PPU College of Engineering and Technology

The Home of Competent Engineers and Researchers

PALESTINE POLYTECHNIC UNIVERSITY

Engineering and technical collage

Electrical and computer department

Communication and Electronic Engineer

Graduation Project

Emerging wireless networks for telemedicine system

Project Supervisor

Dr . Murad Abu Sbieh

Prepared by

Mohammed Awad Munther Hlahlah Mahmoud Titi

2014

Abstract

Patients who are at risk require that their cardiac health to be monitored frequently whether they are indoors or outdoors so that emergency treatment is possible. Telemedicine is widely considered to be part of the inevitable future of the modern practice of medicine. The patient monitoring part will be based on the new emerging standard 802.11ac. This technology is based on MIMO and OFDM technique. It promises to provide Gbps data rate. This high data rate should allow monitoring several patient simultaneously. It should allow the possibility of delivering high quality video signals to special rooms inside hospitals.

The system should be capable of measuring, processing, and transmitting vital parameters over the hospital wireless network. The GSM technology will be used for connecting hospitals for information sharing. In this part, it is intended to design and implement a system for unified blood bank among Palestinian hospitals. It is intended to design, implement and assess the entire system in Palestinian Hospitals.

This allows efficient simultaneous monitoring of patients and also stands behind the importance of that data communication between hospitals needed for the purpose of creating unified blood bank.

وصف المشروع

المرضى الذين هم في خطر تتطلب أوضاع صحتهم القلبية رصدها بشكل متكرر سواء كانوا في داخل أو في خارج المشفى بحيث يكون العلاج في حالات الطوارئ ممكنا. ويعتبر على نطاق واسع التطبيب عن بعد جزءا من المستقبل لممارسة الطب الحديث. وسوف يستند هذا على مراقبة المرضى على تقنية 802.11ac الجديدة. وتستند هذه التقنية على MIMO وتقنية OFDM . انها تستخدم ل تقديم معدل بيانات بالجيجابايت في الثانية. معدل البيانات هذه عالية مما تسمح برصد العديد من المرضى في وقت واحد. مما تسمح ايضا بإمكانية تقديم إشارات الفيديو عالية الجودة إلى غرف خاصة داخل المستشفيات.

يجب أن يكون النظام قادرا على قياس ومعالجة و نقل المعلمات الحيوية عبر الشبكة اللاسلكية داخل المستشفى. سيتم ذلك باستخدام تكنولوجيا الجيل الثالث من الجي اس ام ليتم الربط بين المستشفيات لتبادل المعلومات في هذا الجزء سيتم تصميم نظام لبنك الدم الموحد للمستشفيات الفلسطينية. ويهدف الى تصميم وتنفيذ و تقييم النظام بأكمله في المستشفيات الفلسطينية من خلال قاعدة بيانات موحدة في جميع المستشفيات وهذا يسمح للمتابعة المتزامنة للمرضى و أيضا تقف وراء أهمية أن نقل البيانات بين المستشفيات اللازمة لغرض إنشاء بنك الدم الموحد.

Table of Contents

Figures

Tables

Equations

Chapter one

Introduction

Chapter contents

1.1 Overview

In this project, we aim at investigating the potential of new emerging wireless technologies for real time tele-monitoring of patients as well as connecting hospitals for information sharing. The patient monitoring part will be based on the new emerging standard 802.11ac. This technology is based on MIMO and OFDM technique. It promises to provide Gbps data rate. This high data rate should allow monitoring several patient simultaneously. It should allow the possibility of delivering high quality video signals to special rooms inside hospitals. The system should be capable of measuring, processing, and transmitting vital parameters over the hospital wireless network. The GSM technology will be used for connecting hospitals for information sharing. In this part, it is intended to design and implement a system for unified blood bank among Palestinian hospitals. It is intended to design, implement and assess the entire system in Palestinian Hospitals.

Current monitoring of patient is based mostly on one-way communication: data collected by sensors and are sent via gateway to a central server and made available to health care personal.

The importance of this kind of project stands behind allowing efficient simultaneous monitoring of patients and also stands behind the importance of that data communication between hospitals needed for the purpose of creating unified blood bank.

1.2 Project Motivation

Patients who are at risk require that their cardiac health to be monitored frequently whether they are indoors or outdoors so that emergency treatment is possible. Telemedicine is widely considered to be part of the inevitable future of the modern practice of medicine. It is gaining more and more momentum as a new approach for patients surveillance outside hospitals (homes) to encourage public safety, a ciliate early diagnosis, treatment, and for increased convenience. Defined as the "use of advanced Tele Communication technologies "to exchange the information about the patient's health care status and provide health care services across is now currently being used by doctors, hospitals and other healthcare providers around the world with conventional mode of treatment.

Telemedicine systems are already available to enable the doctor to monitor a patient remotely for home care emergency applications. Nowadays, Wireless networking is an emerging technology that will allow different users to access electronically, regardless of their geographic topography. The use of wireless communication between mobile users has become increasingly popular due to the advancements in computer and wireless technologies. Implementation of wireless technology in the existing heart rate monitoring system eliminates the physical constraints imposed by hard-wired link and allows users to conduct own check up at anytime anywhere.

Moreover, newer cellular access technologies, such as Fifth generation (5G), and others provide much higher data transmission speeds (rates) than basic third generation (3G) GSM cellular system offering future telemedicine solutions endless choices for high-end designs. These relatively new wireless technologies are deployed mostly in or around crowded high income metropolitan areas for our proposed scheme.

Despite of the good advantages for wireless communication system, there are some challenges that may affect system performance. These challenges may come from the surrounding environment. One of the most important challenges faces the wireless communication system is data rate which is a very important consideration in data communication. Data rate indicating how fast data can be sent in bits per second over a channel. Data rate depends on three factors :

1) The bandwidth available.

2) The level of the signals.

3) The quality of the channel (the level of noise).

1.3 Project Objectives

The project objectives are summarized as :

- Develop a tele-monitoring system for Palestinian Hospitals.
- Investigate the potential of new 802.11ac MIMO-OFDM devices to support monitoring health care scenarios.
- Study the performance of 802.11ac networks in terms of coverage, data rate, multimedia support for hospital deployment and health care applications.
- Design and implement a unified blood bank among Palestinian hospitals.

1.4 The main hypothesis

Are as following :

- 802.11ac technology will allow efficient simultaneous monitoring inside hospitals.
- Tele-monitoring of patients inside hospitals will simplify and enhance the services provided by hospitals.
- Unified blood bank will enhance the ability of collaboration among hospitals as well as accelerating the process of rescuing patients.

1.5 Project Phases

The project is organized around two phases.

Phase one primarily involves the design and implementation of the monitoring subsystem, while phase two involves the design and implementation of the connection network for the realization of the unified blood bank. In particular, the project is composed of the following working packages :

Phase one :

The objectives of phase one are :

- Design of sensors and camera circuits.
- Design of the interfacing circuit between sensors and Wi-Fi modems.
- Installation and configuration of the wireless Wi-Fi network.
- Installation and connection of the LCD for displaying reported measurements.

