DESIGN AND DEVELOPMENT OF A NON INVASIVE BLOOD OXYGEN MEASUREMENT SYSTEM

Enrollment Number	071007, 071009, 071135
Name of Students	Saurabh Rawat, Nakul Narang, Amal Singh
Name of Supervisor	**Ms. Vanita Rana**

May 2011

Project Report submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

Table of Contents

Topics		Page
Certificate from the supervisor		IV
Acknowledgement		V
Summary		VI
Chapter 1	**Introduction**	
	1.1 Introduction to pulse oximetry	1
	1.2 Background theory	4
Chapter 2	**Project objective and Specifications**	
	2.1 Project Objective	5
	2.2 Specifications	5
Chapter 3	**Methodology and design concept**	
	3.1 Pulse Oximetry Concept	6
	3.2 Fast Fourier Transform	10
Chapter 4	**Hardware and Software design**	
	4.1 Finger Sensor Part Selection	15
	4.2 Pulse Oximeter Sensor	16
	4.3 ATmega32 Microcontroller	18
	4.3.1 Analog to Digital Converter	21
	4.4 Liquid Crystal Display Unit	23
	4.5 Software Design	26

Chapter 5 **Result and Future prospects**

5.1 Result 29

5.2 Conclusion 30

5.3 Future Prospects 30

5.4 Limitations 31

Appendix A **32**

References **58**

List of figures

Fig 1.1 Avant 9700 2

Fig 3.1 Absorption characteristics 6

Fig 3.2 Finger Probe 7

Fig 3.3 Transmission vs Reflectance 8

Fig 3.4 Light absorption from different parts of finger 8

Fig 3.5 Empirical SpO_2 curve 9

Fig 3.6 Butterfly Structure 14

Fig 4.1 Sensor circuit based on LDR 16

Fig 4.2 Sensor Using Differential Amplifier 17

Fig 4.3 Sensor Using Amplifier 17

List of Tables

Table 1.1 Comparison between various devices 3

Table 1.2 Interface of LCD 25

Certificate

This is to certify that the work titled — "**DESIGN AND DEVELOPRMENT OF A NON INVASIVE BLOOD OXYGEN MEASUREMENT SYSTEM**", submitted by **Saurabh Rawat, Nakul Narang and Amal Singh** in partial fulfillment for the award of degree of Bachelor of Technology in Electronics and Communication Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor	.....
Name of Supervisor	Ms. Vanita Rana
Designation	Senior Lecturer
Date	...23/05/11.....

ACKNOWLEDGEMENT

The zeal to accomplish the task of formulating the project report on "Biomedical Patient Monitoring System" could not have been realised without the support and cooperation of the members of the faculty of ECE Department.

We express our gratitude and sincere thanks to **Ms. Vanita Rana** (Lecturer, Electronics and Communication Department) for providing us the opportunity to undertake the project under her able guidance. She helped us develop novel solutions to every problem and helped us emerge with good engineering acumen.

We are thankful to Retd. Brig. S.P. Ghrera(HOD, Computer Science Department) for his support and guidance from the very start of this endeavour.

We would also like to thank Dr Rajiv Kumar, Ms Pragya Sharma and Ms Meenakshi Sood for their patient hearing of our ideas and opening up our minds to newer horizons by pointing out our flaws, providing critical comments and suggestions to improve the quality of our work and appreciating our efforts.

We would also like to appreciate and thank the Lab Engineers Mr.Manoj Pandey and Mr. Pramod Kumar. Without their help, the completion of the project would have been not possible.

Apart from these, countless events, countless people & several incidents have made a contribution to this project that is indescribable.

(Saurabh Rawat) (Rahul Narang) (Amal Singh)

ABSTRACT

A **pulse oximeter** is a medical device that non-invasively monitors the blood oxygen level of a person as opposed to measuring oxygen level invasively through a blood sample.

This technique is based on differentiating oxygenated (HbO_2) and deoxygenated (Hb) haemoglobin through light absorption characteristics, using IR and optical sensors.

Applications:

- Monitoring patients during anesthesia, intensive care, or those with conditions such as asthma.
- They can warn a pilot of how close he is to hypoxia (deficiency in the amount of oxygen reaching body tissues), similarly other situations where people work at high altitudes.
- A relatively new application in which blood oximetry is used to calculate the stress level of a person as used in Nintendo Wii and Ubisoft's synergy platforms.

The modern day pulse oximeters are very costly and are not easily affordable. Our aim is to develop an affordable pulse oximeter. The sensor is based on dual wavelength LED and a photodiode. We are using the AVR microcontroller family. The code is written in assembly language. The output is displayed on an LCD module.

CHAPTER 1

Introduction

1.1 Introduction to Pulse Oximetry

Oxygen gas is necessary for human life. It is integral for countless biological processes. The transport of oxygen throughout the human body is performed by the circulatory system, and more specifically, haemoglobin in red blood cells. Critical medical information can be obtained by measuring the amount of oxygen in blood, as a percentage of the maximum capacity.

Pulse oximetry has become a standard procedure for the measurement of blood-oxygen saturation in the hospital operating room and recovery room. Oximetry shortens the time passed before the detection of hypoxemia, or deficiency of oxygen. Hypoxemic events have been documented in the critically ill during invasive or diagnostic procedures, and during movement from one location to another. Significant hypoxemic events are also common during cardiac catheterization, or inserting a catheter into a chamber or vessel of a patient's heart. Pulse oximetry also provides an early warning of oxygen ventilator malfunction. Finally, pulse oximetry provides an important function in the intensive care unit, as an early warning system for patient emergencies. Otherwise unsuspected episodes of low blood oxygen can be detected accurately and quickly. Monitoring of a patient through wireless telemetry can be done to view data from numerous remote patients on a single display.

In addition, many screening devices for sleep apnea use pulse oximetry as its most important parameter. By recording oxygen saturation and pulse readings during sleep, pulse oximetry can be an effective and low-cost screening tool that may be used away from the hospital. With the information being easily analyzed and viewed on a computer, it can be a worthwhile, objective view. Wireless pulse oximetry adds many advantages to the traditional wired units. They are more convenient for the patient to use, and can be more comfortable; wireless units don't need to be reconnected each time the patient is moved. Many wireless units are already available from several manufacturers. This product developed in this project provides several advantages

over existing models, primarily adding a longer battery life and a low unit cost. In determining the improvements this design will exhibit, it is important to review existing products in this field. This will clarify the specific advantages this design can provide over previous devices.

PRIOR ART

There are many models and brands of pulse oximeters. A sampling based on popularity and features was chosen. The models described cover the full gamut of possible feature sets, from completely wired and stationary units, to highly portable disaster-relief and triage models, to Bluetooth-compatible wireless units.

Avant 9700

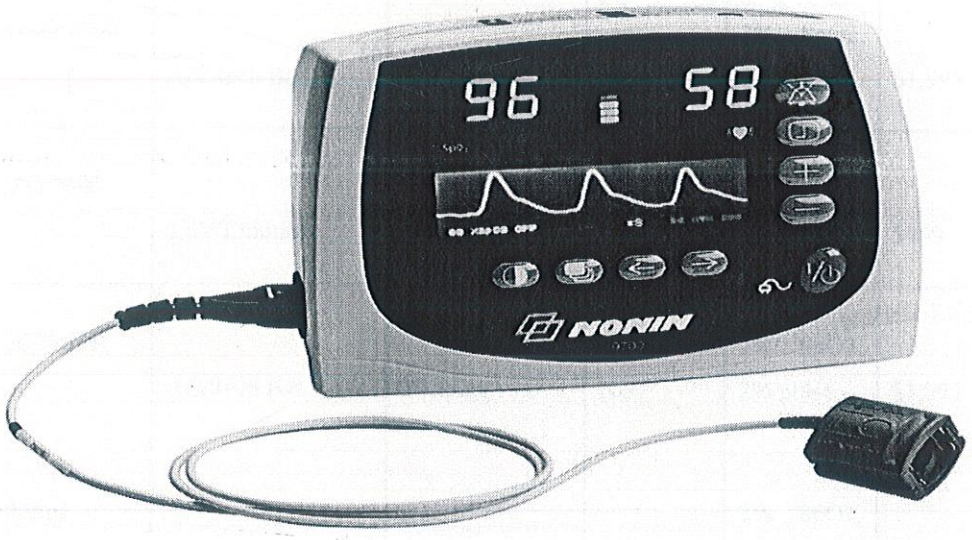


Figure 1.1: Avant 9700

The Avant 9700 is an industry leader pulse oximeter. The sensor is connected to the display via a wire. Oxygen saturation, pulse, and plethysmograph are displayed in bright LED displays. The unit can be either AC powered or battery powered.

GE Trusat

The GE TruSat pulse oximeter is a durable model that displays oxygen saturation and pulse. The LCD display is backlit for easy viewing. It can either be powered by AC or battery power.

Model	Batteries/Power	Battery Life	Pleth Wave	Accuracy	Cost
Avant 4000(Nonin)	Two AA batteries Tx, AC/Rech Bat display	>120 hours Tx, 18 hours display	No	2% SpO2, 3% pulse	$1650
Nonin WristOx	Two 1.5V N-Cell	24 Hours	No	2% SpO2, 3% pulse	$725
Avant 9700	AC/Rech Bat	12 Hours	Yes	2% SpO2, 3% pulse	$1,995
SPO 7500	3.6V Lithium	300 Hours	No	2% SpO2, 3% pulse	$499
GE TruSat	AC/Rech Bat	20 hours	No	2% SpO2, 2% pulse	$1,895
Philips Intellivue	Two AA Batteries	17 Hours	No	2% SpO2, 3% pulse	$1000

Table 1.1: Comparison between various devices

1.2 Background Theory

Current pulse oximeters use a weighted moving average technique lo compute oxygen saturation (SpO_2) values. This method has many limitations including susceptibility to motion artifact, background light, and low perfusion errors. The goal for developing an alternate method for pulse oximetry was to overcome these limitations.

