UNIVERSITY OF ÇUKUROVA
INSITUTE OF NATURAL AND APPLIED SCIENCE

MSc THESIS

Barış DÜNDAR

NEURAL NETWORK-BASED ULTRASONIC LEVEL MEASUREMENT OF FLUIDS

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

ADANA,2010

ÇUKUROVA ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

NEURAL NETWORK-BASED ULTRASONIC LEVEL MEASUREMENT OF FLUIDS

Barış DÜNDAR

YÜKSEK LİSANS TEZİ

ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

Bu tez 29/01/2010 Tarihinde Aşağıdaki Jüri Üyeleri Tarafından Oybirliği İle Kabul Edilmiştir.

İmza
Yrd. Doç. Dr Murat AKSOY
DANIŞMAN

İmza
Yrd. Doç. Dr Mutlu AVCI
ÜYE

İmza
Yrd. Doç. Dr K. Çağatay BAYINDIR
ÜYE

Bu tez Enstitümüz Elektrik-Elektronik Mühendisliği Anabilim Dalında hazırlanmıştır.

Kod No:

Prof. Dr İlhami YEĞİNGİL
Enstitü Müdürü

Not: Bu tezde kullanılan özgün ve başka kaynaktan yapılan bildirişlerin, çizelge, şekil ve fotoğrafların kaynak gösterilmeden kullanımı, 5846 sayılı Fikir ve Sanat Eserleri Kanunundaki hükümlere tabidir

ABSTRACT

MSc THESIS

NEURAL NETWORK-BASED ULTRASONIC LEVEL MEASUREMENT FOR FLUIDS

Barış DÜNDAR

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
INSTITUTE OF NATURAL AND APPLIED SCIENCES
UNIVERSITY OF ÇUKUROVA

Supervisor : Asst. Prof. Dr. Murat AKSOY
Year : 2010, Pages:---
Jury : Asst. Prof. Dr. Murat AKSOY
Asst. Prof. Dr. Mutlu AVCI
Asst. Prof. Dr. K. Çağatay BAYINDIR

This thesis describes the practical implementation of neural network-based ultrasonic level measurement of fluids (in the range of cm to meters) and will be designed a neural network-based ultrasonic range finder sensor for this goal. By a neural network algorithm, this sensor can be advanced including temperature, humidity, and other parameters (such as pressure, CO2 in medium). A Multi Layer Perceptron (MLP) neural network (NN), and Back-Propagation Algorithm which is used as common for it's learning is used in this thesis is . MLP NN is designed by Matlab Neural Network Toolbox. Necessary input-output data set for training and test of MLP NN is obtained using the approximate formula published in JASA [1993] by Owen Cramer Hardware interface uses an Atmel Atmega32 8-bit AVR microcontroller to facilitate the generation of 40 kHz signal burst which is used in the transmitter circuit, and also to process the received signal for measuring the time of flight of reflected waves and exact distance of the obstruction. Level of fluids is momentarily calculated by flight time and speed of reference sound. Temperature and Humidity Sensor (SHT11) provides temperature and humidity data to microcontroller. These data are the inputs of MLP. The outputs of MLP are values of corrected level of fluids. In this project, The program for this device is developed in MCS Electronic / BASCOM-AVR IDE.

Keywords: Microcontroller, Ultrasonic Range Finder, Neural Network, Temperature, Humidity

ÖZ

YÜKSEK LİSANS

SIVI SEVİYESİNİN YAPAY SİNİR AĞI TEMELLİ ULTRASONİK MESAFE ÖLÇER İLE ÖLÇÜMÜ

Barış DÜNDAR

ÇUKUROVA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK VE ELEKTRONİK MÜHENDİSLİĞİ ANA BİLİM DALI

Danışman : Yrd.Doç. Dr. Murat AKSOY
Yıl : 2010, Sayfa:---
Juri : Yrd. Doç. Dr. Murat AKSOY
Yrd. Doç. Dr. Mutlu AVCI
Yrd. Doç. Dr. K. Çağatay BAYINDIR

Bu tez, Sıvıların Yapay Sinir Ağı Temelli Ultrasonik Mesafe Ölçümünün (cm'den mt'ye olan aralıkta) pratik uygulamasını sunacaktır ve bu amaç için bir Yapay Sinir Ağı Temelli Ultrasonik Mesafe Ölçüm Sensör'ü tasarlanacaktır. Bir yapay sinir ağı algoritması ile, bu sensor sıcaklık, nem ve diğer parameterelerin (basınç ve karbondioksit gibi) dahil edilmesiyle geliştirilebilir. Bu tezde, yapay sinir ağının öğrenmesinde yaygın olarak kullanılan Geriye Yayılım Algoritması ile Çok Tabakalı Perseptron (MLP) yapay sinir ağı (NN) modeli kullanılmıştır. MLP NN, Matlab Neural Network Toolbox ile tasarlanmıştır. Tasarlanan ağın eğitimi ve testi için gerekli olan giriş-çıkış data setleri, Owen Cramer tarafından JASA' da [1993] yayımlanan yaklaşım formülünden elde edilmiştir. Verici devresinde kullanılan 40 kHz sinyalin üretimini kolaylaştırmak için donanım arayüzü olarak Atmel Atmega32 8-bit AVR mikrokonroller kullanılmıştır ve mikrokonrolör tarafından yansıyan sinyalin alınması ile uçuş zamanı ve nesneye olan uzaklık hesaplanmıştır. Mikrokontolör'e sıcaklık ve nem bilgileri Sıcaklık ve Nem Sensörü (SHT11) tarafından sağlanmaktadır. Mikrokontrolör tarafından alınan bu bilgiler MLP'nin girişleridir. MLP'nin çıkışı ise sıvı seviyesinin düzeltilmiş olan değeridir. Bu projede, program MCS Electronic / BASCOM-AVR IDE ile geliştirilmiştir.

Anahtar Sözcükler: Mikrokontrolör, Ultrasonik Mesafe Ölçer, Yapay Sinir Ağı, Sıcaklık, Nem

ACKNOWLEDGEMENTS

I would like to express my heartfelt thanks to Assistant Prof. Dr. Murat AKSOY, Assistant Prof. Dr. Mutlu AVCI, Res. As. Zehan KESİLMİŞ and Bülent BÜYÜKGÜZEL for his supervision, guidance, encouragements and extremely useful suggestions throughout this thesis. I am very proud of working with Assistant Prof. Dr. Murat AKSOY as supervisor.

Finally, I would like to thank my wife and family for their endless support and encouragements.

CONTENTS **PAGE**

ABSTRACT…… I

ÖZ……... II

ACKNOWLEDGEMENTS……..... III

CONTENTS…… IV

LIST OF TABLES…….. VII

LIST OF FIGURES …… VIII

1. ELECTRONIC MEASUREMENTS……... 1

1.1. Introduction……... 1

1.2. Physical Variables……..…… 1

1.3. Sensors and Transducers…….. 2

1.3.1. Temperature Sensors…… 4

1.3.2. Humidity Sensors…….. 6

1.4. Instrumentation System..…….. 7

1.5. Summary of Noise Reduction Techniques…… 15

2. ULTRASONIC RANGE FINDER…….. 16

2.1. Introduction…….. 16

2.2. Components of an Ultrasonic Range Finder…….. 17

2.2.1. Ultrasonic Oscillator……. 17

2.2.2. Transmitter Circuit…… 21

2.2.3. Receiver Circuit…… 22

2.2.4. Ultrasonic Transducer……..... 23

2.2.5. Signal Detector …….. 24

2.2.6. Time Measurement Gate Circuit ……..... 27

2.2.7. Alarm Detector…….. 27

2.2.8. Alarm Output Circuit…… 30

2.2.9. Power Circuit……..... 32

2.3. Working Principles of an Ultrasonic Range Finder…… 32

3. ARTIFICAL NEURAL NETWORK…….. 35
3.1. Introduction……... 35
3.2. Definitions and Style of Computation……... 35
3.3. Multilayer Perceptron……. 37
3.3.1. Function of Each PE…….. 39
3.3.2. Training of MLPs……..... 40
3.3.3. Example-1: Training a Neural Network…… 45
3.4. Matlab Neural Network Toolbox for Back-Propagation Algorithm……. 48
3.4.1. Architecture……..... 49
3.4.2. Neuron Model (Tansig, Logsig, Purelin)…….. 49
3.4.3. Feed-Forward Network……. 52
3.4.4. Creating a Network (newff)…….. 52
3.4.5. Initializing Weights (init)…….. 54
3.4.6. Simulation (sim)…… 54
3.4.7. Training……..... 55
3.4.8. Back-Propagation Algorithm……. 55
3.4.9. Batch Training (train)……..... 56
3.4.10. Batch Gradient Descent (traingd)……..... 56
3.4.11. Batch Gradient Descent with Momentum (traingdm)…….. 58
3.4.12. Faster Training…….. 59
3.4.12.1. Levenberg-Marquardt (trainlm)……. 60
3.4.12.2. Reduced Memory Levenber-Marquardt (trainlm)……. 62
3.4.13. Example-2: Training a Neural Network…….. 63
4. THE DESIGN OF THE NEURAL NETWORK-BASED ULTRASONIC MEASUREMENT SYSTEM FOR FLUIDS ……..... 69
4.1. Introduction…….. 69
4.2. Hardware…….. 70
4.2.1. Atmel Atmega32 AVR Microcontroller……... 71
4.2.2. Humidity and Temperature Sensor (SHT11)……..... 73
4.2.3. Ultrasound Motion Sensor (DC-SS014)……. 75
4.3. Training and Testing of Measured Values of Temperature, Humidity, and

Error-Level Measurement for Fluids By Matlab Neural Network Toolbox 76
4.4. Flow Chart…….. 82
4.5. Block Diagram of Neural Network-Based Ultrasonic Range Finder for Fluid…… 84
5. RESULTS……..... 85
6. CONCLUSIONS…….. 86
REFERENCES…… 87
CURRICULUM VITAE……..... 89
APPENDIX A…….. 90
APPENDIX B……... 91
APPENDIX C……..... 93
APPENDIX D……... 94
APPENDIX E……... 96

LIST OF TABLES **PAGES**

Table 1.1. Representative physical variables, symbols, and units…….. 2
Table 1.2. Representative transducers……... 3
Table 1.3. Summarizes a checklist of noise reduction techniques…… 15
Table 4.1. SHT11 list of commands……. 74
Table 4.2. An example showing success of the network……. 81
Table A.1. Physical and chemical transduction principles……. 90
Table B.1. ANN types and applications……. 91
Table E.1. Speed values of sound relative temperature and humidity…… 96

LIST OF FIGURES **PAGES**

Figure 1.1. Generalized block diagram of an instrument applied to a physica measurement……. 1
Figure 1.2. Sensor block diagram…… 4
Figure 1.3. LM335-based temperature sensor circuit…….. 5
Figure 1.4. Relationship of impedance change to humidity…… 7
Figure 1.5. The signal flow diagram……..... 9
Figure 1.6. A comparison of analog and sampled signals…… 11
Figure 1.7. A comparison of analog and quantized signals……. 12
Figure 1.8. An instrument block diagram…… 14
Figure 2.1. An ultrasonic range finder sensor system……. 17
Figure 2.2. Linear and exponential waveforms……... 17
Figure 2.3. Oscillator circuit and its output waveforms ……..……... 19
Figure 2.4. Transmitter circuit and its output waveforms …….. 21
Figure 2.5. Signal amplification circuit……... 22
Figure 2.6. Signal detection circuit…….. 23
Figure 2.7. Serial and parallel equivalent circuits of an ultrasonic transducer…. 24
Figure 2.8. Signal detector and its output waveform…… 25
Figure 2.9. Transmission and reflection signals…… 26
Figure 2.10. Time measurement gate circuit and its output waveforms…… 28
Figure 2.11. Alarm detector Alarm detector and its output waveforms …….. 29
Figure 2.12. Alarm output circuit…….. 31
Figure 2.13. Power circuit……. 32
Figure 2.14. The time from transmission to echo reception……. 33
Figure 2.15. Working principles of a range finder……... 34
Figure 3.1. The style of neural computation……. 36
Figure 3.2. MLP with one hidden layer (d-k-m)…….. 38
Figure 3.3. A PE and the most common nonlinearities…… 38
Figure 3.4. A two-input Pe and its separation surface…….. 39
Figure 3.5. Computing analytically optimal weights for linear Pe……..... 40

Figure 3.6. How to extend LMS to nonlinear Pes with the chain rule…….. 42
Figure 3.7. How to adapt the weights connected to ith PE……... 43
Figure 3.8. Two-layer feed-forward perceptron……..... 45
Figure 3.9. Multilayer networks……. 50
Figure 3.10. Log-sigmoid transfer function……..... 50
Figure 3.11. Tan-sigmoid transfer function…… 51
Figure 3.12. Linear transfer function…….. 51
Figure 3.13. Feed-forward networks……..... 53
Figure 3.14. Perceptron architecture……..... 64
Figure 3.15. Decision boundary for classification problem…… 68
Figure 4.1. The block diagram of the neural network-based an ultrasonic range finder……..... 70
Figure 4.2. Block diagram of atmega32 8-bit microcontroller……. 72
Figure 4.3. Typical application circuit for sht11…….. 73
Figure 4.4. Example measurement sequence for value %35.50 RH……... 75
Figure 4.5. Ultrasound motion sensor……..……... 76
Figure 4.6. The network which was designed……. 78
Figure 4.7. Training of the network…… 79
Figure 4.8. Success of The Network……... 82
Figure 4.9. Flow chart of neural network-based ultrasonic range finder……...…... 83
Figure 4.10. Neural network-based ultrasonic range finder and ultrasonic level measurement system of the fluids…… 84
Figure C.1. SHT11 pin assignment, NC remain floating……. 93
Figure D.1. Timing diagram……..... 94

1. ELECTRONIC MEASUREMENTS

1.1. Introduction

Instruments are the means for monitoring or measuring physical variables. The basic elements of an instrumentation application are shown in Figure 1.1. A physical system produces a measurand , $m(t)$, as time-varying. A transducer transforms a measurand into electrical signal, $s(t)$. This signal is processed by an instrument to yield the desired output information variable, $i(t)$. Producing meaningful information from physical variables requires conversion and processing. Electronic instruments require that physical variable be converted to electrical signals. Electrical signal for desired output information is processed by signal conditioning and signal processing (Schmalzel, 2000).

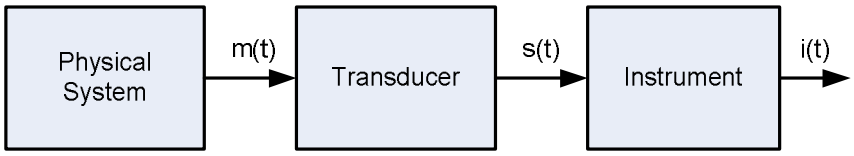


Figure 1.1. Generalized block diagram of an instrument applied to a physical measurement

1.2. Physical Variables

The measurand can be one of many physical variables; the type depends on the application. For example, in process control, typical measurands can include pressure, temperature, humidity and flow. Representative physical variables with corresponding units are summarized in Table 1.1. (Schmalzel, 2000).

Table 1.1. Representative physical variables, symbols, and units

Physical Variable	Symbol	SI Units, Abbreviations
Current	I	Ampere, A
Energy	E	Joule, J
Force	F	Newton, N
Flow	Q	Volume Flow Rate, m^3 / sn
Frequency	f	Hertz, Hz
Length	L	Meter, m
Mass	m	Kilogram, kg
Pressure	P	N / m^2
Power	P	Watt, W
Resistance	R	Ohm, Ω
Temperature	T	Kelvin, K
Time	t	Second, s
Velocity	V	m / s
Voltage	V	Volt, V

1.3. Sensors and Transducers

Transducers convert one form of energy to another. To be useful for an electronic instrument, a transducer must produce an electrical output such as voltage or current to allow required signal conditioning and signal processing steps to be completed. A variety of transducers can be classified according as a measurement requirement. Table 1.2. shows a variety of transducers.

The block diagram of a sensor is showed into Figure 1.2. Sensors are critical components in all measurement and control system. Most, but not all, sensors are transducers, employing one or more transduction mechanisms to produce an electrical output signal.

Table 1.2. Representative transducers

Measurand	Transducer	Operating Principles
Displacement (Length	Resistive, Capacitive, Inductive	Change in resistance, capacitance, or inductance caused by linear or angular displacement of transducer element.
Force	Strain Gage	Resistance, Piezoresistivity
Temperature	Thermistor Thermocouple	Resistance Peltier, Seebeck Effect
Pressure	Diaphragm	Diaphragm motion sensed by a displacement technique
Flow	Differential pressure Turbine	Pressure drop across restriction Angular velocity proportional to flow rate

Sometimes sensors are classified as direct and indirect sensors according to Table 1.2. For example, a mercury thermometer produces a change in volume of mercury in response to a temperature change via thermal expansion. But, the output is a mechanical displacement and not an electrical signal. Another transduction mechanism is required. For this aim, a thermometer is enough. Thermometers transforms mechanical displacement to electrical signal. This could be accomplished using capacitive effects. However, there are more direct temperature sensing methods.

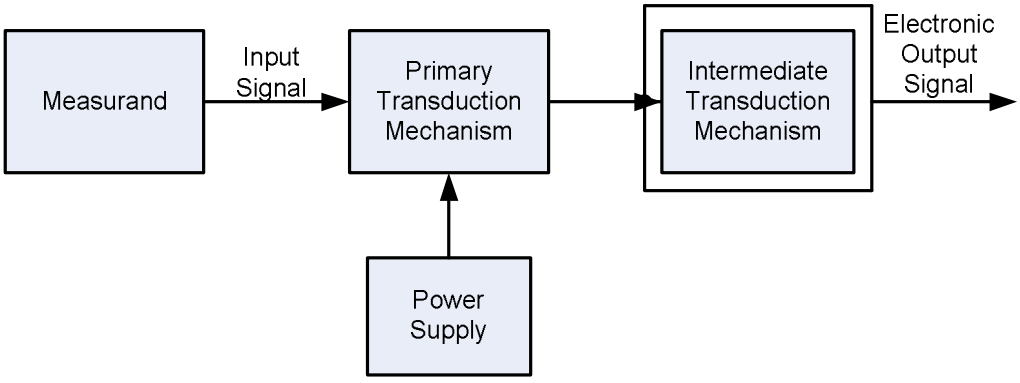


Figure 1.2. Sensor block diagram

The most important parameters for a sensor are sensitivity, stability, and repeatability. Normally, a sensor is only useful if all three of these parameters are tightly specified for a given range of measurand and time of operation. For example, a highly sensitive device is not useful if its output signal drifts greatly during the measurement time and the data obtained is not reliable if the measurement is not repeatable. Other output characteristics, such as selectivity and linearity, can often be compensated for by using, additional, independent sensor input or with signal conditioning circuits. In fact, most sensors have a response to temperature, since most transducing effects are temperature dependent. Sensors are most often classified by the type of measurand, i.e., physical, chemical, or biological. This thesis will given examples for physical sensors.

1.3.1. Temperature Sensors

Heat is one of the many forms of energy, and an amount of heat is measured in the usual energy units of joules. The quantity of heat contained in an object cannot be measured. But, it can be measured by a change of temperature or a change of physical state. In this sense, temperature is a measure of the level of heat for a material. Herein, physical state of a material has to remain unchanged. The relationship between temperature and energy is very similar to that between voltage level and electrical energy (Sinclair, 2001).

Figure 1.3 shows a temperature sensor circuit that uses an LM335. The sensor converts the ambient temperature into an equivalent output voltage and its output increases by approximately 10 mV for every 1 degree Kelvin or decreases. In this circuit, the output of an LM335 is fed into a 741 op-amp (any standard op-amp may be used.) which is configured as a voltage follower. Here, the main function of the op-amp is just to buffer the output of LM335.

Temperature sensors are the most common sensors in an air-conditioning system. Usually, three types of temperature sensors are popular, namely thermocouples, resistance temperature detector (RTDs), and thermistors (Sinclair, 2001).

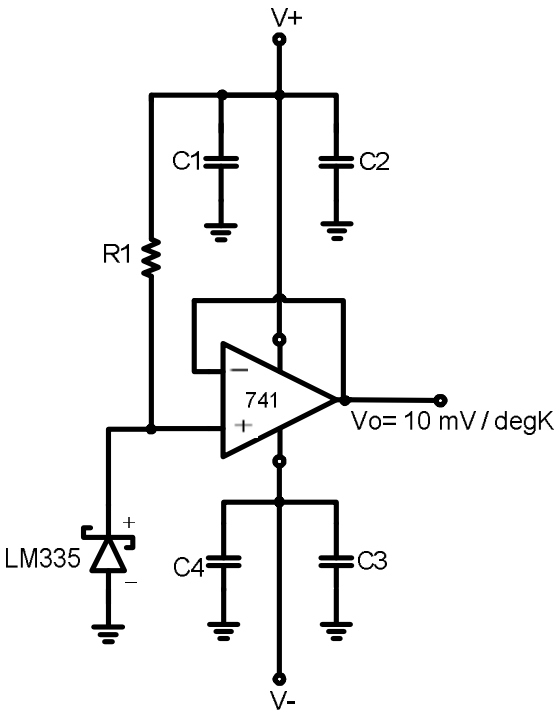


Figure 1.3. LM335-based temperature sensor circuit

Thermocouples make use of the current flowing in a circuit consisting of two dissimilar metals which are joined at different temperatures, a reference temperature and the measured temperature. They are robust but not sensitive, and are normally

used for high-temperature measurements such as in a combustion chamber as suggested by Underwood.

RTDs make use of the variation in electrical resistance of metal with respect to temperature changes. Platinum is the most commonly used metal in RTDs. It offers a wide measurement range from -250 ^{o}C to 650 ^{o}C as its temperature resistance coefficient is linear over this range. The main drawback is its relatively high cost.

1.3.2. Humidity Sensors

These sensors fall into four categories, hygrometers, psychrometers, electronic humidity sensors (resistive and capacitive sensors). Humidity measurement is problem because electromechanical hygrometers have suffered from serious non-linearity. Recently, developed electronic sensors are weak to contamination in the air stream. But yet, certain electronic sensors are steadily improving in terms of accuracy, long-term stability, and tolerance to contaminants.

Hygrometers for measurement of moisture are an earliest developed method. It works with moisture absorption and desorption in medium. They are highly non-linear and weak to drift. They are now replaced by electronic devices.

In a Psychrometers, there two thermometers. First thermometer is dry-bulb other wet-bulb. When air temperature is below freezing, wet-bulb is covered with a thin coating of ice. And, yet may be warmer than the dry-bulb. Relative humidity is computed by difference between wet-bulb and dry-bulb thermometers

Resistive and capacitive sensors are one of the most widespread, commercially available electronic sensors. Today, it is a modern devices and have improved sensitivity, up to %2 in relative humidity. Another advantage is that this type of sensor tends to have better immunity to contaminated air streams than other electronic types (Sinclair, 2001).

Resistive humidity sensors measure the impedance change, which usually has an inverse exponential relationship to humidity (Figure 1.4.). Typically, the impedance change of a medium such as a conductive polymer, salt, or treated substrate is measured (Fontes, 2005).

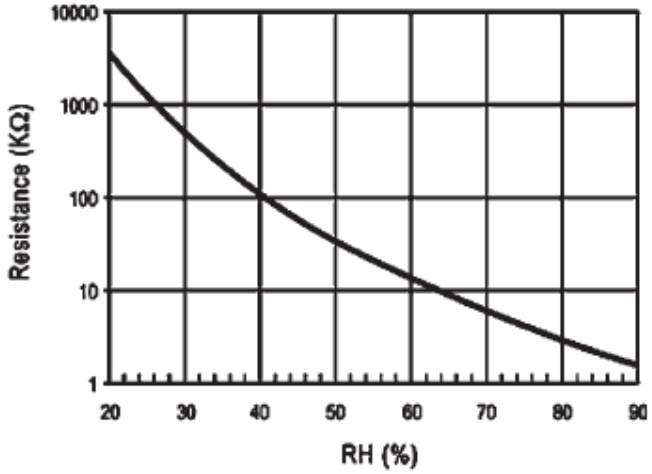


Figure 1.4: Relationship of impedance change to humidity

1.4. Instrumentation System

Although the design of instruments varies greatly, there are common building blocks. Figure 1.5 illustrates block diagram of a digital electronic instrument. The figure depicts a chain of signal processing elements. Each output provides required input information for the next block. In the past, most instruments were purely analog, with analog data being fed directly to analog displays. Currently, however, most instruments being developed contain a digital information processing stage as shown in Figure. 1.5. (Stahlin, 2004).

