;
            // for (int red = 1; red <= 1; red++)
            min_dev = 10000000000;

            for (; XAmbienttemp - Ambienttemp < 1.0; Ambienttemp -= 0.2)
            {

/////////////////////////////////////////////////////////////////////////////
/////////////////////////
                for (int rep = 1; rep <= 3; rep++)
                {
                    double powerstep = (powerin_max - powerin_min) / 10.0;
                    double lossstep = (loss_max - loss_min) / 10.0;
                    for (loss = loss_min; loss <= loss_max; loss += lossstep)
                        for (powerin = powerin_min; powerin <= powerin_max;
powerin += powerstep)
                        {
                            deviation = calculate_deviation(powerin,
Ambienttemp, loss, heatcap, start, end, false);

                            if (deviation < min_dev)
                            {
                                min_dev = deviation;
                                powerin_best = powerin;
```

```
                              loss_best = loss;
                              bestAmbienttemp = Ambienttemp;
                              calculate_deviation(powerin_best,
bestAmbienttemp, loss_best, heatcap, start, end, true);

                          }

                          if (checkBox1.Checked) loss_max = -10000;
                          if (checkBox3.Checked) powerin++;
                          if (checkBox2.Checked) XAmbienttemp = 1000000;



                      }

                  loss_min = loss_best - (1 * lossstep);
                  loss_max = loss_best + (1 * lossstep);
                  powerin_min = powerin_best - (1 * powerstep);
                  powerin_max = powerin_best + (1 * powerstep);
              }


///////////////////////////////////////////////////////////////////////////
///////////////////////////////////////
              double narrow = 1;

              loss_min = loss_best - (lossinterval / narrow);
              loss_max = loss_best + (lossinterval / narrow);
              lossinterval = lossinterval / narrow;

              powerin_min = powerin_best - (powerinterval / narrow);
              powerin_max = powerin_best + (powerinterval / narrow);
              powerinterval = powerinterval / narrow;

              if(checkBox3.Checked) break;
          }

          calculate_deviation(powerin_best, bestAmbienttemp, loss_best,
heatcap, start, end, true);

      }


        private void button2_Click(object sender, EventArgs e)
        {

          D_powerin[colnumber]     = Convert.ToDouble(textBox1.Text);
          D_loss[colnumber]       = Convert.ToDouble(textBox2.Text);
          D_AmbientTemp[colnumber] = Convert.ToDouble(textBox3.Text);

            calculate_deviation(D_powerin[colnumber],
D_AmbientTemp[colnumber], D_loss[colnumber], 3, start, end, true);


        }


        private void MyPlot()
```

```
{
    colnumber = findcol();

    loadparameter();

}


private void radioButton1_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton1.Checked)
    {
        MyPlot();

    }
}

private void radioButton2_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton2.Checked)
    {
        MyPlot();

    }
}

private void radioButton3_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton3.Checked)
    {
        MyPlot();

    }
}

private void radioButton4_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton4.Checked)
    {
        MyPlot();

    }
}

private void radioButton5_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton5.Checked)
    {
        MyPlot();

    }
}

private void radioButton6_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton6.Checked)
    {
        MyPlot();
    }
}

private void radioButton7_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton7.Checked)
    {
        MyPlot();

    }
}


private void radioButton8_CheckedChanged(object sender, EventArgs e)
{

    if (radioButton8.Checked)
    {
        MyPlot();

    }
}


private void radioButton9_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton9.Checked)
    {
        MyPlot();

    }

}

private void radioButton10_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton10.Checked)
    {
        MyPlot();

    }


}

private void radioButton11_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton11.Checked)
    {
        MyPlot();

    }


}
private void radioButton12_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton12.Checked)
    {
        MyPlot();

    }


}
private void radioButton13_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton13.Checked)
    {
        MyPlot();

    }


}
private void radioButton14_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton14.Checked)
    {
        MyPlot();

    }


}
private void radioButton15_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton15.Checked)
    {
        MyPlot();

    }


}
private void radioButton16_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton16.Checked)
    {
        MyPlot();

    }


}
private void radioButton17_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton17.Checked)
    {
        MyPlot();

    }


}
```

```
private void radioButton18_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton18.Checked)
    {
        MyPlot();

    }


}
private void radioButton19_CheckedChanged(object sender, EventArgs e)
{
    if (radioButton19.Checked)
    {
        MyPlot();

    }


}



private void hScrollBar1_Scroll(object sender, ScrollEventArgs e)
{

    chart1.ChartAreas[0].AxisX.Minimum = hScrollBar2.Value ;
    chart1.ChartAreas[0].AxisX.Maximum = hScrollBar1.Value ;

    chart1.Refresh();

}

private void hScrollBar2_Scroll(object sender, ScrollEventArgs e)
{
    chart1.ChartAreas[0].AxisX.Minimum = hScrollBar2.Value ;
    chart1.ChartAreas[0].AxisX.Maximum = hScrollBar1.Value ;

    chart1.Refresh();


}

private void vScrollBar1_Scroll(object sender, ScrollEventArgs e)
{

    chart1.ChartAreas[0].AxisY.Minimum = vScrollBar2.Value/20.0;
    chart1.ChartAreas[0].AxisY.Maximum = vScrollBar1.Value/20.0;

    chart1.Refresh();

}

private void vScrollBar2_Scroll(object sender, ScrollEventArgs e)
{


    chart1.ChartAreas[0].AxisY.Minimum = vScrollBar2.Value/20.0;
```

```
            chart1.ChartAreas[0].AxisY.Maximum = vScrollBar1.Value/20.0;

            chart1.Refresh();
        }
        private void button3_Click(object sender, EventArgs e)
        {

            chart1.ChartAreas[0].AxisY.Minimum = 1000;
            chart1.ChartAreas[0].AxisY.Maximum = 0;
            chart1.Series["Series1"].Points.Clear();
            var list = (object[,])mydata;
            int colnumber = findcol();


            int count = 0;


            for (int i = 3; list[i, colnumber] != null; i++)
            {
                count++;
                if (chart1.ChartAreas[0].AxisY.Minimum > measured[i])
chart1.ChartAreas[0].AxisY.Minimum = Math.Floor(measured[i]);
                if (chart1.ChartAreas[0].AxisY.Maximum <= measured[i])
chart1.ChartAreas[0].AxisY.Maximum = Math.Floor(measured[i]) + 1;
            }
            chart1.ChartAreas[0].AxisX.Minimum = start;
            chart1.ChartAreas[0].AxisX.Maximum = start + count;

            chart1.Series["Series1"].Points.Clear();



            for (int r = 1; r <= 100; r++)
            {
                chart1.Series["Series1"].Points.Clear();
                smooth(measured);

                for (int i = start; i <= (start + count); i++)
                {

                    chart1.Series["Series1"].Points.AddXY(i, measured[i]);
                }
                chart1.Refresh();
            }
        }