Phase two:

The objectives of phase two are :

- Design and implementation of the connection GSM network within hospitals.
- Design and implementation of the database for the realization of the unified blood bank.
- Testing and assessment of the entire system.

1.6 Related Work

Modern telemedicine started developing since the advent of telegraph, radio and telephones and has continued since, albeit at a very slow place and at isolated places, mainly exemplified by its use during American civil war, radio medical services for seafarer's by maritime nations and telegraphic transmission during the earlier part of 19th century.

The previous project (Telemedicine system using GSM and WIFI Network) uses 2.5 G mobiles and WIFI to transfer body temperature and ECG signal as SMS. But, the data rate of 2.5 G is 9.6Kbps and 4G provides 1Gbps.the previous uses cellular phone to transfer data.

This leads to limited service that 2.5G provides. Further, The 2.5G itself provides limited application support compared to 4G GSM .

The paper of [1] , a telemedicine system and mobile telemedicine system .this paper introduces a telemedicine system design, which provides several common services based on wireless communication technologies. The system includes the multifunction portable measure equipment design by using embedded system, the reliable communication and customer services design on the Smartphone terminal, powerful database management design in the medical center, the direct interface design on the Browser-server terminal.

The paper of [2], In this paper we consider the transmission of stored video from a server to a client for medical applications such as, Tele-monitoring, to optimize medical quality of service (m-QOS) and to examine how the client buffer space can be used efficiently and effectively towards reducing the rate variability of the compressed variable bit rate (VBR) video.

The paper of [3], discusses Remote Patient Monitoring Within a Future 5G Infrastructure and Abstract Systems of wearable or implantable medical devices (IMD), sensor systems for monitoring and transmitting physiological recorded signals, will in future health care services be used for purposes of remote monitoring. Today, there exist several constraints, probably preventing the adoption of such services in clinical routine work. Within a future 5G infrastructure, new possibilities will be available due to improved addressing solutions and extended security services in addition to higher bandwidth in the wireless communication link. Thus 5G solutions can represent a paradigm shift regarding remote patient's monitoring and tracking possibilities, with enhancement in transmitting information between patients and health care services. Some aspects of new possibilities are highlighted in describing a realistic scenario within a future 5G framework.

1.7 Time schedule

Week / sk	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
stem finition						🟩	🟩	🟩									
alysis									🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
sign													🟩	🟩	🟩	🟩	🟩
umentation						🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩

Table 1.1 Time schedule

1.8 Estimated Cost of Components.

Component (Item)	No. of items	Exact Cost
Camera	1	300 $
Wi-Fi 802.11ac adaptors	2(75$ for each)	150S
802.11ac Access point	1	150$
Wires and connectors	------	20$
SIM 300 GSM modem	2(300S for each)	600$
Reparatory sensor	1	20S
Temperature sensor	1	10S
LCD DISPLAY	1	300$
Microcontroller	1	250$
POWER SUPPLY	1	10$
Total cost	11	1810S = 1281JD

Table 1.2 Estimated Cost of Components

Chapter Two

Theoretical Background

Chapter contents

2.1 Telemedicine Concept.

2.1.1 What is Telemedicine?

"The use of advanced telecommunication and information technologies to exchange health information and provide health care services across geographic, time, social and cultural barriers". Telemedicine may be as simple as two health professionals discussing a case over the telephone, or as complex as using satellite technology and video conferencing equipment to conduct a real-time consultation between medical specialists in two different countries. It can also involve the use of an unmanned robot (Wikipedia).

Concomitant advances in information technology and medical technology have led to the increasing focus, development and use of telemedicine systems and services around the world, particularly in developed countries. The goals of these systems are:

- Increase the accessibility of and to professional caregivers.
- Increase the quality and continuity of care to patients.
- Increase the focus on preventive medicine through early intervention.
- Reduce the overall cost of healthcare.
- For education and training.
- For providing services to remote areas in case of natural calamities, disasters and military and space operations, remote monitoring.

2.1.2 Telemedicine Vs Telehealth :

Telemedicine is frequently used interchangeably with tele-health in common usage; however the two are not same. The table below summarizes the salient features of both.

Tele-health	Tele-medicine
Teleradiology	Patient records
Telepathlogy	Disease management information access
Telederiratology	Remote monitoring
Telepsychiatry	Patient compliance
Focus on discipline technology	Pilling access

Table 2.1: Summarization of the salient features of Telemedicine and Telehealth

2.1.3 Primary Delivery Methods:

Telemedicine is practiced on the basis of two concepts: real time (synchronous) and store-and-forward (asynchronous).

- **Real Time Telemedicine (synchronous)** is also referred to as "two way interactive television **(IATV)".** It can be as simple as telephone calls or as complex as sophisticate virtual reality (VR) robotic surgery or tele-surgery. In it providers/patients at distant locations interact with each other using communication technology in the form of audiovisual and wireless or microwave signals. Apart from video-conferencing, peripheral sensing devices can also be attached to the patient or the equipment to aid in interactive examination. For example, tele-stethoscope to monitor patients heartbeat or tele-otoscope to examine patients ear. It can also be used for monitoring of long term care patients or patients at their home. In fact, due to the severe cost constraints, quality and continuity of care issues, mal-distribution of physicians in different geographic regions and scarcity of the same, remote home care of chronically ill patients and of long term care patients, is the fastest emerging use of telemedicine. Specialties for which it is used most often are psychiatry, internal medicine, rehabilitation, cardiology, pediatrics, obstetrics and gynecology, neurology.

- **Store and Forward (asynchronous)** technology involves acquiring medical data (images, bio-signals) and transmitting this data to a medical specialist for consultation, evaluation or other purposes. It does not require simultaneous communication between both parties in real time. Tele-radiology and tele-dermatology is the fastest emerging branch that use such kind of services. Overall radiology, pathology and dermatology are most conducive for utilizing this mechanism.

The above mentioned basic telemedicine technologies are utilized for providing various services that spawns numerous specialties and can be broadly categorized as:

- Tele-home Home Health Care
- Tele-surgery.
- Teleradiology.
- General Telemedicine.
- Teleradiology.
- Telemedicine Consulting.
- Teledermatology.
- Emergency Telemedicine.
- Telepathlogy.
- Tele-monitoring.
- Telesurgery.

2.1.4 Telemedicine Structure:

The telemedicine unit basically consists of four modules:

1- Biosignal Acquisition Module: for Biosignal acquisition through sensors and peripheral devices

2- Digital Camera: for digital image or video capturing

3- Processing Unit: comprised of computers of varying complexity

4- Communication Module: Wi-Fi modems, wireless network, GSM modems, Internet, LAN, etc.[4], [5],[6] .

2.2 Wi-Fi 802.11ac

2.2.1 : Wi-Fi 802.11ac adapters

802.11ac is a set of wireless communications specifications proposed for networking. For consumer electronics, 802.11ac will act as an evolution from 802.11g and 802.11n, widely used for wireless networking in home and corporate environments.

802.11ac products will be marketed towards users who require a lot of bandwidth and speed in their wireless networks, whether that's for gaming or streaming of high definition content from the Internet or another source on the network. It is sometimes called 5G Wi-Fi.