Beginning at the bottom of Figure 1.5 is the device under test (DUT). As the primary purpose of the instrument is to gain information about some physical property of the DUT. A connection must be made between the instrument and the DUT. This requirement often imposes design constraints on the instrument. For example, the instrument may need to be portable, or the connection to the DUT may

require a special probe. The design of the thermometer in the earlier example assumes that the mercury reservoir can be immersed into the DUT, such as a fluid. It also assumes that the fluid's temperature is considerably lower than the melting point of glass.

Continuing up from the DUT in Fig. 1.5. is the first transducer in the signal flow of the instrument or the sensor. This is the element that is in physical (not necessarily mechanical) contact with the DUT. The sensor must respond to the physical variable of interest and convert the physical information into an electrical signal. Often, the physical variable of interest is itself an electrical signal. In that case, the sensor is simply an electrical connection. In other cases, however, the physical variable of interest is not electrical (Stahlin, 2004).

Examples of sensors include a piezoelectric crystal that converts pressure to voltage, or a thermocouple that converts temperature into a voltage. The advantage of such sensors is that, by converting the physical phenomenon of interest into an electrical signal.

In the case of a stimulus instrument, the role of this stage is to convert an electrical signal into a physical stimulus of the DUT. In this case, some form of actuator is used. Examples of actuators are solenoids and motors to convert electrical signals into mechanical motion, loudspeakers to convert electrical signals into sound, and heaters to convert electrical signals into thermal energy (Stahlin, 2004).

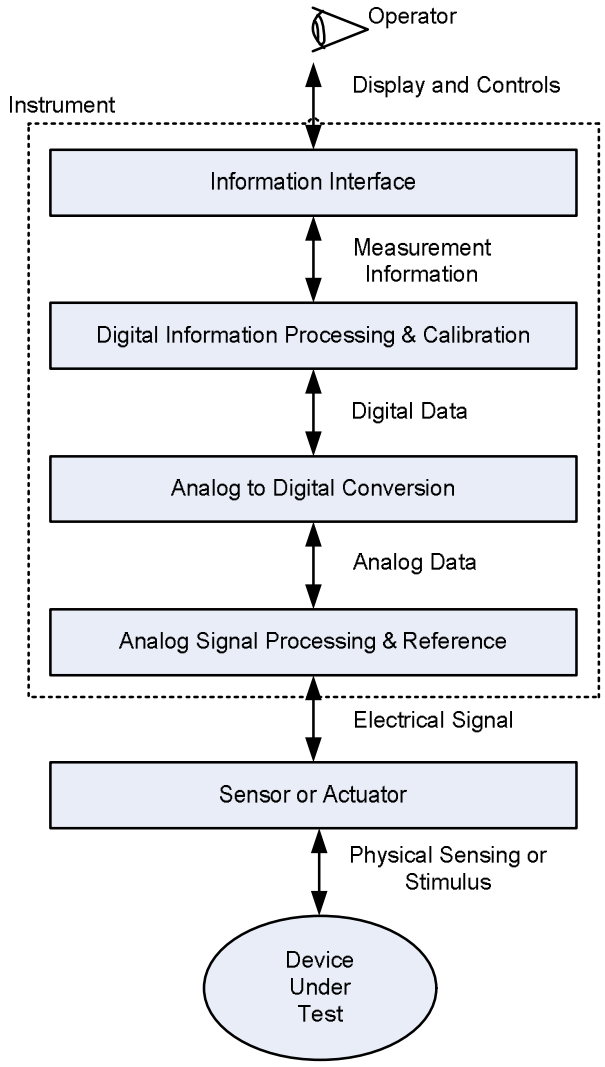


Figure 1.5. The signal flow diagram

The next stage in the signal flow shown in Fig. 1.6 is the analog signal conditioning within the instrument. This stage often contains circuitry that is quite specific to the particular type of instrument. Functions of this stage may include amplification of very low voltage signals coming from the sensor, filtering of noise, mixing of the sensor's signal with a reference signal (to convert the frequency of the signal, for instance), or special circuitry to detect specific features in the input

waveform. A key operation in this stage is the comparison of the analog signal with a reference value.

Ultimately, the value of a measurement depends upon its accuracy. The information created by a measurement is a comparison of the unknown physical variable of the DUT with a reference, or known value. This requires the use of a physical standard or physical quantity whose value is known. A consequence of this is that each instrument must have its own internal reference standard as an integral part of the design if it is to be capable of making a measurement. For example, an instrument designed to measure the time between events or the frequency of a signal must have some form of clock as part of the instrument. Similarly, an instrument that needs to determine the magnitude of an electrical signal must have some form of internal voltage direct reference. The quality of this internal standard imposes limitations on the obtainable precision and accuracy of the measurement.

In the mercury-bulb thermometer example, the internal reference is not a fixed temperature. Rather, the internal reference is a fixed, or known, amount of mercury. In this case, the reference serves as an indirect reference, relying on a well-understood relationship between temperature and volume of mercury. The output of the analog processing section is a voltage or current that is scaled in both amplitude and frequency to be suitable for input to the next stage of the instrument.

For many instruments, the data typically undergo some form of analog-to digital conversion. The purpose of this stage is to convert the continuously varying analog signal into a series of numbers that can be processed digitally. This is accomplished in two basic steps: (1) the signal is sampled, and (2) the signal is quantized, or digitized (Stahlin, 2004).

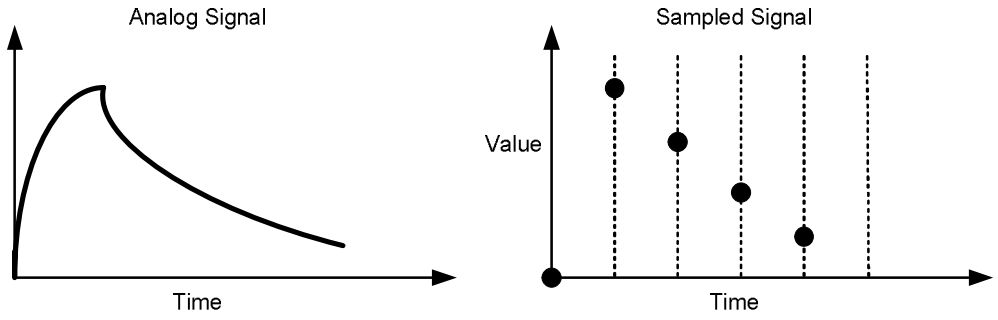


Figure 1.6. A comparison of analog and sampled signals

Sampling is the process of converting a signal that is continuously varying over time to a series of values that are representative of the signal at discrete points in time. Figure 1.6. illustrates an analog signal and the resulting sampled signal. The time between samples is the measure of the sample rate of the conversion. In order to represent an analog signal accurately, the sample rate must be high enough that the analog signal does not change appreciably between samples. Put another way: Given a sequence of numbers representing an analog signal, the maximum frequency that can be detected is proportional to the sample rate of the analog-to-digital conversion.

The second step of analog-to-digital conversion, quantization, is illustrated in Fig. 1.7. As shown in the figure, the principal effect of quantization is to round off the signal amplitude to limited precision. While this is not particularly desirable, some amount of quantization is inevitable since digital computation cannot deal with infinite precision arithmetic. The precision of the quantization is usually measured by the number of bits required by a digital representation of the largest possible signal. If N is the number of bits, then the number of output values possible is 2^n. The output range is from a smallest output of zero to a maximum value of 2^{n-1}. For example, an 8-bit analog-to-digital converter (ADC) could output 2^8, or 256 possible discrete values (Stahlin, 2004).

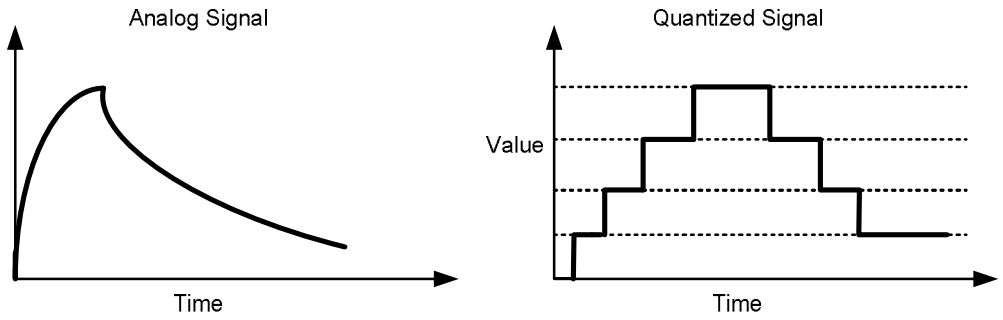


Figure 1.7. A comparison of analog and quantized signals

The output range would be from 0 to 255. If the input range of the converter is 0 to 10 V, then the precision of the converter would be (10-0)/255, or 0.039 V. This quantization effect imposes a tradeoff between the range and precision of the measurement. In practice, the precision of the quantization is a cost and accuracy tradeoff made by the instrument designer. But the phenomenon must be understood by the user when selecting the most appropriate instrument for a given application.

The output of the analog-to-digital conversion stage is thus a succession of numbers. Numbers appear at the output at the sample rate, and their precision is determined by the design of the ADC. These digital data are fed into the next stage of the instrument, the digital processing stage (Stahlin, 2004).

The digital processing stage is essentially a computer. Computer is optimized for the control and computational requirements of the instrument. It usually contains one or more microprocessors and/or digital signal processor circuits. These devices are used to perform calculations on the raw data that come from the ADC. The data are converted into measurement information.

External reference standards are used to check the overall accuracy of instruments. When an instrument is used to measure the value of a Standard DUT, the instrument's reading can be compared with the known true value, with the difference being a measure of the instrument's error. For example, the thermometer's accuracy may be tested by measuring the temperature of water that is boiling or

freezing, since the temperature at which these phase changes occur is defined to be 100°C and 0°C, respectively (Stahlin, 2004).

The source of the error may be due to differences between the instrument's internal reference and the standard DUT or may be introduced by other elements of the signal flow of the instrument. Discrepancies in the instrument's internal reference or nonlinearities in the instrument's signal chain may introduce errors that are repeatable, or systematic. When systematic errors are understood and predictable, a calibration technique can be used to adjust the output of the instrument to more nearly correspond to the true value. For example, if it is known that the markings on the thermometer are off by a fixed distance (determined by measuring the temperature of a reference DUT whose temperature has been accurately determined by independent means), then the indicated temperature can be adjusted by subtracting the known offset before reporting the temperature result. Unknown systematic errors, however, are particularly dangerous, since the erroneous results may be misinterpreted as being correct. These may be minimized by careful experiment design. In critical applications, an attempt is made to duplicate the results via independent experiments.

In many cases the errors are purely random and thus limit the measurement precision of the instrument. In these cases, the measurement results can often be improved by taking multiple readings and performing statistical analysis on the set of results to yield a more accurate estimate of the desired variable's value. The statistical compensation approach assumes that something is known about the nature of the errors. When all understood and repeatable error mechanisms have been compensated, the remaining errors are expressed as a measurement uncertainty in terms of accuracy or precision of the readings.

Besides performing the digital processing of the measurement information, the digital processing stage often controls the analog circuitry, the user interface, and an input/output (I/O) channel to an external computer.

While the design of the signal flow elements focuses on measurement performance, the physical components chosen and their methods of assembly will determine several important specifications of the instrument, namely, its cost,

weight, size, and power consumption. In addition, the instrument designer must consider the compatibility of the instrument with its environment.

Environmental specifications include ranges of temperature, humidity, vibration, shock, chemicals, and pressure. These are often specified at two levels: The first is the range over which the instrument can be expected to operate within specifications, and the second (larger) is the range that will not cause permanent damage to the instrument (Stahlin, 2004).

In order to build an instrument that implements a signal flow like that of Figure 1.5. additional elements such as a mechanical case and power supply are required. A common design of an instrument that implements the signal flow path discussed above is illustrated in Fig. 1.8.. As shown in the figure, the building blocks of the signal flow path are present as physical devices in the instrument. In addition, there are two additional support elements, the mechanical case and package and the power supply (Stahlin, 2004).

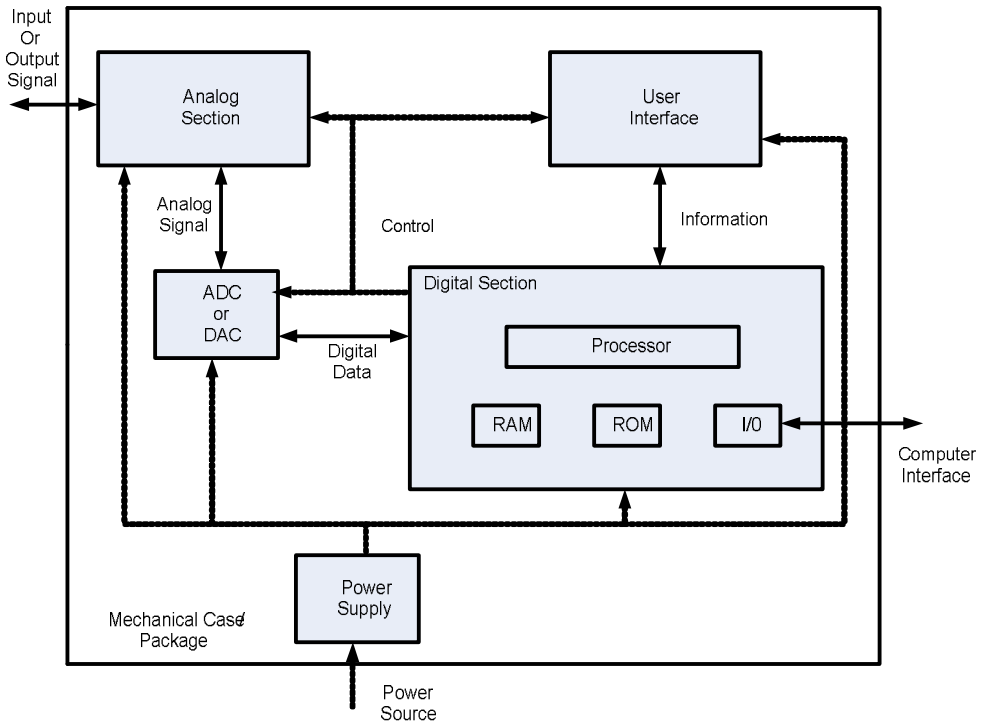


Figure 1.8. An instrument block diagram

1.5. Summary of Noise Reduction Techniques

Elimination of undesired measurement errors benefits from a systematic approach to identifying and solving noise problems. Source, interconnect, and response elements of a measurement system can be treated individually. Table 1.3. shows noise reduction techniques. Some techniques, such as shielding, are applicable to all three. Various combinations of techniques should be tried to achieve best results. There are many choices of grounding techniques that vary depending on whether elements are floating or ground-referred, and based on bandwidth. In general, multiple ground connections that create ground loops should be avoided. Difficult ground loop problems may require isolation or other techniques to interrupt the ground connection between elements (Schmalzel, 2000).

Table 1.3. Summarizes a checklist of noise reduction techniques

Source	Interconnect	Response
Shield enclosures	Shield leads	Shield enclosures
Filter inputs and outputs	Minimize loop are (twist leads)	Filter inputs and outputs
Limit bandwitdh	Keep signal leads near ground	Limit bandwidth
Minimize loop areas	Separate low-,high-level signals Keep signal and ground leads short Low f: Use single ground High f: Use multiple grounds	Minimize loop areas

2. ULTRASONIC RANGE FINDER

2.1. Introduction

Ultrasounds refer to sound above the frequencies of audible sound and nominally include anything over 20 kHz (Yayımlanmamış). Frequencies used for medical diagnostic and imaging extend to 10 MHz and beyond. Higher frequencies have shorter wavelengths. And, these signals reflect from objects more readily. Unfortunately, extremely high frequencies are difficult to generate and measure. Ultrasonic signals can be generated readily. And, detection and measurement of these signals can be received by simple piezoelectric receivers.

Ultrasonic sensing systems are often used in robotics such as obstacle avoidance, distance measurements, map construction of the robot's environment and in industrial environments (Almeida, Freire, Rönnow, Vianna ve Rossi, 2000). The Ultrasonic rangefinder is a low-cost noncontact and easy-to-install measurement device. It is able to meet the every-day needs of commercial production, as well serving a more specialized role in the technologically-advanced aero-space industry, thus placing it firmly in the category of high-level measurement technology. Unlike other level indicators with limited uses, the easy-to-install ultrasonic rangefinder is a highly-accurate device with enough specialized uses to ensure that the needs of the customer are met (www.fine-tek.com).

2.2. Components of an Ultrasonic Range Finder

An ultrasonic range finder system has been developed, based on band-pass to low-pass transformation of the impulse response. The impulse response is a sensitive indicator of time delays between signals. In this case the time delay is caused by the propagation of the sound wave between transmitter and receiver (Holmberg, 1992). The range finder module consists of Ultrasonic Oscillator, Transmitter Circuit, Receiver Circuit, Ultrasonic Transducers (Transmitter and Receiver), Signal Detector, Time Measurement Gate Circuit, Alarm Detector, Alarm Output Circuit, and Power

Supply (Inoue, 1998). Microcontroller-based ultrasonic range finder consist of microcontroller, ultrasonic transmitter and receiver circuits, and power supply.

An ultrasonic range finder sensor system is illustrates in Figure 2.1. (Inoue, 1998).

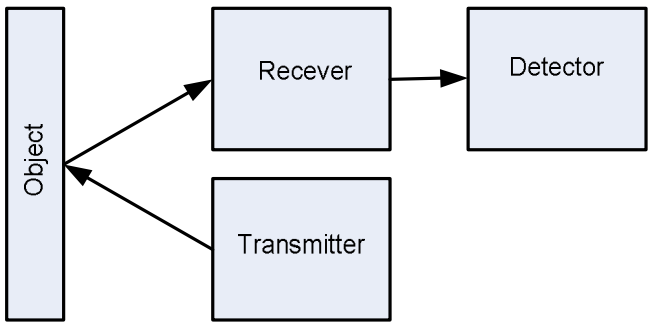


Figure 2.1. An ultrasonic range finder sensor system

2.2.1. Ultrasonic Oscillator

When driven with a constant current I, a capacitance C charges or discharges at a constant rate, yielding a linear transient or ramp of the type of Figure 2.2.

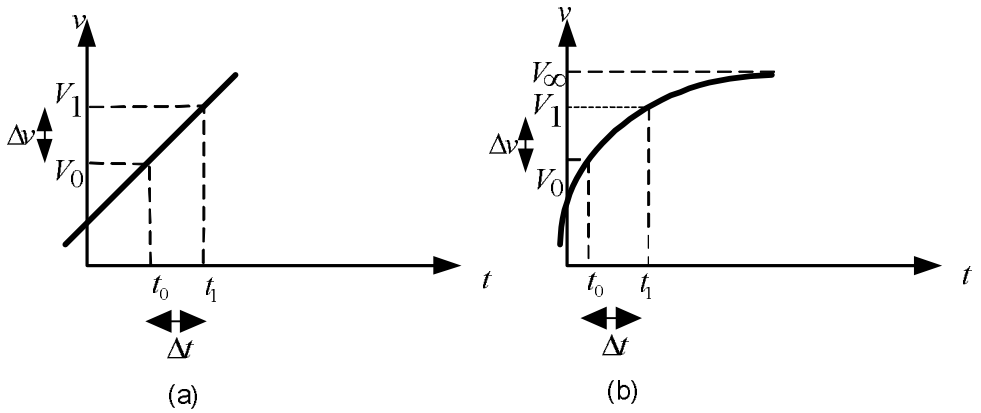


Figure 2.2. Linear and exponential waveforms

Engineers often describe this ramp via the easy-to-remember relationship

$$C\Delta v = I\Delta t \tag{2.1}$$

or “cee delta vee equals aye delta tee”. This allows us to estimate the time it ties to effect a constant-rate change Δv as

$$\Delta t = \frac{C}{I}\Delta v \tag{2.2}$$

An exponential transient occurs when C is charged or discharged via a series resistance R. With reference to Figure 2.2. (a), the instantaneous capacitance voltage is

$$v(t) = V_\infty + (V_0 - V_\infty)e^{(t-t_0)/t} \tag{2.3}$$

where V_0 is the initial voltage, V_∞ is the steady-state voltage that would be reached in the limit $t \to \infty$, and $t = R_{thevenin}C = RC$ is the time constant governing the transient. This equation holds regardless of the values and polarities of V_0 and V_∞. The transient reaches a specified intermediate value V_1 at t_1 such that Eq.(2.3). Taking the natural logarithm of both sides and solving for $\Delta t = t_1 - t_0$ allows us to estimate the time it takes to charge or discharge C from V_0 to V_1 as

$$\Delta t = t\ \ln\frac{V_\infty - V_0}{V_\infty - V_1}. \tag{2.4}$$

Figure 2.3 illustrates an ultrasonic oscillator. IC2 is the circuit to make oscillate the ultrasonic frequency of 40 kHz. The circuit makes $R_B \rangle R_A$ to bring the duty (Ratio of ON/OFF) of the oscillation wave close %50.