       }
}
```

B: Video to Heating Curve Converter Software

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Runtime.InteropServices;
using System.IO;
using System.Text;

namespace JockerSoft.Media
{
       /// <summary>
       /// Summary description for Form1.
       /// </summary>
       public class Form1 : System.Windows.Forms.Form
    {
              private System.Windows.Forms.PictureBox pictureBox1;
              private System.Windows.Forms.Button button5;
              private System.Windows.Forms.TrackBar trackBar1;
              private System.Windows.Forms.TextBox textBox1;
              private System.Windows.Forms.TextBox textBox2;
              private System.Windows.Forms.Label label1;
              private System.Windows.Forms.Label label2;
              private System.Windows.Forms.Button button1;
              private System.Windows.Forms.OpenFileDialog openFileDialog1;
        private System.Windows.Forms.Button button2;
        private RichTextBox richTextBox2;
        private Button button8;
        private Button button10;
        private Button button11;
        private NumericUpDown numericUpDown1;
        private NumericUpDown numericUpDown2;
        private System.Windows.Forms.DataVisualization.Charting.Chart chart1;
        private Button button3;
              /// <summary>
              /// Required designer variable.
        ///
        ///
              /// </summary>
              private System.ComponentModel.Container components = null;
        public string excellpath;
              public Form1()
              {
                    //
                    // Required for Windows Form Designer support
                    //
                    InitializeComponent();

                    //
                    // TODO: Add any constructor code after
InitializeComponent call
                    //
              }
```

```
		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		protected override void Dispose( bool disposing )
		{
			if( disposing )
			{
				if (components != null)
				{
					components.Dispose();
				}
			}
			base.Dispose( disposing );
		}

		#region Windows Form Designer generated code
		/// <summary>
		/// Required method for Designer support - do not modify
		/// the contents of this method with the code editor.
		/// </summary>
		private void InitializeComponent()
		{
            System.Windows.Forms.DataVisualization.Charting.ChartArea
chartArea2 = new System.Windows.Forms.DataVisualization.Charting.ChartArea();
            System.Windows.Forms.DataVisualization.Charting.Series series2 =
new System.Windows.Forms.DataVisualization.Charting.Series();
            this.pictureBox1 = new System.Windows.Forms.PictureBox();
            this.button5 = new System.Windows.Forms.Button();
            this.trackBar1 = new System.Windows.Forms.TrackBar();
            this.textBox1 = new System.Windows.Forms.TextBox();
            this.textBox2 = new System.Windows.Forms.TextBox();
            this.label1 = new System.Windows.Forms.Label();
            this.label2 = new System.Windows.Forms.Label();
            this.button1 = new System.Windows.Forms.Button();
            this.openFileDialog1 = new System.Windows.Forms.OpenFileDialog();
            this.button2 = new System.Windows.Forms.Button();
            this.richTextBox2 = new System.Windows.Forms.RichTextBox();
            this.button8 = new System.Windows.Forms.Button();
            this.button10 = new System.Windows.Forms.Button();
            this.button11 = new System.Windows.Forms.Button();
            this.numericUpDown1 = new System.Windows.Forms.NumericUpDown();
            this.numericUpDown2 = new System.Windows.Forms.NumericUpDown();
            this.chart1 = new
System.Windows.Forms.DataVisualization.Charting.Chart();
            this.button3 = new System.Windows.Forms.Button();

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.trackBar1)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.numericUpDown1)).BeginInit()
;

((System.ComponentModel.ISupportInitialize)(this.numericUpDown2)).BeginInit()
;

((System.ComponentModel.ISupportInitialize)(this.chart1)).BeginInit();
            this.SuspendLayout();
            //
            // pictureBox1
            //
            this.pictureBox1.Anchor =
((System.Windows.Forms.AnchorStyles)((((System.Windows.Forms.AnchorStyles.Top
| System.Windows.Forms.AnchorStyles.Bottom)
            | System.Windows.Forms.AnchorStyles.Left)
            | System.Windows.Forms.AnchorStyles.Right)));
            this.pictureBox1.BackColor =
System.Drawing.SystemColors.ControlDark;
            this.pictureBox1.Location = new System.Drawing.Point(8, 133);
            this.pictureBox1.Name = "pictureBox1";
            this.pictureBox1.Size = new System.Drawing.Size(800, 480);
            this.pictureBox1.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.CenterImage;
            this.pictureBox1.TabIndex = 4;
            this.pictureBox1.TabStop = false;
            this.pictureBox1.SizeChanged += new
System.EventHandler(this.pictureBox1_SizeChanged);
            this.pictureBox1.Click += new
System.EventHandler(this.pictureBox1_Click);
            //
            // button5
            //
            this.button5.Location = new System.Drawing.Point(241, 54);
            this.button5.Name = "button5";
            this.button5.Size = new System.Drawing.Size(69, 23);
            this.button5.TabIndex = 5;
            this.button5.Text = "excel";
            this.button5.Click += new
System.EventHandler(this.button5_Click);
            //
            // trackBar1
            //
            this.trackBar1.Anchor =
((System.Windows.Forms.AnchorStyles)(((System.Windows.Forms.AnchorStyles.Bott
om | System.Windows.Forms.AnchorStyles.Left)
            | System.Windows.Forms.AnchorStyles.Right)));
            this.trackBar1.Location = new System.Drawing.Point(8, 85);
            this.trackBar1.Maximum = 1000;
            this.trackBar1.Name = "trackBar1";
            this.trackBar1.Size = new System.Drawing.Size(65, 45);
            this.trackBar1.TabIndex = 6;
            this.trackBar1.TickFrequency = 5;
            this.trackBar1.TickStyle = System.Windows.Forms.TickStyle.Both;
            this.trackBar1.Scroll += new
System.EventHandler(this.trackBar1_Scroll);
            //
            // textBox1
            //
            this.textBox1.Location = new System.Drawing.Point(8, 28);
            this.textBox1.Name = "textBox1";
            this.textBox1.Size = new System.Drawing.Size(130, 20);
            this.textBox1.TabIndex = 7;
            this.textBox1.Text = "C:\\Users\\seco\\Desktop\\tez
data\\test\\inclined\\2022_09_17\\oil9\\4000_01.wmv";
            this.textBox1.Enter += new
System.EventHandler(this.textBox1_Enter);
```

```
            this.textBox1.MouseUp += new
System.Windows.Forms.MouseEventHandler(this.textBox1_MouseUp);
            //
            // textBox2
            //
            this.textBox2.Location = new System.Drawing.Point(161, 28);
            this.textBox2.Name = "textBox2";
            this.textBox2.Size = new System.Drawing.Size(150, 20);
            this.textBox2.TabIndex = 8;