There is many types of adaptors, but the type needed in our project is the WiFi USB Adaptor which has been designed to connect your notebook or desktop computer to an 802.11ac network to give you lag-free HD streaming of multimedia or maximum throughput on wireless data transfers. This, of course, is tremendously important for those trying to wirelessly game or who just want the fastest speeds that their network can provide. Of course the biggest feature of the adapter is its ability to connect with the last routers on the market using 802.11ac speeds. This dual-band adapter functions over 2.4GHz and 5.0GHz frequency bands for speeds up to 1200 Mbps.

2.2.2 Wi-Fi 802.11ac In terms of Speed, Reliability, and Quality.

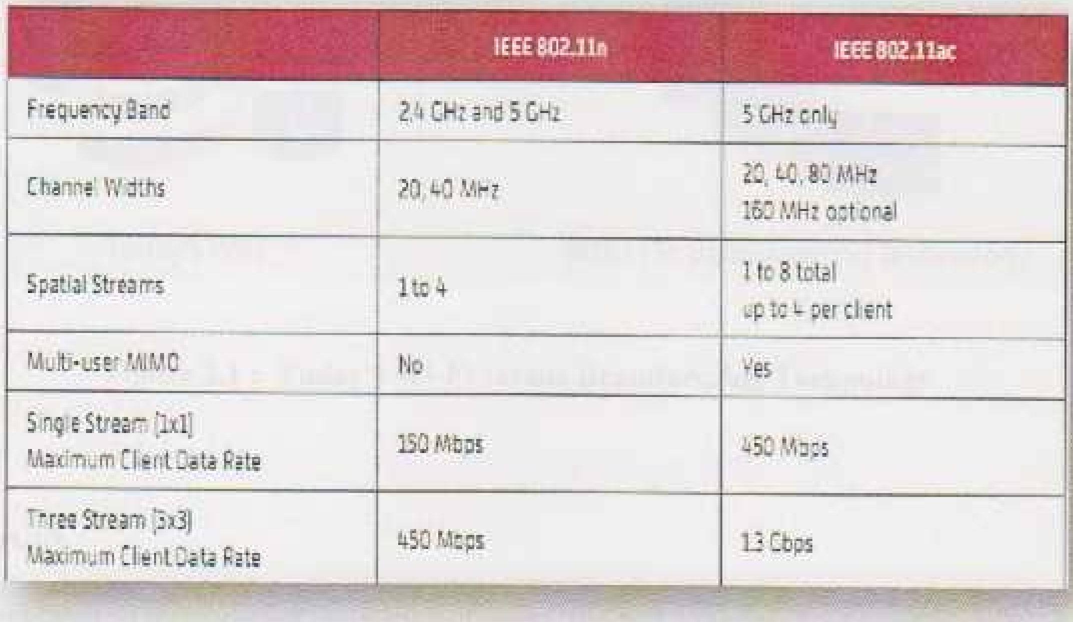
To meet these growing needs from the ever growing proliferation of WiFi devices, there is a new wireless standard under development.

The new standard, known as 802.11ac, promises extraordinary improvements in speed, reliability, and quality.

Speed

The most powerful 802.11n devices cap out today at a maximum link rate of 450 megabits per second at close range, with declining performance as the range increases. In contrast, the new 802.11ac standard can achieve more than three times the performance of the current standard, with speeds up to 1.35 Gigabits per second. What's more, 802.11ac has the capability to maintain a higher level of performance at any range, compared with its predecessors. This increase in speed is achieved by providing wider frequency bands, faster processing, and multiple antennas. Think of it in terms of an automotive highway, where a transition is made from a highway with four lanes to one with eight lanes, while simultaneously upgrading to a Ferrari on an auto-bahn highway with no speed limit. The result is faster traffic and significantly reduced congestion to enable a decrease in overall travel time.

The 3X speed improvement achieved by the new standard means that the 450 Mbps performance from today's fastest 3 antenna 802.11n device can be achieved by single antenna 802.11ac device – with similar power consumption. This means that a typical tablet with single antenna 802.11n 150Mbps WiFi can now support 450 Mbps with 802.11ac – without any increase in power consumption or decrease in battery life.



	IEEE 802.11n	IEEE 802.11ac
Frequency Band	2.4 GHz and 5 GHz	5 GHz only
Channel Widths	20, 40 MHz	20, 40, 80 MHz 160 MHz optional
Spatial Streams	1 to 4	1 to 8 total up to 4 per client
Multi-user MIMO	No	Yes
Single Stream (1x1) Maximum Client Data Rate	150 Mbps	450 Mbps
Three Stream (3x3) Maximum Client Data Rate	450 Mbps	1.3 Gbps

Table 2.2 : Feature Enhancement comparison : 802.11n / 802.11ac

Reliability

Speed is largely irrelevant if the connection lacks reliability. For example, most users have experienced the irritating video buffering during video playback, which causes frozen or jittery screens. By increasing the bandwidth capacity and improved processing, 802.11ac enables far more bandwidth to be available for consumption by wireless devices, which helps avoid interference, and improves the speed for demanding applications such as hi-definition video streaming.

The result is more effective coverage with fewer dead zones. Another feature that is expected to boost the reliability of the connection at required speed and range is the much improved "beam forming" standard, which provides directional signal transmission and reception. Previous standards can only receive and transmit omnidirectional signals, which are subject to significant levels of interference, due to the fact that the signals are transmitted indiscriminately in every possible direction. With beam forming, there's an understanding of the relative location of the device, and the signal is correspondingly strengthened in that direction.

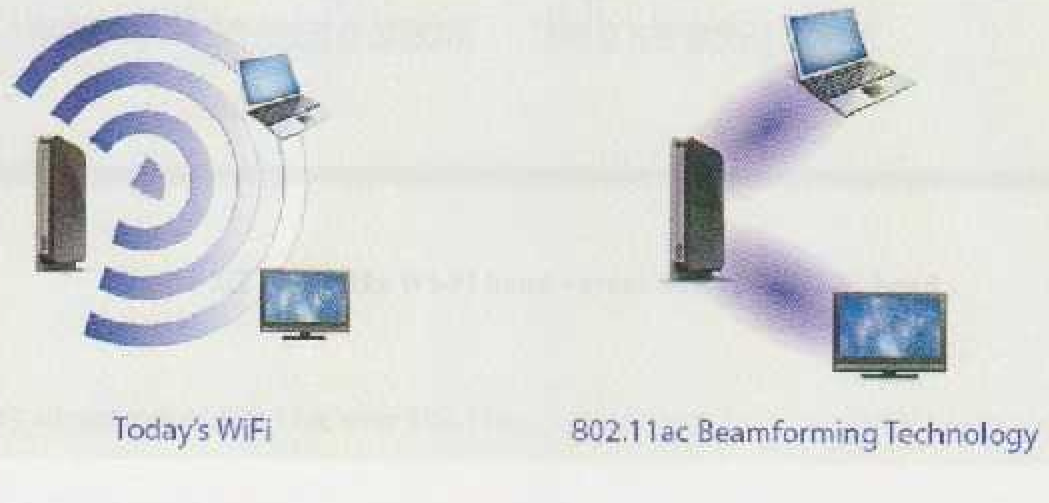


Figure 2.1 : Today's Wi-Fi versus Beamforming Technology

Quality

In addition to its increased capacity and range, the 802.11ac standard operates in the 5 GHz wireless spectrum, which is less prone to interference. Though more widespread in usage (all 802.11b and g devices operate exclusively in the 2.4 GHz spectrum). 2.4 GHz only has three non-overlapping channels for transmission, which are crowded due to the vast number of interfering devices, including other WiFi access points, microwave ovens, cordless phones, Bluetooth devices, and baby monitors. As a result, the environment is noisy, which increases interference and degrades performance. In contrast, the 5 GHz channel is much cleaner with less interference, with 23 non-overlapping channels – 8 times more than what is available in the 2.4 GHz spectrum – which makes it far more suitable for applications such as video streaming and gaming, which are very sensitive to packet loss and delays.