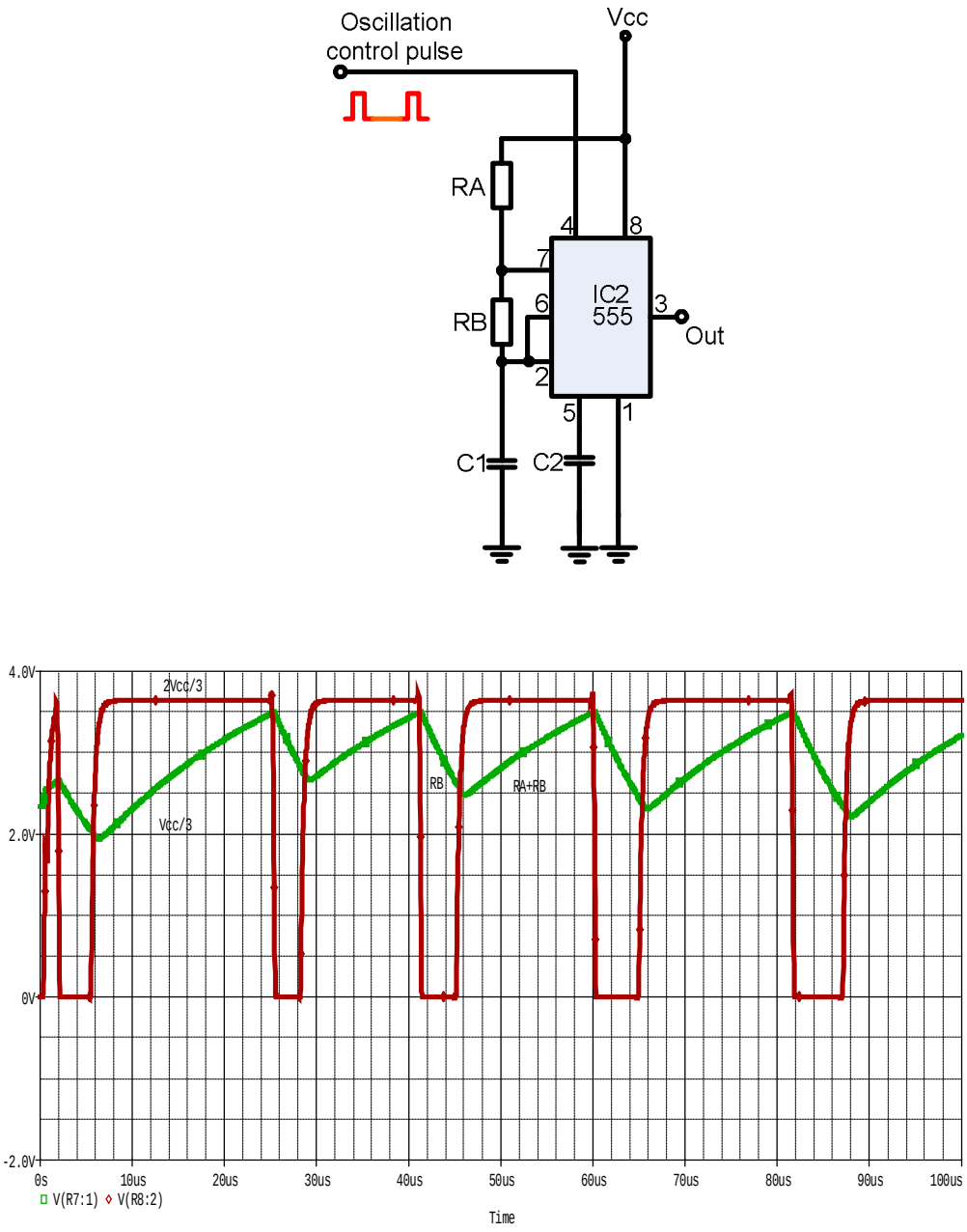


Figure 2.3. Oscillator circuit and its output waveforms

The frequency of the ultrasonic oscillator circuit (f) is found as

$$V_\infty = V_{cc}, V_0 = \frac{V_{cc}}{3}, V_1 = \frac{2V_C}{3} \Rightarrow t_1 = C\left(R_A + R_B\right) In \frac{V_{cc} - \frac{V_{cc}}{3}}{V_{cc} - \frac{2V_{cc}}{3}} = C\left(R_A + R_B\right) In2$$

(2.5)

$$V_\infty = 0, V_0 = \frac{2V_{cc}}{3}, V_1 = \frac{V_C}{3} \Rightarrow t_2 = CR_B In \frac{0 - \frac{2V_{cc}}{3}}{0 - \frac{V_{cc}}{3}} = CR_B In2 \quad (2.6)$$

$$T = t_1 + t_2 = C\left(R_A + R_B\right) In2 + CR_B \ln 2 = C\left(R_A + 2R_B\right) In2$$

(2.7)

$$\Rightarrow f = \frac{1}{T} = \frac{1}{C\left(R_A + 2R_B\right) In2}$$

The frequency of the ultrasonic must be adjusted to the resonant frequency of the ultrasonic sensor. Therefore, the oscillation frequency able to adjust by making the R_B the variable resistor (V_{R1}). When the reset terminal is the H level, IC2 works in the oscillation. The ultrasonic of 40 kHz is sent out for the 1 millisecond and pauses for the 68 milliseconds. The calculation example of the frequency is shown below the condition (Inoue, 1998):

$$R_A = 12k\Omega, R_B = 12k\Omega, C = 1nF \quad (2.8)$$

$$T_L = 0.69R_B C = 0.69 \times 12 \times 10^3 \Omega \times 10^{-9} F = 8.28m \sec. \quad (2.9)$$

$$T_H = 0.69\left(R_A + R_B\right)C = 0.69 \times \left(12 \times 10^3 \Omega + 12 \times 10^3\right) \times 10^{-9} F = 16.56m \sec.$$

(2.10)

$$f = \frac{1}{\left(T_L + T_H\right)} = \frac{1}{\left(8.28ms + 16.56ms\right)} \cong 40kHz. \quad (2.11)$$

(Yayımlanmamış)

2.2.2. Transmitter Circuit

The transmitter circuit is made up of several inverters and two transistors. The first inverter outputs the negative part of the ultrasonic wave. The transistors are to drive the CMOS inverters. The two inverters are connected in parallel to increase electric power transmission. The phase of the sensor is shifted by 180° between the positive and negative terminals of the sensor. The voltage applied on to the transmitter is twice the from the single inverter input (have a positive and negative peak to peak value). Because it is cutting the direct current with the capacitor, about twice of voltage of the inverter output are applied to the sensor. A transmitter is shown into Figure 2.4 (Inoue, 1998).

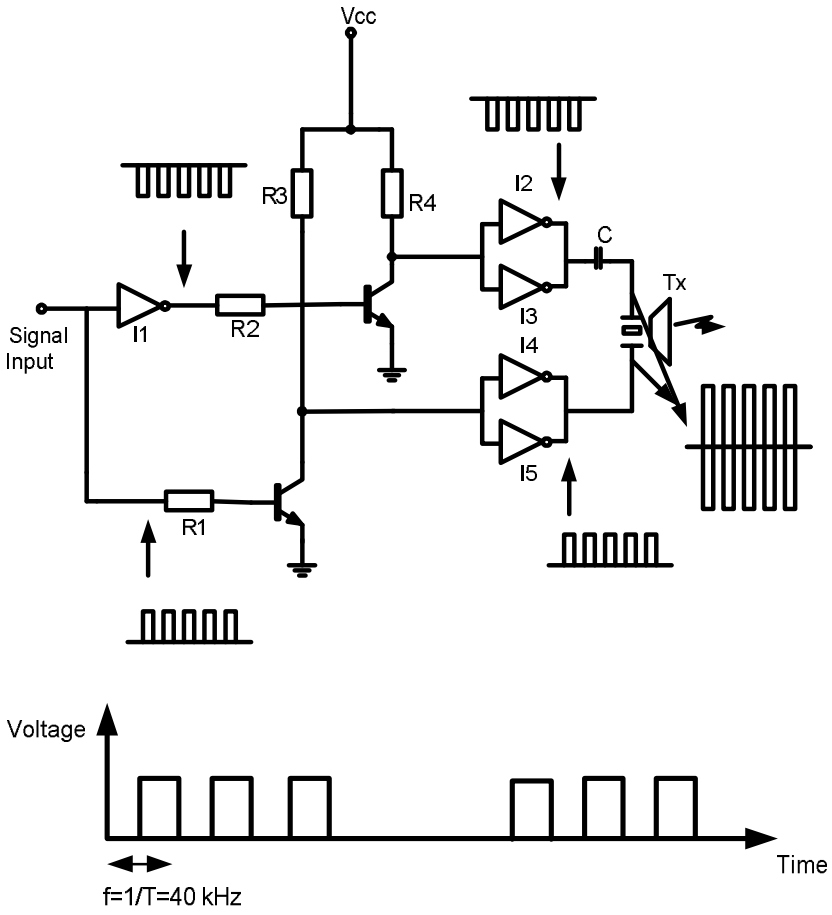


Figure 2.4. Transmitter circuit and its output waveforms

2.2.3. Receiver Circuit

The receiver circuit consists of two main parts, the signal amplification and detection circuitry. Figure 2.5. illustrates a receiver circuit.

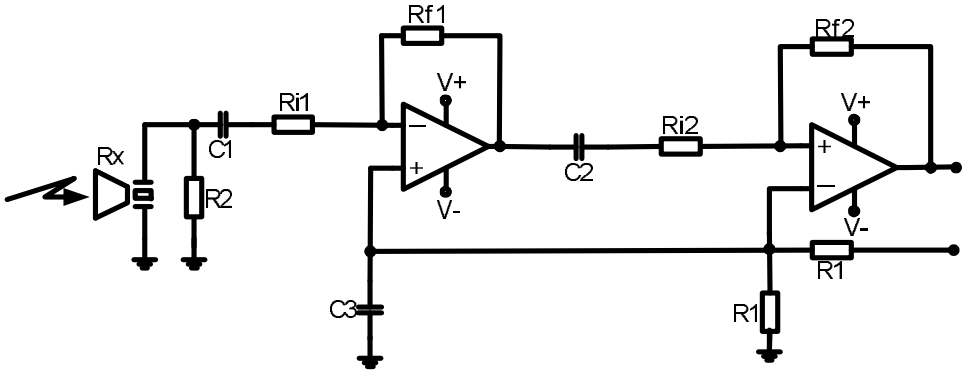


Figure 2.5. Signal amplification circuit

Upon receiving an ultrasonic signal through a reception sensor, the signal is amplified by 1000 times. The first stage will amplify the original signal by 100 times (40 dB) and the gain of second stage is 10 (20 dB) (Inoue, 1998).

$$A_1 = \frac{R_f}{R_i} = \frac{1M}{10k} = 100 \tag{2.12}$$

$$A_2 = \frac{R_f}{R_i} = \frac{100k}{10k} = 10 \tag{2.13}$$

$$A = A_1 \times A_2 = 10 \times 100 = 100 \tag{2.14}$$

$$A_{dB} = 20\log A = 20\log 1000 = 60dB \tag{2.15}$$

After the amplification circuit, the signal will go through a detection circuit consisting of a half wave rectification circuit (Figure 2.6.). This circuit is

implemented by two 1CV5 diodes. The detection is done to detect the received ultrasonic signal. It is the half-wave rectification circuit which used the Shottky barrier diodes. The DC voltage according to the level of the detection signal is gotten by the capacitor behind the diode. the Shottky barrier diodes are used because the high frequency characteristic is good.

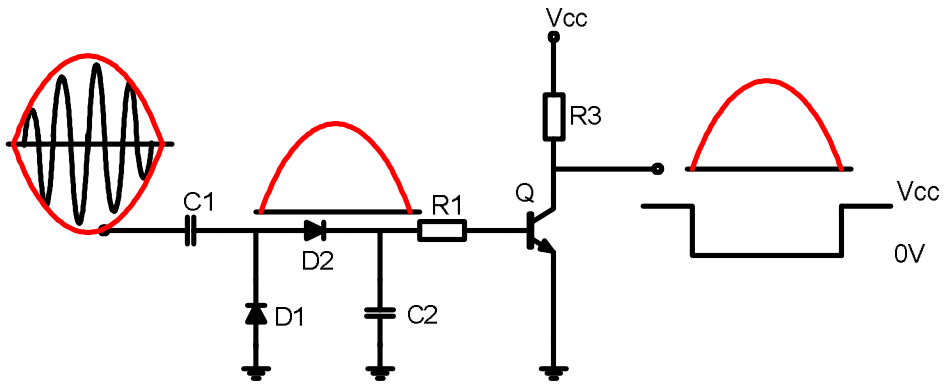


Figure 2.6. Signal detection circuit

The rectified signal goes to the transistor. When there is no signal, output is 3.3V (HIGH). If a signal is present, it will cause the output voltage to drop to 0V (LOW). The output is fed into Rx pin of the microcontroller to generate an interrupt when a falling edge is detected (Inoue, 1998)..

2.2.4. Ultrasonic Transducer

Figure 2.7. shows a simplified model of the series and parallel equivalent circuits of an ultrasonic transducer (Castles, 2008).

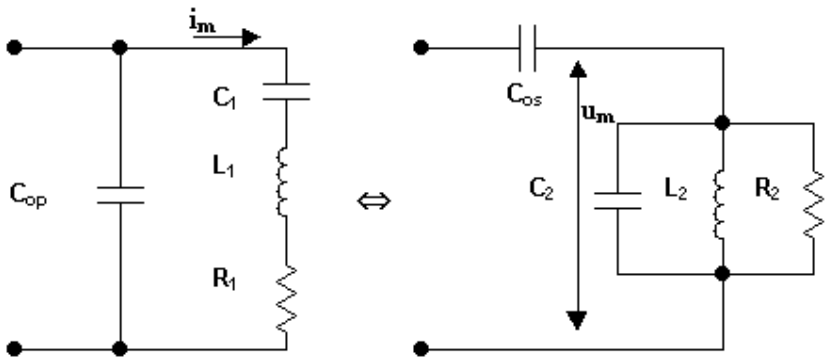


Figure 2.7. Serial and parallel equivalent circuits of an ultrasonic transducer

There are two main types of transducers used to transmit ultrasonic signals. They are the Piezo type and the electrostatic type. The main difference between a receiving and transmitting transducer is the impedance. The transmitter generally has a low impedance and a receiver has a higher impedance.

Piezo-electric ultrasonic output transducers are devices that have a complex impedance . They require a rather high voltage drive (~tens of Volts) to achieve maximum output and must be driven at their resonant frequency for maximum power output. The transducers can also be driven at low voltage by paralleled inverter gates or power amplifiers. This method is quite adequate for short range signals (Castles, 2008).

2.2.5. Signal Detector

This circuit is the circuit which detects the ultrasonic which returned from the object. The output of the detection circuit is detected as illustrated Figure 2.8. using the comparator

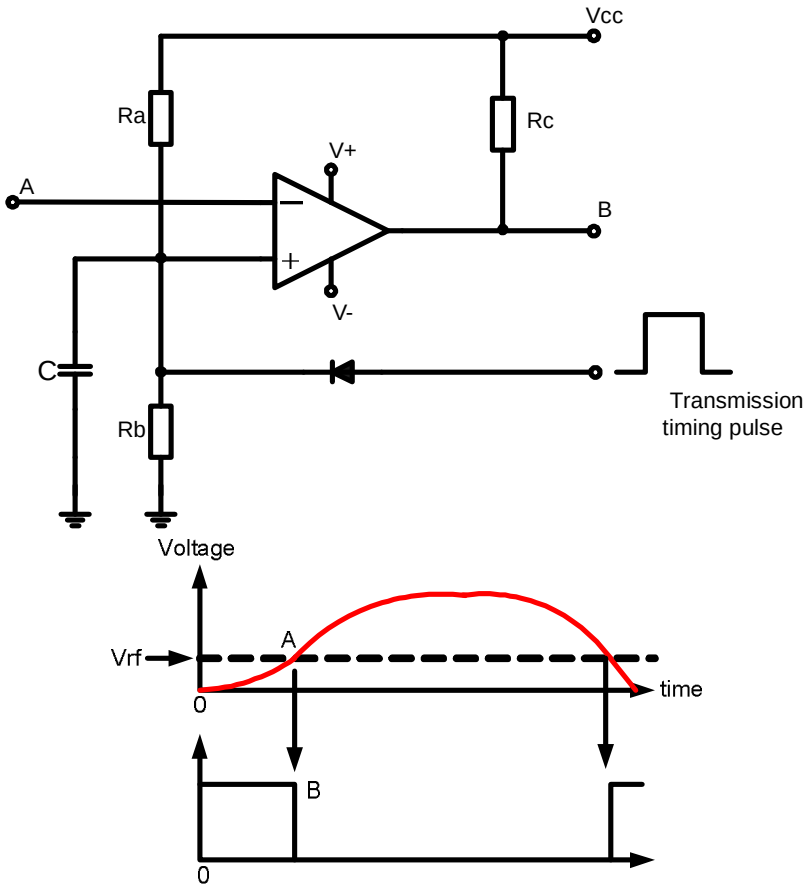


Figure 2.8. Signal detector and its waveform

At the circuit this time, the operational amplifier amplifies the difference between the positive input and the negative input and contrasts.. When the positive input becomes higher a little than the negative input, the difference is tens of thousands of times amplified and the output becomes the same as the power supply almost.(It is the saturation state) Oppositely, when the positive input becomes lower a little than the negative input, the difference is tens of thousands of times amplified and the output becomes 0 V almost (It is in the off condition). This operation is the same as the operation of the comparator. However, because the inner circuit is different about the comparator and the operational amplifier, the comparator can not be used as the operational amplifier. If reference voltage is

$$V_{rf} = \frac{R_b V_{CC}}{R_a + R_b} = \frac{47k\Omega \times 9V}{1M\Omega \times 47k\Omega} = 0.4V, \quad (2.16)$$

the rectified ultrasonic signal becomes more than 0.4 V, the output of the signal detector becomes the L level (approximately 0 V).

There is another device in this circuit. It is the diode (D) which connects with the side of the positive input. The pulse sending-out timing signal of the transmitter is applied to this diode. So, it makes not detect the transmission signal which was crowded when sending out the ultrasonic signal from the transmitter and going around to the reception sensor, making the voltage of the positive input of the signal detector rise in the pulse sending-out timing signal (Figure 2.9.). The transmission signal has the remaining signal even if it stops the transmission timing pulse. So, it makes the falling of the transmission timing pulse gentle with the capacitor (C) and it is preventing from the mis-detection by the remaining signal.

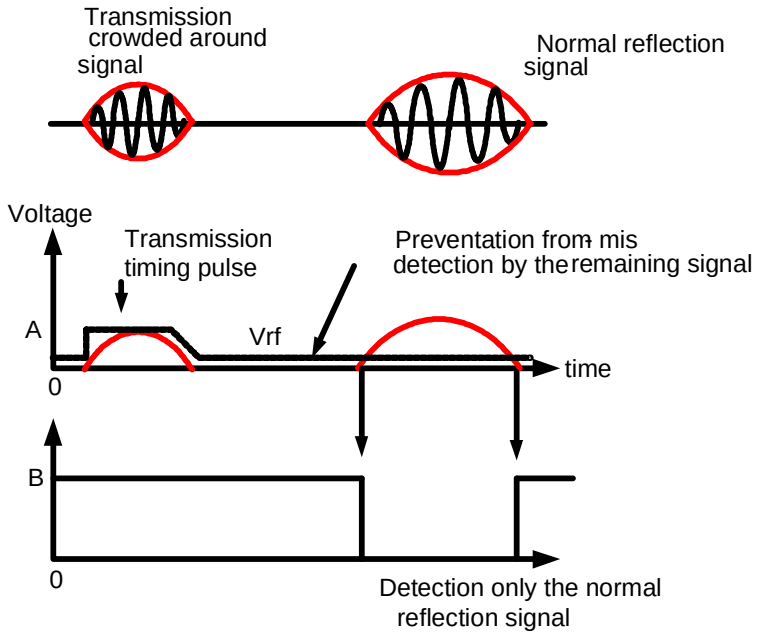


Figure 2.9. Transmission and reflection signals

The value of this capacitor is the one point which decides the efficiency of the equipment. The detection start time becomes late when the value of this capacitor is big and can not do the short distance. The equipment this time makes the transmission pulse long (about 1 millisecond) to make detect possible to the about 10-m distance and makes the capacitor of the detector big a little. Therefore, the shortest distance becomes about 40 cm. To detect the short distance, making T_L in IC1 short, the value of the capacitor of the signal detector must be made small. Way, in the time that the ultrasonic goes and returns in the 30 cm distance. It is the 1.75 milliseconds.

2.2.6. Time Measurement Gate Circuit

After ultrasonic signal receive by ultrasonic receiver circuit and detected by signal detector circuit, time of flight is measured by this circuit into Figure 2.10. It is using the SR (the set and the reset) flip-flop.

The set condition is the time which begins to let out the ultrasonic with the transmitter. It uses the transmission timing pulse. The reset condition is the time which detected the signal with the signal detector of the receiver circuit. That is, the time that the output of SR-FF (D) is in the ON condition becomes the time which returns after letting out the ultrasonic (Castles, 2008).

2.2.7. Alarm detector

This circuit into Figure 2.11. is used to detect signal. It sets the time using the 555 timer circuit. The transmission timing pulse is used for the trigger of the timer. The 555 timers begin the operation when the input trigger becomes the L level. So, it is inputting the transmission timing pulse through the inverter.

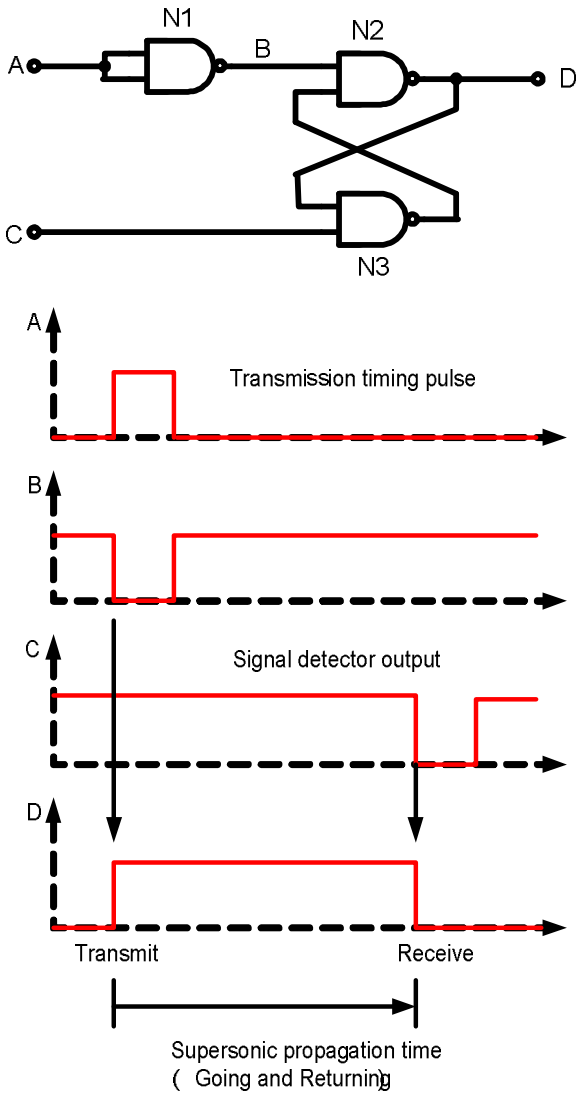


Figure 2.10. Time measurement gate circuit and its waveforms

The alarm output (E) becomes the L level if the output of the timer (D) becomes the H level and the output of the measurement gate circuit (C) becomes the H level. When the timer does in the time-out, the output of the timer (D) becomes the L level. So, after that, even if the output of the measurement gate circuit (C) becomes the H level, the alarm output (E) is as the H level. If the D point becomes the H level before the C point becomes the L level, the wrong pulse with the alarm output (the E

point) is output. In the actual operation, because that the C point becomes the L level is earlier than the operation beginning of the timer.