The underlying physical basis of the pulse oximeter is Beer's law which relates the intensity of light transmitted to the amount of solute concentration with equation:

$$I_{trans} = I_{in} e^{-Dca}$$

where I_{trans} is the intensity of transmitted light; I_i is the intensity of incident light, D is the distance which light is transmitted through the solute, c is the concentration of solute, and a is the extinction or absorption coefficient of the solute. To compute the oxygen saturation, a two-solute concentration is assumed. With CO representing the oxygen haemoglobin, and C representing the reduced or deoxyhaemoglobin, a measure blood oxygen saturation levels can be defined. Since oxyhaemoglobin absorbs more infrared light and deoxyhaemoglobin absorbs more red light, the absorption of infrared light relative to the red light increase with oxyhaemoglobin. The ratio of the absorption coefficient can be used to determine the oxygen saturation of blood. The ratio of the absorption coefficients can be used to determine the oxygen saturation of blood. This measurement indicates the cardiopulmonary state of the patient.

This technique is based upon the different red and infrared light absorption characteristics of oxygenated(HbO2) and deoxygenated(Hb) haemoglobin.

Chapter 2

Project Objectives and Specifications

2.1 Project Objectives

- To understand and appreciate the process of designing and developing an embedded system.
- To design an embedded system capable of determining the oxygen concentration in blood.
- Have low power consumption, and a functional life of a good number of hours, so as to last through an operation.
- Be small and lightweight, such that it is not a major annoyance when clipped on.

2.2 Design specifications:

The probe had to fulfill the following requirements:

- Durable enough to withstand the heavy use that it may encounter in hospitals of the developing world.
- Cheap enough to be sold for under Rs 1500.
- Suitable for use with a variety of patients (most age and ethnic groups)
- The probe must be able to interface with the Microcontroller device; in conjunction with the microcontroller , it must be able to obtain an oxygen saturation reading
- The probe must be convenient for medical personnel to use.

Chapter 3

Concept and Methodology

3.1 Pulse Oximeter Concept

The main operation of a pulse oximeter is the determination of a person's functional oxygen saturation. Arterial oxygen saturation, or SaO_2, is the percentage of functional arterial haemoglobin that is oxygenated. Functional haemoglobins are a type of haemoglobin that is able to bind with oxygen. Non-functional haemoglobins cannot bind with oxygen. An example of non-functional haemoglobin is carboxyhaemoglobin (COHb), which binds easily with carbon monoxide. When a functional haemoglobin binds with four oxygen molecules, it is considered an oxygenated haemoglobin (HbO_2). When it is carrying less than four oxygen molecules, it is considered reduced (Hb). Functional oxygen saturation measured with a pulse oximeter is often called SpO_2 because it is an estimation based peripheral measurements and an assumption that only HbO_2 and Hb are present in the blood. The presence of non-functional haemoglobins such as COHb can cause erroneous measurements. Therefore, SpO_2 is a different measurement than SaO_2.

The figure below illustrates the differences in absorption of oxygenated haemoglobin and deoxygenated haemoglobin.

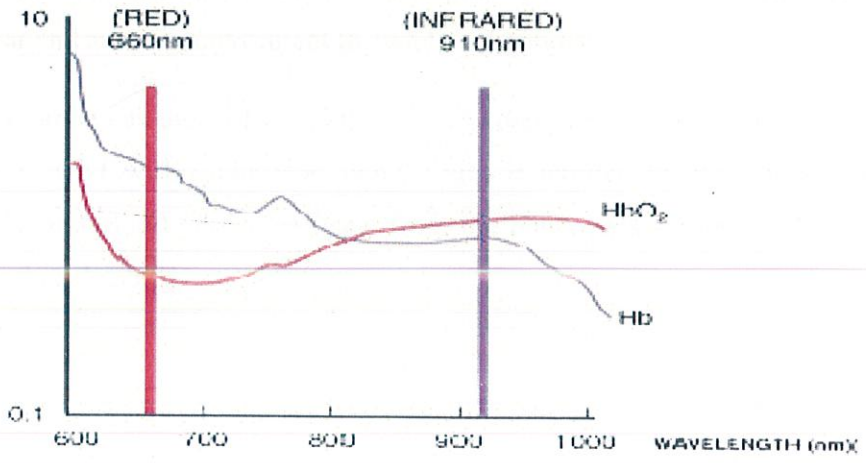


Figure 3.1: Absorption characteristics

Both red and infrared light are shone through a part of the body that is translucent, and has good blood flow (typically the ear, finger or toe). A photodetector at the opposite end determines the strength of the resulting red and infrared signals.

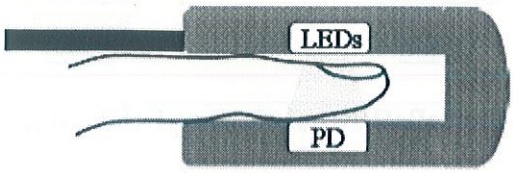


Figure 3.2: Finger Probe

Because the flow of blood is pulsatile in nature, the transmitted light changes with time. A normal finger has light absorbed from bloodless tissue, venous blood, and arterial blood. The volume of arterial blood changes with pulse, so the absorption of light also changes. The light detector will therefore see a large DC signal representing the residual arterial blood, venous blood, and bloodless tissue. A small portion of the detected signal (~1%), will be an AC signal representing the arterial pulse. Because this is the only AC signal, the arterial portion of the signal can be differentiated.

There are two approaches to developing an oximeter probe. The first uses transmitted light, the second uses reflected light. The difference is in the way the elements within the probe are positioned. A transmittance probe has two LEDs on one side and a photodiode (light detector) on the other. The tissue to be analyzed (commonly a finger or an ear) is inserted between the two. Transmittance probes are commonly placed on a finger or ear and are very convenient to attach and remove.

A reflectance probe has the LEDs and the photodiode(s) on the same side. It must be placed over a point with underlying bone. Light is emitted by the LEDs, passes through tissue and blood vessels, reflects from the bone, passes through the tissues again, and is then detected.

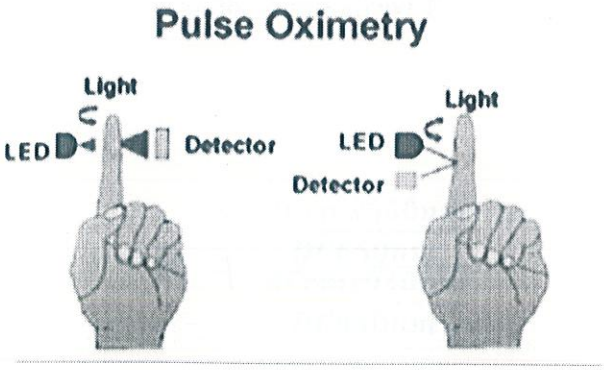


Figure 3.3: Transmission vs Reflectance

Probes utilizing reflectance have the advantage that, regardless of the patient's size (infants to very large adults), the attachment site is always similar. However, attachment of a reflectance probe is, in general, more difficult than attachment of a transmittance probe, and proper attachment of a reflectance probe is essential to ensure its signal quality. Also, properly attaching a reflectance probe is quite time consuming which may lead to improper use. Both the transmittance and the reflectance probes are used clinically, though the transmittance probe is more common due to the convenience of attachment and better signal quality.

This AC signal is separated with simple filtering and an RMS value can be calculated.

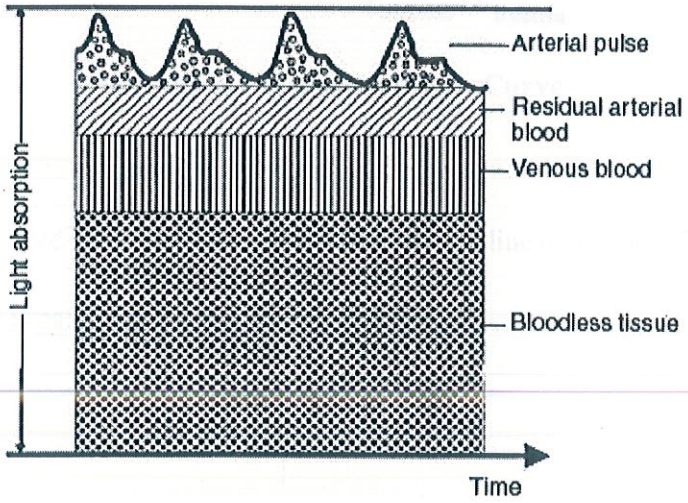


Figure 3.4: Light absorption from different parts of finger

An intermediate value, known as the Normalized R ratio, is calculated using these signals.

$$R = \frac{\frac{ACrms660nm}{DC660nm}}{\frac{ACrms940nm}{DC940nm}}$$

This value represents a ratio of reduced to oxygenated arterial haemoglobin. Using this value, a value of oxygen saturation is calculated based on empirical data.

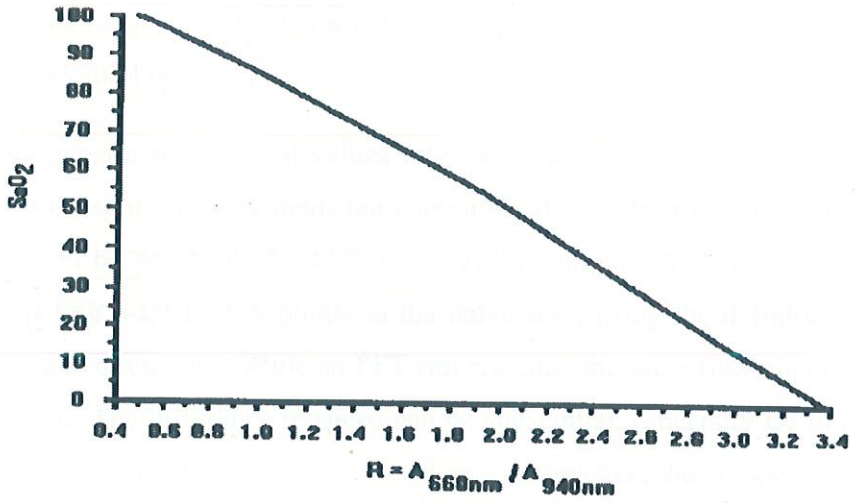


Figure 3.5: Empirical SpO_2 Curve

This empirical curve can be closely approximated by a linear equation.