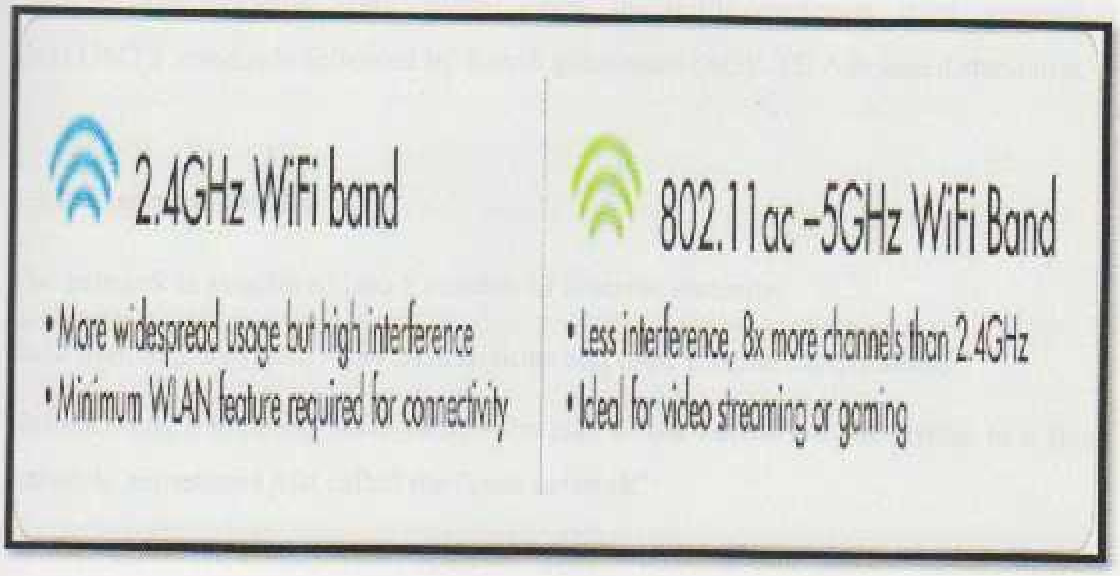

2.4GHz WiFi band
• More widespread usage but high interference
• Minimum WLAN feature required for connectivity
802.11ac -5GHz WiFi Band
• Less interference, 8x more channels than 2.4GHz
• Ideal for video streaming or gaming

Figure 2.2 : 2.4GHz Wi-Fi band versus 802.11ac Wi-Fi band

Key advantages of 802.11ac over 802.11n:

• Gigabit speed wireless with approximately 3 times the performance of 802.11n
• Better performance at any range with fewer dead spots
• More reliable connections for media streaming with beam-forming
• More WiFi bandwidth on your mobile
• Only utilizes the 5 GHz Band, which is less prone to interference
• Backward compatible to 802.11 a & n, which also use the 5 GHz band.[7].

2.3 GSM technology :

2.3.1 Introduction:

The GSM standard was developed as a replacement for first generation (1G) analog cellular networks, and originally described a digital, circuit-switched network optimized for full duplex voice telephony. This was expanded over time to include data communications, first by circuit-switched transport, then packet data transport via GPRS (General Packet Radio Services) and EDGE (Enhanced Data rates for GSM Evolution or EGPRS).

Further improvements were made when the 3GPP developed third generation (3G) UMTS standards followed by fourth generation (4G)LTE Advanced standards.

2.3.2 Technical details:

The network is structured into a number of discrete sections:

- Base Station Subsystem – the base stations and their controllers explained
- Network and Switching Subsystem – the part of the network most similar to a fixed network, sometimes just called the "core network"
- GPRS Core Network – the optional part which allows packet-based Internet connections
- Operations support system (OSS) – network maintenance

Base station subsystem:

GSM is a cellular network, which means that cell phones connect to it by searching for cells in the immediate vicinity. There are five different cell sizes in a GSM network—macro, micro, Pico, femto, and umbrella cells. The coverage area of each cell varies according to the implementation environment. Macro cells can be regarded as cells where the base station antenna is installed on a mast or a building above average rooftop level. Micro cells are cells whose antenna height is under average rooftop level; they are typically used in urban areas. Pico cells are small cells whose coverage diameter is a few dozen meters; they are mainly used indoors. Femto cells are cells designed for use in residential or small business environments and connect to the service provider's network via a broadband internet connection. Umbrella cells are used to cover shadowed regions of smaller cells and fill in gaps in coverage between those cells.

Cell horizontal radius varies depending on antenna height, antenna gain, and propagation conditions from a couple of hundred meters to several tens of kilometers. The longest distance the GSM specification supports in practical use is 35 kilometers (22 mi). There are also several implementations of the concept of an extended cell,[10] where the cell radius could be double or even more, depending on the antenna system, the type of terrain, and the timing advance. Indoor coverage is also supported by GSM and may be achieved by using an indoor Pico cell base station, or an indoor repeater with distributed indoor antennas fed through power splitters, to deliver the radio signals from an antenna outdoors to the separate indoor distributed antenna system. These are typically deployed when significant call capacity is needed indoors, like in shopping centers or airports. However, this is not a prerequisite, since indoor coverage is also provided by in-building penetration of the radio signals from any nearby cell.

GSM carrier frequencies:

Into GSM frequency ranges for 2G and UMTS frequency bands for 3G), with most 2G GSM networks operating in the 900 MHz or 1800 MHz bands. Where these bands were already allocated, the 850 MHz and 1900 MHz bands were used instead (for example in Canada and the United States). In rare cases the 400 and 450 MHz frequency bands are assigned in some countries because they were previously used for first-generation systems.

Most 3G networks in Europe operate in the 2100 MHz frequency band. For more information on worldwide GSM frequency usage, see GSM frequency bands.

Regardless of the frequency selected by an operator, it is divided into timeslots for individual phones. This allows eight full-rate or sixteen half-rate speech channels per radio frequency. These eight radio timeslots (or burst periods) are grouped into a TDMA frame. Half-rate channels use alternate frames in the same timeslot. The channel data rate for all8 channels is 270.833 Kbit/s, and the frame duration is 4.615 ms. The transmission power in the handset is limited to a maximum of 2 watts in GSM 850/900 and 1 watt in GSM 1800/1900.