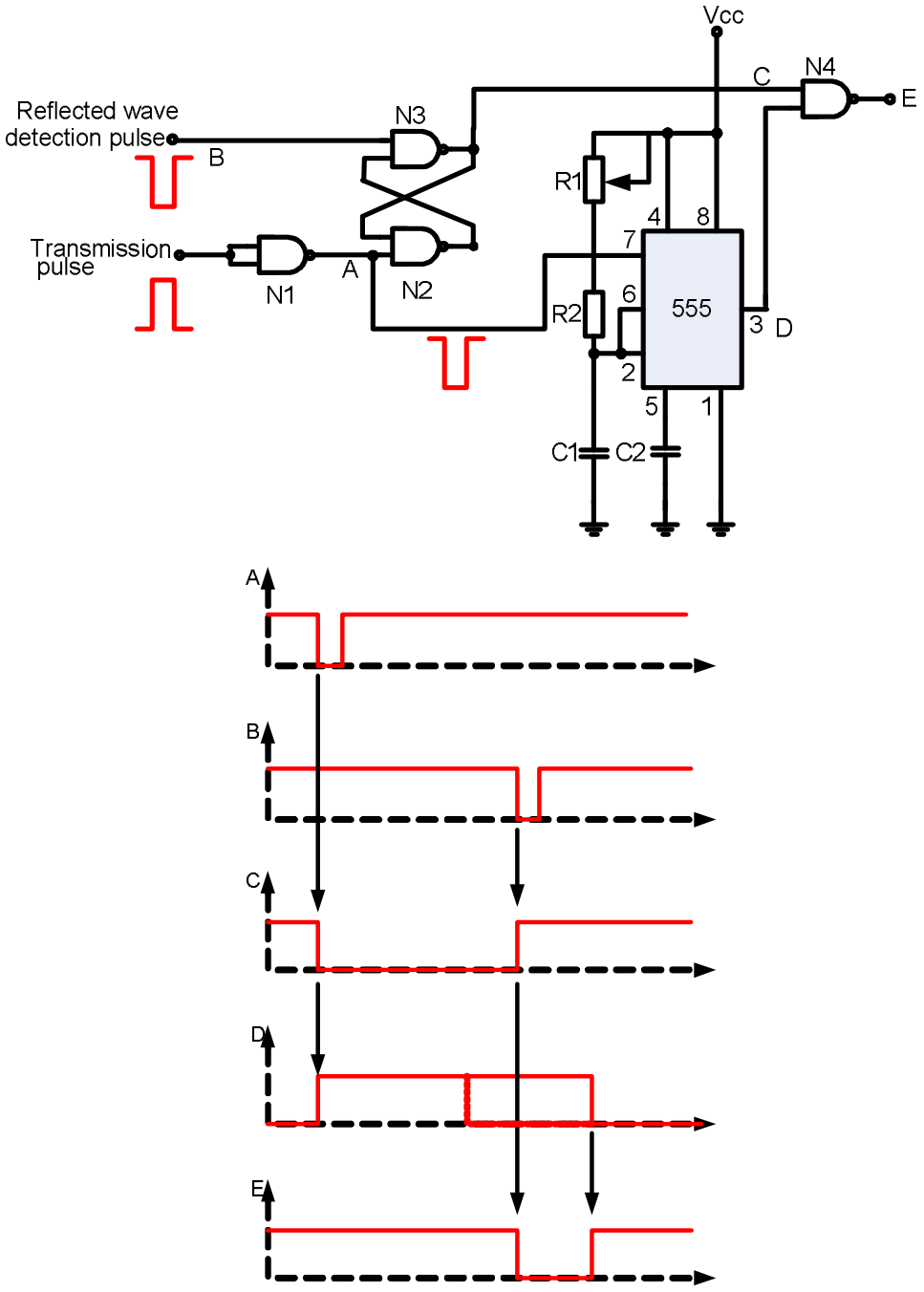


Figure 2.11. Alarm detector and its output waveforms

It made the range of the alarm from about cm to mt. The most short distance is limited by the transmission pulse mis-detection prevention circuit. The most long distance is limited by the transmission level of the ultrasonic, receiver sensitivity and then the interval of the transmission of the transmission pulse.

When the ambient temperature is 20°C, the propagation speed of the sound wave is 343.5 m/second. In the time to be propagated by 20 cm (the going and returning), time of flight is calculated as follows.

$$\frac{20cm}{2\times 343.5m/s} = 29.1m\sec. \tag{2.17}$$

In the time of the alarm detection timer, it makes change with the variable resistor. It is possible to calculate as follows in the minimum time and the maximum time . Since the 555 timer is the monostable mode, a pulse is generated at the output. The pulse with of time t, which is the time it makes to charge C to 2/3 of supply voltage, is given by

$$V_\infty = V_{cc}, V_0 = 0, V_1 = \frac{2V_C}{3} \Rightarrow t = R_2C_1\, In\frac{V_{cc}-0}{V_{cc}-\frac{2V_{cc}}{3}} = R_2CIn3 = 1.1R_2C_1 \tag{2.18}$$

2.2.8. Alarm Output Circuit

The output of the alarm detector (Figure 2.12.) isn't as long as it works a relay. The alarm output circuit makes constant time output last making the output of the alarm detector a trigger. It works a relay for about 1 second using 555 timer. (Castles, 2008)

$$T = 1,1R_1C_1 = 1,1\times 100k\Omega\times 10mF = 1.1\sec onds \tag{2.19}$$

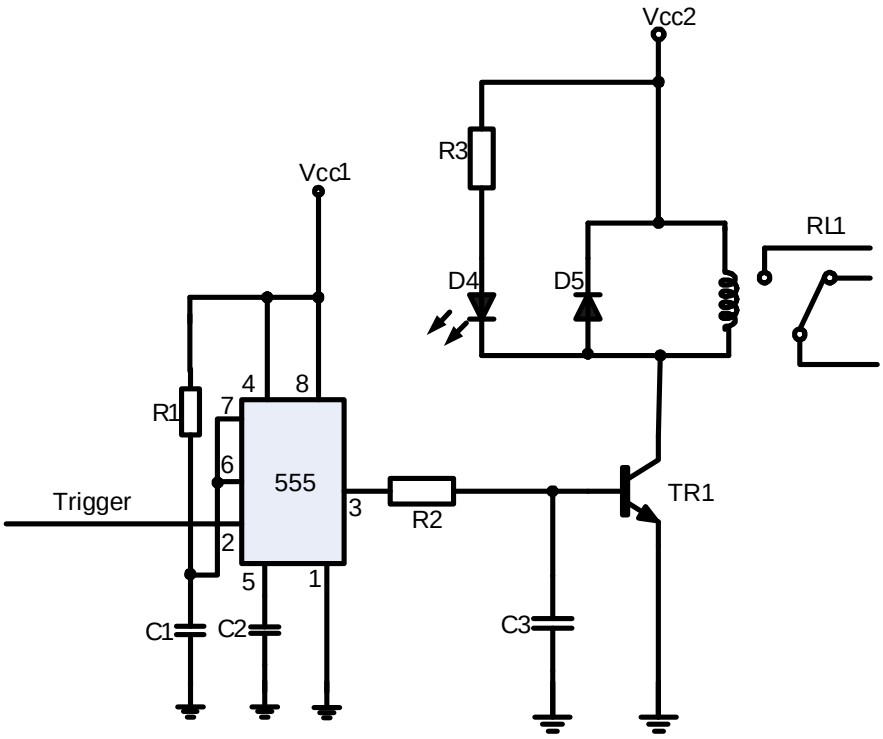


Figure 2.12. Alarm output circuit

The output of the timer (the H level in case of operation) is inputted to the transistor and a relay is driven. The diode (D5) which is connected with the coil of the relay in parallel is to protect a transistor from the opposite electromotive force which occurs with the relay . Also, the LED(D4) lights up at the same time as the relay works. Making it be possible to see this LED from outside by installing it in the case. It is to make it easy to set the distance of the alarm detector. The capacitor (C3) to be putting between the base of the transistor and the ground is put to delay the release (the non-operation) of the relay when the timer does in the time-out. After the timer does in the time-out, the following trigger signal is inputted a maximum of 70 milliseconds later. It is because the interval of the transmission of the ultrasonic is the about 70 milliseconds. When detecting the obstacle continuously, the output of the timer becomes the L level condition for the about 70 milliseconds every second. Even if the output of the timer becomes the L level in short time, by the electric charge to have stored the capacitor (C24), the relay continues to work. The value of

the capacitor is due to the release electric current of the relay. In case of the relay which was used this time, it is doing the continuous operation at 220 mF (Castles, 2008).

2.2.9. Power Circuit

In this thesis, 7805 voltage regulator circuit is used as power supply. Its output voltage is +5V stable. Filtering towards high frequencies is provided by capacitors. Figure 2.13. illustrates a power circuit.

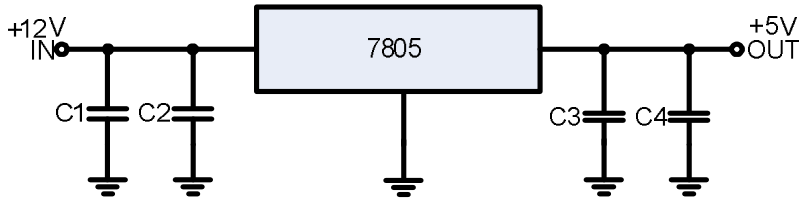


Figure 2.13. Power circuit (Castles, 2008)

2.3. Working Principles of an Ultrasonic Range Finder

Distance is computed by measuring the time taken for the ultrasound to echo back to the ultrasonic sensor. Ideally, the object should have a large surface area and does not absorb the ultrasound.

The principle of operation of the ultrasonic sensor system is to use the ultrasonic pulses which are transmitted by the transducer to the surface to be monitored and are reflected back to the transducer, the time period between transmission and reception of the sound pulses is directly proportional to the distance between the transducer and surface (Figure 2.14.). A micro-controller computes this time period for all echoes received and analyses them to determine which is the correct reflection from the material surface, it uses this data as the basis for giving control outputs and displays in usable engineering units (Renesas, 2004).

The distance d is determined from the velocity of sound v $(334.1m/s)$ and the time period t by the formula:

$$d = \frac{vt}{2} \tag{2.20}$$

Figure 2.15. shows the working principles of the ultrasonic rangefinder. Tx pin of microcontroller is used to transmit 40kHz ultrasound for 0.5msec and Rx pin of microcontroller is used to detect the reflected wave. After transmission, Timer C is started to keep track of the number of counts in Timer Counter C (TCC) so as to compute the distance of the object (Renesas, 2004).

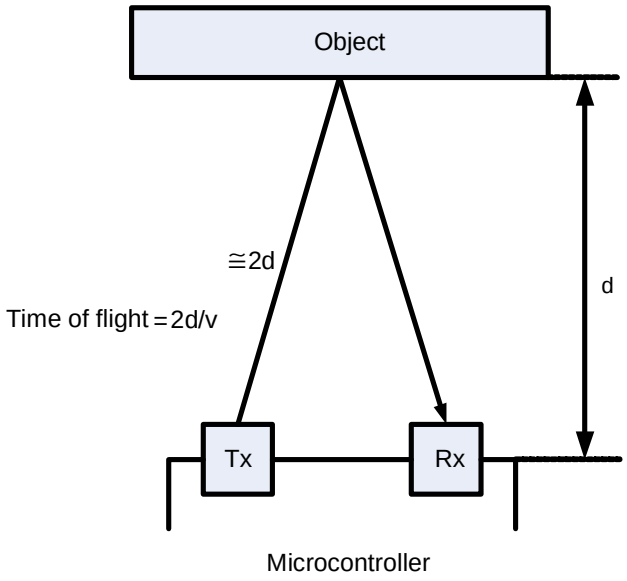


Figure 2.14. The time from transmission to echo reception (Renesas, 2004).

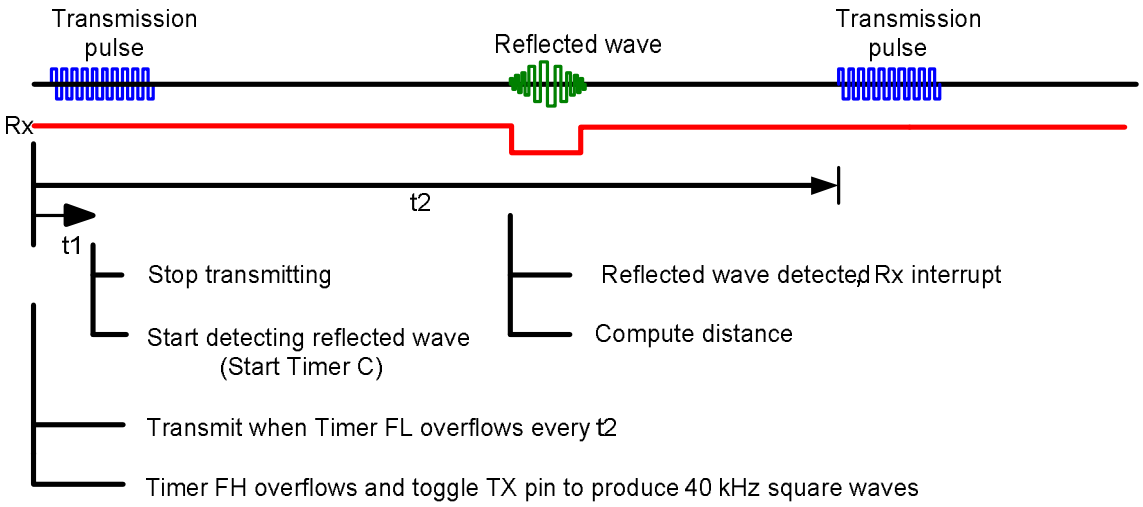


Figure 2.15. Working principles of a range finder (Renesas, 2004).

3. ARTIFICAL NEURAL NETWORK

3.1. Introduction

Artifical Neural Network (ANN) is well known that biological systems can perform tasks. It performs these tasks without recourse. Learning capability of biologic neurons reflects its ability to learn external stimulus ant to generalize. Such properties of nervous systems make them attractive as computation that can be designed to process complex data.

ANNs are mathematical models inspired from biological nervous systems. They are attractive as computation devices that can accept a large number of inputs. Then, they learn solely from training samples. As mathematical models for biological nervous systems, ANNs are useful in establishing relationships between inputs and outputs of any kind of system.

An ANN roughly is a collection of artificial neurons. An artificial neuron is a mathematical model of a biological neuron in its simplest form. From our understanding, biological neurons are viewed as elementary units for information processing in any nervous system. The mathematical model of an artificial neuron is based on the following theses:

- Neurons are the elementary units in a nervous system at which information processing occurs.
- Incoming information is in the form of signals that are passed between neurons through connection links.
- Each connection link has a proper weight that multiplies the signal transmitted.
- Each neuron has an internal action, depending on a bias threshold, resulting in an activation function being applied to the weighted sum of the input signals to produce an output signal (Principe, 2000).

3.2. Definitions and Style of Computation

An ANN is an adaptive, most often nonlinear system that learns to perform a function (an input/output map) from data. Adaptive means that the system

parameters are changed during operation. This operation is called as the training phase. The ANN is built with a systematic step-by-step procedure. To optimize a performance criterion is provided by this procedure. A performance criterion is commonly referred to as the learning rule

The input/output training data are fundamental in neural network technology. Because the nonlinear nature of the neural network processing elements (PEs), to achieve practically any desired input/output map. There is a neural computation style into Figure 3.1. (Principe, 2000)..

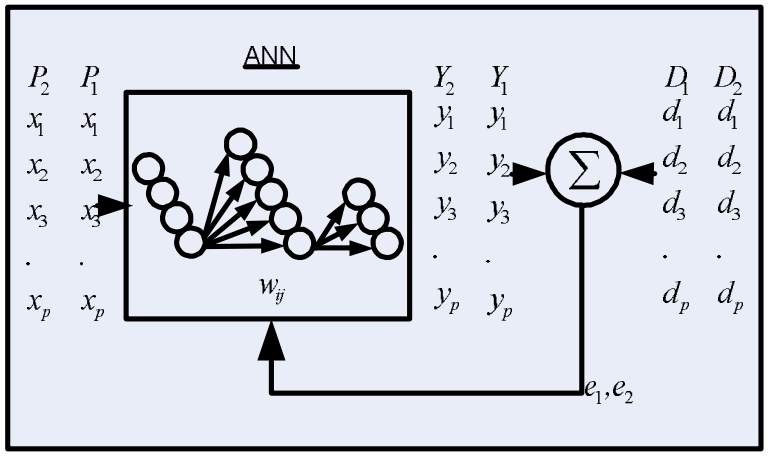


Figure 3.1. The style of neural computation

An input is presented to the network and a corresponding desired or target response set at the output. This is called as “Supervised Training”. An error is composed from the difference between the desired response and the system output. This error information is fed back to the system and adjusts the system parameters in a learning rule. If one does not have data that cover a significant portion of the operating conditions neural network technology is probably not the right solution. On the other hand, if there is plenty of data and the problem is poorly understood to derive an approximate model, then neural network technology is a good choice. In ANNs, the designer chooses the network topology, the performance function, the learning rule, and the criterion to stop the training phase. After inputs perform to the

system which is designed, the system automatically adjusts the parameters until the desired criterions reach (Principe, 2000).

3.3. Multilayer Perceptron

Multilayer perceptrons (MLP) are one of the most popular topologies and learning paradigms. They are a layered arrangement of nonlinear PEs as shown in Figure 3.2. The layer that receives the input is called the input layer. And, the layer that produces the output is the output layer. The layers that do not have direct access to the external world are called hidden layers. A layered network with just the input and output layers is called the perceptron. Each connection between PEs is weighted by a scalar, w_i called a weight, which is adapted during learning. The PEs in the MLP are composed of an adder followed by a smooth saturating nonlinearity of the sigmoid type (Figure 3.3.). The most common saturating nonlinearities are the logistic function and the hyperbolic tangent. The threshold is used in other nets. The importance of the MLP is that it is a universal mapper (implements arbitrary input/output maps) when the topology has at least two hidden layers and sufficient number of PEs .

Even MLPs with a single hidden layer are able to approximate continuous input/output maps. This means that rarely we will need to choose topologies with more than two hidden layers. But these are existence proofs, so the issue that we must solve as engineers is to choose how many layers and how many PEs in each layer are required to produce good results (Principe, 2000).

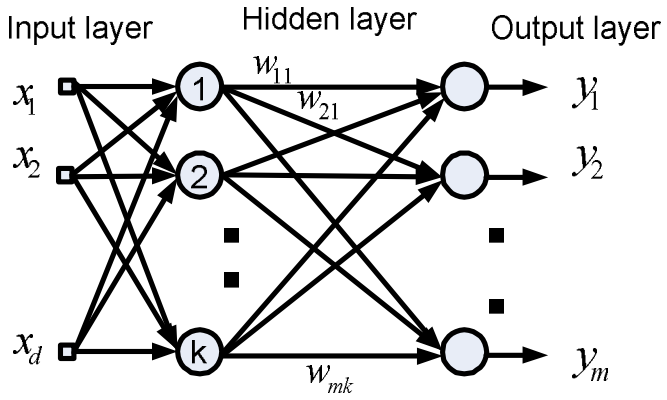


Figure 3.2. MLP with one hidden layer (d-k-m)

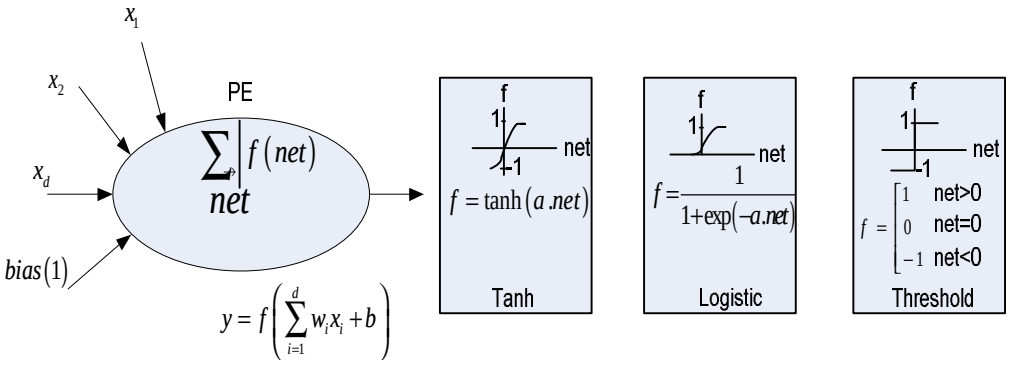


Figure 3.3. A PE and the most common nonlinearities

Many problems in engineering can be thought of in terms of a transformation of an input space, containing the input, to an output space where the desired response exists. For instance, dividing data into classes can be thought of as transforming the input into 0 and 1 responses that will code the classes. Likewise, identification of an unknown system can also be framed as a mapping (function approximation) from the input to the system output. The MLP is highly recommended for these applications (Principe, 2000).

3.3.1. Function of Each PE

Let us study briefly the function of a single PE with two inputs. If the nonlinearity is the threshold nonlinearity we can immediately see that the output is simply 1 and –1. The surface that divides these subspaces is called a separation surface, and in this case it is a line of equation:

$$y(w_1, w_2) = w_1 x_1 + w_2 x_2 + b = 0 \tag{3.1}$$

PE weights and the bias control the orientation and position of the separation line, respectively (Figure. 3.4.). In many dimensions the separation surface becomes an hyperplane of dimension one less than the dimensionality of the input space. So, each PE creates a dichotomy in the input space. For smooth nonlinearities the separation surface is not crisp; it becomes fuzzy but the same principles apply. In this case, the size of the weights controls the width of the fuzzy boundary (larger weights shrink the fuzzy boundary. The perceptron input/output map is built on linear separation surfaces, so the perceptron gives zero classification error only for linearly separable classes (Principe, 2000).

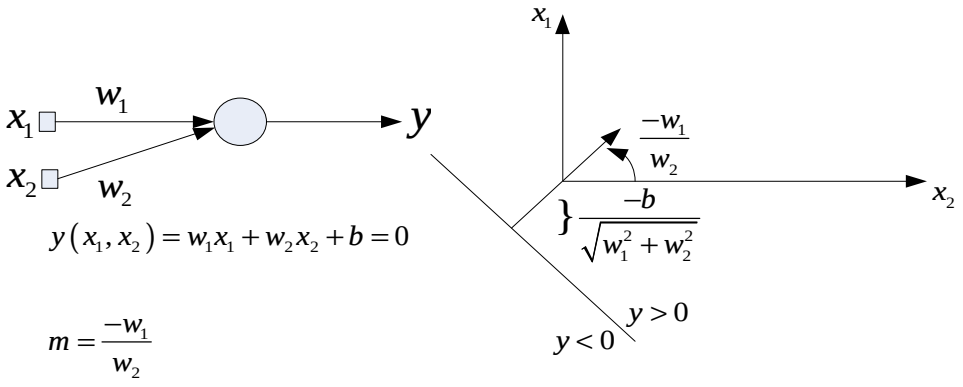
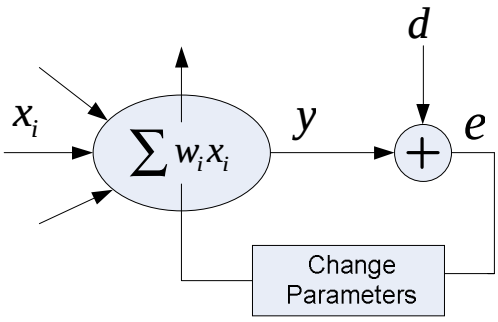


Figure 3.4. A two-input pe and its separation surface

3.3.2. Training of MLPs

One fundamental issue is how to adapt the weights w_i of the MLP to achieve a given input/output map. The core ideas are provided by the least mean square (LMS) algorithm. Let us review the theory here. Assume that linear PE ($f(net) = net$) is had and that one wants to adapt the weights as to minimize the square difference between the desired signal and the PE response (Figure 3.5.).



$$J = \sum_p \left(d_p - y_p\right)^2$$

P is the pattern number

$$\min J \rightarrow w_{opt} = R^{-1}P$$

Figure 3.5. Computing analytically optimal weights for linear Pe

This problem has an analytical solution known as the squares. The optimal weights are obtained as the product of the inverse of the input autocorrelation function (R^{-1}) and the cross-correlation vector (P) between the input and and the desired response. The analytical solution is equivalent to a search fort he minimum of the quadratic performance $J(w_i)$ using gradient descent, where the weights at each iteration k are adjusted by

$$w_i(k+1) = w_i(k) - h\nabla J_i(k), \nabla J_i = \frac{\partial J}{\partial w_i} \tag{3.2}$$

Where h is a small constant called the step size, and $\nabla J(k)$ is the gradient of the performance surface at iteration k. Bernand Widrow in the late 1960s proposed a very efficient estimate to compute the gradient at each iteration

$$\nabla J_i(k) = \frac{\partial}{\partial w_i} J(k), \qquad \approx \frac{1}{2}\frac{\partial}{\partial w_i}\left(e^2(k)\right) = -e(k)x_i(k) \tag{3.3}$$

which when substituted into Eq. 3.3 produces the so-called LMS algorithm. He showed that the LMS converged to the analytic solution provided the step size h is small enough. Since it is a steepest descent procedure, the largest step size is limited by the inverse of the largest value of the input autocorrelation matrix. The larger the step size (below this limit), the faster is the convergence, but the final values will "rattle" around the optimal value in a basin that has a radius proportional to the step size. Hence, there is a fundamental trade-off between speed of convergence and accuracy in the final weight values. One great appeal of the LMS algorithm is that it is very efficient (just one multiplication per weight) and requires only local quantities to be computed (Principe, 2000).