            this.textBox2.Text = "c:\\frame.bmp";
            //
            // label1
            //
            this.label1.Location = new System.Drawing.Point(9, 9);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(100, 16);
            this.label1.TabIndex = 9;
            this.label1.Text = "video file";
            //
            // label2
            //
            this.label2.Location = new System.Drawing.Point(162, 9);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(100, 16);
            this.label2.TabIndex = 10;
            this.label2.Text = "save frame to";
            //
            // button1
            //
            this.button1.Location = new System.Drawing.Point(134, 28);
            this.button1.Name = "button1";
            this.button1.Size = new System.Drawing.Size(24, 20);
            this.button1.TabIndex = 11;
            this.button1.Text = "...";
            this.button1.Click += new
System.EventHandler(this.button1_Click);
            //
            // button2
            //
            this.button2.Location = new System.Drawing.Point(8, 54);
            this.button2.Name = "button2";
            this.button2.Size = new System.Drawing.Size(69, 23);
            this.button2.TabIndex = 12;
            this.button2.Text = "get&&show frame";
            this.button2.Click += new
System.EventHandler(this.button2_Click);
            //
            // richTextBox2
            //
            this.richTextBox2.Location = new System.Drawing.Point(317, 4);
            this.richTextBox2.Name = "richTextBox2";
            this.richTextBox2.Size = new System.Drawing.Size(241, 123);
            this.richTextBox2.TabIndex = 21;
            this.richTextBox2.Text = "";
            //
            // button8
            //
            this.button8.Location = new System.Drawing.Point(83, 54);
            this.button8.Name = "button8";
            this.button8.Size = new System.Drawing.Size(75, 23);
            this.button8.TabIndex = 22;
            this.button8.Text = "button8";
            this.button8.UseVisualStyleBackColor = true;
            this.button8.Click += new
System.EventHandler(this.button8_Click);
            //
            // button10
            //
            this.button10.Location = new System.Drawing.Point(564, 4);
            this.button10.Name = "button10";
            this.button10.Size = new System.Drawing.Size(75, 23);
            this.button10.TabIndex = 28;
            this.button10.Text = "Clear";
            this.button10.UseVisualStyleBackColor = true;
            this.button10.Click += new
System.EventHandler(this.button10_Click);
            //
            // button11
            //
            this.button11.Location = new System.Drawing.Point(564, 33);
            this.button11.Name = "button11";
            this.button11.Size = new System.Drawing.Size(75, 23);
            this.button11.TabIndex = 29;
            this.button11.Text = "Copy";
            this.button11.UseVisualStyleBackColor = true;
            this.button11.Click += new
System.EventHandler(this.button11_Click);
            //
            // numericUpDown1
            //
            this.numericUpDown1.Location = new System.Drawing.Point(564, 62);
            this.numericUpDown1.Name = "numericUpDown1";
            this.numericUpDown1.Size = new System.Drawing.Size(46, 20);
            this.numericUpDown1.TabIndex = 30;
            this.numericUpDown1.Value = new decimal(new int[] {
            20,
            0,
            0,
            0});
            //
            // numericUpDown2
            //
            this.numericUpDown2.Location = new System.Drawing.Point(564, 85);
            this.numericUpDown2.Name = "numericUpDown2";
            this.numericUpDown2.Size = new System.Drawing.Size(46, 20);
            this.numericUpDown2.TabIndex = 31;
            this.numericUpDown2.Value = new decimal(new int[] {
            28,
            0,
            0,
            0});
            //
            // chart1
            //
            chartArea2.Name = "ChartArea1";
            this.chart1.ChartAreas.Add(chartArea2);
            this.chart1.Location = new System.Drawing.Point(762, 12);
            this.chart1.Name = "chart1";
            series2.ChartArea = "ChartArea1";
            series2.ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Point;
            series2.Name = "Series1";
            this.chart1.Series.Add(series2);
            this.chart1.Size = new System.Drawing.Size(564, 571);
            this.chart1.TabIndex = 32;
            this.chart1.Text = "chart1";
            this.chart1.Click += new System.EventHandler(this.chart1_Click);
            //
            // button3
            //
            this.button3.Location = new System.Drawing.Point(666, 9);
            this.button3.Name = "button3";
            this.button3.Size = new System.Drawing.Size(75, 23);
            this.button3.TabIndex = 33;
            this.button3.Text = "button3";
            this.button3.UseVisualStyleBackColor = true;
            this.button3.Click += new
System.EventHandler(this.button3_Click);
            //
            // Form1
            //
            this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
            this.ClientSize = new System.Drawing.Size(1370, 749);
            this.Controls.Add(this.button3);
            this.Controls.Add(this.chart1);
            this.Controls.Add(this.numericUpDown2);
            this.Controls.Add(this.numericUpDown1);
            this.Controls.Add(this.button11);
            this.Controls.Add(this.button10);
            this.Controls.Add(this.button8);
            this.Controls.Add(this.richTextBox2);
            this.Controls.Add(this.button2);
            this.Controls.Add(this.button1);
            this.Controls.Add(this.label2);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.textBox2);
            this.Controls.Add(this.textBox1);
            this.Controls.Add(this.button5);
            this.Controls.Add(this.pictureBox1);
            this.Controls.Add(this.trackBar1);
            this.Name = "Form1";
            this.Text = "Form1";
            this.Load += new System.EventHandler(this.Form1_Load);

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.trackBar1)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.numericUpDown1)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.numericUpDown2)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.chart1)).EndInit();
            this.ResumeLayout(false);
            this.PerformLayout();

              }
              #endregion

              /// <summary>
              /// The main entry point for the application.
              /// </summary>
              [STAThread]
              static void Main()
              {
                    Application.Run(new Form1());
              }
        int XforGradient = 688;
        int MyGX = 291;
        int MyGY = 319;

        int count = 0;
        string datafile;
        double[,] mycoldata1 = new double[10000, 1];
        double[,] mycoldata2 = new double[10000, 1];

        object mydata;


        private void GetGradient()
        {
            //refresh the picture box
            pictureBox1.Refresh();