Network switching subsystem (NSS) :

is the component of a GSM system that carries out call switching and mobility management functions for mobile phones roaming on the network of base stations. It is owned and deployed by mobile phone operators and allows mobile devices to communicate with each other and telephones in the wider public switched telephone network **(PSTN).**

The GPRS core network:

is the central part of the general packet radio service **(GPRS)** which allows 2G, 3G and **WCDMA** mobile networks to transmit IP packets to external networks such as the Internet. The GPRS system is an integrated part of the GSM network switching subsystem

Operations support systems (OSS)

are computer systems used by telecommunications service providers. The term OSS most frequently describes "network systems" dealing with the telecom network itself, supporting processes such as maintaining network inventory, provisioning services, configuring network components, and managing faults. The complementary term, business support systems or BSS, is a newer term and typically refers to "business systems" dealing with customers, supporting processes such as taking orders, processing bills, and collecting payments. The two systems together are often abbreviated OSS/BSS, BSS/OSS or simply B/OSS.[8].

2.4 OFDM and MIMO

2.4.1 MIMO concepts:

in radio, multiple-input and multiple-output, or MIMO is the use of multiple antennas at both the transmitter and receiver to improve communication performance.

MIMO technology has attracted attention in wireless communications, because it offers significant increases in data throughput and link range without additional bandwidth or increased transmit power. It achieves this goal by spreading the same total transmit power over the antennas to achieve an array gain that improves the spectral efficiency (more bits per second per hertz of bandwidth) and/or to achieve a diversity gain that improves the link reliability (reduced fading). Because of these properties, MIMO is an important part of modern wireless communication standards such as IEEE 802.11n (Wi-Fi), 4G,3GPP Long Term Evolution, WiMAX and HSPA+.

2.4.2 MIMO function:

MIMO can be sub-divided into three main categories, precoding, spatial multiplexing or SM, and diversity coding. Precoding is multi-stream beamforming, in the narrowest definition. In more general terms, it is considered to be all spatial processing that occurs at the transmitter. In (single-stream) beamforming, the same signal is emitted from each of the transmit antennas with appropriate phase and gain weighting such that the signal power is maximized at the receiver input. The benefits of beamforming are to increase the received signal gain, by making signals emitted from different antennas add up constructively, and to reduce the multipath fading effect. In line-of-sight propagation, beamforming results in a well defined directional pattern. However, conventional beams are not a good analogy in cellular networks, which are mainly characterized by multipath propagation. When the receiver has multiple antennas, the transmit beamforming cannot simultaneously maximize the signal level at all of the receive antennas, and precoding with multiple streams is often beneficial. Note that precoding requires knowledge of channel state information (CSI) at the transmitter and the receiver.

Spatial multiplexing

requires MIMO antenna configuration. In spatial multiplexing, a high rate signal is split into multiple lower rate streams and each stream is transmitted from a different transmit antenna in the same frequency channel. If these signals arrive at the receiver antenna array with sufficiently different spatial signatures and the receiver has accurate CSI, it can separate these streams into (almost) parallel channels. Spatial multiplexing is a very powerful technique for increasing channel capacity at higher signal-to-noise ratios (SNR). The maximum number of spatial streams is limited by the lesser of the number of antennas at the transmitter or receiver. Spatial multiplexing can be used without CSI at the transmitter, but can be combined with precoding if CSI is available. Spatial multiplexing can also be used for simultaneous transmission to multiple receivers, known as space-division multiple access or multi-user MIMO, in which case CSI is required at the transmitter. The scheduling of receivers with different spatial signatures allows good separability.

Diversity Coding techniques are used when there is no channel knowledge at the transmitter. In diversity methods, a single stream (unlike multiple streams in spatial multiplexing) is transmitted, but the signal is coded using techniques called space-time coding. The signal is emitted from each of the transmit antennas with full or near orthogonal coding. Diversity coding exploits the independent fading in the **multiple** antenna links to enhance signal diversity. Because there is no channel knowledge, there is no beamforming or array gain from diversity coding. Diversity coding can be combined with spatial multiplexing when some channel knowledge is available at the transmitter.

2.4.3 OFDM concepts:

Orthogonal Frequency-Division Multiple Access (OFDMA) is a multi-user version of the popular orthogonal frequency-division multiplexing (OFDM) digital modulation scheme. Multiple access is achieved in OFDMA by assigning subsets of subcarriers to individual users . Claimed advantages over OFDM with time-domain statistical multiplexing:

- Allows simultaneous low-data-rate transmission from several users.
- Pulsed carrier can be avoided.
- Lower maximum transmission power for low data rate users.
- Shorter delay, and constant delay.
- Contention-based multiple access (collision avoidance) is simplified.
- Further improves OFDM robustness to fading and interference.
- Combat narrow-band interference.

Orthogonality of Sub-Channel Carriers :

OFDM communications systems are able to more effectively utilize the frequency spectrum through overlapping sub-carriers. These sub-carriers are able to partially overlap without interfering with adjacent sub-carriers because the maximum power of each sub-carrier corresponds directly with the minimum power of each adjacent channel. Below, we illustrate the frequency domain of an OFDM system graphically. As you can see from the figure, each sub-carrier is represented by a different peak. In addition, the peak of each sub-carrier corresponds directly with the zero crossing of all channels. Note that OFDM channels are different from band limited FDM channels how they apply a pulse-shaping filter. With FDM systems, a sinc-shaped pulse is applied in the time domain to shape each individual symbol and prevent ISI. With OFDM systems, a sinc-shaped pulse is applied in the frequency domain of each channel. As a result, each sub-carrier remains orthogonal to one another.

Figure 2.3 : OFDM subcarriers orthogonal to one another

Transmitter/Receiver Implementation

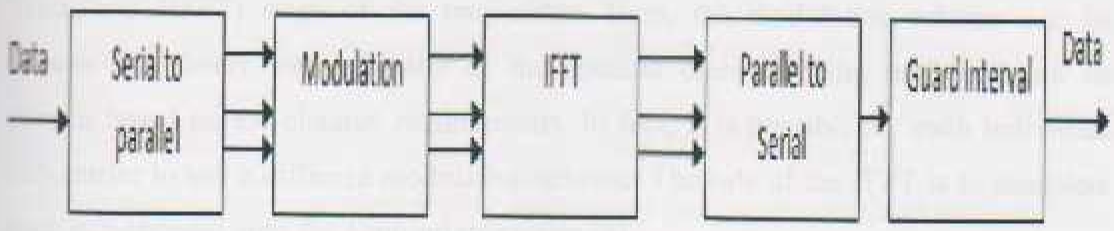


Figure 2.4: OFDM Transmitter

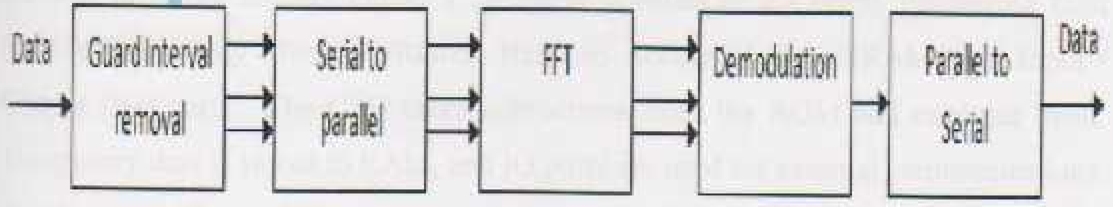


Figure 2.5 : OFDM Receiver

Serial to Parallel Conversion:

In an OFDM system, each channel can be broken into various sub-carriers. The use of sub-carriers makes optimal use out of the frequency spectrum but also requires additional processing by the transmitter and receiver. This additional processing is necessary to convert a serial bit stream into several parallel bit streams to be divided among the individual carriers. Once the bit stream has been divided among the individual sub-carriers, each sub-carrier is modulated as if it was an individual channel before all channels are combined back together and transmitted as a whole. The receiver performs the reverse process to divide the incoming signal into appropriate sub-carriers and then demodulating these individually before reconstructing the original Bit stream.