The LMS algorithm can be framed as a computation of partial derivatives of the cost with respect to the unknowns, i.e., the weight values. In fact, with the chain rule one writes

$$\frac{\partial J}{\partial w_i} = \frac{\partial J}{\partial y}\frac{\partial y}{\partial w_i} = \frac{\partial}{\partial y}\left(\sum (d-y)^2\right)\frac{\partial}{\partial w_i}\left(\sum w_i x_i\right) = e x_i \tag{3.4}$$

The LMS is obtained algorithm for the linear PE. What happens if the PE is nonlinear? If the nonlinearity is differentiable (smooth), here same method still can be apply, because of the chain rule, which prescribes that Figure 3.6.

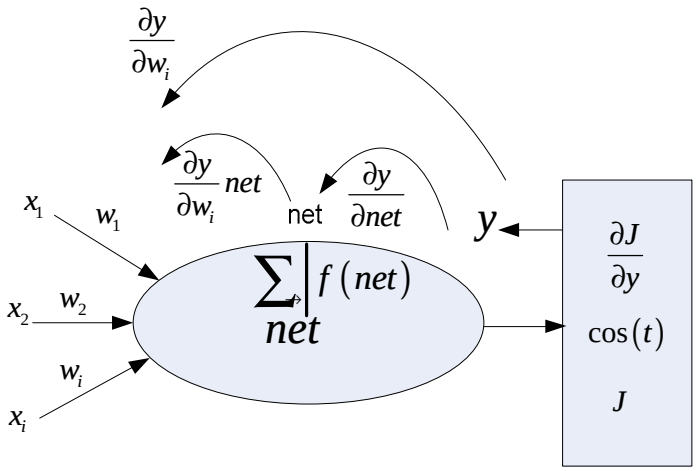


Figure 3.6. How to extend LMS to nonlinear Pes with the chain rule

$$\frac{\partial J}{\partial w_i} = \frac{\partial J}{\partial y}\frac{\partial y}{\partial net}\frac{\partial}{\partial w_i} net = -(d-y)f'(net)x_i = -ef'(net)x_i \tag{3.5}$$

where $f'(net)$ is the derivative of the nonlinearity computed at the operating point. Equation 3.5 is known as the delta rule, and it will train the perceptron. Note that throughout the derivation we skipped the pattern index p for simplicity, but this rule is applied for each input pattern. However, the delta rule cannot train MLPs since it requires the knowledge of the error signal at each PE.

The principle of the ordered derivatives can be extended to multilayer networks, provided we organize the computations in flows of activation and error propagation. The principle is very easy to understand, but a little complex to formulate in equation form (Principe, 2000).

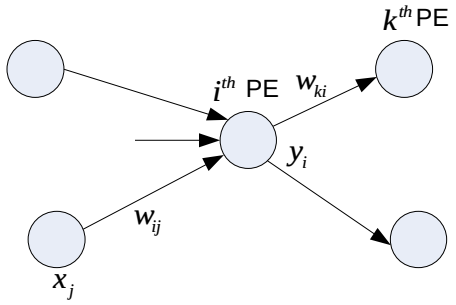


Figure 3.7. How to adapt the weights connected to ith PE

Suppose that to adapt the weights connected to a hidden layer PE is wanted, the i th PE (Figure 3.7.). One can decompose the computation of the partial derivative of the cost with respect to the weight w_{ij} as

$$\frac{\partial J}{\partial w_{ij}} = \frac{\partial J}{\partial y_i}\frac{\partial y_i}{\partial net_i}\frac{\partial}{\partial w_{ij}} net_i \tag{3.6}$$

i.e., the partial derivative with respect to the weight is the product of the partial derivative with respect to the PE state – part 1 in Eq. 3.6 – times the partial derivative of the local activation to the weights – part 2 in Eq. 3.4 This last quantity is exactly the same as fort he nonlinear PE ($f'(net_i)x_j$), so the big issue is the computation of $\frac{\partial J}{\partial y}$. For an output PE, $\frac{\partial J}{\partial y}$ becomes the infected error e in Eq. (3.4) For the hidden ith PE $\frac{\partial J}{\partial y}$ is evaluated by summing all the errors that reach the PE from the top layer through the topology when the injected errors e_k are clamped at the top layer, or in an equation

$$\frac{\partial J}{\partial y_i} = \left(\sum_k \frac{\partial J}{\partial y_k}\frac{\partial y_k}{\partial net_k}\frac{\partial}{\partial y_i} net_k \right) = \sum_k e_k f'(net_k) w_{ki} \tag{3.7}$$

Substituting back in Eq. 3.6. Finally,

$$\frac{\partial J}{\partial w_{ij}} = -x_j f'(net_i)\left(\sum_k e_k f'(net_k) w_{ki}\right) \qquad (3.8)$$

1 2

is got.

This equation embodies the back-propagation training algorithm. It can be rewritten as the product of a local activation (part 1) and a local error (part 2), exactly as the LMS and the delta rules. But now the local error is a composition of errors that flow through the topology, which becomes equivalent to the existence of a desired response at the PE .

There is an intrinsic flow in the implementation of the back-propagation algorithm: first, inputs are applied to the net and activations computed everywhere to yield the output activation. Second, the external errors are computed by subtracting the net output from the desired response. Third, these external errors are utilized in Eq. 3.8 to compute the local errors for the layer immediately preceding the output layer, and the computations chained up to the input layer. Once all the local errors are available, Eq. 3.1 can be used to update every weight. These three steps are then repeated for other training patterns until the error is acceptable. Step three is equivalent to injecting the external errors in the dual topology and back- propagating them up to the input layer [Haykin, 1994]. The dual topology is obtained from the original one by reversing data flow and substituting summing junctions by splitting nodes and vice versa. The error at each PE of the dual topology is then multiplied by the activation of the original network to compute the weight updates. So, effectively the dual topology is being used to compute the local errors which makes the procedure highly efficient. This is the reason back-propagation trains a network of N weights with a number of multiplications proportional to N, ($O(N)$), instead of ($O(N^2)$) for previous methods of computing partial derivatives known in control theory. Using the dual topology to implement back-propagation is the best and most general method to program the algorithm in a digital computer (Principe, 2000).

3.3.3. Example-1: Training a Neural Network

To respond with a desired output $d = 0.9$ at the output y_1 to the augmented input vector $x = [1, x_1, x_2]^T = [1,1,3]^T$ will be train the two-layer perceptron network shown in Figure 3.8. The network weights have been initialized as shown. Assuming sigmoidal activation functions at the outputs of the hidden and output nodes and learning gain $h = 0.1$ and no momentum term, here will be analyzed a single feed-forward and back-propagation network with activation function $f(z) = \frac{1}{1+e^{-z}}$ (Nguyen, Prasad, L.Walker, and A. Walker, 2002)

The weight matrices fort he specified network are

$$d = 0.9 \tag{3.9}$$

$$x = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 3 \end{bmatrix} \tag{3.10}$$

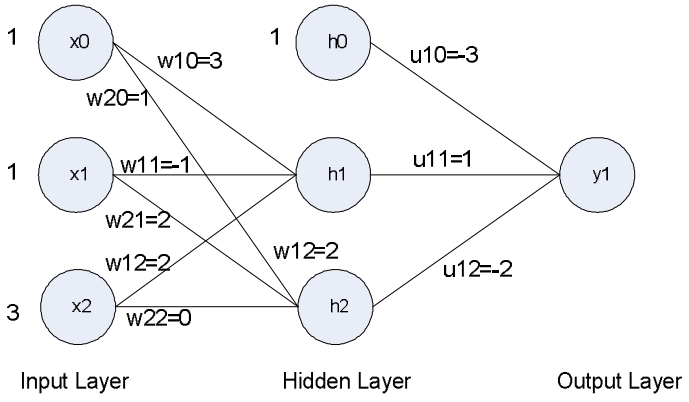


Figure 3.8. Two-layer feed-forward perceptron

$$W = \begin{bmatrix} w_{10}, w_{11,} w_{12} \\ w_{20}, w_{21}, w_{22} \end{bmatrix} = \begin{bmatrix} 3,-1,2 \\ 1,2,0 \end{bmatrix} \tag{3.11}$$

$$u = \begin{bmatrix} u_{10} \\ u_{11} \\ u_{12} \end{bmatrix} = \begin{bmatrix} -3 \\ 1 \\ -2 \end{bmatrix} \tag{3.12}$$

The output of the hidden node is given by $a_i = w_j \bullet x$ and $h_j = f(a_j) = \frac{1}{1+e^{-a_j}}$, from which we calculate

$$a_1 = (3)(1) + (-1)(1) + (2)(3) = 8 \tag{3.13}$$

$$a_2 = (1)(1) + (2)(1) + (0)(3) = 3 \tag{3.14}$$

$$h = [h_0, h_1, h_2]^T = [1, 0.9997, 0.9526]^T \tag{3.15}$$

The network output is therefore

$$y_1 = f(u.h) = f((-3)(1) + (1)(0.9997) - (2)(0.9526)) = f(-3.9055) = 0.0197 \tag{3.16}$$

The error signals are computed as follows:

$$d_{y1} = (y_1 - d)(y_1)(1 - y_1) = (0.0197 - 0.9)(0.0197)(1 - 0.0197) = -0.0170 \tag{3.17}$$

$$d_{h_1} = (h_1)(1 - h_1)(d_{y_1})(u_{11}) = (0.9997)(1 - 0.9997)(-0.0170)(1) = -5.098x10^{-6} \tag{3.18}$$

$$d_{h_2} = (h_2)(1 - h_2)(d_{y_1})(u_{12}) = (0.9526)(1 - 0.9526)(-0.0170)(-2) = 8.056x10^{-4} \tag{3.19}$$

Compute all the nine weight updates:

$$1.\Delta w_{10} = -h\left(d_{h_1}\right)\left(x_0\right) = -\left(0.1\right)\left(-5.098x10^{-6}\right)\left(1\right) = 5.098x10^{-7} \tag{3.20}$$

$$2.\Delta w_{11} = -h\left(d_{h_1}\right)\left(x_1\right) = -\left(0.1\right)\left(-5.098x10^{-6}\right)\left(1\right) = 5.098x10^{-7} \tag{3.21}$$

$$3.\Delta w_{12} = -h\left(d_{h_1}\right)\left(x_2\right) = -\left(0.1\right)\left(-5.098x10^{-6}\right)\left(3\right) = 1.5294x10^{-6} \tag{3.22}$$

$$4.\Delta w_{20} = -h\left(d_{h_2}\right)\left(x_0\right) = -\left(0.1\right)\left(8.056x10^{-4}\right)\left(1\right) = -8.056x10^{-5} \tag{3.23}$$

$$5.\Delta w_{21} = -h\left(d_{h_2}\right)\left(x_1\right) = -\left(0.1\right)\left(8.056x10^{-4}\right)\left(1\right) = -8.056x10^{-5} \tag{3.24}$$

$$6.\Delta w_{20} = -h\left(d_{h_2}\right)\left(x_2\right) = -\left(0.1\right)\left(8.056x10^{-4}\right)\left(3\right) = -2.417x10^{-4} \tag{3.25}$$

$$7.\Delta u_{10} = -h\left(d_{y_1}\right)\left(h_0\right) = -\left(0.1\right)\left(-0.0170\right)\left(1\right) = 0.0017 \tag{3.26}$$

$$8.\Delta u_{11} = -h\left(d_{y_1}\right)\left(h_1\right) = -\left(0.1\right)\left(-0.0170\right)\left(0.9997\right) = 0.0017 \tag{3.27}$$

$$9.\Delta u_{12} = -h\left(d_{y_1}\right)\left(h_2\right) = -\left(0.1\right)\left(-0.0170\right)\left(0.9526\right) = 0.0016 \tag{3.28}$$

Using the delta rule, the new weights are update as

$$W = \begin{bmatrix} w_{10} + \Delta w_{10}, w_{11} + \Delta w_{11}, w_{12} + \Delta w_{12} \\ w_{20} + \Delta w_{20}, w_{21} + \Delta w_{21}, w_{22} + \Delta w_{22} \end{bmatrix} \tag{3.29}$$

$$u = \begin{bmatrix} u_{10} + \Delta u_{10} \\ u_{11} + \Delta u_{11} \\ u_{12} + \Delta u_{12} \end{bmatrix} \tag{3.30}$$

In this example, we see that since the weight updates are very small, the network is very nearly trained to obtain the desired output $d = 0.9$. Repeated

computations will make he output y_1 tend toward any desired near-zero tolerance (Nguyen, Prasad, L.Walker and A. Walker, 2002).

3.4. Matlab Neural Network Toolbox for Back-Propagation Algorithm

Back-Propagation was created by generalizing the Widrow-Hoff learning rule to multiple-layer networks and nonlinear differentiable transfer functions. Input vectors and the corresponding target vectors are used to train a network until it can approximate a function. Input and output vectors must be defined by designer. Networks with biases, a sigmoid layer, and a linear output layer are capable of approximating any function with a finite number of discontinuities. Standard back-propagation is a gradient descent algorithm, as is the Widrow-Hoff learning rule. A gradient descent algorithm moves network weights along the negative of the gradient of the performance function. The term back-propagation refers to the manner in which the gradient is computed for nonlinear multilayer networks. There are a number of variations on the basic algorithm that are based on other standard optimization techniques, such as conjugate gradient and Newton methods. The Neural Network Toolbox implements a number of these variations. This chapter explains how to use each of these routines and discusses the advantages and disadvantages of each. Properly trained back-propagation networks tend to give reasonable answers when presented with inputs that they have never seen. Typically, a new input leads to an output similar to the correct output for input vectors used in training that are similar to the new input being presented. This generalization property makes it possible to train a network on a representative set of input/target pairs and get good results without training the network on all possible input/output pairs. There are two features of the Neural Network Toolbox that are designed to improve network generalization - regularization and early stopping. These features and their use are discussed later in this chapter (Demuth, 2002).

The primary objective of this chapter is to explain how to use the back-propagation training functions in the toolbox to train feed-forward neural networks to solve specific problems. There are generally four steps in the training process:

- Assemble the training data
- Create the network object
- Train the network
- Simulate the network response to new inputs ((Demuth, 2002).

3.4.1. Architecture

This section presents the architecture of the network that is most commonly used with the back-propagation algorithm - the multilayer feed-forward network. The routines in the Neural Network Toolbox can be used to train more general networks; some of these will be briefly discussed in later sections.

3.4.2. Neuron Model (tansig, logsig, purelin)

An elementary neuron with R inputs is shown below. Each input is weighted with an appropriate w. The sum of the weighted inputs and the bias forms the input to the transfer function f. Neurons may use any differentiable transfer function f to generate their output. Multilayer Networks often use the log-sigmoid transfer function logsig. Figure 3.9. illustrates an multilayer networks (Demuth, 2002).

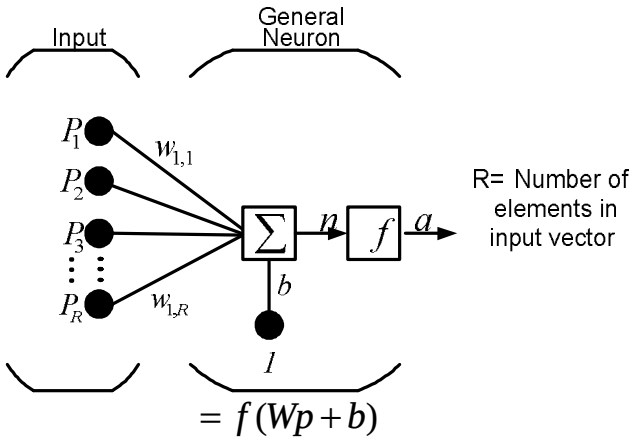


Figure 3.9. Multilayer networks

Figure 3.10. illustrates an log-sigmoud transfer function

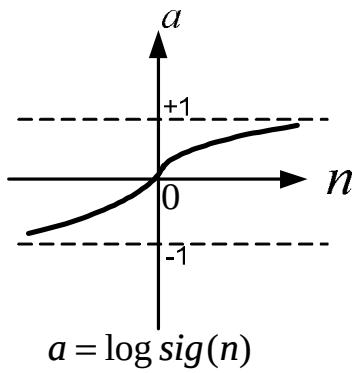


Figure 3.10. Log-sigmoid transfer function

The function logsig generates outputs between 0 and 1 as the neuron's net input goes from negative to positive infinity. Alternatively, multilayer networks may use the tan-sigmoid transfer function tansig. Figure 3.11. illustrates an tan-sigmoud transfer function (Demuth, 2002).

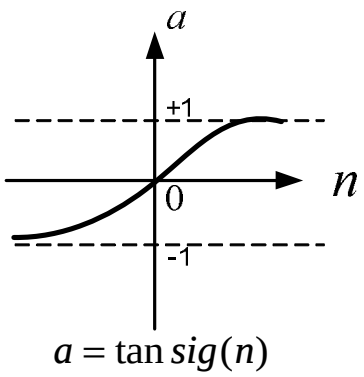


Figure 3.11. Tan-sigmoid transfer function

Figure 3.12. illustrates an linear transfer function.

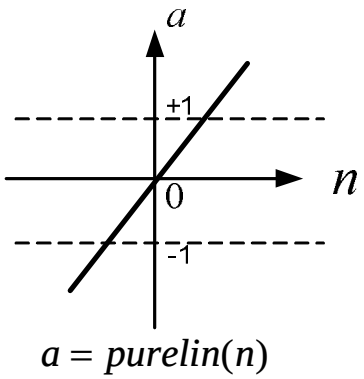


Figure 3.12. Linear transfer function

If the last layer of a multilayer network has sigmoid neurons, then the outputs of the network are limited to a small range. If linear output neurons are used the network outputs can take on any value.

In back-propagation it is important to be able to calculate the derivatives of any transfer functions used. Each of the transfer functions above, tansig, logsig, and purelin, have a corresponding derivative function: dtansig, dlogsig, and dpurelin. To get the name of a transfer function's associated derivative function, call the transfer function with the string 'deriv'.

Tansig('deriv') 3.31)

Ans=dtansig (3.32)

The three transfer functions described here are the most commonly used transfer functions for back-propagation, but other differentiable transfer functions can be created and used with back-propagation if desired (Demuth, 2002).

3.4.3. Feed-Forward Network

A single-layer network of S logsig neurons having R inputs is shown in Figure 3.13. in full detail on the left and with a layer diagram on the right. Feed-forward networks often have one or more hidden layers of sigmoid neurons followed by an output layer of linear neurons. Multiple layers of neurons with nonlinear transfer functions allow the network to learn nonlinear and linear relationships between input and output vectors. The linear output layer lets the network produce values outside the range –1 to +1. On the other hand, if it is wanted to constrain the outputs of a network (such as between 0 and 1), then the output layer should use a sigmoid transfer function (such as logsig).

3.4.4. Creating a Network (newff)

The first step in training a feed-forward network is to create the network object. The function newff creates a feed-forward network. It requires four inputs and returns the network object. The first input is an R by 2 matrix of minimum and maximum values for each of the R elements of the input vector. The second input is an array containing the sizes of each layer. The third input is a cell array containing the names of the transfer functions to be used in each layer. The final input contains the name of the training function to be used. For example, the following command creates a two-layer network. There is one input vector with two elements. The values for the first element of the input vector range between -1 and 2, the values of the second element of the input vector range between 0 and 5. There are three neurons in the first layer and one neuron in the second (output) layer.

The transfer function in the first layer is tan-sigmoid, and the output layer transfer function is linear. The training function is traingd (which is described in a later section).

net=newff([-1 2; 0 5],[3,1],{'tansig','purelin'},'traingd'); (3.33)

This command creates the network object and also initializes the weights and biases of the network; therefore the network is ready for training. There are times when you may want to reinitialize the weights, or to perform a custom initialization. The next section explains the details of the initialization process (Demuth, 2002).

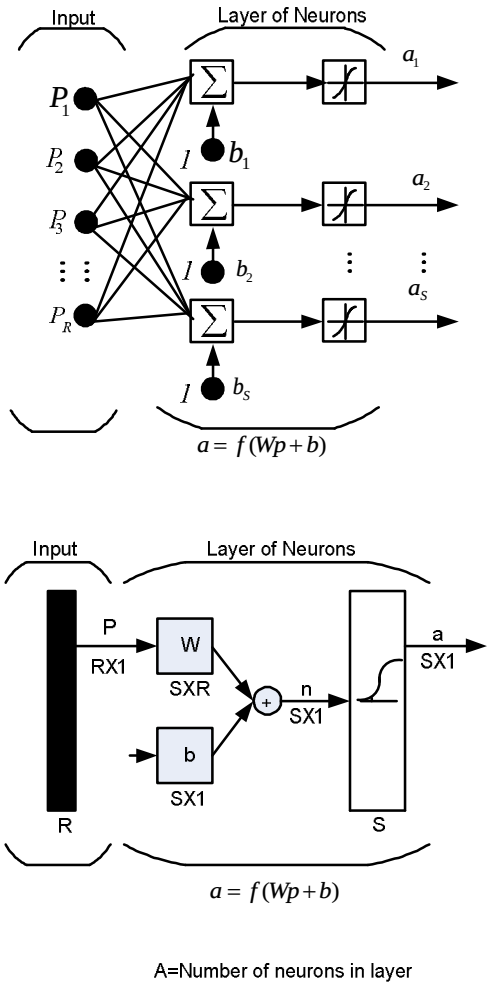


Figure 3.13. Feed-forward networks

3.4.5. Initializing Weights (init)

Before training a feed-forward network, the weights and biases must be initialized. The newff command will automatically initialize the weights, but you may want to reinitialize them. This can be done with the command init. This function takes a network object as input and returns a network object with all weights and biases initialized. Here is how a network is initialized (or reinitialized):

```
net = init(net);
```
(3.34)

3.4.6. Simulation (sim)

The function sim simulates a network. sim takes the network input p, and the network object net, and returns the network outputs a. Here is how you can use sim to simulate the network we created above for a single input vector:

```
p = [1;2];
```
(3.35)

```
a = sim(net,p)
```
(3.36)

```
a =-0.1011
```
(3.37)

If these commands is tried, output may be different, depending on the state of random number generator when the network was initialized. Below, sim is called to calculate the outputs for a concurrent set of three input vectors. This is the batch mode form of simulation, in which all of the input, vectors are place in one matrix. This is much more efficient than presenting the vectors one at a time.

```
p = [1 3 2;2 4 1];
```
(3.38)

```
a=sim(net,p)
```
(3.39)

```
a =-0.1011 -0.2308 0.4955
```
(3.40)

3.4.7. Training

Once the network weights and biases have been initialized, the network is ready for training. The network can be trained for function approximation (nonlinear regression), pattern association, or pattern classification. The training process requires a set of examples of proper network behavior - network inputs p and target outputs t. During training the weights and biases of the network are iteratively adjusted to minimize the network performance function net.performFcn. The default performance function for feed-forward networks is mean square error mse - the average squared error between the network outputs a and the target outputs t (Demuth, 2002).

3.4.8. Back-Propagation Algorithm

There are many variations of the back-propagation algorithm, several of which we discuss in this chapter. The simplest implementation of back-propagation learning updates the network weights and biases in the direction in which the performance function decreases most rapidly - the negative of the gradient. One iteration of this algorithm can be written:

$$x_{k+1} = x_k - a_k g_k \tag{3.41}$$

where x_k is a vector of current weights and biases, g_k is the current gradient, and a_k is the learning rate.

There are two different ways in which this gradient descent algorithm can be implemented: incremental mode and batch mode. In the incremental mode, the gradient is computed and the weights are updated after each input is applied to the network. In the batch mode all of the inputs are applied to the network before the weights are updated. The next section describes the batch mode of training; incremental training will be discussed in a later section.

3.4.9. Batch Training (train)

In batch mode the weights and biases of the network are updated only after the entire training set has been applied to the network. The gradients calculated at each training example are added together to determine the change in the weights and biases.

3.4.10. Batch Gradient Descent (traingd)

The batch steepest descent training function is traingd. The weights and biases are updated in the direction of the negative gradient of the performance function. If you want to train a network using batch steepest descent, you should set the network trainFcn to traingd, and then call the function train. There is only one training function associated with a given network.