            //create a new Bitmap object
            Bitmap map = (Bitmap)pictureBox1.Image;
            //create a graphics object
            Graphics g = Graphics.FromImage(map);
            //create a pen object and setting the color and width for the pen
            Pen p = new Pen(Color.Black, 1);

            //draw line between  point p1 and p2

            int GradientTop = 0;
            int GradientBottom = 0;


            Color PixelColor;
            for (int i = 1; i < map.Height; i++)
            {

                PixelColor = map.GetPixel(XforGradient, i);


                if (PixelColor.R == 0 && PixelColor.G == 0 && PixelColor.B ==
0)
                {
                    if (GradientTop == 0)
                    {
                        GradientTop = i;
                    }
                    else
                    {
                        if (GradientBottom == 0)
                        {
                            GradientBottom = i;
                        }
                    }
                }

            }

        }



        private double pixtotemp(int pixel, double smintemp, double smaxtemp)
        {
            double temp = -300;
            double gradient_heighet = (pictureBox1.Height * 202.0 / 240.0);
            double deltatemp = smaxtemp - smintemp;


            temp = deltatemp * (gradient_heighet - (pixel -
(pictureBox1.Height * 18.0 / 240.0))) / gradient_heighet;
            temp += smintemp;

            return temp;
        }



        private double searchcolor(int time, int x, int y)
        {
            pictureBox1.Refresh();
            Bitmap map = (Bitmap)pictureBox1.Image;
            Graphics g = Graphics.FromImage(map);
            Pen p = new Pen(Color.Black, 1);

            double subjectred = 0;
            double subjectgreen = 0;
            double subjectblue = 0;


            Color SubjectPixCol;
            int RED_Total = 0;
            int GREEN_Total = 0;
            int BLUE_Total = 0;


            for (int ax = -1; ax <= 1; ax++)
                for (int ay = -1; ay <= 1; ay++)
                {
                    SubjectPixCol = map.GetPixel(x + ax, y + ay);

                    RED_Total += SubjectPixCol.R;
                    GREEN_Total += SubjectPixCol.G;
                    BLUE_Total += SubjectPixCol.B;
                }

            SubjectPixCol = Color.FromArgb(RED_Total/9 , GREEN_Total/9,
BLUE_Total/9);
            subjectred   = RED_Total / 9.0;
            subjectblue  = BLUE_Total / 9.0;
            subjectgreen = GREEN_Total / 9.0;


            for (int ax = -2; ax <= 2; ax++)
                for (int ay = -2; ay <= 2; ay++)
                {

                    g.DrawLine(p, new Point(x + ax, y + ay), new Point(x + ax
+ 1, y + ay));
                }

            Color PixelColor;






            double min = 100000;
            int min_i= -1;
            double currentdist = 0;
            for (int i = pictureBox1.Height*18 / 240 ; i <
pictureBox1.Height*220 /240; i++)
            {

                PixelColor = map.GetPixel(XforGradient, i);


                currentdist = ((PixelColor.R - subjectred) * (PixelColor.R -
subjectred)) +
                               ((PixelColor.G - subjectgreen) * (PixelColor.G
- subjectgreen)) +
                               (2*(PixelColor.B - subjectblue) *
(PixelColor.B - subjectblue)) ;



                if (min > currentdist)
                {
                    min_i = i;
                    min = currentdist;
                }

            }


                g.DrawLine(p, new Point(XforGradient - 10, min_i), new
Point(XforGradient + 10, min_i));

                double smaxtemp, smintemp;
                smintemp = decimal.ToDouble(numericUpDown1.Value);
                smaxtemp = decimal.ToDouble(numericUpDown2.Value);
                double temp = pixtotemp(min_i, smintemp, smaxtemp);

                richTextBox2.AppendText(time + "\t" + temp.ToString() +
"\n");

            pictureBox1.Image = map;
            //dispose pen and graphics object
            p.Dispose();
            g.Dispose();

            return temp;
        }

        private int bw_color(int x, int y)
        {
            //refresh the picture box
            pictureBox1.Refresh();
            //create a new Bitmap object
            Bitmap map = (Bitmap)pictureBox1.Image;
            //create a graphics object

            Color SubjectPixCol;

            SubjectPixCol = map.GetPixel(x, y );

            Color PixelColor;
            int min = 100000;
            int min_i = -1;
            int currentdist = 0;
            for (int i = pictureBox1.Height * 18 / 240; i <
pictureBox1.Height * 220 / 240; i++)
            {

                PixelColor = map.GetPixel(XforGradient, i);

                currentdist = ((PixelColor.R - SubjectPixCol.R) *
(PixelColor.R - SubjectPixCol.R)) +
                              ((PixelColor.G - SubjectPixCol.G) *
(PixelColor.G - SubjectPixCol.G)) +
                              ((PixelColor.B - SubjectPixCol.B) *
(PixelColor.B - SubjectPixCol.B));



                if (min > currentdist)
                {
                    min_i = i;
                    min = currentdist;
                }

            }



            int bwc;
            bwc = (min_i - 35) *10/16;
            bwc = 255 - bwc;
            return bwc;

        }


              private void button2_Click(object sender, System.EventArgs e)
              {
            pictureBox1.Height = 480;
            pictureBox1.Width = 800;

                    try
                    {
                           this.pictureBox1.Image =
FrameGrabber.GetFrameFromVideo(this.textBox1.Text, 0.2d, new
Size(pictureBox1.Width, pictureBox1.Height));
                // this.pictureBox1.Image =
FrameGrabber.MyGetFrameFromVideo(this.textBox1.Text, 100, new
Size(pictureBox1.Width, pictureBox1.Height));
                    }
                    catch (JockerSoft.Media.InvalidVideoFileException ex)
                    {
                           MessageBox.Show(ex.Message, "Extraction failed");
                    }
              }


        private object GetFromExcel(string sheetName, string c1, string c2)
        {
            Microsoft.Office.Interop.Excel.Application oXL = null;

            Microsoft.Office.Interop.Excel._Workbook oWB = null;

            Microsoft.Office.Interop.Excel._Worksheet oSheet = null;

            Microsoft.Office.Interop.Excel.Range range = null;

            object value = null;
            try
            {
                oXL = new Microsoft.Office.Interop.Excel.Application();

                oWB = oXL.Workbooks.Open(excellpath+"\\MyExcel.xlsx");

                oSheet = String.IsNullOrEmpty(sheetName) ?
(Microsoft.Office.Interop.Excel._Worksheet)oWB.ActiveSheet :
(Microsoft.Office.Interop.Excel._Worksheet)oWB.Worksheets[sheetName];

                //data = double.oSheet.get_Range(c1, c2).Value2.ToString() ;
                //values = oSheet.get_Range(c1, c2).Value2.ToString();
                //object[,] values1 =
(object[,])oWB.Worksheets["sheet1"].Cells["A1:J10"];
                range = oSheet.get_Range(c1, c2);

                value = range.Value;