Modulation with the Inverse FFT:

The modulation of data into a complex waveform occurs at the Inverse Fast Fourier Transform (IFFT) stage of the transmitter. Here, the modulation scheme can be chosen completely independently of the specific channel being used and can be chosen based on the channel requirements. In fact, it is possible for each individual sub-carrier to use a different modulation scheme. The role of the IFFT is to modulate each sub-channel onto the appropriate carrier.[9]

2.5 Microcontrollers

2.5.1 : Definitions and features

As you probably already know, a computer consists of a Central Processing Unit (CPU), Read Only Memory (ROM), Random Access Memory (RAM) and Input / Output (IO) ports. The CPU takes instructions from the ROM and executes them. Temporary data is stored in RAM, and IO ports are used for external communications. A microcontroller is different to a microprocessor in that it has all these things on a single chip. While internally these chips are very complex, the external interface can actually be very simple – a fully functional microcontroller may be contained in a chip that has as few as 8 pins.

Another definition for microcontroller A microcontroller is designed for a very specific task to control a particular system and is used in automatically controlled products and devices, such as automobile engine control.

There are many types of microcontrollers, one of these types is Arduino which becomes one of the most popular microcontrollers . There are many different types of Arduino microcontrollers which differ not only in design and features, but also in size and processing capabilities.

There are many features that are common to all Arduino boards, making them very versatile. All Arduino boards are based around the ATMEGA AVR series microcontrollers from ATMEL which feature both analog and digital pins. Arduino also created software which is compatible with all Arduino microcontrollers. The software, also called "Arduino", can be used to program any of the Arduino microcontrollers by selecting them from a drop-down menu. Being open source, and based around C, Arduino users are not necessarily restricted to this software, and can use a variety of other software to program the microcontrollers.

There are many microcontroller types and architectures different in length of register and instruction word. We can mention here the most known types of microcontrollers:

* PIC (8-bit PIC16, PIC18, 16-bit dsPIC33/PIC24) , Intel 8051 , MIPS , ATMega .

There are many additional manufacturers who use the open-source schematics provided by Arduino to make their own boards (either identical to the original, or with variations to add to the functionality). For example, the most popular board, the Diecimilla / Duemilanove (and now the Uno) has dozens of look-alike boards from other suppliers which differ slightly (different USB port, color etc) from the original.

The main advantages of Arduino microcontrollers:

- Inexpensive - Arduino boards are relatively inexpensive compared to other microcontroller platforms.
- Simple, clear programming environment - The Arduino programming environment is easy-to-use for beginners, yet flexible enough for advanced users.
- Open source and extensible software - The Arduino software and is published as open source tools, available for extension by experienced programmers.

- Open source and extensible hardware - The Arduino is based on Atmel's ATMEGA8 and ATMEGA168 microcontrollers.

2.5.2 : Some types of microcontrollers

Here there are some of Arduino types includes some details and pictures :

Arduino Mega 1280 / 2560

The most recent addition to the Arduino lineup is the Arduino MEGA. This board is physically larger than all the other boards and offers significantly more digital and analog pins. The MEGA uses a different processor allowing greater program size and more.

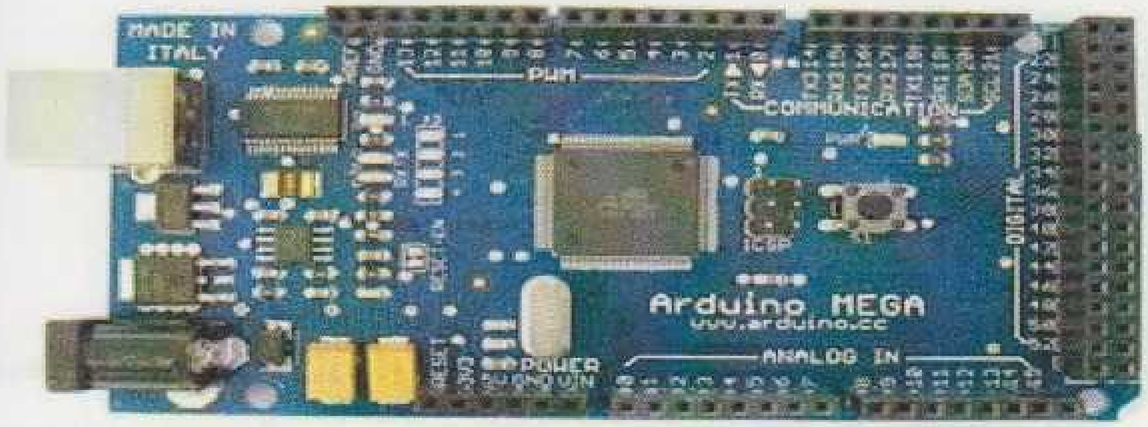


Figure 2.6 : Arduino Mega 1280 / 2560

Arduino Nano / Nano Lite

The last breadboard mountable Arduino is the Arduino Nano. This microcontroller distinguishes itself from the others by having the USB to serial chip and connector onboard. The Nano has 8 analog pins and 14 digital pins. There are the ISCP headers to re-flash the ATMega chip. There is also the Arduino Nano Lite which does not include the downward facing pin headers.

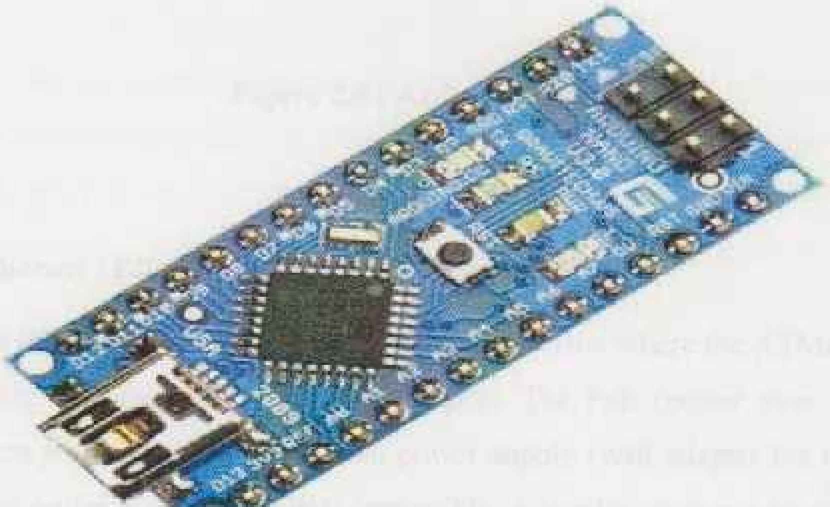


Figure 2.7 : Arduino Nano / Nano Lite

Arduino Fio

The Arduino Fio is a bit of a one-off board and is essentially an Arduino Mini with a built-in LiPo charger and XBee headers.