There are seven training parameters associated with traingd: epochs, show, goal, time, min_grad, max_fail, and lr. The learning rate lr is multiplied times the negative of the gradient to determine the changes to the weights and biases. The larger the learning rate, the bigger the step. If the learning rate is made too large, the algorithm becomes unstable. If the learning rate is set too small, the algorithm takes a long time to converge.

The training status is displayed for every show iteration of the algorithm. The other parameters determine when the training stops. The training stops if the number of iterations exceeds epochs, if the performance function drops below goal, if the magnitude of the gradient is less than mingrad, or if the training time is longer than time seconds. We discuss max_fail, which is associated with the early stopping technique, in the section on improving generalization (Demuth, 2002).

The following code creates a training set of inputs p and targets t. For batch training, all of the input vectors are placed in one matrix.

```
p = [-1 -1 2 2;0 5 0 5];
```
(3.42)

```
t = [-1 -1 1 1];
```
(3.43)

Next, the feed-forward network is created. Here the function minmax is used to determine the range of the inputs to be used in creating the network.

```
net=newff(minmax(p),[3,1],{'tansig','purelin'},'traingd');
```
(3.44)

At this point, some of the default training parameters can modify following commands.

```
net.trainParam.show = 50;
```
(3.45)

```
net.trainParam.lr = 0.05;
```
(3.46)

```
net.trainParam.epochs = 300;
```
(3.47)

```
net.trainParam.goal = 1e-5;
```
(3.48)

The above commands are not necessary to use the default training parameters Now the network can be trained.

```
[net,tr]=train(net,p,t);
```
(3.49)

```
TRAINGD, Epoch 0/300, MSE 1.59423/1e-05, Gradient 2.76799/1e-10
TRAINGD, Epoch 50/300, MSE 0.00236382/1e-05, Gradient0.0495292/1e-10
TRAINGD, Epoch 100/300, MSE 0.000435947/1e-05, Gradient0.0161202/1e-10
TRAINGD, Epoch 150/300, MSE 8.68462e-05/1e-05, Gradient0.00769588/1e-10
TRAINGD, Epoch 200/300, MSE 1.45042e-05/1e-05, Gradient0.00325667/1e-10
TRAINGD, Epoch 211/300, MSE 9.64816e-06/1e-05, Gradient0.00266775/1e-10
TRAINGD, Performance goal met.
```

Now the trained network can be simulated to obtain its response to the inputs in the training set.

```
a = sim(net,p)
```
(3.50)

a = -1.0010 -0.9989 1.0018 0.9985 (3.51)

3.4.11. Batch Gradient Descent with Momentum (traingdm)

In addition to traingd, there is another batch algorithm for feed-forward networks that often provides faster convergence - traingdm, steepest descent with momentum. Momentum allows a network to respond not only to the local gradient, but also to recent trends in the error surface. Acting like a low-pass filter, momentum allows the network to ignore small features in the error surface. Without momentum a network may get stuck in a shallow local minimum. With momentum a network can slide through such a minimum.

Momentum can be added to back-propagation learning by making weight changes equal to the sum of a fraction of the last weight change and the new change suggested by the back-propagation rule. The magnitude of the effect that the last weight change is allowed to have is mediated by a momentum constant, mc, which can be any number between 0 and 1. When the momentum constant is 0, a weight change is based solely on the gradient. When the momentum constant is 1, the new weight change is set to equal the last weight change and the gradient is simply ignored. The gradient is computed by summing the gradients calculated at each training example, and the weights and biases are only updated after all training examples have been presented. If the new performance function on a given iteration exceeds the performance function on a previous iteration by more than a predefined ratio max_perf_inc (typically 1.04), the new weights and biases are discarded, and the momentum coefficient mc is set to zero. The batch form of gradient descent with momentum is invoked using the training function traingdm. The traingdm function is invoked using the same steps shown above for the traingd function, except that the mc, lr and max_perf_inc learning parameters can all be set. In the following code we recreate our previous network and retrain it using gradient descent with momentum. The training parameters for traingdm are the same as those for traingd, with the addition of the momentum factor mc and the maximum performance increase

max_perf_inc. (The training parameters are reset to the default values whenever net.trainFcn is set to traingdm.)

```
p = [-1 -1 2 2;0 5 0 5];
t = [-1 -1 1 1];
net=newff(minmax(p),[3,1],{'tansig','purelin'},'traingdm');
net.trainParam.show = 50;
net.trainParam.lr = 0.05;
net.trainParam.mc = 0.9;
net.trainParam.epochs = 300;
net.trainParam.goal = 1e-5;
[net,tr]=train(net,p,t);
TRAINGDM, Epoch 0/300, MSE 3.6913/1e-05, Gradient 4.54729/1e-10
TRAINGDM, Epoch 50/300, MSE 0.00532188/1e-05, Gradient0.213222/1e-10
TRAINGDM, Epoch 100/300, MSE 6.34868e-05/1e-05, Gradient0.0409749/1e-10
TRAINGDM, Epoch 114/300, MSE 9.06235e-06/1e-05, Gradient0.00908756/1e-10
TRAINGDM, Performance goal met.
a = sim(net,p)
a =-1.0026 -1.0044 0.9969 0.9992
```

3.4.12. Faster Training

The previous section presented two back-propagation training algorithms: gradient descent, and gradient descent with momentum. These two methods are often too slow for practical problems. In this section several high performance algorithms that can converge from ten to one hundred times faster than the algorithms discussed previously will be discussed. All of the algorithms in this section operate in the batch mode and are invoked using train. These faster algorithms fall into two main categories. The first category uses heuristic techniques, which were developed from an analysis of the performance of the standard steepest descent algorithm. One heuristic modification is the momentum technique, which was presented in the

previous section. This section discusses two more heuristic techniques: variable learning rate back-propagation, traingda; and resilient back-propagation trainrp. The second category of fast algorithms uses standard numerical optimization techniques. Later in this section we present three types of numerical optimization techniques for neural network training: conjugate gradient (traincgf, traincgp, traincgb, trainscg), quasi-Newton (trainbfg, trainoss), and Levenberg-Marquardt (trainlm) (Demuth, 2002).

3.4.12.1. Levenberg-Marquardt (trainlm)

Like the quasi-Newton methods, the Levenberg-Marquardt algorithm was designed to approach second-order training speed without having to compute the Hessian matrix. When the performance function has the form of a sum of squares (as is typical in training feed-forward networks), then the Hessian matrix can be approximated as:

$$H = J^T J \tag{3.52}$$

and the gradient can be computed as:

$$g = J^T e \tag{3.53}$$

where is the Jacobian matrix that contains first derivatives of the network errors with respect to the weights and biases, and **e** is a vector of network errors. The Jacobian matrix can be computed through a standard back-propagation technique that is much less complex than computing the Hessian matrix. The Levenberg-Marquardt algorithm uses this approximation to the Hessian matrix in the following Newton-like update:

$$x_{k+1} = x_k - \left[J^T J + mI\right]^{-1} J^T e \tag{3.54}$$

In the following code, we reinitialize our previous network and retrain it using the Levenberg-Marquardt algorithm. The training parameters for trainlm are epochs, show, goal, time, min_grad, max_fail, mu, mu_dec, mu_inc, mu_max, mem_reduc. The first six parameters earlier have been discussed. The parameter mu is the initial value for μ. This value is multiplied by mu_dec whenever the performance function is reduced by a step. It is multiplied by mu_inc whenever a step would increase the performance function. If mu becomes larger than mu_max, the algorithm is stopped. The parameter mem_reduc is used to control the amount of memory used by the algorithm. It is discussed in the next section. The parameters show and epoch are set to 5 and 300, respectively.

```
p = [-1 -1 2 2;0 5 0 5];
t = [-1 -1 1 1];
net=newff(minmax(p),[3,1],{'tansig','purelin'},'trainlm');
net.trainParam.show = 5;
net.trainParam.epochs = 300;
net.trainParam.goal = 1e-5;
[net,tr]=train(net,p,t);
TRAINLM, Epoch 0/300, MSE 2.7808/1e-05, Gradient 7.77931/1e-10
TRAINLM, Epoch 4/300, MSE 3.67935e-08/1e-05, Gradient 0.000808272/1e-10
TRAINLM, Performance goal met.
a = sim(net,p)
a = -1.0000 -1.0000 1.0000 0.9996
```

This algorithm appears to be the fastest method for training moderate-sized feed-forward neural networks (up to several hundred weights). It also has a very efficient Matlab implementation, since the solution of the matrix equation is a built-in function, so its attributes become even more pronounced in a Matlab setting. Levenberg-Marquardt algorithm is a high performance algorithm. So, it is used for simulation of the network.

3.4.12.2. Reduced Memory Levenberg-Marquardt (trainlm)

The main drawback of the Levenberg-Marquardt algorithm is that it requires the storage of some matrices that can be quite large for certain problems. The size of the Jacobian matrix is Qxn, where Q is the number of training sets and n is the number of weights and biases in the network. It turns out that this matrix does not have to be computed and stored as a whole. For example, if we were to divide the Jacobian into two equal submatrices we could compute the approximate Hessian matrix as follows:

$$H = j^T J = \begin{bmatrix} J_1^T J_2^T \end{bmatrix} \begin{bmatrix} J_1 \\ J_2 \end{bmatrix} = J_1^T J_1 + J_2^T J_2 \tag{3.55}$$

Therefore, the full Jacobian does not have to exist at one time. The approximate Hessian can be computed by summing a series of subterms. Once one subterm has been computed, the corresponding submatrix of the Jacobian can be cleared. When using the training function trainlm, the parameter mem_reduc is used to determine how many rows of the Jacobian are to be computed in each submatrix. If mem_reduc is set to 1, then the full Jacobian is computed, and no memory reduction is achieved. If mem_reduc is set to 2, then only half of the Jacobian will be computed at one time. This saves half of the memory used by the calculation of the full Jacobian. There is a drawback to using memory reduction. A significant computational overhead is associated with computing the Jacobian in submatrices. If enough memory İS available, then it is better to set mem_reduc to 1 and to compute the full Jacobian. If you have a large training set, and you are running out of memory, then you should set mem_reduc to 2, and try again. If you still run out of memory, continue to increase mem_reduc. Even if memory reduction is used, the Levenberg-Marquardt algorithm will always compute the approximate Hessian matrix, which has dimensions nxn. If network is very large, then memory may run out. If this is the

case, then to try trainscg, trainrp, or one of the conjugate gradient algorithms will be wanted (Demuth, 2002).

3.4.13. Example-2: Training a Neural Network

Figure 3.14. illustrates a perceptron architecture (Hagan, Demuth and Beale, 2002)

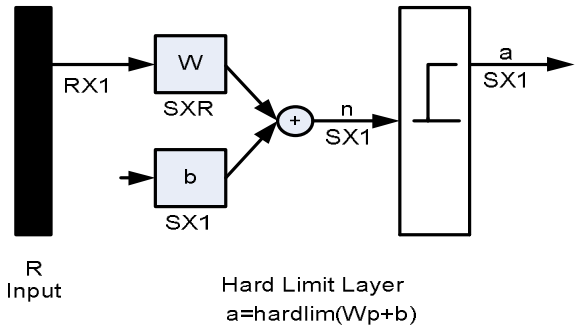


Figure 3.14. Perceptron architecture

$$a = hard\lim(Wp + b) \tag{3.56}$$

$$W = \begin{bmatrix} {}_1w^T \\ {}_2w^T \\ . \\ . \\ . \\ {}_Sw^T \end{bmatrix} \tag{3.57}$$

$$a_i = hard\lim(n_i) = hard\lim({}_iw^T p + b_i) \tag{3.58}$$

Decision Boundary of the network is

$$_i w^T p + b_i = 0 \tag{3.59}$$

The decision boundry is always orthogonal to the weight vector. Single-layer perceptrons can only classify linearly separable vectors. The perceptron learning rule is

$$W^{new} = W^{old} + ep^T \tag{3.60}$$

$$b^{new} = b^{old} + e \tag{3.61}$$

where $e = t - a$. Here

t=Desired value
a=Output of the network

Now the bias is found for each perceptron by picking a point on the decision boundry and satisfying Eq.(3.62)

$$_1 w^T p + b = 0 \tag{3.62}$$

$$b = -_1 w^T p \tag{3.63}$$

The following classification problem can solve the perceptron rule.

$$\left\{p_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, t_1 = 0\right\}, \left\{p_2 = \begin{bmatrix} 1 \\ -2 \end{bmatrix}, t_2 = 1\right\}, \left\{p_3 = \begin{bmatrix} -2 \\ 2 \end{bmatrix}, t_3 = 0\right\}, \left\{p_4 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, t_4 = 0\right\} \tag{3.64}$$

$$W(0) = [0,0], b(0) = 0. \tag{3.65}$$

Calculating is started the perceptron's output a for the first input vector p_1 ,using the initial weights and bias.

$$a = hard\lim(W(0)p_1 + b(0)) = hard\lim\left([0,0]\begin{bmatrix}2\\2\end{bmatrix} + 0\right) = hard\lim(0) = 1 \quad (3.66)$$

The output a does not equal the target value t_1, so the perceptron rule is used to find new weights and biases based on the error.

$$e = t_1 - a = 0 - 1 = -1 \quad (3.67)$$

$$W(1) = W(0) + ep_1^T = [0,0] + (-1)[2,2] = [-2,-2] \quad (3.68)$$

$$b(1) = b(0) + e = 0 + (-1) = -1 \quad (3.69)$$

Now using the updated weights and bias is applied for the second input vector p_2 (Hagan, Demuth and Beale, 2002).

$$a = hard\lim(W(1)p_2 + b(1)) = hard\lim\left([-2,-2]\begin{bmatrix}1\\-2\end{bmatrix} - 1\right) = hard\lim(1) = 1 \quad (3.70)$$

This time the output a is equal to the target t_2. Application of the perceptron rule will not result in any changes.

$$W(2) = W(1) \quad (3.71)$$

$$b(2) = b(1) \quad (3.72)$$

The last input vector p_4 is moved.

$$a = hard\lim(W(3)p_4 + b(3)) = hard\lim\left(\begin{bmatrix}-2,-2\end{bmatrix}\begin{bmatrix}-1\\1\end{bmatrix} - 1\right) = hard\lim(-1) = 0 \tag{3.73}$$

This time the output a does not equal the appropriate target t_4. The perceptron rule will result in a new set of values for W and b.

$$e = t_4 - a = 1 - 0 = 1 \tag{3.74}$$

$$W(4) = W(3) + ep_4^T = \begin{bmatrix}-2,-2\end{bmatrix} + (1)\begin{bmatrix}-1,-1\end{bmatrix} = \begin{bmatrix}-3,-1\end{bmatrix} \tag{3.75}$$

$$b(4) = b(3) + e = -1 + 1 = 0 \tag{3.76}$$

Now the firs vector p_1 must check again. This time the output a is equa to the associated target t_1.

$$a = hard\lim(W(4)p_1 + b(4)) = hard\lim\left(\begin{bmatrix}-3,-1\end{bmatrix}\begin{bmatrix}2\\2\end{bmatrix} + 0\right) = hard\lim(-8) = 0 \tag{3.77}$$

Therefore there are no changes.

$$W(5) = W(4) \tag{3.78}$$

$$b(5) = b(4) \tag{3.79}$$

The second presentation of p_2 results in an error and therefore a new set of weight and bias values.

$$a = hard\lim(W(5)p_2 + b(5)) = hard\lim\left([-3,-1]\begin{bmatrix}1\\-2\end{bmatrix} + 0\right) = hard\lim(-1) = 0 \tag{3.80}$$

Here are those new values:

$$e = t_2 - a = 1 - 0 = 1 \tag{3.81}$$

$$W(6) = W(5) + ep_2^T = [-3,-1] + (1)[1,-2] = [-2,-3] \tag{3.82}$$

$$b(6) = b(5) + e = 0 + 1 = 1 \tag{3.83}$$

Cycling through each input vector once more results in no errors.

$$a = hard\lim(W(6)p_3 + b(6) = hard\lim\left([-2,-3]\begin{bmatrix}-2\\2\end{bmatrix} + 1\right) = 0 = t_3 \tag{3.84}$$

$$a = hard\lim(W(6)p_4 + b(6) = hard\lim\left([-2,-3]\begin{bmatrix}-1\\1\end{bmatrix} + 1\right) = 1 = t_4 \tag{3.85}$$

$$a = hard\lim(W(6)p_1 + b(6) = hard\lim\left([-2,-3]\begin{bmatrix}2\\2\end{bmatrix} + 1\right) = 0 = t_1 \tag{3.86}$$

$$a = hard\lim(W(6)p_2 + b(6) = hard\lim\left([-2,-3]\begin{bmatrix}1\\2\end{bmatrix} + 1\right) = 1 = t_2 \tag{3.87}$$

Therefore the algorithm has converged. The final solution is:

$$W = [-2,-3], b = 1 \tag{3.88}$$

Now the training data and the decision boundary of the solution. The decision boundary is given by

$$n = Wp + b = w_{1,1}p_1 + w_{1,2}p_2 + b = -2p_1 - 3p_2 + 1 = 0 \tag{3.89}$$

To find the p_2 intercept of the decision boundary, set $p_1 = 0$:

$$p_2 = -\frac{b}{w_{1,2}} = -\frac{1}{-3} = \frac{1}{3} \text{ if } p_1 = 0. \tag{3.90}$$

To find the p_1 intercept, set $p_2 = 0$

$$p_1 = -\frac{b}{w_{1,1}} = -\frac{1}{-2} = \frac{1}{2} \text{ if } p_2 = 0 \tag{3.91}$$

The resulting decision boundary is illustrated in Figure 3.15.

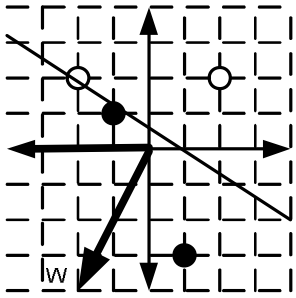


Figure 3.15. Decision boundary for classification problem

The decision boundary falls across one of the training vectors. This is acceptable, given the problem definition, since the hard limit function returns 1 when given an input of 0, and the target for the vector in question is indeed 1 (Hagan, Demuth and Beale, 2002).

4. THE DESIGN OF THE NEURAL NETWORK-BASED ULTRASONIC MEASUREMENT SENSOR FOR FLUIDS

4.1. Introduction

Ultrasonic sensing systems are often used in robotics such as obstacle avoidance, distance measurements, map construction of the robot's environment and in industrial environments (Almeida, 2000). The Ultrasonic range finder is a low-cost noncontact and easy-to-install measurement device. It is able to meet the every-day needs of commercial production, as well serving a more specialized role in the technologically-advanced aero-space industry, thus placing it firmly in the category of high-level measurement technology. Unlike other level indicators with limited uses, the easy-to-install ultrasonic range finder is a highly-accurate device with enough specialized uses to ensure that the needs of the customer are met (www.fine-tek.com). The range finder module consists of a microcontroller, a transmitter, a receiver, a direct-receive inhibit circuit, and an interface (Grill, 1999).

An ultrasonic range measurement system has been developed, based on band-pass to low-pass transformation of the impulse response. The impulse response is a sensitive indicator of time delays between signals. In this case the time delay is caused by the propagation of the sound wave between transmitter and receiver.

The use of ultrasonic sensors for range detection is based on the propagation velocity of sound in the air medium. Thus, if the propagation velocity is change, an error of range measurement will occur. Some of the parameters causing velocity changes are parameters such as temperature and humidity (Holmberg, 1992).

In this thesis, microcontroller-based an ultrasonic range finder circuit which accurately allows level measurement for fluids will be designed. The use of ultrasonic sensors for range detection is based on the propagation velocity of sound in the air. Some of the parameters causing velocity changes such as temperature and humidity. For this reason, a neural network algorithm will be used in this work. This algorithm will evaluate data which are generated by SH11 and by ultrasonic transducer. In this design, SH11 sensor is used to measure humidity and temperature. Ultrasonic trans-

ducers are used to generate ultrasonic signals for sending and receive returned signals, respectively. A microcontroller is used to control and get the data from transducers. Basic Programming Language is used to develop an algorithm for this study and it will be loaded into AVR microcontroller. This project is divided into two design. These are hardware and software. The block diagram of the neural network-based an ultrasonic range finder circuit is given in Figure.4.1.

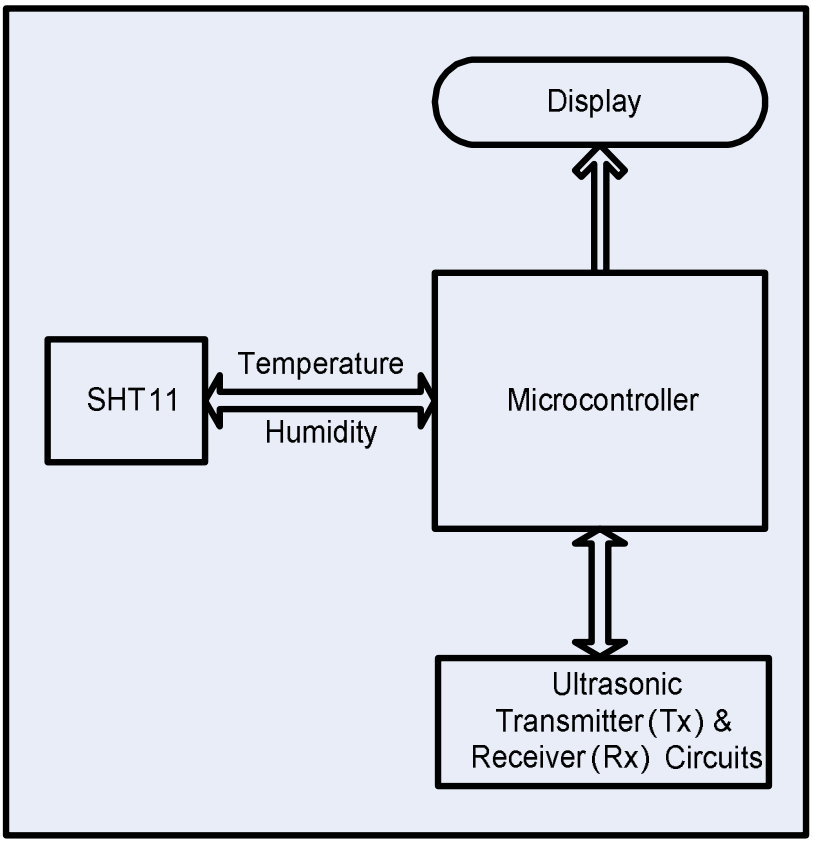


Figure 4.1. The block diagram of the neural network-based an ultrasonic range finder

4.2. Hardware

In this thesis are Atmel Atmega32 AVR microcontroller, transmitter, receiver, temperature and humidity sensor and 2x16 LCD as hardware.

4.2.1. Atmel Atmega32 AVR Microcontroller

In this thesis, high-performance, low-power AVR 8-bit Microcontroller, Atmega32 (Figure.4.2.) from Atmel Corporation, which has many special features compared to any other microconroller is employed as microcontroller. This is widely suitable for all 8-bit applications. It is used in a wide range of embedded applications, and the experience has been so positive that most of the embedded system designers and developers intend to use only Atmel AVRs for all in-house microconroller projects. The main role of an AVR Microconroller is used to measure error-level for fluids, humidity and temperature in medium. Then, it corrects error-level for fluids by an neural algorithm.