The SpO_2 value is calculated using the formula below:

$$SpO_2 = 110 - 25R$$

3.2 Fast Fourier Transform

In order to calculate blood oxygen levels, pulse oximeters, in general, use similar methods which include the use of Fast-Fourier Transform (FFT) analysis of red light signals and infrared light signals, and their comparative absorption through a translucent part of a patient" s body. For the digital signal processing implementation, this FFT computation is the final result desired, where all previous processes lead to this end result

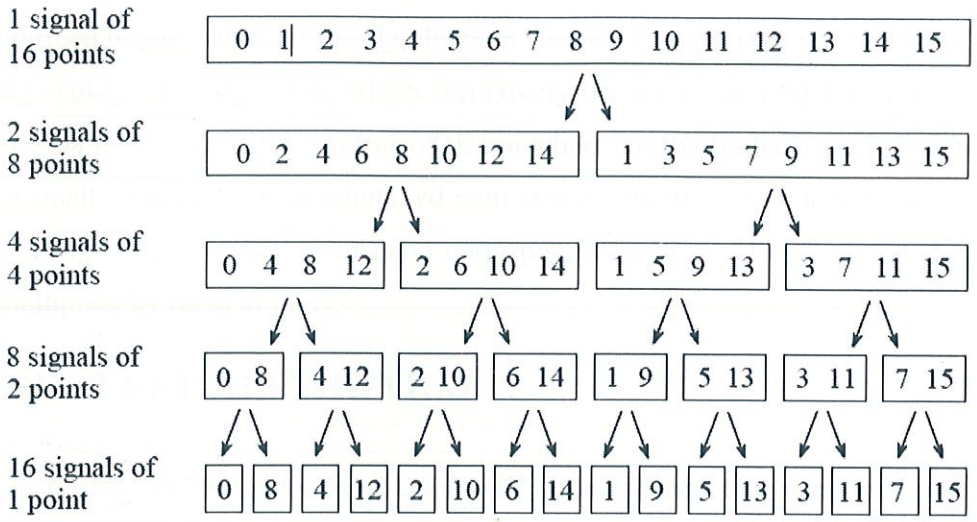
A **fast Fourier transform (FFT)** is an efficient algorithm to compute the discrete Fourier transform (DFT) and its inverse. There are many distinct FFT algorithms involving a wide range of mathematics, from simple complex-number arithmetic to group theory and number theory.

A DFT decomposes a sequence of values into components of different frequencies. This operation is useful in many fields but computing it directly from the definition is often too slow to be practical. An FFT is a way to compute the same result more quickly: computing a DFT of N points in the naive way, using the definition, takes $O(N^2)$ arithmetical operations, while an FFT can compute the same result in only $O(N \log N)$ operations. The difference in speed can be substantial, especially for long data sets where N may be in the thousands or millions—in practice, the computation time can be reduced by several orders of magnitude in such cases, and the improvement is roughly proportional to $N / \log(N)$. This huge improvement made many DFT-based algorithms practical; FFTs are of great importance to a wide variety of applications, from digital signal processing and solving partial differential equations to algorithms for quick multiplication of large integers.

The FFT operates by decomposing an N point time domain signal into N time domain signals each composed of a single point. The second step is to calculate the N frequency spectra corresponding to these N time domain signals. Lastly, the N spectra are synthesized into a single frequency spectrum.

The following figure shows an example of the time domain decomposition used in the FFT . In this example, a 16 point signal is decomposed through four separate stages.

The first stage breaks the 16 point signal into two signals each consisting of 8 points. The second stage decomposes the data into four signals of 4 points. This pattern continues until there are N signals composed of a single point. An **interlaced decomposition** is used each time a signal is broken in two, that is, the signal is separated into its even and odd numbered samples. There are *Log* stages required in this decomposition, i.e., $2N$ a 16 point signal (24) requires 4 stages, a 512 point signal (27) requires 7 stages, a 4096 point signal (212) requires 12 stages, etc.



Let $x_0, \ldots, x_{N-1}$ be complex numbers. The DFT is defined by the formula

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i 2\pi k \frac{n}{N}} \qquad k = 0, \ldots, N-1.$$

Evaluating this definition directly requires $O(N^2)$ operations: there are N outputs X_k, and each output requires a sum of N terms. An FFT is any method to compute the same results in O(N log N) operations. There are various algorithms to calculate FFT but in this project we have used the Butterfly Structure Algorithm.

THE BUTTERFLY STRUCTURE

In the context of fast Fourier transform algorithms, a **butterfly** is a portion of the computation that combines the results of smaller discrete Fourier transforms (DFTs) into a larger DFT, or vice versa (breaking a larger DFT up into subtransforms). In the context of fast Fourier transform algorithms, a **butterfly** is a portion of the computation that combines the results of smaller discrete Fourier transforms (DFTs) into a larger DFT, or vice versa (breaking a larger DFT up into subtransforms). The name "butterfly" comes from the shape of the data-flow diagram in the radix-2 case, as described below. Most commonly, the term "butterfly" appears in the context of the Cooley–Tukey FFT algorithm, which recursively breaks down a DFT of composite size $n = rm$ into r smaller transforms of size m where r is the "radix" of the transform. These smaller DFTs are then combined with size-r butterflies, which themselves are DFTs of size r (performed m times on corresponding outputs of the sub-transforms) pre-multiplied by roots of unity.

RADIX 2 FFT ALGORITHM

Let us consider the computation of the $N = 2^{\nu}$ point DFT by the divide-and conquer approach. We split the N-point data sequence into two $N/2$-point data sequences $f_1(n)$ and $f_2(n)$, corresponding to the even-numbered and odd-numbered samples of $x(n)$, respectively, that is,

$$f_1(n) = x(2n)$$

$$f_2(n) = x(2n+1), \qquad n = 0,1,\ldots,\frac{N}{2}-1$$

Thus $f_1(n)$ and $f_2(n)$ are obtained by decimating $x(n)$ by a factor of 2, and hence the resulting FFT algorithm is called a *decimation-in-time algorithm.*

Now the N-point DFT can be expressed in terms of the DFT's of the decimated sequences as follows:

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, \qquad k = 0,1,\ldots, N-1$$

$$= \sum_{n \text{ even}} x(n) W_N^{kn} + \sum_{n \text{ odd}} x(n) W_N^{kn}$$

$$= \sum_{m=0}^{(N/2)-1} x(2m) W_N^{2mk} + \sum_{m=0}^{(N/2)-1} x(2m+1) W_N^{k(2m+1)}$$

But $W_N^2 = W_{N/2}$. With this substitution, the equation can be expressed as

$$X(k) = \sum_{m=0}^{(N/2)-1} f_1(m) W_{N/2}^{km} + W_N^k \sum_{m=0}^{(N/2)-1} f_2(m) W_{N/2}^{km}$$

$$= F_1(k) + W_N^k F_2(k), \qquad k = 0,1,\ldots, N-1$$

where $F_1(k)$ and $F_2(k)$ are the $N/2$-point DFTs of the sequences $f_1(m)$ and $f_2(m)$, respectively.

Since $F_1(k)$ and $F_2(k)$ are periodic, with period $N/2$, we have $F_1(k+N/2) = F_1(k)$ and $F_2(k+N/2) = F_2(k)$. In addition, the factor $W_N^{k+N/2} = -W_N^k$. Hence the equation may be expressed as

$$X(k) = F_1(k) + W_N^k F_2(k), \qquad k = 0,1,\ldots, \frac{N}{2} - 1$$

$$X(k + \frac{N}{2}) = F_1(k) - W_N^k F_2(k), \qquad k = 0,1,\ldots, \frac{N}{2} - 1$$

We observe that the direct computation of $F_1(k)$ requires $(N/2)^2$ complex multiplications. The same applies to the computation of $F_2(k)$. Furthermore, there are $N/2$ additional complex multiplications required to compute $W_N^k F_2(k)$. Hence the computation of $X(k)$ requires $2(N/2)^2 + N/2 = N^2/2 + N/2$ complex multiplications. This first step results in a reduction of the number of multiplications from N^2 to $N^2/2 + N/2$, which is about a factor of 2 for N large.

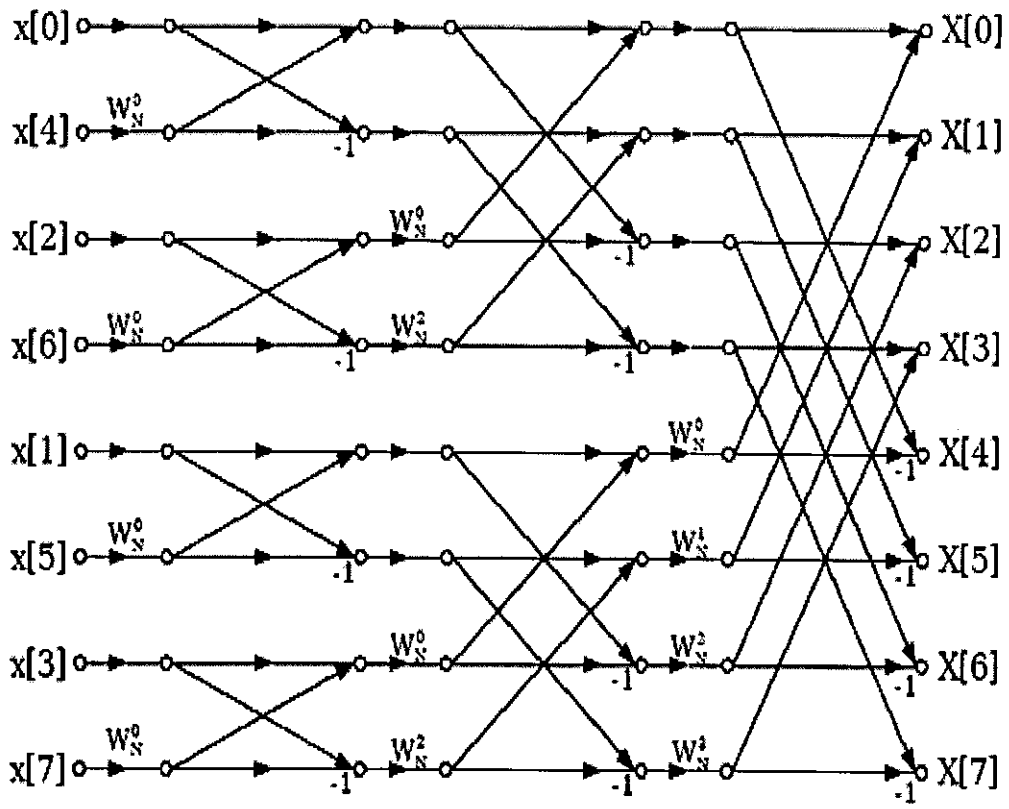


Figure 3.6: Butterfly Structure

Chapter 4

Hardware and software design

4.1 Finger Sensor Part Selection:

Red LED

The red LED, in particular, has strict requirements. Due to its position on the light extinction curve, unpredictability in the LED wavelength can alter the accuracy of the final SPO2 calculation. For this reason, having a narrow bandwidth, or spectral line half-width, is necessary. Having a wavelength of 660nm is important, because the most research has been done with this wavelength, as it is easy to work with. The deviation from the 660nm wavelength can be accounted for with a wavelength coding resistor. For the sake of simplifying the project, this was not implemented. The maximum pulsed current is crucial, as it is necessary to pulse the LED brightly in order to obtain a quality signal. The angle of the light beam is also a consideration, as a wide angle can waste energy. Any light not received by the light detector is wasted light. Size and mounting ability is also an issue. Too large of an LED would not easily fit in a compact sensor.