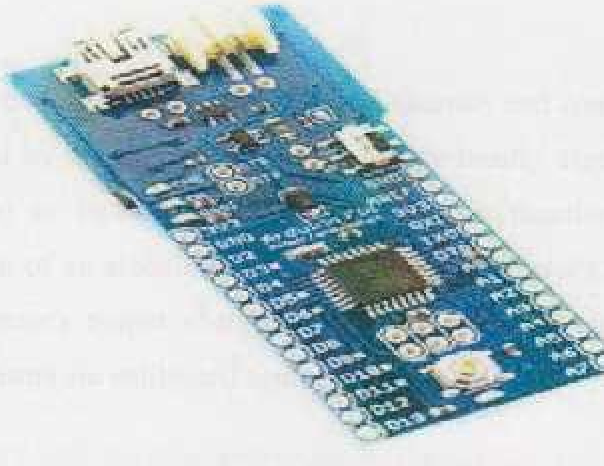


Figure 2.8 : Arduino Fio

Arduino Ethernet / Ethernet PoE

The Arduino Ethernet is essentially a normal Arduino Uno where the ATMega8 chip and USB plug are changed for an Ethernet port. The PoE (power over Ethernet) version means you don't need a separate power supply (wall adapter for example), although your router must also be PoE compatible. A similar setup can be done using a standard shield-compatible Arduino and an Ethernet shield.

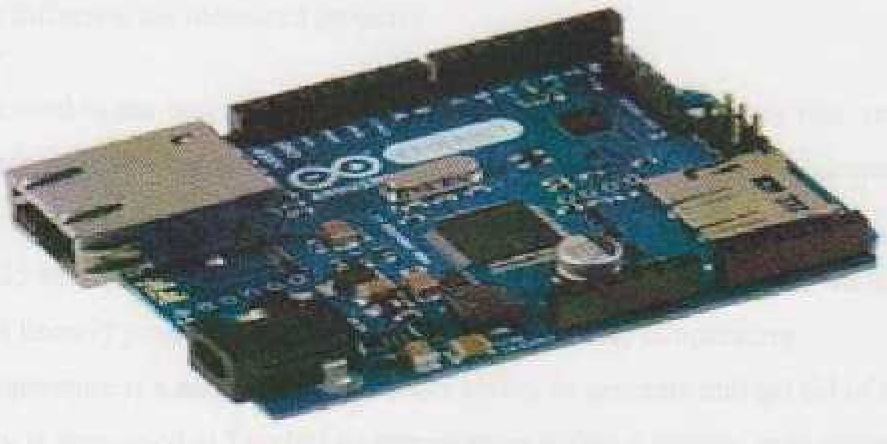


Figure 2.9 : Arduino Ethernet / Ethernet PoE

2.6 : sensors

A sensor is a converter that measures a physical quantity and converts it into a signal which can be read by an observer or by an (today mostly electronic) instrument , which responds to an input quantity by generating a functionally related output usually in the form of an electrical or optical signal. A sensor's sensitivity indicates how much the sensor's output changes when the measured quantity changes. For accuracy, most sensors are calibrated against known standards.

Sensors are used in everyday objects such as touch-sensitive elevator buttons (tactile sensor) and lamps which dim or brighten by touching the base. There are also innumerable applications for sensors of which most people are never aware. Applications include cars, machines, aerospace, medicine, manufacturing and robotics.

A good sensor obeys the following rules:

- Is sensitive to the measured property only
- Is insensitive to any other property likely to be encountered in its application
- Does not influence the measured property

Here, we need to use two types of sensors temperature and respiratory rate sensor . In this project, temperature has been measured from the skin surface of the human body, using the LM35 temperature sensor which is shown in Figure 2.9.

The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature.

Body temperature is a measure of the body's ability to generate and get rid of heat.

The body is very good at keeping its temperature within a narrow, safe range in spite of large variations in temperatures outside the body.

- Temperature measurement locations in the body . The body temperature can be measured in many locations on the body. The skin surface, mouth, ear, armpit, and rectum are the most commonly used places. Temperature can also be measured on your forehead. In this project, the temperature is measured from the skin surface.

- The normal body temperature. Most people think of a "normal" body temperature as an oral temperature of 37°C. This is an average of normal body temperatures. The temperature may actually be 1°F (0.6°C) or more above or below 37°C. The normal body temperature changes by as much as 0.6°C (1°F) throughout the day, depending on how active you are and the time of day. There are several types of sensors to measure the temperature, such as:

1. Thermistors.
2. The Thermocouple.
3. LM35 sensors.

The main features of the LM35 temperature sensor are:

- Calibrated directly in ° Celsius (Centigrade).
- Linear + 10.0 mV/°C scale factor.
- 0.5°C accuracy guarantee able (at +25°C).
- Rated for full −55° to +150°C range.
- Suitable for remote applications.
- Low cost due to wafer-level trimming.
- Operates from 4 to 30 volts.
- Less than 60 µA current drain.
- Low impedance output, 0.1 W for 1 mA load.

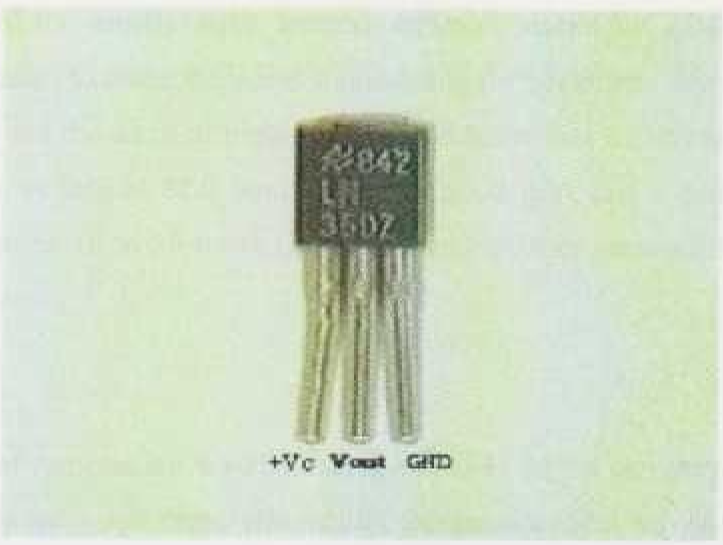


Figure 2.10: LM35 Temperature Sensor

2.7 Cameras

A camera is an optical instrument that records images that can be stored directly, transmitted to another location, or both. These images may be still photographs or moving images such as videos or movies. The term camera comes from the word camera obscura (Latin for "dark chamber"), an early mechanism for projecting images.