The ATmega32 provides the following features: 32K bytes of In-System Programmable Flash Program memory with Read-While-Write capabilities, 1024 bytes EEPROM, 2K byte SRAM, 32 general purpose I/O lines, 32 general purpose working registers, a JTAG interface for Boundary-Scan, On-chip Debugging support and programming, three flexible Timer/Counters with compare modes, Internal and External Interrupts, a serial programmable USART, a byte oriented Two-wire Serial Interface, an 8-channel, 10-bit ADC with optional differential input stage with programmable gain (TQFP package only), a programmable Watchdog Timer with Internal Oscillator, an SPI serial port, and six software selectable power saving modes. The Idle mode stops the CPU while allowing the USART, Two-wire interface, A/D Converter, SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next External Interrupt or Hardware Reset. In Power-save mode, the Asynchronous Timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except Asynchronous Timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low-power consumption. In Extended Stand-

by mode, both the main Oscillator and the Asynchronous Timer continue to run. Figure 4.2. shows block diagram of Atmega32 Microcontroller (ATMEL, 2009).

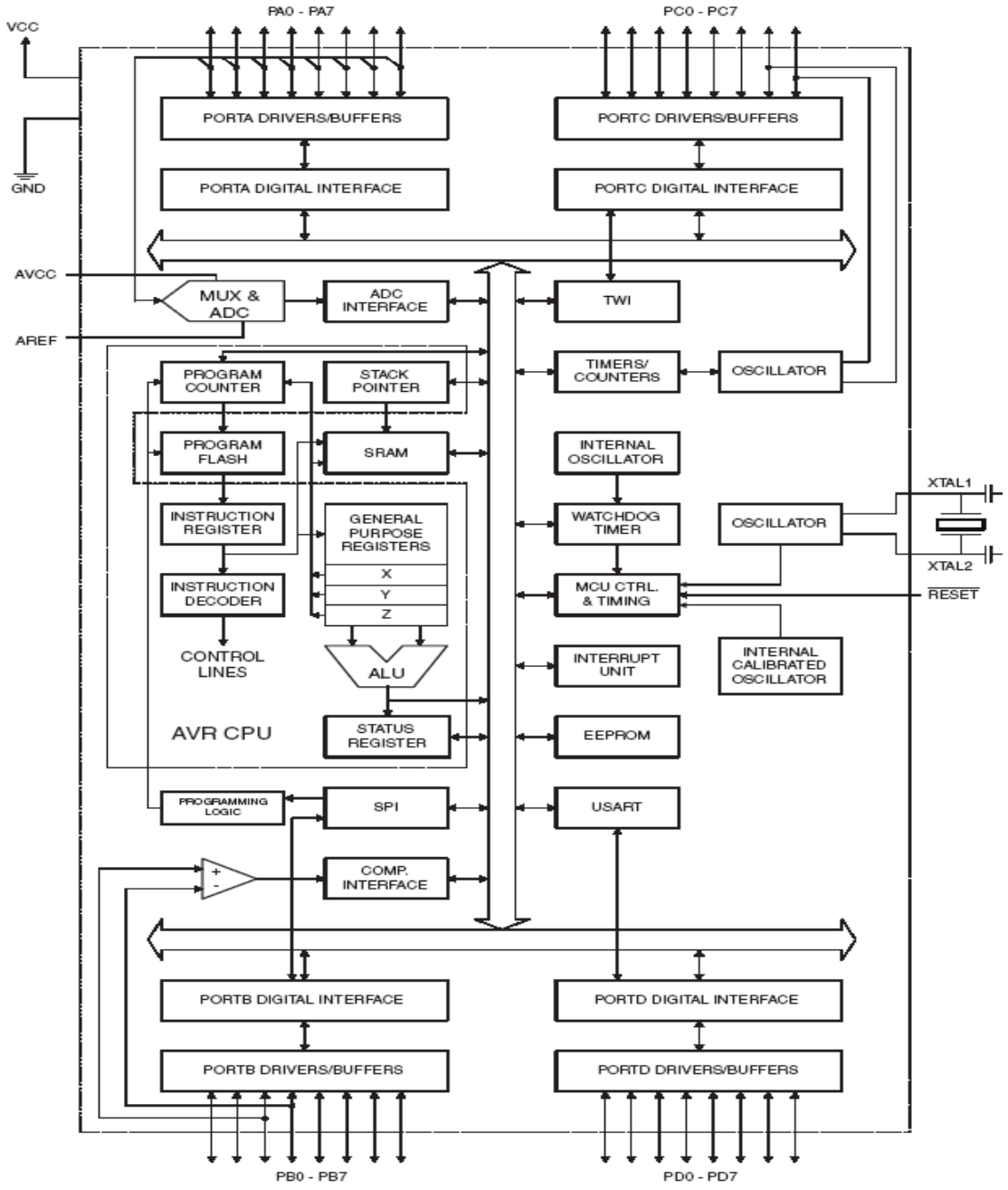


Figure 4.2. Block diagram of atmega32 8-bit microcontroller

4.2.2. Humidity and Temperature Sensor (SHT11)

SHT11 is Sensirion's family of surface mountable relative humidity and temperature sensors. The sensors integrated sensor elements plus signal processing on a tiny foot print and provide a fully calibrated digital output. A unique capacitive sensor element is used for measuring relative humidity while temperature is measured by a band-gap sensor. The sensor is seamlessly coupled to a 14-bit analog to digital converter and a serial interface circuit. This results in superior signal quality, a fast response time and insensitivity to external disturbances (EMC) (Sensirion, 2003).

The serial interface of the SHT11 is optimized for sensor readout and effective power consumption. The sensor cannot be addressed by I^2C protocol; however, the sensor can be connected to an I^2C bus without interference with other devices connected to the bus. The controller must switch between the protocols.

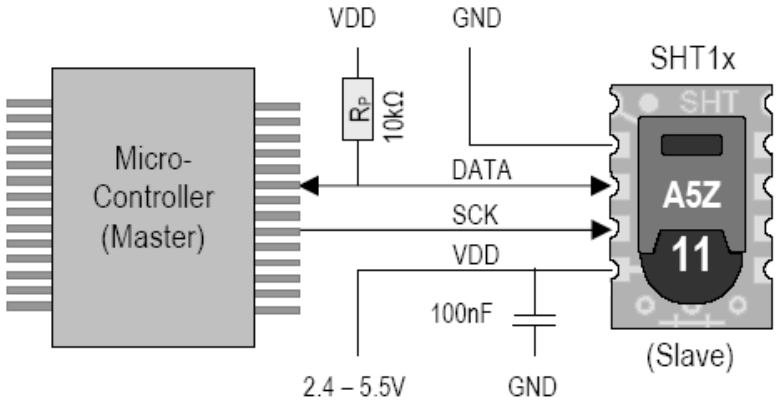


Figure 4.3. Typical application circuit for sht11 (Sensirion, 2003).

The subsequent command consists of three address bits (only '000' is supported) and five command bits. The SHT11 indicates the proper reception of a command by pulling the DATA pin low (ACK bit) after the falling edge of the 8th SCK clock. The DATA line is released (and goes high) after the falling edge of the 9th SCK clock.

Table 4.1. SHT11 list of commands (Sensirion, 2003).

Command	Code
Reserved	0000x
Measure Temperature	**00011**
Measure Relative Humidity	**00101**
Read Status Register	00111
Write Status Register	00110
Reserved	0101x-1110x
Soft reset, resets the interface, clears the status register to default values. Wait minimum 11 ms before next command	**11110**

After issuing a measurement command (‘00000101’ for relative humidity, ‘00000011’ for temperature) the controller has to wait for the measurement to complete. This takes a maximum of 20/80/320 ms for a 8/12/14 bit measurement. The time varies with the speed of the internal oscillator and can be lower by up to %30. To signal the completion of a measurement, the SHT11 pulls data line low and enters Idle Mode (Sensirion, 2003).

To compensate for the non-linearity of the humidity sensor and to obtain the full accuracy it is recommended to convert the readout with the following formula:

$$RH_{linear} = c_1 + c_2 SO_{RH} + c_3 SO_{RH}^2 \tag{4.1}$$

where SO_{RH} is the sensor readout for RH and with $c_1 = -4$, $c_2 = 0.0405$, $c_3 = -2.8\times10^{-6}$ for 12 bit SO_{RH}

For temperatures significantly different from $25^{o}C$ the temperature coefficient of the RH sensor should be considered:

$$RH_{true} = \left(T_{^{o}C} - 25\right)\left(t_1 + t_2 SO_{RH}\right) + RH_{linear} \tag{4.2}$$

with $t_1 = 0.01; t_2 = 0.00008$. The temperature sensor is very linear by design. Use

following formulas or table to convert from digital 14 bit readout to temperature:

$$T_{^oC} = SO_T \times 0.01 - 40 \tag{4.3}$$

where SO_T is the 14 bit sensor readout for temperature.

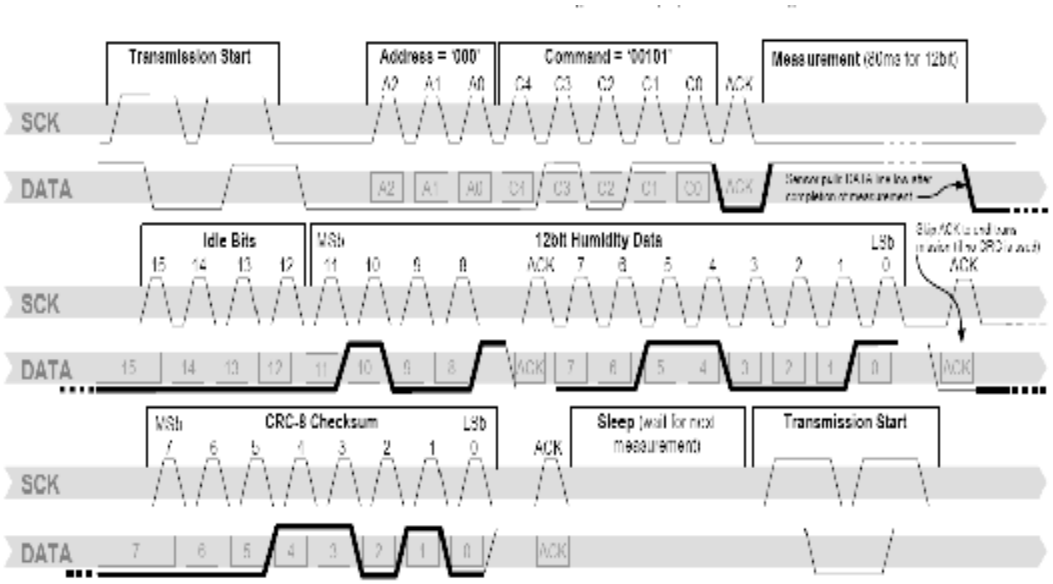


Figure 4.4. Example measurement sequence for value “0000””0100”0011””0001”=%35 RH.

4.2.3. Ultrasound Motion Sensor (DC-SS014)

This sensor is ultrasonic transmitter and receiver circuits and is used to detect the move of human or object. Suitable for indoor and outdoor burglar-proof application, vehicle burglar-proof application, ATM surveillance camera, warehouse surveillance camera, and safety warning applications in dangerous site where high voltage and high temperature exist. Its main technical specifications are

- Power Voltage: 6-12V DC
- Quiescent current: Less than 2 Ma

- Output level: High 5V
- Output level: Low 0V
- Sensing Angle: No greater than 15 degree
- Sensing distance: 2mm-3m

Figure 4.5. shows its schematic (Sure Electronic, 2008).

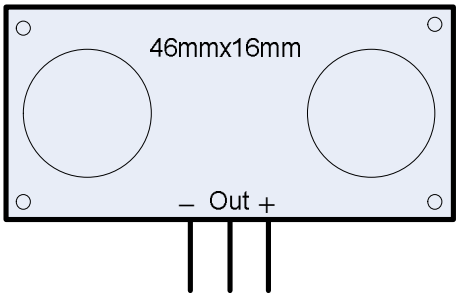


Figure 4.5. Ultrasound motion sensor

4.3. Training and Testing of Measured Values of Temperature, Humidity, and Error-Level Measurement for Fluids By Matlab Neural Network Toolbox

In this thesis, a multi-layer perceptron (MLP) neural network (NN), and Back-Propagation Algorithm which is used as common for it's learning is used. Back-Propagation Algorithm is generalization of the delta rule which is used for learning algorithm of single-layer neural networks. MLP NN is designed by Matlab Neural Network Toolbox. The network has three inputs layer, hidden layer, and single output layer. Input parameters are temperature, humidity, and inaccurate level measurement for fluids. Level measurement for fluids calculates by flight time and speed of sound. Speed of sound changes as relative temperature and humidity, and other parameters. The output of network which was trained is desired accurate level measurement for fluids. We provided necessary data set for training and testing of the network using the approximate formula published in JASA [1993] by Owen Cramer (httpwww.sengpielaudio.comcalculator-airpressure.htm\Speed of Sound Calculator. mht)

$$f\left(t, p, x_w, x_c\right) = a_0 + a_1 t + a_2 t^2 + \left(a_3 + a_4 t + a_5 t^2\right) x_w + \left(a_6 + a_7 t + a_8 t^2\right) p + \left(a_9 + a_{10} t + a_{11} t^2\right) x_c + a_{12} x_w^2 + a_{13} p^2 + a_{14} x_x^2 + a_{15} x_w p x_c \tag{4.4}$$

where the t is the celsius temperature, p is the pressure, x_w is the water vapor mole fraction, and x_c is the carbon dioxide mole fraction. The water vapor mole fraction is

$$x_w = \frac{hfp_{sv}}{p} \tag{4.5}$$

where h is the relative humidity expressed as a fraction, f is the enhancement factor, and p_{sv} is the saturation vapor pressure of water vapor in air. The enhancement factor is

$$f = 1.00062 + 3.14\times10^{-8} p + 5.6\times10^{-7} t^2 \tag{4.6}$$

and

$$p_{sv} = e^{1.281\times10^{-5} T^2 - 1.950\times10^{-2} T + 34.049 - 6.353\times10^{3}/t} \text{ (PA)} \tag{4.7}$$

The coefficients are:

$$\begin{aligned} &a_0 = 331.502, a_1 = 0.603, a_2 = -0.0005, a_3 = 51.471 \\ &a_4 = 0.149, a_5 = -0.0007, a_6 = -0.182\times10^{-7}, a_7 = 3.73\times10^{-8} \\ &a_8 = -2.93\times10^{-10}, a_9 = -85.209, a_{10} = -0.228, a_{11} = 5.91\times10^{-5} \\ &a_{12} = -2.835, a_{13} = -2.15\times10^{-13}, a_{14} = 29.180, a_{15} = 0.0004 \end{aligned} \tag{4.8}$$

The above equations are only valid over the temperature range 0^oC to 30^oC.

For training and testing of the network, data set of input temperatures changes in the interval [0-30] ^oC, and these values was got as one each degree. For each degree, relative humidity is applied in the interval %[45-55]. These values was got as

one each. The vessel is at altitude of 30 cm. For training and testing of the network, output data set of the network is in the interval [0-30] cm. These values was got as one each. For every one cm, values of temperature, humidity, and measured inaccurate level measurement for fluids compose network data set. Finally, numbers of 10571 dataset for training and testing of the network is obtained. Following commands for training and testing of the network is created by Matlab Neural Network Toolbox.

```
a=load.data(1:8998,1:3); % Training input data
b=load.data(1:8998,4); % Training target data
c=load.data(8999:10571,1:3); %Test input data
d=load.data(8999:10571,4); %Test target data
[an,mina,maxa,bn,minb,maxb]=premnmx(a',b'); %
[cn,minc,maxc,dn,mind,maxd]=premnmx(c',d');
net=newff(minmax(an),[3 17 1],{'tansig','tansig','tansig'},'trainlm');
net.trainParam.epochs=4000; % number of iteration
net.trainParam.lr=0.01; % Learning rate
net.trainParam.goal=0.00000001; % The error goal
net=train(net,an,bn)
m=sim(net,an);
g=postmnmx(m',minb,maxb) %Output of the network for b
k=sim(net,cn);
l=postmnmx(k',mind,maxd) %Output of the network for d
```

Figure 4.6. shows the network which was designed.

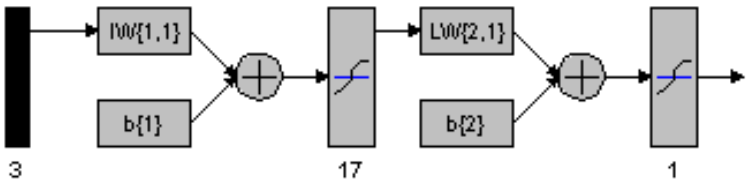


Figure 4.6. The network which was designed

The transfer function of the network is given Eq. 4.9.

$$d_{output} = \tan sig\left[\sum_{j=1}^{17}\left(w_{2j}\tan sig\left[\sum_{i=1}^{3}\left(w_{1ij}x_i\right)+b_{1j}\right]+b_2\right)\right] \quad (4.9)$$

d_{output}:The output of the network

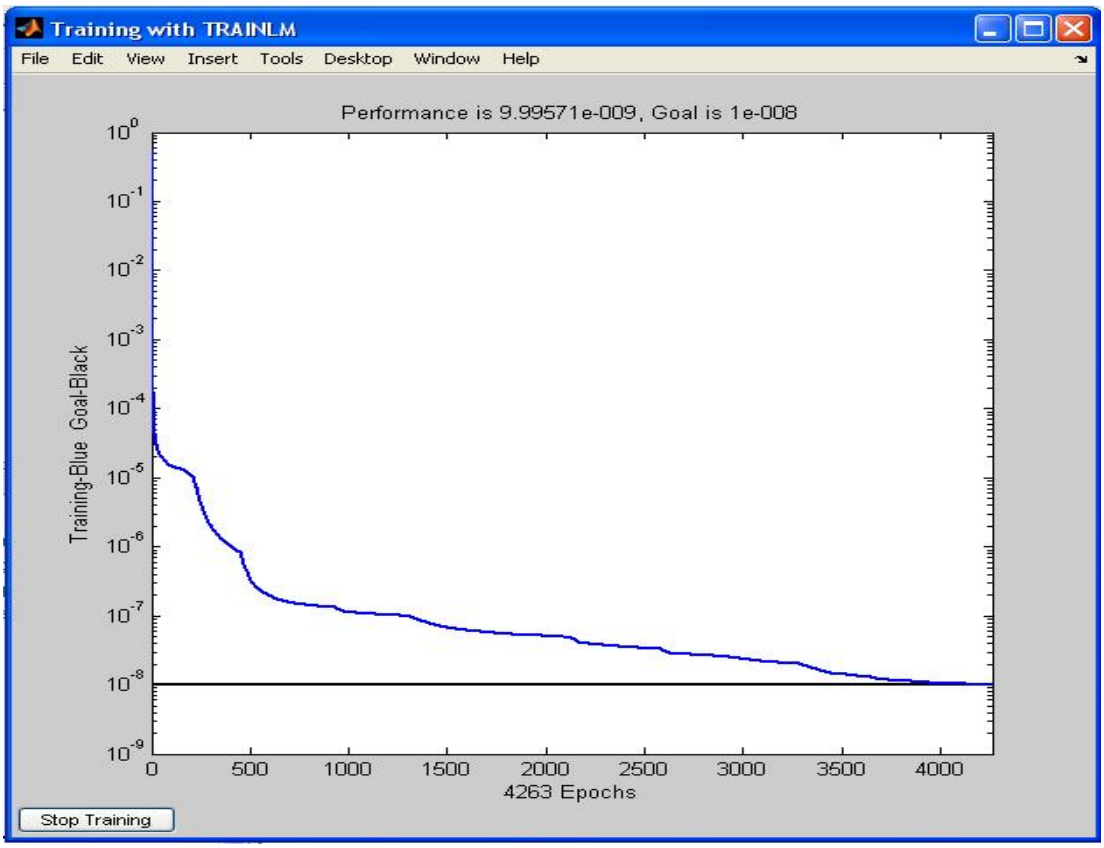


Figure 4.7. Training of the network

Minimum-error goal and number of iteration as a result of training of the network is calculated Matlab Neural Network Toolbox. This simulation illustrates in the Figure 4.7. The training was stopped after 4263 iterations because of validation error. At result of training, the final weights and biases are calculated as

$$LW\{2,1\} = \begin{bmatrix} -4.2064 \\ 2.2359 \\ 13.5349 \\ -1.6748 \\ -6.4146 \\ -1.6966 \\ 3.1708 \\ -18.617 \\ 0.0416 \\ 0.3273 \\ 8.5028 \\ -0.0040 \\ 0.0002 \\ -0.3150 \\ 5.6074 \\ -10.7572 \\ 4.6628 \end{bmatrix}_{17x1}, \quad IW\{1,1\} = \begin{bmatrix} 0.2037 & 0.0032 & -2.4854 \\ -0.2347 & -0.0435 & -9.3875 \\ -0.5174 & -0.0255 & 32.5548 \\ -0.5773 & -0.2701 & 2.2179 \\ 1.9584 & 0.1840 & -9.0824 \\ 0.6449 & 0.0972 & -2.0539 \\ 0.1625 & 0.0199 & -0.1044 \\ 0.0036 & -0.0043 & -3.6362 \\ 0.0324 & 0.0098 & 2.4355 \\ -0.2681 & -0.0471 & 1.6717 \\ -0.7735 & -0.0145 & 9.1335 \\ 0.4417 & -0.4705 & 0.1765 \\ 2.5624 & -1.6456 & 0.4816 \\ -0.2054 & -0.0497 & 1.5226 \\ -0.0991 & -0.0120 & 0.2226 \\ 7.1465 & 0.4248 & -55.4251 \\ -0.0075 & -0.0064 & -4.0668 \end{bmatrix}_{17x3}$$

$$b\{1\} = \begin{bmatrix} -3.4395 \\ 10.8037 \\ -33.7219 \\ 4.2183 \\ -10.2514 \\ -3.0605 \\ 0.2403 \\ 4.6121 \\ -1.0889 \\ 0.6774 \\ 9.9761 \\ 0.2597 \\ 0.9932 \\ 0.6827 \\ -0.1892 \\ -49.1260 \\ 4.3794 \end{bmatrix}_{17x1}, \quad b\{2\} = \begin{bmatrix} -4.3323 \end{bmatrix}_{1x1} \tag{4.10}$$

The network which is trained is tested by these weights and biases. Consequently, the network provided desired the results. Table 4.2. denotes success of the network and Figure 4.8.

Table 4.2. An example showing success of the network

Real Level Measurement of Fluids (cm)	The Output of The Nework (cm)
0	0,000224
1	1,0011
2	2,0001
3	3,0004
8	7,9984
12	12,001
14	14,001
17	17,001
20	20,001
21	21
25	25
26	26
27	26,878
28	27,923
30	29,973

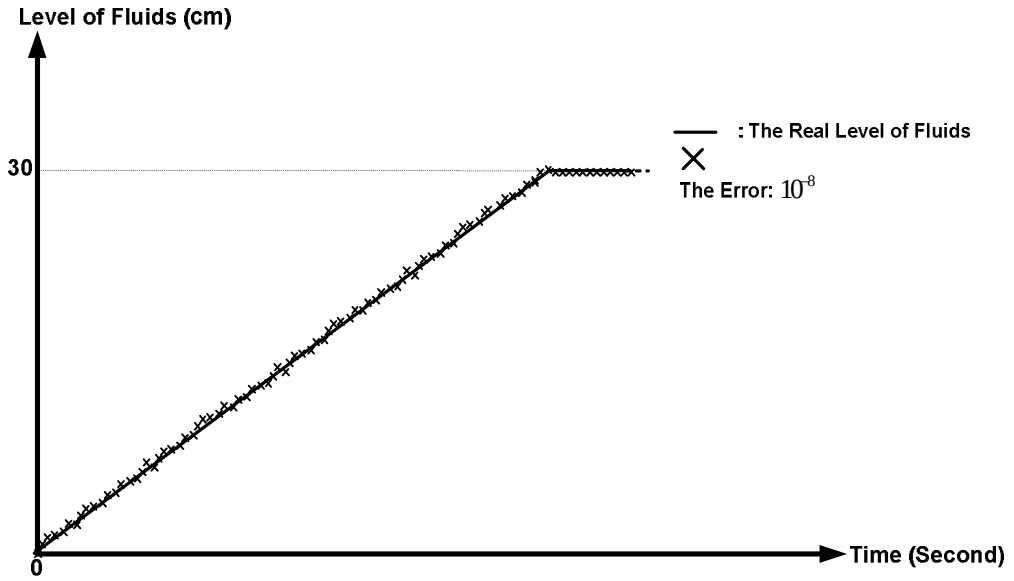


Figure 4.8. Success of the network

4.4. Flow Chart

The software is used to control and get the from ultrasonic circuits and temperature and humidity sensor. In this thesis, BASCOM-AVR as compiler and Basic Programming Language is for software. The flow chart is shown in Figure 4.9.