                MessageBox.Show("Get Done!");
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.ToString());
            }
            finally
            {
                if (oWB != null)
                    oWB.Close();
            }
            return value;
        }


        private void UpdateExcel(string sheetName, int row, int col, string
data)
        {
            Microsoft.Office.Interop.Excel.Application oXL = null;

            Microsoft.Office.Interop.Excel._Workbook oWB = null;

            Microsoft.Office.Interop.Excel._Worksheet oSheet = null;

            try
            {
                oXL = new Microsoft.Office.Interop.Excel.Application();

                oWB = oXL.Workbooks.Open(excellpath + "\\MyExcel.xlsx");

                oSheet = String.IsNullOrEmpty(sheetName) ?
(Microsoft.Office.Interop.Excel._Worksheet)oWB.ActiveSheet :
(Microsoft.Office.Interop.Excel._Worksheet)oWB.Worksheets[sheetName];

                oSheet.Cells[row, col] = data;

                oWB.Save();

                MessageBox.Show("Done!");
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.ToString());
            }
            finally
            {
                if (oWB != null)
                    oWB.Close();
            }
        }

        private void UpdateExcel1(string sheetName, string c1, string c2,
double[,] data)
        {
            Microsoft.Office.Interop.Excel.Application oXL = null;

            Microsoft.Office.Interop.Excel._Workbook oWB = null;

            Microsoft.Office.Interop.Excel._Worksheet oSheet = null;
            try
            {
                oXL = new Microsoft.Office.Interop.Excel.Application();

                oWB = oXL.Workbooks.Open(excellpath + "\\MyExcel.xlsx");

                oSheet = String.IsNullOrEmpty(sheetName) ?
(Microsoft.Office.Interop.Excel._Worksheet)oWB.ActiveSheet :
(Microsoft.Office.Interop.Excel._Worksheet)oWB.Worksheets[sheetName];

                oSheet.get_Range(c1, c2).Value2 = data;

                oWB.Save();

                MessageBox.Show("Done!");
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.ToString());
            }
            finally
            {
                if (oWB != null)
                    oWB.Close();
            }
        }

             private void button5_Click(object sender, System.EventArgs e)
             {


            string letters =  "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

            int lastcolid = 0;

           mydata=GetFromExcel("Sheet1","A1","GZ2000");

           UpdateExcel1("Sheet1","A2","A"+(count+2).ToString(),mycoldata1);

           var list = (object[,])mydata;

            for(int i = 2;i<=400;i++)
                if (list[1,i] == null)
                {
                    lastcolid = i-1;
                    break;
                }


            string c1 = letters[lastcolid] + "2";

            string c2 = letters[lastcolid] + (count+2).ToString();

           UpdateExcel1("Sheet1", c1, c2, mycoldata2);

           UpdateExcel("Sheet1",1, lastcolid+1, datafile);
        }

                private void trackBar1_Scroll(object sender, System.EventArgs e)
                {
                        try
                        {
                  this.pictureBox1.Image =
FrameGrabber.GetFrameFromVideo(this.textBox1.Text,
((double)this.trackBar1.Value) / this.trackBar1.Maximum, new
Size(pictureBox1.Width, pictureBox1.Height));
                        }
                        catch (JockerSoft.Media.InvalidVideoFileException ex)
                        {
                                MessageBox.Show(ex.Message, "Extraction failed");
                        }
                }

                private void textBox1_Enter(object sender, System.EventArgs e)
                {
                        this.textBox1.SelectAll();
                }

                private void textBox1_MouseUp(object sender,
System.Windows.Forms.MouseEventArgs e)
                {
                        this.textBox1.SelectAll();
                }

                private void button1_Click(object sender, System.EventArgs e)
                {
                        if (this.openFileDialog1.ShowDialog() == DialogResult.OK)
                                this.textBox1.Text =
this.openFileDialog1.FileName;
                }


        private void pictureBox1_Click(object sender, EventArgs e)
        {
            if (e is MouseEventArgs)
            {
                var mea = e as MouseEventArgs;
                MyGX = mea.X;
                MyGY = mea.Y;

                searchcolor(0,MyGX, MyGY);
            }

        }

        private void button8_Click(object sender, EventArgs e)
        {
            chart1.ChartAreas[0].AxisY.Minimum = 1000;

            chart1.ChartAreas[0].AxisY.Maximum = 0;

            excellpath = Path.GetDirectoryName(this.textBox1.Text);

            richTextBox2.Clear();
            double strl =
FrameGrabber.MyGetstreamLengthFromVideo(this.textBox1.Text, 540);

            datafile = Path.GetFileName(this.textBox1.Text);

            chart1.Series["Series1"].Points.Clear();
            double temp;

            count = 0;

            for (int i = 1; i < strl; i += 1)
            {
                count++;
                try
                {
                    this.pictureBox1.Image =
FrameGrabber.MyGetFrameFromVideo(this.textBox1.Text, i, new
Size(pictureBox1.Width, pictureBox1.Height));
                }
                catch (JockerSoft.Media.InvalidVideoFileException ex)
                {
                    MessageBox.Show(ex.Message, "Extraction failed");
                }
                catch (StackOverflowException)
                {
                    MessageBox.Show("The target image size is too big",
"Extraction failed");
                }


                    temp = searchcolor(i, MyGX, MyGY);

                    if (chart1.ChartAreas[0].AxisY.Minimum > temp)
chart1.ChartAreas[0].AxisY.Minimum = Math.Floor(temp);

                    if (chart1.ChartAreas[0].AxisY.Maximum < temp)
chart1.ChartAreas[0].AxisY.Maximum = Math.Floor(temp)+1;


                    mycoldata1[count,0] = i;

                    mycoldata2[count,0] = temp;


                    chart1.Series["Series1"].Points.AddXY(i, temp);
                    chart1.Refresh();

            }

        }


        private void Form1_Load(object sender, EventArgs e)
        {

        }
        private void pictureBox1_SizeChanged(object sender, EventArgs e)
        {
            XforGradient = (pictureBox1.Width * 344) / 400;
        }

        private void button10_Click(object sender, EventArgs e)
        {
            richTextBox2.Clear();
        }

        private void button11_Click(object sender, EventArgs e)
        {

            richTextBox2.SelectAll();

            richTextBox2.Copy();
        }

        private void chart1_Click(object sender, EventArgs e)
        {


        }

        private void button3_Click(object sender, EventArgs e)
        {
            excellpath = Path.GetDirectoryName(this.textBox1.Text);

            Test.Form2 f2 = new Test.Form2(excellpath);

            f2.Show();

        }

    }
}
```