IR LED

The IR LED has looser requirements. Due to its position on the light extinction curve, wavelength accuracy is not a prime concern. Having a wavelength of 940nm is important, because the most research has been done with this wavelength, as it is easy to work with.

Photo-detector

The light sensor has to be responsive to both wavelengths of light. One of the most important parameters of interest is the rise and fall time. The faster the times, the lower the duty cycle can be for pulsing the LEDs. In this design, and integrated photo-detector and op-amp pair 25 were chosen for simplicity of design, fast rise and fall times, and a wide spectrum response.

4.2 Pulse Oximeter Sensor

We tried various methodologies for the pulse oximeter sensor:

Sensor based on LDR

We tried to implement the pulse oximeter sensor based on a pair of LED and LDR but it didn't give appropriate readings due to interference of light and other surroundings and the circuit for that is as given:

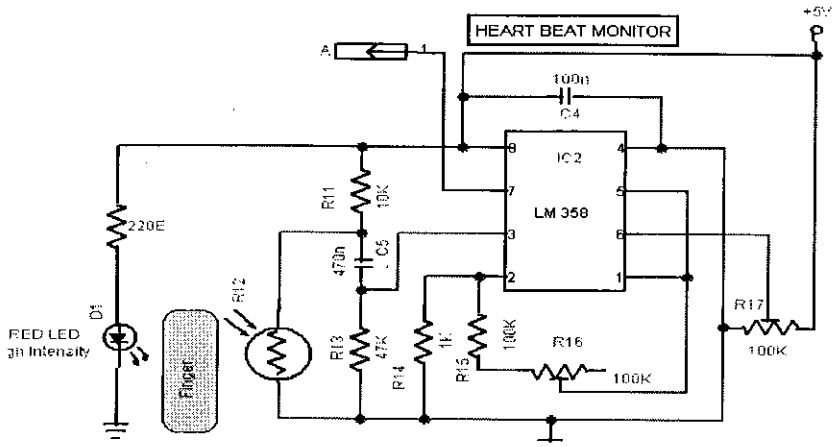


Figure 4.1: Sensor circuit based on LDR

Then we tried another circuit that was based on the concept of separating out the AC and DC signals using a band pass filter with a bandwidth of 0.5-5 Hz, and an amplifier.

But we could not separate AC and DC signals reliably. Hence the results were not accurate.

The circuit diagram for the above is as follows

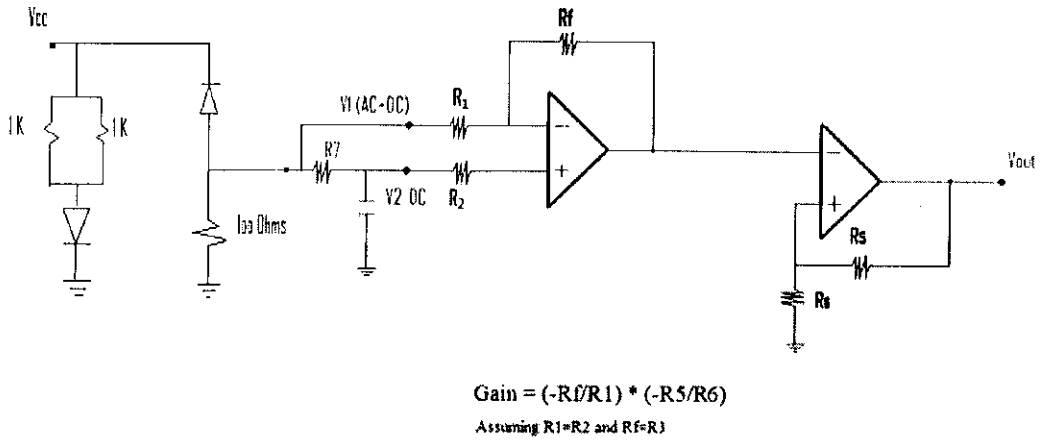


Figure 4.2: Sensor using Differential Amplifier

Then we came across a research paper which discusses calculating SPO2 values with the help of FFT algorithms and thus eliminates the need to separate AC and DC signals.

Therefore the circuit required for implementation of the pulse oximeter sensor requires only a dual wavelength LED, photodiode and a basic amplifier circuit.

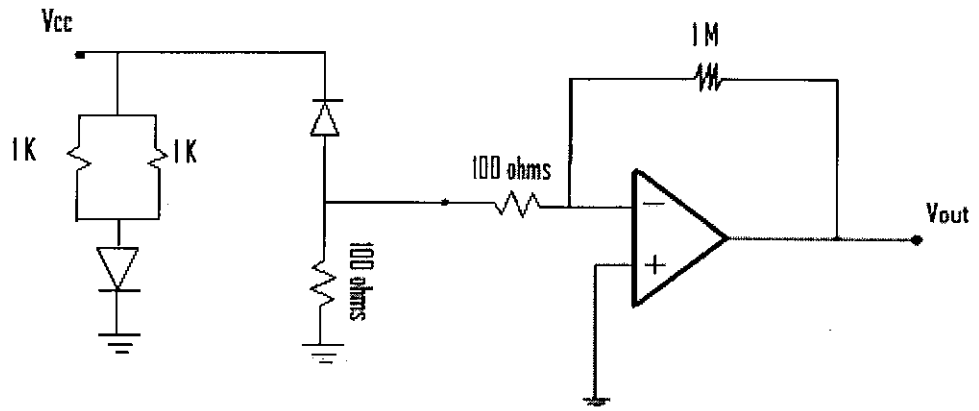


Figure 4.3: Sensor using Amplifier

4.3 ATmega32 Microcontroller

ATmega32 is a microcontroller by Atmel Corporation. It is an 8 bit microcontroller and has a memory of 32 K. It is based on the AVR architecture.

The AVR family of microcontrollers was chosen because of the following reasons.

1. On board Crystal oscillator of 1 MHz is provided. This greatly simplifies circuit designing and hence reduces the overall size of the product as well as the total power consumed.
2. An analog to digital convertor is also provided which is needed in this project. This further reduces the size of the final product.
3. AVR family is widely supported and extensive documentation is available online.

ATmega32 features:

• High-performance, Low-power AVR® 8-bit Microcontroller

• Advanced RISC Architecture

– 131 Powerful Instructions – Most Single-clock Cycle Execution

– 32 x 8 General Purpose Working Registers

– Fully Static Operation

– Up to 16 MIPS Throughput at 16 MHz

– On-chip 2-cycle Multiplier

• High Endurance Non-volatile Memory segments

– 32K Bytes of In-System Self-programmable Flash program memory

– 1024 Bytes EEPROM

– 2K Byte Internal SRAM

– Write/Erase Cycles: 10,000 Flash/100,000 EEPROM

– Data retention: 20 years at 85°C/100 years at 25°C

– Optional Boot Code Section with Independent Lock Bits

In-System Programming by On-chip Boot Program

True Read-While-Write Operation

– Programming Lock for Software Security

• JTAG (IEEE std. 1149.1 Compliant) Interface

– Boundary-scan Capabilities According to the JTAG Standard

– Extensive On-chip Debug Support

– Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface

• Peripheral Features

– Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes

– One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode

– Real Time Counter with Separate Oscillator

– Four PWM Channels

– 8-channel, 10-bit ADC

-8 Single-ended Channels

-7 Differential Channels in TQFP Package Only

-2 Differential Channels with Programmable Gain at 1x, 10x, or 200x

– Byte-oriented Two-wire Serial Interface

– Programmable Serial USART

– Master/Slave SPI Serial Interface

– Programmable Watchdog Timer with Separate On-chip Oscillator

– On-chip Analog Comparator

• Special Microcontroller Features

– Power-on Reset and Programmable Brown-out Detection

– Internal Calibrated RC Oscillator

– External and Internal Interrupt Sources

– Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby

• I/O and Packages

– 32 Programmable I/O Lines

– 40-pin PDIP, 44-lead TQFP, and 44-pad QFN/MLF

• Operating Voltages

– 2.7 - 5.5V for ATmega32L

– 4.5 - 5.5V for ATmega32

• Speed Grades

– 0 - 8 MHz for ATmega32L

– 0 - 16 MHz for ATmega32

• Power Consumption at 1 MHz, 3V, 25°C for ATmega32L

– Active: 1.1 mA

– Idle Mode: 0.35 mA

– Power-down Mode: < 1 μA

4.3.1 Analog to Digital Converter

Features

- 10-bit Resolution
- 0.5 LSB Integral Non-linearity
- ±2 LSB Absolute Accuracy
- 13 - 260 µs Conversion Time
- Up to 15 kSPS at Maximum Resolution
- 8 Multiplexed Single Ended Input Channels
- 7 Differential Input Channels
- 2 Differential Input Channels with Optional Gain of 10x and 200x
- Optional Left adjustment for ADC Result Readout
- 0 - VCC ADC Input Voltage Range
- Selectable 2.56V ADC Reference Voltage
- Free Running or Single Conversion Mode
- ADC Start Conversion by Auto Triggering on Interrupt Sources
- Interrupt on ADC Conversion Complete
- Sleep Mode Noise Canceler

The ATmega32 features a 10-bit successive approximation ADC. The ADC is connected to an 8-channel Analog Multiplexer which allows 8 single-ended voltage inputs constructed from the pins of Port A. The single-ended voltage inputs refer to 0V (GND).