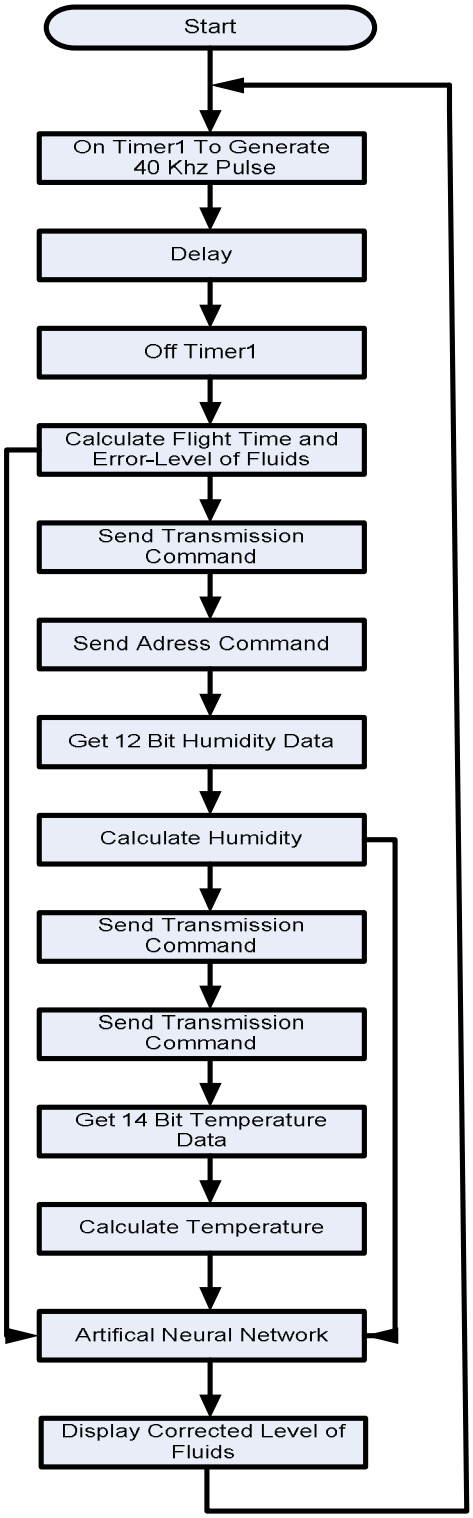


Figure 4.9. Flow chart of neural network-based ultrasonic range finder

4.5. The Block Diagram of Neural Network-Based Ultrasonic Range Finder for Fluids

Figure 4.10. summarizes the neural network-based ultrasonic range finder system for level measurement for fluids.

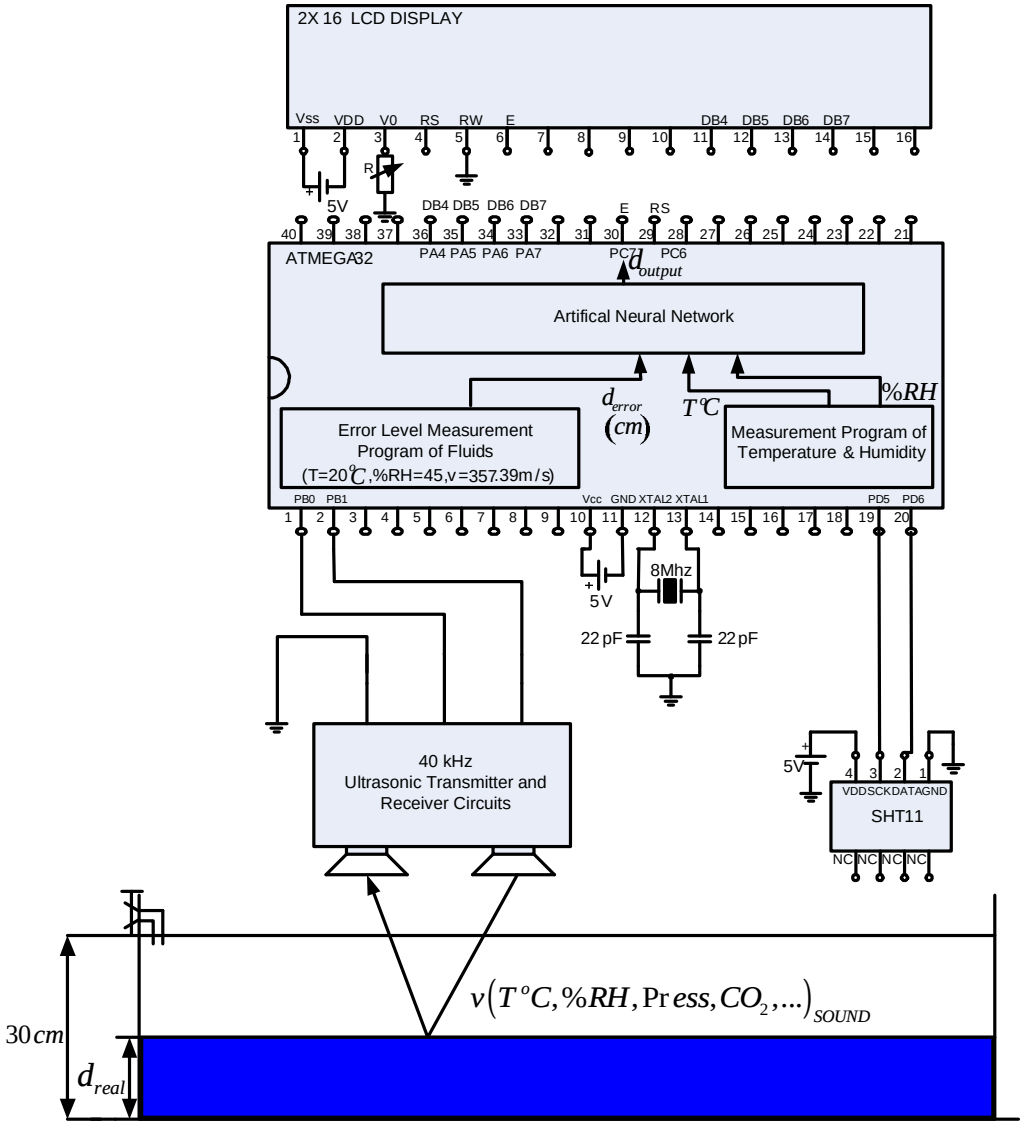


Figure 4.10. Neural network-based ultrasonic range finder and ultrasonic level measurement system of the fluids

5. RESULTS

In this project, 40 kHz pulses are generated by microcontroller. Reflected pulses is received by microcontroller. Microcontroller calculates flight of time. Error-level measurements of fluids are calculates by speed of reference sound and flight of times. Values of temperature and humidity is provided by Temperature and Humidity (SHT11). Error-level measurements of fluids and values of temperature and humidity are the inputs of MLP NN. The outputs of MLP NN are corrected level measurements of fluids. Levels of fluids in interval of [0,30] cm which had measured at of error. 10^{-8}

The measured values are sent to 2x16 LCD display. The Neural Network-Based Ultrasonic Range Finder System for Fluids found to be working efficiently, in the laboratory environment, where stray noises are less. However in open environment, it has produced less accurate results. Software algorithms can be employed to eliminate this problem (including CO2, pressure and other parameters in medium, and efficient algorithms) in future.

6. CONCLUSIONS

The objective of this project is to design and implement an ultrasonic measurement device. As described in this thesis a system is developed that can detect calculate ultrasonic level measurements of fluids sufficiently accuracy. With respect to the requirements for an ultrasonic range finder the following can be concluded.

- The system is able to detected objects with the sensing range
- The system can calculated level measurement of fluids and the distance of the obstruction with sufficiently accuracy.
- This device has the capability to interact with other peripheral if used as a secondary device.
- This offers a low cost and efficient solution for non-contact type distance measurement.

The range finder has numerous applications. It can be used for automatic guided vehicle, positioning of robots as well as measuring generic distance, liquid level in tanks, and the depth of snow banks. The device can serve as a motion detector in production lines. The ultrasonic detection range relates with size, figure, material and position of the object. The bigger the reflector is, the better the reflectance is, and the stronger the reflection signal is. The ultrasonic distance measurement is an untouchable detection mode. Compared with else detection modes, it does not get much influenced by ray, temperature and colour etc., and it has the great capability to adapt to various circumstance and ambient conditions. A restricted target angle (it requires a near-perpendicular surface) and large beam, which can create poor resolution, seem to be the range finder' s only limitations. Also, there is a blind are and distance limitation in ultrasonic distance measurement. Despite these drawback, the device to be extremely useful is found using neural network or other algorithm.

REFERENCES

ALMEIDA A. R., FREIRE E. O., RÖNNOW, C. A., ROSI R. M., 2000. Neural Network-Based Geometric Reference Recognition Applied to Ultrasound Echo Signals.

ATMEL, 2009

CASTLES D., 2008. Generating, Detecting and Processing Ultrasonic Signals, http://www.ee.latrobe.edu.au/~djc/UltraSonics/Ultrasonics.htm, Latrobe University, Melbourne.

ECE Lab, 2006

FINETEK, Ultrasonic Level Indicator, www.fine-tek.com, 2009.

FONTES J., 2005. Sensor Technology, Elsevier, Oxford, 703p.

FRANCO, S., 2002. Design With Operational Amplifiers And Analog Integrated Circuits, State University, San Francisco, 658 p.

GMO INTERNET, Circuit Explanation for Ultrasonic Range Meter, http://www.interq.or.jp/japan/se-inoue/e_sonic1_3.htm#3, 1998.

GRILL, W., 1999. Riverhead Systems, Controller Supports Differential Monitor Display, Littleton.

HAGAN M.T., DEMUTH H.T., BEALE M., 2002. Neural Network Design, China Machine Press, Ohio State University, 734p.

HOLMBERG P., 1992. Robust Ultrasonic Range Finder- an FFT Analysis

PRINCIPE J.C., 2000. Electrical Engineering Handbook, Crc Press LLC, University of Florida, 20: 545-559.

RENESAS, 2004.

SCHMALZEL, J.L., 2000. The Electrical Engineering Handbook. Crc Press LLC, Boca Raton-Florida, 108: 2564-2572.

SENGPIEL E., http://www.sengpielaudio.com/calculator-airpressure.htm, 2008

SENSIRION, 2003

SINCLAIR, I. R., 2001. Sensors and Transducers.India, 703p.

SMITH, R.L., 2000. The Electrical Engineering Handbook. Crc Press LLC, Boca Raton-Florida, 56: 1303-1313.

STAHLIN, B., 2004. Electronic Instrument Handbook, Agilent Technologies, Loveland, Colorado, 664p.

SURE ELECTRONIC, 2008

WALKER E.A., PRASAD R., 2002. Fuzzy and Neural Control, Hall/CRC, 290p.

CURRICULUM VITAE

I was born in Adana, Turkey, in 1983. I completed the high school education in Adana, in 2000. I graduated from Nigde University, The Faculty of Engineering and Architecture, department of Electric & Electronic Engineering in 2006. Now, I have been working as an electronic engineering officer in Turkish Coast Guard Command.

APPENDIX A

Table A.1. Physical and chemical transduction principle

Primary Signal	Secondary Signal					
	Mechanical	Thermal	Electrical	Magnetic	Radiant	Chemical
Mechanical	(Fluid) mechanical and acoustic effects (e.g., diaphragm, gravity balance, echo sounder)	Friction effects (e.g., friction calorimeter) Cooling effects (e.g., thermal flow meters)	Piezoelectricity Piezoresistivity Resistive, capacitive, and inductive effects	Magneto-mechanical effects (e.g., piezo-magnetic effect)	Photoelastic systems (stress-induced birefringence) Interferometers Sagnac effect Doppler effect	
Thermal	Thermal expansion (bimetal strip, liquid-in-glass and gas thermometers, resonant frequency) Radiometer effect (light mill)		Seebeck effect Thermoresistance Pyroelectricity Thermal (Johnson) noise		Thermooptical effects (e.g., in liquid crystals) Radiant emission	Reaction activation (e.g., thermal dissociation)
Electrical	Electrokinetic and electro-mechanical effects (e.g., piezoelectricity, electro-meter, Ampere's law)	Joule (resistive) heating Peltier effect	Charge collectors Langmuir probe	Biot-Savart's law	Electrooptical effects (e.g., Kerr effect) Pockel's effect Electroluminescence	Electrolysis Electromigration
Magnetic	Magnetomechanical effects (e.g., magnetorestriction, magnetometer)	Thermomagnetic effects (e.g., Righi-Leduc effect) Galvanomagnetic effects (e.g., Ettingshausen effect)	Thermomagnetic effects (e.g., Ettingshausen-Nernst effect) Galvanomagnetic effects (e.g., Hall effect, magnetoresistance)		Magnetooptical effects (e.g., Faraday effect) Cotton-Mouton effect	
Radiant	Radiation pressure	Bolometer thermopile	Photoelectric effects (e.g., photovoltaic effect, photoconductive effect)		Photorefractive effects Optical bistability	Photosynthesis, -dissociation
Chemical	Hygrometer Electrodeposition cell Photoacoustic effect	Calorimeter Thermal conductivity cell	Potentiometry Conductimetry Amperometry Flame ionization Volta effect Gas-sensitive field effect	Nuclear magnetic resonance	(Emission and absorption) spectroscopy Chemiluminiscence	

Source: T. Grandke and J. Hesse, Introduction, Vol. 1: *Fundamentals and General Aspects, Sensors: A Comprehensive Survey,* W. Gopel, J. Hesse, and J. H. Zemel, Eds., Weinheim, Germany: VCH, 1989. With permission.

© 2000 by CRC Press LLC

APPENDIX B

Table B.1. ANN types and applications(Principe, 2000)

Application	Topology	Supervised Learning	Unsupervised Learning
Association	Hopfield [Zurada,1992; Haykin, 1994]	-	Hebbian [Zurada, 1992; Haykin, 1994; Kung,1993]
	Multilayer perceptron [Zurada, 1992; Haykin, 1994; Bishop, 1995]	Back-propagation [Zurada, 1992; Haykin, 1994; Bishop, 1995]	-
	Linear associative mem. [Zurada, 1992; Haykin, 1994]	-	Hebbian
Pattern Recognition	Multilayer perceptron [Zurada, 1992; Haykin, 1994; Bishop, 1995]	Back-propagation	-
	Radial basis functions [Zurada, 1992; Bishop, 1995]	Least mean square	k-means [Bishop, 1995]
Feature extraction	Competitive [Zurada, 1992; Haykin, 1994]	-	Competitive
	Kohonen [Zurada, 1992; Haykin, 1994]	-	Kohonen
	Multilayer perceptron [Kung, 1993]	Back-propagation	-
	Principal comp. Anal. [Zurada, 1992; Kung, 1993]	-	Oja's [Zurada, 1992; Kung, 1993]
Prediction, System ID	Time-lagged networks [Zurada, 1992; Kung, 1993; de Vries and Principe, 1992]	Back-propagation through time [Zurada, 1992]	-
	Fully recurrent nets [Zurada, 1992]	-	-

It is always risky to establish a taxonomy of a technology, but motivation is one of providing a quick overview of the application areas and the most popular topologies and learning paradigms

It is clear that multilayer perceptrons (MLPs), the back-propagation algorithm and its extensions — time-lagged networks (TLN) and back-propagation through time (BPTT), respectively — hold a prominent position in ANN technology. It is therefore only natural to spend most of our overview presenting the theory and tools of back-propagation learning. It is also important to notice that Hebbian learning (and its extension, the Oja rule) is also a very useful (and biologically plausible) learning mechanism. It is an unsupervised learning method since there is no need to specify the desired or target response to the ANN (Principe, 2000).

The two-layer neural network is a typical multilayer perceptron (MLP), a multi-layer neural network whose neurons perform the same function on inputs, usually a composite of the weighted sum and a differentiable nonlinear activation function, or transfer function, such as a hyperbolic tangent function. Multi-layer perceptrons are the most commonly used neural network structures for a broad range of applications (Principe, 2000).

APPENDIX C

Pin	Name	Comment
1	GND	Ground
2	DATA	Serial Data, bidirectional
3	SCK	Serial Clock, input only
4	VDD	Source Voltage
NC	NC	Must be left unconnected

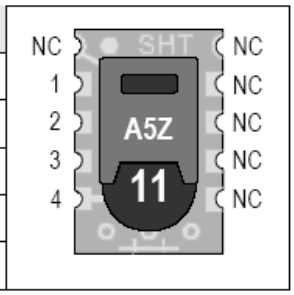


Figure C.1. SHT11 pin assignment, NC remain floating (Sensirion, 2003)

APPENDIX D

Serial clock input (SCK) is used to synchronize the communication between microcontroller and SHT11. Since the interface consists of fully static logic there is no minimum SCK frequency. Serial data (DATA) tri-state pin is used to transfer data in and out of the sensor. For sending a command to the sensor, DATA is valid on the rising edge of the serial clock (SCK) and must remain stable while SCK is high. After the falling edge of SCK the DATA value may be changed. For safe communication DATA valid shall be extended T_{SU} and T_{HO} before the rising and after the falling edge of SCK, respectively, in Figure D.1 (Sensirion, 2003).

.

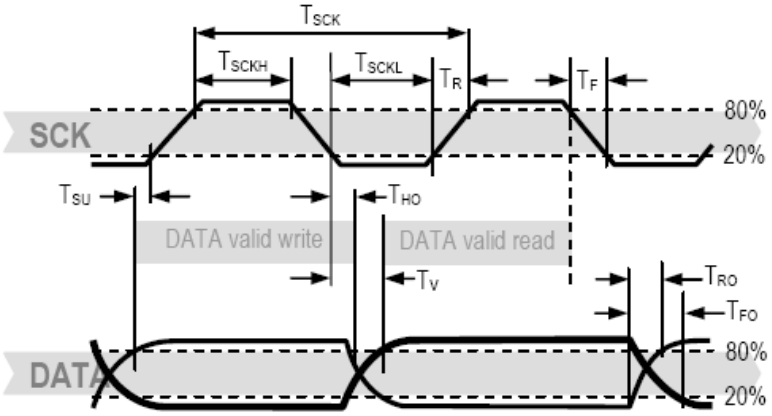


Figure D.1. Timing diagram

Table D.1. SHT11 I/O signal characteristics (Sensirion, 2003).

	Parameter	Conditions	min	typ	max	Units
F_{SCK}	SCK Frequency	VDD > 4.5V	0	0.1	5	MHz
		VDD < 4.5V	0	0.1	1	MHz
T_{SCKx}	SCK hi/low time		100			ns
T_R/T_F	SCK rise/fall time		1	200	*	ns
T_{FO}	DATA fall time	OL = 5pF	3.5	10	20	ns
		OL = 100pF	30	40	200	ns
T_{RO}	DATA rise time		**	**	**	ns
T_V	DATA valid time		200	250	***	ns
T_{SU}	DATA setup time		100	150	***	ns
T_{HO}	DATA hold time		10	15	****	ns

* $T_{R_max} + T_{F_max} = (F_{SCK})^{-1} - T_{SCKH} - T_{SCKL}$

** T_{R0} is determined by the $R_P * C_{bus}$ time-constant at DATA line

*** T_{V_max} and T_{SU_max} depend on external pull-up resistor (R_P) and total bus line capacitance (Cbus) at DATA line

**** $T_{H0_max} < T_V - \max(T_{R0}, T_{F0})$

APPENDIX E

Table E.1. represents speed values of sound relative temperature and humidity obtained by approximate formula of Owen Cramer (www.sengpielaudio.comcalculator-airpressure.htm\Calculation speed of sound in humid air and the air pressure density of water density - sengpielaudio.mht)

Table E.1. Speed values of sound relative temperature and humidity

Values of Humidity (%RH)	Values of Temperature (Degree)																
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	Speed Values of Sound (m/s)																
45	335	335,9	336,8	337,7	338,6	339,6	340,6	341,6	342,6	343,63	344,71	345,81	346,95	348,13	349,33	350,57	351,85
46	335,1	336	336,9	337,8	338,7	339,7	340,7	341,7	342,7	343,78	344,87	345,99	347,14	348,32	349,54	350,8	352,09
47	335,2	336	337	337,9	338,8	339,8	340,8	341,8	342,9	343,92	345,03	346,16	347,32	348,52	349,75	351,02	352,33
48	335,2	336,1	337	338	338,9	339,8	340,9	341,9	343	344,07	345,19	346,33	347,5	348,71	349,96	351,25	352,57
49	335,3	336,2	337,1	338,1	339	340	341	342,1	343,1	344,22	345,35	346,5	347,69	348,91	350,17	351,47	352,81
50	335,4	336,3	337,2	338,2	339,1	340,1	341,2	342,2	343,3	344,37	345,51	346,67	347,87	349,11	350,38	351,69	353,05
51	335,5	336,4	337,3	338,3	339,3	340,3	341,3	342,3	343,4	344,52	345,67	346,84	348,05	349,3	350,59	351,92	353,29
52	335,5	336,5	337	338,4	339,4	340,4	341,4	342,5	343,6	344,67	345,82	347,01	348,24	349,5	350,8	352,14	353,52
53	335,6	336,6	337,5	338,5	339,5	340,5	341,5	342,6	343,7	344,82	345,98	347,18	348,42	349,69	351,01	352,36	353,76
54	335,7	336,6	337,6	338,6	339,6	340,6	341,6	342,7	343,8	344,97	346,14	347,35	348,6	349,89	351,21	352,58	354
55	335,8	336,7	337,7	338,7	339,7	340,7	341,8	342,9	344	345,12	346,3	347,53	348,78	350,08	351,42	352,81	354,24

Values of Humidity (%RH)	Values of Temperature (Degree)													
	17	18	19	20	21	22	23	24	25	26	27	28	29	30
	Speed Values of Sound (m/s)													
45	353,2	354,5	355,9	357,4	358,9	360,4	362	363,7	365,4	367,16	368,99	370,88	372,84	374,87
46	353,4	354,8	356,2	357,7	359,2	360,8	362,4	364,1	365,8	367,6	369,46	371,38	373,37	375,43
47	353,7	355,1	356,5	358	359,5	361,1	362,8	364,5	366,2	368,04	369,93	371,88	373,9	375,99
48	353,9	355,4	356,8	358,3	359,9	361,5	363,1	364,9	366,6	368,49	370,4	372,37	374,42	376,54
49	354,2	355,6	357,1	358,6	360,2	361,8	363,5	365,3	367,1	368,93	370,87	372,87	374,95	377,1
50	354,5	355,9	357,4	358,9	360,5	362,2	363,9	365,7	367,5	369,37	371,33	373,37	375,47	377,65
51	354,7	356,2	357,7	359,2	360,9	362,5	364,3	366	367,9	369,81	371,8	373,86	376	378,21
52	355	356,4	358	359,5	361,2	362,9	364,6	366,4	368,3	370,25	372,27	374,36	376,52	378,76
53	355,2	356,7	358,3	359,9	361,5	363,2	365	366,8	368,7	370,69	372,74	374,85	377,04	379,32
54	355,5	357	358,5	360,2	361,8	363,6	365,4	367,2	369,1	371,13	373,2	375,35	377,57	379,87
55	355,7	357,2	358,8	360,5	362,2	363,9	365,7	367,6	369,6	371,57	373,67	375,84	378,09	380,42