C: Code of the Computer Side Software for Function Generator

```
//Here are some useful articles when creating new PC applications for COM
ports:
//(links valid as of June 3, 2008)
//
//"SerialPort Class"
//http://msdn.microsoft.com/en-us/library/system.io.ports.serialport.aspx
//
//"SerialPort Members"
//http://msdn.microsoft.com/en-
us/library/system.io.ports.serialport_members.aspx
//
//"SerialPort.DataReceived Event"
//http://msdn.microsoft.com/en-
us/library/system.io.ports.serialport.datareceived.aspx
//
//"How to: Make Thread-Safe Calls to Windows Forms Controls"
//http://msdn.microsoft.com/en-us/library/ms171728(VS.80).aspx
//

using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO.Ports;

namespace Csharp_Simple_CDC_Demo
{
    public partial class Form1 : Form
    {
        //Create a delegate function for this thread that will take
        //  in a string and will write it to the txtDataReceived textbox
        delegate void SetTextCallback(string text);


/*****************************************************************************
          Function:
            public Form1()

          Summary:
            The main contructor for the Form1 class.

          Description:
            The main contructor for the Form1 class.  This function creates
and
            initializes all of the form objects.

          Precondition:
            None

          Parameters:
            None

          Return Values:
            None

          Remarks:
            None

***************************************************************************/
        public Form1()
        {
            InitializeComponent();

            //Initialize the COM ports list
            UpdateCOMPortList();
        }


/****************************************************************************
          Function:
            private void UpdateCOMPortList()

          Summary:
            This function updates the COM ports listbox.

          Description:
            This function updates the COM ports listbox.  This function is
launched
            periodically based on its Interval attribute (set in the form
editor under
            the properties window).

          Precondition:
            None

          Parameters:
            None

          Return Values:
            None

          Remarks:
            None

***************************************************************************/
        private void UpdateCOMPortList()
        {
            int i;
            bool foundDifference;

            i = 0;
            foundDifference = false;

            //If the number of COM ports is different than the last time we
            //  checked, then we know that the COM ports have changed and we
            //  don't need to verify each entry.
            if (lstCOMPorts.Items.Count == SerialPort.GetPortNames().Length)
            {
                //Search the entire SerialPort object.  Look at COM port name
                //  returned and see if it already exists in the list.
                foreach (string s in SerialPort.GetPortNames())
                {
                    //If any of the names have changed then we need to update
                    //  the list
                    if (lstCOMPorts.Items[i++].Equals(s) == false)
                    {
                        foundDifference = true;
                    }
                }
            }
            else
            {
                foundDifference = true;
            }

            //If nothing has changed, exit the function.
            if (foundDifference == false)
            {
                return;
            }

            //If something has changed, then clear the list
            lstCOMPorts.Items.Clear();

            //Add all of the current COM ports to the list
            foreach (string s in SerialPort.GetPortNames())
            {
                lstCOMPorts.Items.Add(s);
            }
            //Set the listbox to point to the first entry in the list
            lstCOMPorts.SelectedIndex = 0;
        }


/****************************************************************************
          Function:
            private void timer1_Tick(object sender, EventArgs e)

          Summary:
            This function updates the COM ports listbox.

          Description:
            This function updates the COM ports listbox.  This function is 
launched
            periodically based on its Interval attribute (set in the form 
editor under
            the properties window).

          Precondition:
            None

          Parameters:
            object sender     - Sender of the event (this form)
            EventArgs e       - The event arguments

          Return Values:
            None

          Remarks:
            None

***************************************************************************/
        private void timer1_Tick(object sender, System.EventArgs e)
        {
            //Update the COM ports list so that we can detect
            //  new COM ports that have been added.
            UpdateCOMPortList();
        }



/****************************************************************************
          Function:
            private void btnConnect_Click(object sender, EventArgs e)

          Summary:
            This function opens the COM port.

          Description:
            This function opens the COM port.  This function is launched when
the
            btnConnect button is clicked.  In addition to opening the COM
port, this
            function will also change the Enable attribute of several of the
form
            objects to disable the user from opening a new COM port.

          Precondition:
            None

          Parameters:
            object sender     - Sender of the event (this form)
            EventArgs e       - The event arguments

***************************************************************************/
        private void btnConnect_Click(object sender, System.EventArgs e)
        {
            //This section of code will try to open the COM port.
            //  Please note that it is important to use a try/catch
            //  statement when opening the COM port.  If a USB virtual
            //  COM port is removed and the PC software tries to open
            //  the COM port before it detects its removal then
            //  an exeception is thrown.  If the execption is not in a
            //  try/catch statement this could result in the application
            //  crashing.
            try
            {
                //Get the port name from the application list box.
                //  the PortName attribute is a string name of the COM
                //  port (e.g. - "COM1").
                serialPort1.PortName =
lstCOMPorts.Items[lstCOMPorts.SelectedIndex].ToString();

                //Open the COM port.
                serialPort1.Open();

                //Change the state of the application objects
                btnConnect.Enabled = false;
                lstCOMPorts.Enabled = false;
                btnClose.Enabled = true;

                //Clear the textbox and print that we are connected.
                txtDataReceived.Clear();
                txtDataReceived.AppendText("Connected.\r\n");
            }
            catch
            {
                //If there was an exception, then close the handle to
                //  the device and assume that the device was removed
                btnClose_Click(this, null);
            }
        }


/****************************************************************************
          Function:
            private void btnClose_Click(object sender, EventArgs e)

          Summary:
            This function closes the COM port.

          Description:
            This function closes the COM port.  This function is launched
when the
            btnClose button is clicked.  This function can also be called
directly
            from other functions.  In addition to closing the COM port, this
            function will also change the Enable attribute of several of the
form
            objects to enable the user to open a new COM port.

          Precondition:
            None

          Parameters:
            object sender     - Sender of the event (this form)
            EventArgs e       - The event arguments


***************************************************************************/
        private void btnClose_Click(object sender, System.EventArgs e)
        {
            //Reset the state of the application objects
            btnClose.Enabled = false;
            btnConnect.Enabled = true;
            lstCOMPorts.Enabled = true;

            //This section of code will try to close the COM port.
            //  Please note that it is important to use a try/catch
            //  statement when closing the COM port.  If a USB virtual
            //  COM port is removed and the PC software tries to close
            //  the COM port before it detects its removal then
            //  an exeception is thrown.  If the execption is not in a
            //  try/catch statement this could result in the application
            //  crashing.
            try
            {
                //Dispose the In and Out buffers;
                serialPort1.DiscardInBuffer();
                serialPort1.DiscardOutBuffer();

                //Close the COM port
                serialPort1.Close();
            }
            //If there was an exeception then there isn't much we can