The device also supports 16 differential voltage input combinations. Two of the differential inputs (ADC1, ADC0 and ADC3, ADC2) are equipped with a programmable gain stage, providing amplification steps of 0 dB (1x), 20 dB (10x), or

46 dB (200x) on the differential input voltage before the A/D conversion. Seven differential analog input channels share a common negative terminal (ADC1), while any other ADC input can be selected as the positive input terminal. If 1x or 10x gain is used, 8-bit resolution can be expected. If 200x gain is used, 7-bit resolution can be expected.

Operation

The ADC converts an analog input voltage to a 10-bit digital value through successive approximation. The minimum value represents GND and the maximum value represents the voltage on the AREF pin minus 1 LSB. Optionally, AVCC or an internal 2.56V reference voltage may be connected to the AREF pin by writing to the REFSn bits in the ADMUX Register. The internal voltage reference may thus be decoupled by an external capacitor at the AREF pin to improve noise immunity.

The analog input channel and differential gain are selected by writing to the MUX bits in ADMUX. Any of the ADC input pins, as well as GND and a fixed bandgap voltage reference, can be selected as single ended inputs to the ADC. A selection of ADC input pins can be selected as positive and negative inputs to the differential gain amplifier.

If differential channels are selected, the differential gain stage amplifies the voltage difference between the selected input channel pair by the selected gain factor. This amplified value then becomes the analog input to the ADC. If single ended channels are used, the gain amplifier is bypassed altogether.

The ADC is enabled by setting the ADC Enable bit, ADEN in ADCSRA. Voltage reference and input channel selections will not go into effect until ADEN is set. The ADC does not consume power when ADEN is cleared, so it is recommended to switch off the ADC before entering power saving sleep modes.

The ADC generates a 10-bit result which is presented in the ADC Data Registers, ADCH and ADCL. By default, the result is presented right adjusted, but can optionally be presented left adjusted by setting the ADLAR bit in ADMUX.

If the result is left adjusted and no more than 8-bit precision is required, it is sufficient to read ADCH. Otherwise, ADCL must be read first, then ADCH, to ensure that the content of the Data Registers belongs to the same conversion. Once ADCL is read, ADC access to Data Registers is blocked. This means that if ADCL has been read, and a conversion completes before ADCH is read, neither register is updated and the result from the conversion is lost. When ADCH is read, ADC access to the ADCH and ADCL Registers is re-enabled. The ADC has its own interrupt which can be triggered when a conversion completes. When ADC access to the Data Registers is prohibited between reading of ADCH and ADCL, the interrupt will trigger even if the result is lost. After the conversion is complete (ADIF is high), the conversion result can be found in the ADC Result Registers (ADCL, ADCH).

For single ended conversion, the result is where VIN is the voltage on the selected input pin and VREF the selected voltage reference 0x000 represents analog ground, and 0x3FF represents the selected reference voltage minus one LSB.

If differential channels are used, the result is where VPOS is the voltage on the positive input pin, VNEG the voltage on the negative input pin, GAIN the selected gain factor, and VREF the selected voltage reference. The result is presented in two's complement form, from 0x200 (-512d) through 0x1FF (+511d). Note that if the user wants to perform a quick polarity check of the results, it is sufficient to read the MSB of the result (ADC9 in ADCH). If this bit is one, the result is negative, and if this bit is zero, the result is positive.

4.4 Liquid Crystal Display Unit

The LCD is used as a display unit in our project. It displays the SPO2 reading in percentage and provides a useful interface for the user. The LCD we have used is a single row 16 character display. The basic pin configuration is shown below:

Pins Description

1 Ground

2 Vcc

3 Contrast Voltage

4 "R/S" _Instruction/Register Select

5 "R/W" _Read/Write LCD Registers

6 "E" Clock

7 - 14 Data I/O Pins

The Ground and Vcc pins are used to supply the ground and supply voltage respectively.

The contrast pin is used to specify the contrast (or "darkness") of the characters on the LCD screen and has important significance when displaying many characters. In our system we have grounded this pin.

The "R/S" bit is used to select whether data or an instruction is being transferred between the microcontroller and the LCD. If the Bit is set, then the byte at the current LCD "Cursor" Position can be read or written. When the Bit is reset, either an instruction is being sent to the LCD or the execution status of the last instruction is read back (whether or not it has completed).

The R/W bit is used to select read or write operation. This line is pulled low in order to write commands or character data to the module or pulled high to read character or status information from the registers.

The "E" Clock is used to initiate the data transfer within the LCD.

The LCD deals with ASCII code that are send 4 or 8 bits at a time. Thus the interface is a parallel bus, allowing simple and fast reading/writing of data to and from the LCD. The transfer of data is via the Data I/O pins.

Interface

Pin No.	Symbol	Description	Function
1	VSS	Ground	0V (GND)
2	VCC	Power Supply for logic circuit	+5 V
3	VEE	LCD contrast adjustment	
4	RS	Instruction/data Register selection	RS=0:Instruction Register RS=1:Data Register
5	R/W	Read/Write selection	R/W=0: Register write R/W=1: Register read
6	E	Enable Signal	
7	DB0	Data Input/Output Lines	8 BIT:DB0-DB7
8	DB1		
9	DB2		
10	DB3		
11	DB4		
12	DB5		
13	DB6		
14	DB7		
15	LED+	Supply Voltage for LED+	+5V

Table 4.1: Interface of LCD

4.5 Software Design

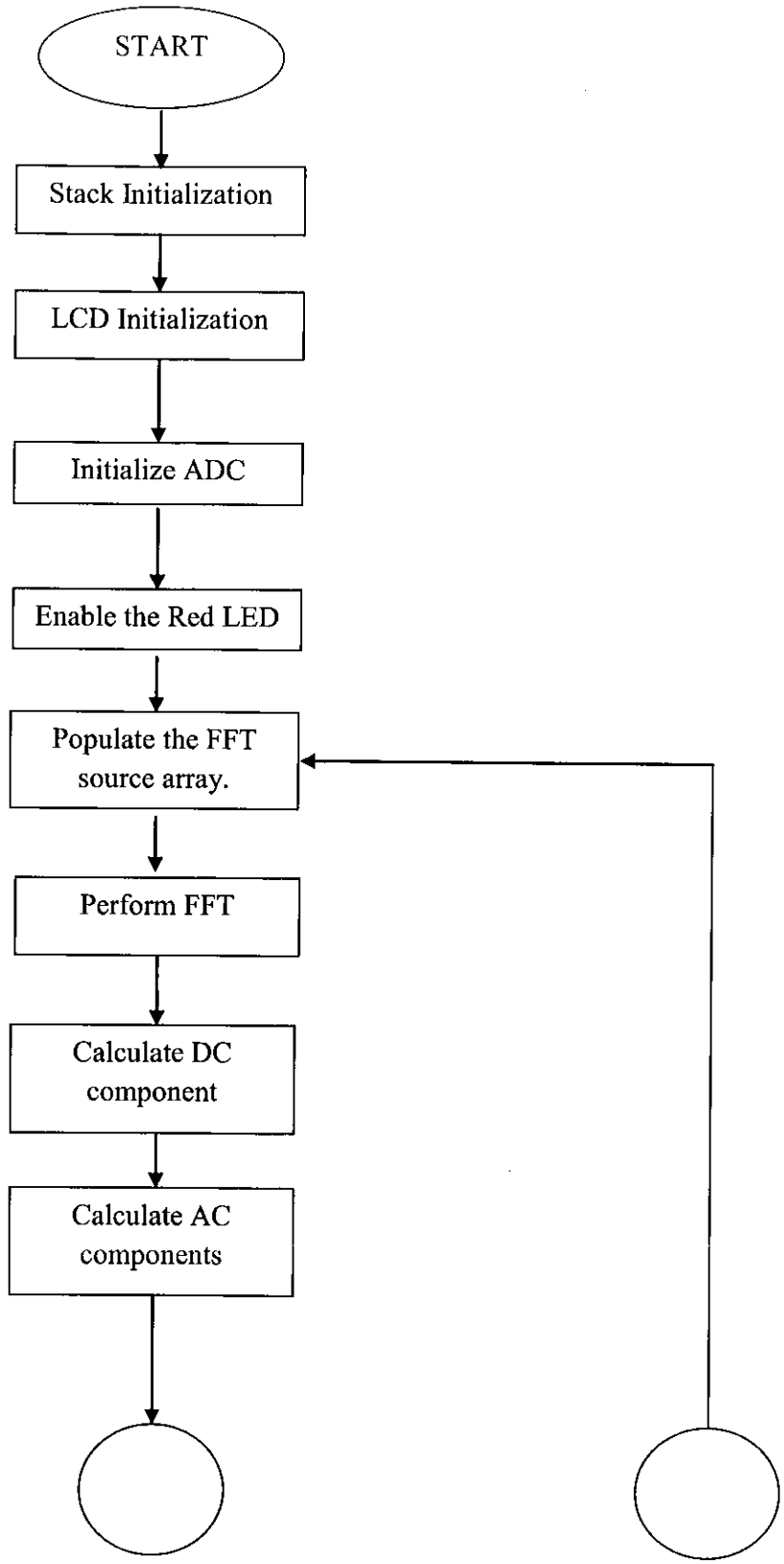
The software implements a butterfly-2-radix FFT algorithm with the use of Hemming window. The cardiac signal has a frequency of:

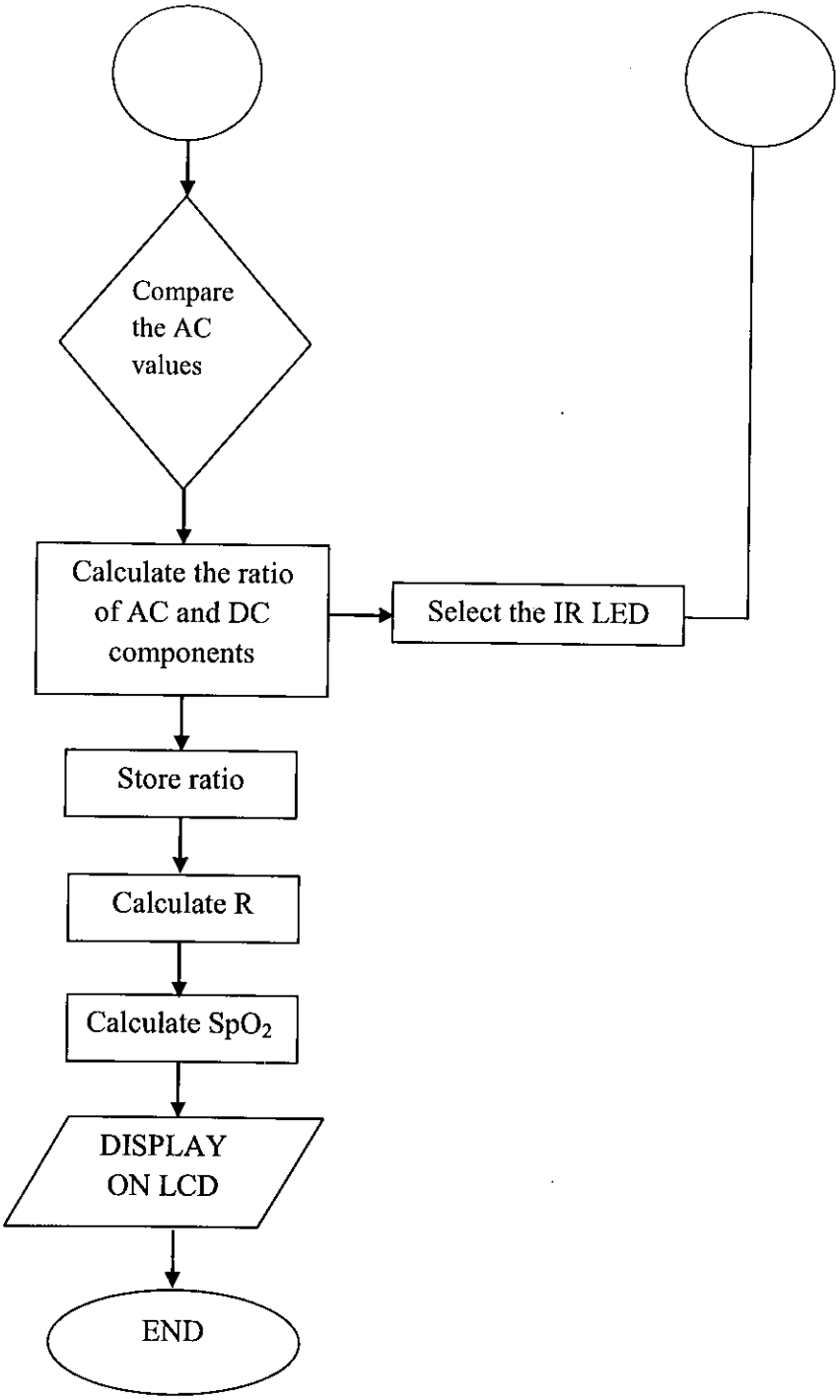
72/60 = 1.2 Hz

72 is the normal rate of heartbeat for an average human being. After increasing the range to 220 beats per minute we get a frequency range of 72-140 bpm, which gives us the frequency range of 1.2-2 Hz. The software should be able to:

a. Initialize the stack.
b. Initialize the LCD module.
c. Initialize the on board ADC in free running mode.
d. Enable the Red LED.
e. Populate FFT source array with 64, 2 byte values from A/D conversion with the help of ADC.
f. Perform FFT on the source array.
g. Find out DC component by considering the 0^{th} index of the FFT output array (0^{th} frequency component).
h. Find out the AC component at first index and second index of FFT output array.
i. Compare the above two values and determine which is greater. This value is the required AC amplitude.
j. Calculate the ratio of the AC and DC component.
k. Now select the IR led and repeat from (e).
l. Now calculate the ratio of the ratios that we calculated from above.
m. Multiply this value with 25 and subtract 110 from it.
n. Display this value (which is the SpO2) on the LCD.

FLOW DIAGRAM OF CODE




Compare the AC values
Calculate the ratio of AC and DC components
Select the IR LED
Store ratio
Calculate R
Calculate SpO_2
DISPLAY ON LCD
END

Chapter 5

Result and future prospects

5.1 Result:

The SpO2 values in a normal individual, calculated by a reliable pulse oximeter vary from 87-99, with 87 being the value in case of critical condition, hence our pulse oximeter should output results in this range.

We were getting SpO2 values with a range of individuals in the range 94-97, though the values were not stable.

These results were not highly accurate due t the low quality of sensor components. The accuracy can be improved greatly if we use higher quality components, which would also increase the cost.

The overall cost of the project is as follows.

Components	**Price(in Rs.)**
ATmega32 Microcontroller	150
LCD module	250
Dual wavelength LED	235
Photodiode	30
Sensor brackets	200
Op amps and resistors	35
Total Cost	**900**

5.2 Conclusion

The project "Pulse Oximeter" has been completed as a mandatory task in the partial fulfillment requirement for the degree of Bachelor of Electronics and Communication Engineering. The undertaken project, in essence was carried out as an application of biomedical instrumentation. The pulse oximeter sensor was used to extract the signal from the finger .Use of microcontroller and different hardware and proper interfacing among these devices led to the development of the "Pulse Oximeter". Different signal conditioning equipments like an amplifier helped to strengthen the signal which was low in amplitude and frequency. The system thus designed was a standalone system that can be used just by providing it with the power supply. From the project so concluded, it is found that the results obtained were found to have some errors. This is due to the presence of poor quality photo detector present in the circuit.

We were able to construct the whole project within our targeted price limit which was Rs. 1500. The design of a low-cost microcontroller based device for measuring the oxygen level has been described. The device has the advantage that it can be used by non-professional people at home to measure the oxygen level easily and safely.

Irrespective of these errors our system works satisfactorily displaying the SpO2 level in the desired range. The team has put all its sincere efforts in developing the project; however there does remain certain room for future enhancements. Hence it can be concluded that most of the objective of the project has been achieved and we have been successful in developing the "Pulse Oximeter".

This project has been a learning experience for us and we were intimated with the intricacies of developing an embedded system from scratch.

5.3 Future Prospects

This project can be further enhanced by incorporating a heart beat monitor so it becomes a complete package providing oxygen level and heartbeat rate information on a single console.

We can also implement a wireless interface to be connected with a PC or mobile to provide easy and accessible information.

A controller area network can also be implemented using Zigbee or some other protocol and can be used in hospitals for a scalar, integrated patient monitoring system.

5.4 Limitations

Although using FFT technique minimizes the errors introduced by motion of subject but still it's not highly reliable.

Hypothermia, anemia, shock, are among conditions under which the readings will be unreliable.

A pulse oximeter cannot determine between O2 and CO saturation, since both reflect similar amounts of red light. A patient with, say, a true O2 saturation of 80% and CO of 15% (possibly found in, say, a heavy smoker) will still have a pulse oximeter reading of 95%.

An oximeter is simply another tool rather than a guaranteed diagnosis, but it is the easiest to use out of all methods of determining blood oxygen saturation.