            //  do.  The port is no longer available.
            catch { }
        }


/************************************************************************
          Function:
            private void serialPort1_DataReceived(  object sender,

SerialDataReceivedEventArgs e)

          Summary:
            This function prints any data received on the COM port.

          Description:
            This function is called when the data is received on the COM
port.  This
            function attempts to write that data to the txtDataReceived
textbox.  If
            an exception occurs the btnClose_Click() function is called in
order to
            close the COM port that caused the exception.

          Precondition:
            None

          Parameters:
            object sender     - Sender of the event (this form)
            SerialDataReceivedEventArgs e       - The event arguments

***********************************************************************/
        private void serialPort1_DataReceived(object sender,
SerialDataReceivedEventArgs e)
        {
            //The ReadExisting() function will read all of the data that
            //  is currently available in the COM port buffer.  In this
            //  example we are sending all of the available COM port data
            //  to the SetText() function.
            //
            //  NOTE: the <SerialPort>_DataReceived() function is launched
            //  in a seperate thread from the rest of the application.  A
            //  delegate function is required in order to properly access
            //  any managed objects inside of the other thread.  Since we
            //  will be writing to a textBox (a managed object) the delegate
            //  function is required.  Please see the SetText() function for
            //  more information about delegate functions and how to use
them.
            try
            {
                SetText("\r\nReceived: ");
                SetText(serialPort1.ReadExisting());
            }
            catch
            {
                //If there was an exception, then close the handle to
                //  the device and assume that the device was removed
                btnClose_Click(this, null);
            }
        }


/*****************************************************************************
          Function:
            private void SetText(string text)

          Summary:
            This function prints the input text to the txtDataReceived
textbox.

          Description:
            This function prints the input text to the txtDataReceived
textbox.  If
            the calling thread is the same as the thread that owns the
textbox, then
            the AppendText() method is called directly.  If a thread other
than the
            main thread calls this function, then an instance of the delegate
function
            is created so that the function runs again in the main thread.

          Precondition:
            None

          Parameters:
            string text     - Text that needs to be printed to the textbox


****************************************************************************/
        private void SetText(string text)
        {
            //InvokeRequired required compares the thread ID of the
            //  calling thread to the thread ID of the creating thread.
            //  If these threads are different, it returns true.  We can
            //  use this attribute to determine if we can append text
            //  directly to the textbox or if we must launch an a delegate
            //  function instance to write to the textbox.
            if (txtDataReceived.InvokeRequired)
            {
                //InvokeRequired returned TRUE meaning that this function
                //  was called from a thread different than the current
                //  thread.  We must launch a deleage function.

                //Create an instance of the SetTextCallback delegate and
                //  assign the delegate function to be this function.  This
                //  effectively causes this same SetText() function to be
                //  called within the main thread instead of the second
                //  thread.
                SetTextCallback d = new SetTextCallback(SetText);

                //Invoke the new delegate sending the same text to the
                //  delegate that was passed into this function from the
                //  other thread.
                Invoke(d, new object[] { text });
            }
            else
            {
                //If this function was called from the same thread that
                //  holds the required objects then just add the text.
                txtDataReceived.AppendText(text);
            }
        }


/****************************************************************************
          Function:
            private void btnSendData_Click(object sender, EventArgs e)

          Summary:
            This function will attempt to send the contents of txtData over
the COM port

          Description:
            This function is called when the btnSendData button is clicked.
It will
            attempt to send the contents of txtData over the COM port.  If
the attempt
            is unsuccessful this function will call the btnClose_Click() in
order to
            close the COM port that just failed.

          Precondition:
            None

          Parameters:
            object sender     - Sender of the event (this form)
            EventArgs e       - The event arguments

***************************************************************************/
        private void btnSendData_Click(object sender, System.EventArgs e)
        {
            //This section of code will try to write to the COM port.
            //  Please note that it is important to use a try/catch
            //  statement when writing to the COM port.  If a USB virtual
            //  COM port is removed and the PC software tries to write
            //  to the COM port before it detects its removal then
            //  an exeception is thrown.  If the execption is not in a
            //  try/catch statement this could result in the application
            //  crashing.
            try
            {
                //Write the data in the text box to the open serial port
                serialPort1.Write("Start");
            }
            catch
```

```
    {
        //If there was an exception, then close the handle to
        //  the device and assume that the device was removed
        btnClose_Click(this, null);
    }
}

private void button1_Click(object sender, System.EventArgs e)
{
    //This section of code will try to write to the COM port.
    //  Please note that it is important to use a try/catch
    //  statement when writing to the COM port.  If a USB virtual
    //  COM port is removed and the PC software tries to write
    //  to the COM port before it detects its removal then
    //  an exeception is thrown.  If the execption is not in a
    //  try/catch statement this could result in the application
    //  crashing.
    try
    {
        //Write the data in the text box to the open serial port
        serialPort1.Write("Stop");
    }
    catch
    {
        //If there was an exception, then close the handle to
        //  the device and assume that the device was removed
        btnClose_Click(this, null);
    }
}

private void button2_Click(object sender, System.EventArgs e)
{
    try
    {

        serialPort1.Write(txtData.Text);
    }
    catch
    {
        btnClose_Click(this, null);
    }
}

private void txtData_TextChanged(object sender, System.EventArgs e)
{

}

private void button3_Click(object sender, System.EventArgs e)
{
    try
    {

        serialPort1.Write("AA005000" );
    }
    catch
    {
        btnClose_Click(this, null);
```

```
            }
        }

        private void numericUpDown1_ValueChanged(object sender,
System.EventArgs e)
        {
            try
            {
                string data="";
                string value;
                value = numericUpDown1.Value.ToString();


                if (value.Length == 3) data = "CC000" + value;
                if (value.Length == 4) data = "CC00" + value;
                if (value.Length == 5) data = "CC0" + value;
                serialPort1.Write(data);

                value = numericUpDown2.Value.ToString();
                if (value.Length == 3) data = "BB0000" + value;
                if (value.Length == 4) data = "BB000" + value;
                if (value.Length == 5) data = "BB00" + value;
                serialPort1.Write(data);

            }
            catch { }
        }

        private void numericUpDown2_ValueChanged(object sender,
System.EventArgs e)
        {
            try
            {
                string data = "";
                string value;
                value = numericUpDown1.Value.ToString();


                if (value.Length == 3) data = "CC000" + value;
                if (value.Length == 4) data = "CC00" + value;
                if (value.Length == 5) data = "CC0" + value;
                serialPort1.Write(data);

                value = numericUpDown2.Value.ToString();
                if (value.Length == 3) data = "BB0000" + value;
                if (value.Length == 4) data = "BB000" + value;
                if (value.Length == 5) data = "BB00" + value;
                serialPort1.Write(data);

            }
            catch { }
        }
    }
}
```

D: Code of the Microcontroler Side Software for Function Generator

```
/* USER CODE BEGIN Header */
/**
********************************************************************
**********
  * @file           : main.c
  * @brief          : Main program body

********************************************************************
***********
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2020 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under Ultimate Liberty license
  * SLA0044, the "License"; You may not use this file except in compliance with
  * the License. You may obtain a copy of the License at:
  *                             www.st.com/SLA0044
  *
********************************************************************
**********
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "usb_device.h"
```

```
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "usbd_cdc_if.h"
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */
/* USER CODE END PTD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
/* USER CODE END PM */
/* Private variables ---------------------------------------------------------*/
TIM_HandleTypeDef htim1;
/* USER CODE BEGIN PV */
char *SendBuffer;
uint8_t COMMAND_T[20] = "ABCDEFGHIJK";
uint8_t COMMAND_V[20] = "0123456789";
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_TIM1_Init(void);
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */
/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */
  /* USER CODE END 1 */
  /* MCU Configuration--------------------------------------------------------*/
  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();
  /* USER CODE BEGIN Init */
  /* USER CODE END Init */
  /* Configure the system clock */
  SystemClock_Config();
  /* USER CODE BEGIN SysInit */
  /* USER CODE END SysInit */
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USB_DEVICE_Init();
  MX_TIM1_Init();
  /* USER CODE BEGIN 2 */
  HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
  HAL_TIMEx_PWMN_Start(&htim1, TIM_CHANNEL_1);
  /* USER CODE END 2 */
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */
    /* USER CODE BEGIN 3 */
			HAL_GPIO_TogglePin(GPIOC,GPIO_PIN_13);
			HAL_Delay(1000);
  }
  /* USER CODE END 3 */
}
/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  /** Configure the main internal regulator output voltage
  */
  __HAL_RCC_PWR_CLK_ENABLE();
__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE
_SCALE2);
  /** Initializes the CPU, AHB and APB busses clocks
  */
  RCC_OscInitStruct.OscillatorType =
RCC_OSCILLATORTYPE_HSI|RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue =
RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLM = 15;
  RCC_OscInitStruct.PLL.PLLN = 144;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  RCC_OscInitStruct.PLL.PLLQ = 5;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB busses clocks
  */
  RCC_ClkInitStruct.ClockType =
RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_HSI;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) != HAL_OK)
  {
    Error_Handler();
  }
}
/**
  * @brief TIM1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM1_Init(void)
{
  /* USER CODE BEGIN TIM1_Init 0 */
  /* USER CODE END TIM1_Init 0 */
  TIM_MasterConfigTypeDef sMasterConfig = {0};
  TIM_OC_InitTypeDef sConfigOC = {0};
  TIM_BreakDeadTimeConfigTypeDef sBreakDeadTimeConfig = {0};

  /* USER CODE BEGIN TIM1_Init 1 */

  /* USER CODE END TIM1_Init 1 */
  htim1.Instance = TIM1;
  htim1.Init.Prescaler = 0;
  htim1.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim1.Init.Period = 0;
  htim1.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim1.Init.RepetitionCounter = 0;
  htim1.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_PWM_Init(&htim1) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim1, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sConfigOC.OCMode = TIM_OCMODE_PWM1;
  sConfigOC.Pulse = 0;
  sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
  sConfigOC.OCNPolarity = TIM_OCNPOLARITY_HIGH;
  sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
  sConfigOC.OCIdleState = TIM_OCIDLESTATE_RESET;
  sConfigOC.OCNIdleState = TIM_OCNIDLESTATE_RESET;
  if (HAL_TIM_PWM_ConfigChannel(&htim1, &sConfigOC,
TIM_CHANNEL_1) != HAL_OK)
  {
    Error_Handler();
  }
  sBreakDeadTimeConfig.OffStateRunMode = TIM_OSSR_DISABLE;
  sBreakDeadTimeConfig.OffStateIDLEMode = TIM_OSSI_DISABLE;
  sBreakDeadTimeConfig.LockLevel = TIM_LOCKLEVEL_OFF;
  sBreakDeadTimeConfig.DeadTime = 0;
  sBreakDeadTimeConfig.BreakState = TIM_BREAK_DISABLE;
  sBreakDeadTimeConfig.BreakPolarity = TIM_BREAKPOLARITY_HIGH;
  sBreakDeadTimeConfig.AutomaticOutput =
TIM_AUTOMATICOUTPUT_DISABLE;
  if (HAL_TIMEx_ConfigBreakDeadTime(&htim1, &sBreakDeadTimeConfig) !=
HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM1_Init 2 */

  /* USER CODE END TIM1_Init 2 */
  HAL_TIM_MspPostInit(&htim1);
}
/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_RESET);

  /*Configure GPIO pin : PC13 */
  GPIO_InitStruct.Pin = GPIO_PIN_13;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);


}
/* USER CODE BEGIN 4 */
int POW10(int n)
{
        int result = 1 ;
        for (int i = 0; i<n; i++)
        {
                result = result * 10;
        }
        return result;
}
void ExecCommand(int CT, int CV)
{
        if ( CT == 0 )TIM1->CCR1 = CV;
        if ( CT == 1 )TIM1->BDTR=0x4400+CV;
        if ( CT == 2 ){TIM1->ARR = CV;TIM1->CCR1 = CV/2;}

        HAL_GPIO_TogglePin(GPIOC,GPIO_PIN_13);
}

int ToCommand(uint8_t *buf)
{
        int CommandType = 0;
        int CommandValue = 0;

  if(buf[0] == COMMAND_T[0])  CommandType = 0;
        if(buf[0] == COMMAND_T[1])  CommandType = 1;
        if(buf[0] == COMMAND_T[2])  CommandType = 2;
        if(buf[0] == COMMAND_T[3])  CommandType = 3;
        if(buf[0] == COMMAND_T[4])  CommandType = 4;

        if(buf[0] == COMMAND_T[5])  CommandType = 5;

        if(buf[0] == COMMAND_T[6])  CommandType = 6;

        if(buf[0] == COMMAND_T[7])  CommandType = 7;


        for(int i = 1 ; i<=6; i++ )

        {

    if(buf[i] == COMMAND_V[1])  CommandValue += POW10(6-i);

    if(buf[i] == COMMAND_V[2])  CommandValue += 2*POW10(6-i);

    if(buf[i] == COMMAND_V[3])  CommandValue += 3*POW10(6-i);

    if(buf[i] == COMMAND_V[4])  CommandValue += 4*POW10(6-i);

    if(buf[i] == COMMAND_V[5])  CommandValue += 5*POW10(6-i);

    if(buf[i] == COMMAND_V[6])  CommandValue += 6*POW10(6-i);

    if(buf[i] == COMMAND_V[7])  CommandValue += 7*POW10(6-i);

    if(buf[i] == COMMAND_V[8])  CommandValue += 8*POW10(6-i);

    if(buf[i] == COMMAND_V[9])  CommandValue += 9*POW10(6-i);


        }


        ExecCommand(CommandType,CommandValue);

        return CommandValue;

}

char *MySprintf(char * out, char * inp)

{

        out ="BBB";
        int a = strlen(out);
        for (int i = 1; i<=strlen(inp);i++ )
    out[a+i-1] = inp[i];
        return out;
}
void CDC_ReceiveCallBack(uint8_t *buf, uint32_t len)
{
        //char* sss;
        //char dddd[40] = "AAAA";
        uint8_t myC[20] = "10";
        uint8_t myBuf[20] = "Hello World! \r\n";

        if (buf[1] == myC[1])
                CDC_Transmit_FS(myBuf,strlen((char* )myBuf));
        else
          CDC_Transmit_FS(buf, len);
         //HAL_Delay(200);
        // uint8_t *seco;
         //SendBuffer= "123456789";
//      char * alo = "sdfasdfasdf";
//      sss = MySprintf(dddd,alo);
  //sprintf(SendBuffer,"ALO%s",alo);
        //strcpy (SendBuffer, "FF-- \r\n");
         //*seco = *SendBuffer;
         int cc = ToCommand(buf);
		// seco = (uint8_t*)(sss);
		//CDC_Transmit_FS(seco, strlen((char *) seco));
}
/* USER CODE END 4 */
/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state */
  /* USER CODE END Error_Handler_Debug */
}
#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

CURRICULUM VITAE

Surname, Name : Öztürk, Seçkin

EDUCATION

Degree	**Institution**	**Year of Graduation**
MS	METU Physics	2004
BS	METU Micro and Nanotechnology	2010
High School	Adana Anatolian High School, Adana	1999

WORK EXPERIENCE

Year	**Place**
2005-	METU Central Laboratory

FOREIGN LANGUAGES

Advanced English

PUBLICATIONS

1. Kantarcioglu M., Yildirim G., Akpinar Oktar P., Yanbakan S., Ozer Z.B., Yurtsever Sarica D., Tasdelen S., Bayrak E., Akin Bali D.F., Ozturk S., Akcali K.C., Ezer U., Kurekci A.E. "Coffee-Derived Exosome-Like Nanoparticles: Are They the Secret Heroes? ", Turk J Gastroenterol, 34(2), 161-169 (2023)

2. Ozturk Koc S., Galioglu S., Ozturk S., Akata Kurc B., Koc E., Salamov B.G. "New Mixed Conductivity Mechanisms in the Cold Plasma Device Based on Silver-Modified Zeolite Microporous Electronic Materials", Journal of Electronic Materials, 47, 2791-2799 (2018)

3. Soldatkin O.O., Kucherenko I.S., Shelyakina M.S., Soy E., Kirdeciler K., Ozturk S., Jaffrezic-Renault N., Akata Kurc B., Dzyadevych S.V., Soldatkin A.P. "Application of Different Zeolites for Improvement of the Characteristics of a pH-FET Biosensor Based on Immobilized Urease", Electroanalysis, 25(2), 468-474 (2013)

4. Kucherenko I.S., Soldatkin O.O, Soy E., Kirdeciler K., Ozturk S., Akata Kurc B., Jaffrezic-Renault N., Soldatkin A.P, Dzyadevych S.V. "Effect of Different Modifications of BEA-Zeolites on Operational Characteristics of Conductometric Biosensor", Materials Science and Engineering:C, 32(6), 1648-1653 (2012)

5. Kucherenko I.S., Soldatkin O.O, Kasap B.O., Ozturk S., Akata Kurc B., Soldatkin A.P, Dzyadevych S.V. "Elaboration of Urease Adsorption on Silicalite for Biosensor Creation", Electroanalysis, 24(6), 1380-1385 (2012)

6. Kirdeciler K., Soy E., Ozturk S., Kucherenko I., Soldatkin O., Dzyadevych S., Akata Kurc B. "A Novel Urea Conductometric Biosensor Based on Zeolite Immobilized Urease", Talanta, 85(3), 1435-1441 (2011)

7. Ozturk S., Akata Kurc B. "Oriented Assembly and Nanofabrication of Zeolite a Monolayers", Microporous and Mesoporous Materials, 126(3), 228-233 (2009)