***

Appendix A

Program to calculate SpO_2 value

```
;Program for atmega32.
;It controls two LEDs and accepts the photo sensor's output
;at ADC pin, car22ulates the SPO2 values and displays it on LCD.
;fft routines taken from elam-chan.org

.include "m32def.inc"

#ifndef FFT_N
#define FFT_N 64
#endif
#define FFT_B 6

;.def lcd_rs PORTD7
;.def lcd_en PORTD6
;.def lcd_rw PORTD5

.def var1 = r17
.def var0 = r16
.def tmp0 = r18
.def tmp1 = r19
.def tmp2 = r20
.def T0L = r0
.def T0H = r1
.def T2L = r2
.def T2H = r3
.def T4L = r4
.def T4H = r5
.def T6L = r6
.def T6H = r7
```

```
.def T8L = r8
.def T8H = r9
.def T10L = r10
.def T10H = r11
.def T12L = r12
.def T12H = r13
.def T14L = r14
.def T14H = r15
.def AL = r16
.def AH = r17
.def BL = r18
.def BH = r19
.def CL = r20
.def CH = r21
.def DL = r22
.def DH = r23
.def EL = r24
.def EH = r25
.def XL = r26
.def XH = r27
.def YL = r28
.def YH = r29
.def ZL = r30
.def ZH = r31

.macro          ldiwdh,dl, abs
                ldi     \dl, lo8(\abs)
                ldi     \dh, hi8(\abs)
.endm

.macro          subiw       dh,dl, abs
                subi    \dl, lo8(\abs)
                sbci    \dh, hi8(\abs)
.endm
```

```
.macro      addw dh,dl, sh,sl
            add  \dl, \sl
            adc  \dh, \sh
.endm

.macro      addd d3,d2,d1,d0, s3,s2,s1,s0
            add  \d0, \s0
            adc  \d1, \s1
            adc  \d2, \s2
            adc  \d3, \s3
.endm

.macro      subw dh,dl, sh,sl
            sub  \dl, \sl
            sbc  \dh, \sh
.endm

.macro      subd d3,d2,d1,d0, s3,s2,s1,s0
            sub  \d0, \s0
            sbc  \d1, \s1
            sbc  \d2, \s2
            sbc  \d3, \s3
.endm

.macro      lddw dh,dl, src
            ldd  \dl, \src
            ldd  \dh, \src+1
.endm

.macro      ldw  dh,dl, src
            ld   \dl, \src
            ld   \dh, \src
.endm
```

```
.macro      stw   dst, sh,sl
            st    \dst, \sl
            st    \dst, \sh
.endm

.macro      clrw  dh, dl
            clr   \dh
            clr   \dl
.endm

.macro      lsrw  dh, dl
            lsr   \dh
            ror   \dl
.endm

.macro      asrw  dh, dl
            asr   \dh
            ror   \dl
.endm

.macro      lslw  dh, dl
            lsl   \dl
            rol   \dh
.endm

.macro      pushw       dh, dl
            push  \dh
            push  \dl
.endm
```

```
.macro      popw    dh, dl
            pop     \dl
            pop     \dh
.endm

.macro      lpmw    dh,dl, src
            lpm     \dl, \src
            lpm     \dh, \src
.endm

.macro      rjne    lbl
            breq    99f
            rjmp    \lbl
.endm

;Fractional Multiply (19clk) ----------------------------------------------------------
.macro      FMULS16 d3,d2,d1,d0 ,s1h,s1l, s2h,s2l
            fmuls   \s1h, \s2h
            movw    \d2, T0L
            fmul    \s1l, \s2l
            movw    \d0, T0L
            adc     \d2, EH         ;EH: zero reg.
            fmulsu  \s1h, \s2l
            sbc     \d3, EH
            add     \d1, T0L
            adc     \d2, T0H
            adc     \d3, EH
            fmulsu  \s2h, \s1l
            sbc     \d3, EH
            add     \d1, T0L
            adc     \d2, T0H
            adc     \d3, EH
.endm
```

```
.macro      SQRT32          ; 32bit square root (526..542clk)
            clr   T6L
            clr   T6H
            clr   T8L
            clr   T8H
            ldi   BL, 1
            ldi   BH, 0
            clr   CL
            clr   CH
            ldi   DH, 16
90:         lsl   T2L
            rol   T2H
            rol   T4L
            rol   T4H
            rol   T6L
            rol   T6H
            rol   T8L
            rol   T8H
            lsl   T2L
            rol   T2H
            rol   T4L
            rol   T4H
            rol   T6L
            rol   T6H
            rol   T8L
            rol   T8H
            brpl  91f
            add   T6L, BL
            adc   T6H, BH
            adc   T8L, CL
            adc   T8H, CH
            rjmp  92f
```

```
91:                 sub   T6L, BL
                    sbc   T6H, BH
                    sbc   T8L, CL
                    sbc   T8H, CH
92:                 lsl   BL
                    rol   BH
                    rol   CL
                    andi  BL, 0b11111000
                    ori   BL, 0b00000101
                    sbrc  T8H, 7
                    subi  BL, 2
                    dec   DH
                    brne  90b
                    lsr   CL
                    ror   BH
                    ror   BL
                    lsr   CL
                    ror   BH
                    ror   BL
.endm

tbl_window:
        .dc.w   2621, 2693, 2910, 3270, 3768, 4401, 5161, 6042, 7036, 8132,
                9320, 10588, 11926, 13318, 14753, 16216
        .dc.w   17694, 19171, 20634, 22069, 23462, 24799, 26068, 27256,
                28352, 29345, 30226, 30987, 31619, 32117, 32477, 32694
        .dc.w   32766, 32694, 32477, 32117, 31619, 30987, 30226, 29345,
                28352, 27256, 26068, 24799, 23462, 22069, 20634, 19171
        .dc.w   17694, 16216, 14753, 13318, 11926, 10588, 9320, 8132, 7036,
                6042, 5161, 4401, 3768, 3270, 2910, 2693

tbl_cos_sin:
        .dc.w   32767, 0, 32609, 3211, 32137, 6392, 31356, 9511, 30272, 12539,
                28897, 15446, 27244, 18204, 25329, 20787
```

```
        .dc.w   23169, 23169, 20787, 25329, 18204, 27244, 15446, 28897,
                12539, 30272, 9511, 31356, 6392, 32137, 3211, 32609
        .dc.w   0, 32766, -3211, 32609, -6392, 32137, -9511, 31356, -12539,
                30272, -15446, 28897, -18204, 27244, -20787, 25329
        .dc.w   -23169, 23169, -25329, 20787, -27244, 18204, -28897, 15446,
                -30272, 12539, -31356, 9511, -32137, 6392, -32609, 3211

tbl_bitrev:
        .dc.w   0*4, 32*4, 16*4, 48*4, 8*4, 40*4, 24*4, 56*4, 4*4, 36*4, 20*4,
                52*4, 12*4, 44*4, 28*4, 60*4
        .dc.w   2*4, 34*4, 18*4, 50*4, 10*4, 42*4, 26*4, 58*4, 6*4, 38*4, 22*4,
                54*4, 14*4, 46*4, 30*4, 62*4

array_srcandout:
        .dc.w   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0
        .dc.w   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0

array_bfly:
        .dc.w   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0
        .dc.w   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0


.org $0000

        jmp     RESET                   ;reset interrupt vector

RESET:

        ldi     r16, high(RAMEND)       ;stack initialization

        out     SPH, r16

        ldi     r16, low(RAMEND)
```

```
out     SPL, r16
ldi     r16, $ff
out     DDRD, r16
out     DDRC, r16
ldi     r16, $00
out     PORTD, r16
call    lcd_init
ldi     r16, $01
call    lcd_command
ldi     r16, $80
call    lcd_command
call    delay
ldi     r19, $bf
out     ADCSR, r19          ;division factor 128
ldi     r19, $c0
out     ADMUX, r19
ldi     r19, 0x87
out     ADCSR, r19          ;division factor 128
ldi     r19, 0xc0
out     ADMUX, r19
;sbi    ADCSR, 6
clr     r25
out     DDRB, r25
```

```
ldi     r25, 0x01           ;portb pin1 select red led
out     PORTB, r25
call    generate_src
call    get_ac_dc           ;max red ac amplitude and dc
call    store_ac_dc
pop     r16
pop     r17
pop     r18
pop     r19
call    div16u
pop     r21
pop     r22
ldi     r25, 0x02           ;portb pin2 select ir led
out     PORTB, r25
call    generate_src
call    get_ac_dc           ;max ir ac amplitude and dc
call    store_ac_dc
pop     r16
pop     r17
pop     r18
pop     r19
call    div16u
pop     r21
```

```
        pop     r22
        pop     r18
        pop     r19
        pop     r16
        pop     r17
        call    div16u
        pop     r21
        pop     r22
        pop     r16
        pop     r17
        ldi     r17, 25
        mul     r16, r17
        ldi     r17, 110
        sub     r0, r17
        mov     r16, r0
        call    mk_decu8        ;convert SpO2 into decimal string
loop:
        jmp     loop
```

```
;-----------------------------------------------------------------

;8 bit binary to decimal

mk_decu8:

                clr     var1                ;8 bit entry

mk_decu16:                                  ;16 bit entry

                clr     tmp2                ;digit counter

                inc     tmp2                ; decimal string generating loop

                clr     tmp0                ;var1 /= 10;

                ldi     tmp1,16

                lsl     var0

                rol     var1

                rol     tmp0

                cpi     tmp0,10

                brcs    PC+3

                subi    tmp0,10

                inc     var0

                dec     tmp1

                brne    PC-8

                subi    tmp0,-'0'           ;Push the remainder (a decimal digit)

                push    tmp0

                cp      var0,tmp1           ;if(var =! 0)

                cpc     var1,tmp1           ; continue digit loop;

                brne    PC-16
```

```
        cp      tmp2,len            ;Adjust string length
        brcc    PC+5
        inc     tmp2
        ldi     var0,' '
        push    var0
        rjmp    PC-5
        pop     var0                ;Put decimal string
        call    lcd_data            ; Put a char (var0) to memory,
                                    ;console or any display device
        dec     tmp2
        brne    PC-3
        ret
;---------------------------------------------------------------------------
;16 bit unsigned division
div16u:
        clr     r20                 ;initialize variables
        clr     r21                 ; mod = 0;
        ldi     r22,16              ; r22 = 16
        lsl     r16                 ;var1 = var1 << 1
        rol     r17                 ;/
        rol     r20                 ;mod = mod << 1 + carry;
        rol     r21                 ;/
        cp      r20,r18             ;if (mod => var2) {
```

```
        cpc     r21,r19             ; mod -= var2; var1++;
        brcs    PC+4                ; }
        inc     r16
        sub     r20,r18
        sbc     r21,r19
        dec     r22                 ;if (r22 > 0)
        brne    PC-11               ; continue loop;
        pushw   r17, r16            ;quotient
        pushw   r21, r20            ;remainder
        ret
;-------------------------------------------------------------------------
;fill array_srcandout with sensor output signal samples
generate_src:
        ldiw    ZH,ZL, array_srcandout
        ldi     r19, 63
check:
        dec     r19
        brmi    exit_sampling
        jmp     sample
exit_sampling:
        ret
sample:
        sbi     ADCSR, 6            ;start conversion
```

```
noop:       nop
            sbrs    ADCSR, 4
            jmp     noop
            ldi     r16, ADCL
            ldi     r17, ADCH
            stw     Z+, r17,r16
            jmp     check
;-------------------------------------------------------------------------------
;get dc and ac components by using fft of 64 sample points
;with hamming window and butterfly technique
get_ac_dc:
            call    generate_src
            ldiw    EH, EL, array_srcandout
            ldiw    DH, DL, array_bfly
            call    fft_input
            ldiw    EH, EL, array_bfly
            call    fft_execute
            ldiw    EH, EL, array_bfly
            ldiw    DH, DL, array_srcandout
            call    fft_output
            ret
```

```
;-------------------------------------------------------------------------

;store dc and ac components from fft output

store_ac_dc:

                ldiw    AH, AL, array_srcandout

                ldw     BH, BL, A+

                ldiw    CH,CL, A+

                ldiw    DH, DL, A

                cp      DH, CH

                brlo    case1

                breq    case2

case1:          movw    EL, CL

case2:          movw    EL, DL

                pushw   BH, BL          ;dc component

                pushw   EH, EL          ;max ac amplitude

                ret

;-------------------------------------------------------------------------

;delay of 100x50 clocks, approx 5ms

delay:

                push    r16

                ldi     r16, $32

here:           dec     r16

                brbs    1, back

                push    r17
```

```
                ldi     r17, $64
there:          dec     r17
                brbc    1, there
                pop     r17
                jmp     here
back:           pop     r16
                ret
;-----------------------------------------------------------------------------
;function for issuing a command to lcd
;command byte is taken from r16
lcd_command:
                ldi     r17, $00
                out     PORTD, r17
                call    delay
                out     PORTC, r16
                nop
                ldi     r17, 0b01000000         ;enable pulse
                out     PORTD, r17
                call    delay
                ldi     r17, 0b00000000
                out     PORTD, r17
                call    delay
                ret
```

```
;-------------------------------------------------------------------------

;function for issuing display data to lcd

;display byte is taken from r16

lcd_data:

        ldi     r21, 0b10000000

        out     PORTD, r21

        call    delay

        out     PORTC, r16          ;send display byte

        nop

        ldi     r21, 0b11000000

        out     PORTD, r21

        call    delay               ;lcd rs = 1 for data

        ldi     r21, 0b10000000     ;enable pulse

        out     PORTD, r21

        call    delay

        ret

;-------------------------------------------------------------------------

;function for lcd initialization

lcd_init:

        push    r16

        ldi     r16, $38

        call    lcd_command

        ldi     r16, $01
```

```
                call    lcd_command
                ldi     r16, $0c
                call    lcd_command
                ldi     r16, $80
                call    lcd_command
                ldi     r16, $06
                call    lcd_command
                pop     r16
                ret
;---------------------------------------------------------------------------
;fft routines
fft_input:
                pushw   T2H,T2L
                pushw   AH,AL
                pushw   YH,YL
                movw    XL, EL          ;X = array_srcandout;
                movw    YL, DL          ;Y = array_bfly;
                clr     EH              ;Zero

                ldiw    ZH, ZL, tbl_window  ;Z = &tbl_window[0];
                ldiw    AH, AL, FFT_N       ;A = FFT_N;
1:              lpmw    BH, BL, Z+          ;B = *Z++; (window)
                ldw     CH, CL, X+          ;C = *X++; (I-axis)
```

```
FMULS16     DH, DL, T2H, T2L, BH, BL, CH, CL      ;D = B * C;
            stw     Y+, DH, DL                    ;*Y++ = D;
            stw     Y+, DH, DL                    ;*Y++ = D;
            subiw   AH, AL, 1                     ;while(--A)
            brne    1b
            popw    YH,YL
            popw    AH, AL
            popw    T2H, T2L
            clr     r1
            ret
fft_execute:
            pushw   T2H, T2L
            pushw   T4H, T4L
            pushw   T6H, T6L
            pushw   T8H, T8L
            pushw   T10H, T10L
            pushw   T12H, T12L
            pushw   T14H, T14L
            pushw   AH, AL
            pushw   YH, YL
            movw    ZL, EL                        ;Z = array_bfly;
            ldiw    EH,EL, 1                      ;E = 1;
```

```
                ldiw    XH,XL, FFT_N/2          ;X = FFT_N/2;
1:              ldi     AL, 4                   ;T12 = E; (angular speed)
                mul     EL, AL
                movw    T12L, T0L
                mul     EH, AL
                add     T12H, T0L
                movw    T14L, EL                ;T14 = E;
                pushw   EH,EL
                movw    YL, ZL                  ;Z = &array_bfly[0];
                mul     XL, AL                  ;Y = &array_bfly[X];
                addw    YH,YL, T0H,T0L
                mul     XH, AL
                add     YH, T0L
                pushw   ZH,ZL
2:              clrw    T10H,T10L               ;T10 = 0 (angle)
                clr     EH                      ;Zero reg.
3:              lddw    AH,AL, Z+0              ;A = *Z - *Y; *Z++ += *Y;
                asrw    AH,AL
                lddw    DH,DL, Y+0
                asrw    DH,DL
                movw    CL, AL
                subw    AH,AL, DH,DL
                addw    CH,CL, DH,DL
```

```
stw     Z+, CH,CL

lddw    BH,BL, Z+0                  ;B = *Z - *Y; *Z++ += *Y;

asrw    BH,BL

lddw    DH,DL, Y+2

asrw    DH,DL

movw    CL, BL

subw    BH,BL, DH,DL

addw    CH,CL, DH,DL

stw     Z+, CH,CL

movw    T0L, ZL

ldiw    ZH,ZL, tbl_cos_sin          ;C = cos(T10); D = sin(T10);

addw    ZH,ZL, T10H,T10L

lpmw    CH,CL, Z+

lpmw    DH,DL, Z+

movw    ZL, T0L

FMULS16     T4H,T4L,T2H,T2L, AH,AL, CH,CL
                                    ;*Y++ = A * C + B * D;

FMULS16     T8H,T8L,T6H,T6L, BH,BL, DH,DL

addd    T4H,T4L,T2H,T2L, T8H,T8L,T6H,T6L

stw     Y+, T4H,T4L

FMULS16     T4H,T4L,T2H,T2L, BH,BL, CH,CL
                                    ;*Y++ = B * C - A * D;

FMULS16     T8H,T8L,T6H,T6L, AH,AL, DH,DL
```

```
subd    T4H,T4L,T2H,T2L, T8H,T8L,T6H,T6L
stw     Y+, T4H,T4L
addw    T10H,T10L, T12H,T12L        ;T10 += T12; (next angle)
sbrs    T10L, FFT_B + 1
rjmp    3b
ldi     AL, 4                       ;Y += X,  Z += X
mul     XL, AL
addw    YH,YL, T0H,T0L
addw    ZH,ZL, T0H,T0L
mul     XH, AL
add     YH, T0L
add     ZH, T0L
ldi     EL, 1                       ;while(--T14)
subw    T14H,T14L, EH,EL
rjne    2b
popw    ZH,ZL
popw    EH,EL
lslw    EH,EL                       ;E *= 2;
lsrw    XH,XL                       ;while(X /= 2)
adiw    XL, 0
rjne    1b
popw    YH,YL
popw    AH,AL
```

```
                popw   T14H,T14L
                popw   T12H,T12L
                popw   T10H,T10L
                popw   T8H,T8L
                popw   T6H,T6L
                popw   T4H,T4L
                popw   T2H,T2L
                ret
fft_output:
                pushw  T2H,T2L
                pushw  T4H,T4L
                pushw  T6H,T6L
                pushw  T8H,T8L
                pushw  T10H,T10L
                pushw  AH,AL
                pushw  YH,YL
                movw   T10L, EL              ;T10 = array_bfly;
                movw   YL, DL                ;Y = array_output;
                ldiw   ZH,ZL, tbl_bitrev     ;Z = tbl_bitrev;
                clr    EH                    ;Zero
                ldiw   AH,AL, FFT_N / 2
                                             ;A = FFT_N / 2
1:              lpmw   XH,XL, Z+             ;X = *Z++;
```

```
addw   XH,XL, T10H,T10L                ;X += array_bfly
ldw    BH,BL, X+                       ;B = *X++
ldw    CH,CL, X+                       ;C = *X++
FMULS16    T4H,T4L,T2H,T2L, BH,BL, BH,BL
                                       ;T4:T2 = B * B
FMULS16    T8H,T8L,T6H,T6L, CH,CL, CH,CL
                                       ;T8:T6 = C * C
addd   T4H,T4L,T2H,T2L, T8H,T8L,T6H,T6L
                                       ;T4:T2 += T8:T6
SQRT32                                 ;B = sqrt(T4:T2)
stw    Y+, BH,BL                       ;*Y++ = B
subiw  AH,AL, 1                        ;while(--A)
rjne   1b
popw   YH,YL
popw   AH,AL
popw   T10H,T10L
popw   T8H,T8L
popw   T6H,T6L
popw   T4H,T4L
popw   T2H,T2L
clr    r1
ret
```

```
fmuls_f:
        movw  CL, EL                              ;C = E
        clr   EH                                  ;Zero
        FMULS16   ZH,ZL,XH,XL, CH,CL, DH,DL       ;Z:X = C * D
        movw  EL, ZL
        clr   r1
        ret
```

***

References

1. L. M. Schnapp, "Pulse Oximetry: Uses and Abuses," *Chest*, 1990, volume 98, pgs 1244-1250. http://www.chestjournal.org/cgi/reprint/98/5/1244. [Accessed September 2, 2007]

2. Signal Processing Using Fourier & Wavelet Transform for Pulse Oximery, J.M. Kim, S.H. Kim, D.J. Lee, H.S. Lim, Department of biomedical Engineering, College of Medicine, Chung-nam National University, Daesa-dong, Jung-gu, Tae-jeon, 30 1-72 **1**, Korea, knewton@chollian.net

3. C. Zamarron, "Utility of Oxygen Saturation and Heart Rate Spectral Analysis Obtained from Pulse Oximetric Recording in the Diagnosis of Sleep Apnea Syndrome," *Chest*, 2003, volume 123, pgs 1567-1576. Available Chestjournal.com,

4. J. G. Webster, et al, *Design of Pulse Oximeters*, Series in Medical Physics and Biomedical Engineering. Boca Raton: CRC Press, 1997.

5. Yitzhak Mendelson, "Pulse Oximetry", in Wiley Encyclopedia of Biomedical Engineering, John Wiley & Sons, Inc, 2006.

6. Design of Pulse Oximeters, Oliver Wieben
7. http://ieeexplore.ieee.org/iel2/691/6333/00247420.pdf?tp=&isnumber=&arnumber=247420

8. Pulse Oximetry: Principles and Limitations; James E. Sinex, MD
9. http://www.analog.com/library/analogdialogue/archives/41-1/pulse_oximeter.html

10. J. G.Webster, *Design of Pulse Oximeters*. Bristol, U.K.: Inst. Phys..,
11. 1997.
12. http://www.chestjournal.org/cgi/reprint/123/5/1567. [Accessed September 15, 2007].

13. http://www.nonin.com/documents/Avant%209700%20Brochure.pdf

14. http://www.gehealthcare.com/usen/oximetry/docs/TruSat%20Brochure.pdf

15. http://www.nonin.com/documents/Avant%204000%20Brochure.pdf

16. http://www.nonin.com/documents/3100%20WristOx%20Brochure.pdf

17. http://www.spomedical.com/7500.php

18. http://www.medical.philips.com/main/products/patient_monitoring/products/philips_i
19. nfo_center/

20. http://alivetec.com/images/pulseoxhand.jpg, http://alivetec.com/products.html