There are many types of digital cameras Broadly speaking Digital cameras can be segregated into two major types: Consumer and Digital Single Lens Reflex (DSLR). A third type, the Prosumer camera, usually offers all the features of the consumer camera with some of the features of the DSLR.

Digital Consumer Cameras

These are made as an all-in-one device. They typically have a fixed lens, and are often loaded with a lot of gimmicks to make them attractive to the average consumer.

- Digital Single Lens Reflex Cameras (Digital SLR)
  Digital SLR focuses on giving high manual control to the user, and when combined with possibility to change lenses, gives a high degree of flexibility to the user. Entry-

level Digital SLRs usually have several different automatic modes (similar to Consumer cameras) to make them less intimidating for beginners. The high degree of quality control and the array of high-quality optic lenses has a correspondingly huge drawback: cost. A Digital SLR tends to cost a good deal, and a professional level Digital SLR can be 10 to 20 times the cost of a Consumer camera.... before adding the cost of the lens.

Another types of cameras are Wi-Fi cameras. A Wi-Fi digital camera offers users the ability to do things a bit more efficiently. In essence they provide some of the conveniences of a camera phone with the higher quality images and photographic Features of standalone digital cameras.

A Wi-Fi digital camera allows you to:

1- Download and save pictures to your computer wirelessly. No card reader or USB cables needed.

2- Transfer pictures to a compatible printer quickly and without wires or a computer.

3- Wirelessly transfer pictures to photo sharing sites.

4- Remote control of camera functions via computer in some instances.

The most critical thing necessary in order to use Wi-Fi digital camera will be a network connection. A wireless home network or a hotspot should do. Buyers should be sure to read the literature on the camera to understand exactly how that individual device can connect and to which sites it will easily transfer photos wirelessly.

Figure 2.11 WiFi cameras

Chapter Three

System conceptual design

Chapter contents

3.1 Introduction

In this chapter we will describe our system main components and the design concepts. We will illustrate the general block diagram, system main components. Then describe in some details the inner blocks and circuits and how they are related to other components.

3.2 Block Diagram

The system is comprised of the following entities:
sensors and camera circuits, interfacing circuit between sensors and Wi-Fi modems, the wireless Wi-Fi network, the computer for displaying reported measurements, the connection GSM network within hospitals, the database for the realization of the unified blood bank.

The figure 3.1 describes the general block diagram of the system .

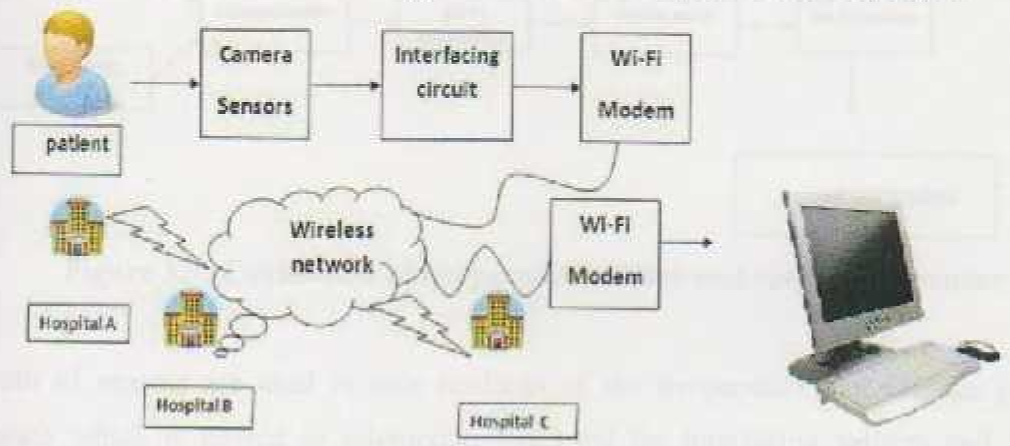


Figure 3.1 : General block diagram

3.2 .1 Camera system :

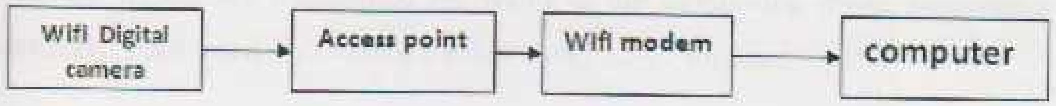


Figure 3.2: The block diagram of Wi-Fi scenario

camera will capture pictures and videos and send them via Wi-Fi transmitter to a Wi-Fi flash that connected to a computer.

3.2 . 2 Sensor system :

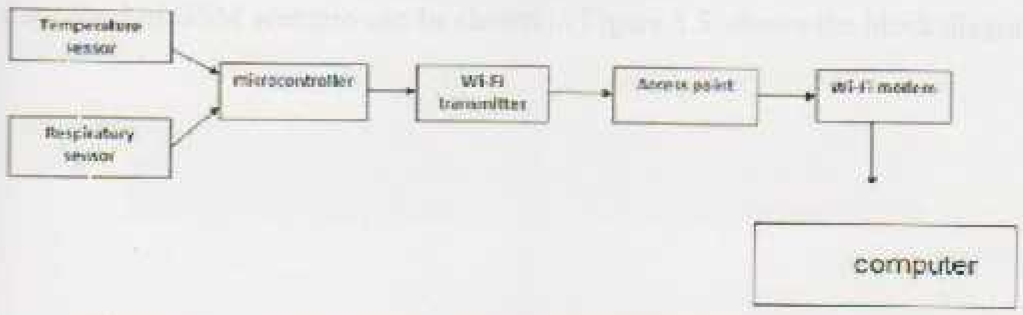


Figure 3.3: Connection of temperature sensor and respiratory sensor

Both of sensors are used to take readings of the temperature and average patient breath which is passed to microcontroller used for interfacing sensors and Wi-Fi transceivers and computer.

3.3 System scenarios

The communication part of the project has two scenarios. The first, explained in Figure 3.4, is inside the hospital. It uses Wi-Fi network which connects camera in the room of the patient that will capture pictures and videos and send them via Wi-Fi transmitter to Wi-Fi access point which passes packets to a Wi-Fi receiver to a computer in the monitoring room. in the monitoring room of the hospital various sensors are attached to the patient's body, these sensors connected a microcontroller which acquires signals from the sensors and performs certain processing on them. Then the signals are transmitted by Wi-Fi to the monitoring room. monitoring is constantly performed by reading signals every 15-25 minutes.

The second scenario, as shown in Figure 3. 6 based on GSM technology which is used to transmit information data from a specific GSM modem located in the hospital A for example, and connected to a microcontroller. The GSM modem communicates with another GSM modem of hospital B. According to a specific programming, an SMS is produced to describe the blood type needed for patients. The transmission is just done if upnormal conditions are measured in one of connected hospitals. The whole system operation can be described by the following flowcharts for each scenario separately. Figure 3.4 shows the flowchart of the Wi-Fi scenario while the flow chart of GSM scenario can be shown in Figure 3.5 shows the block diagram.

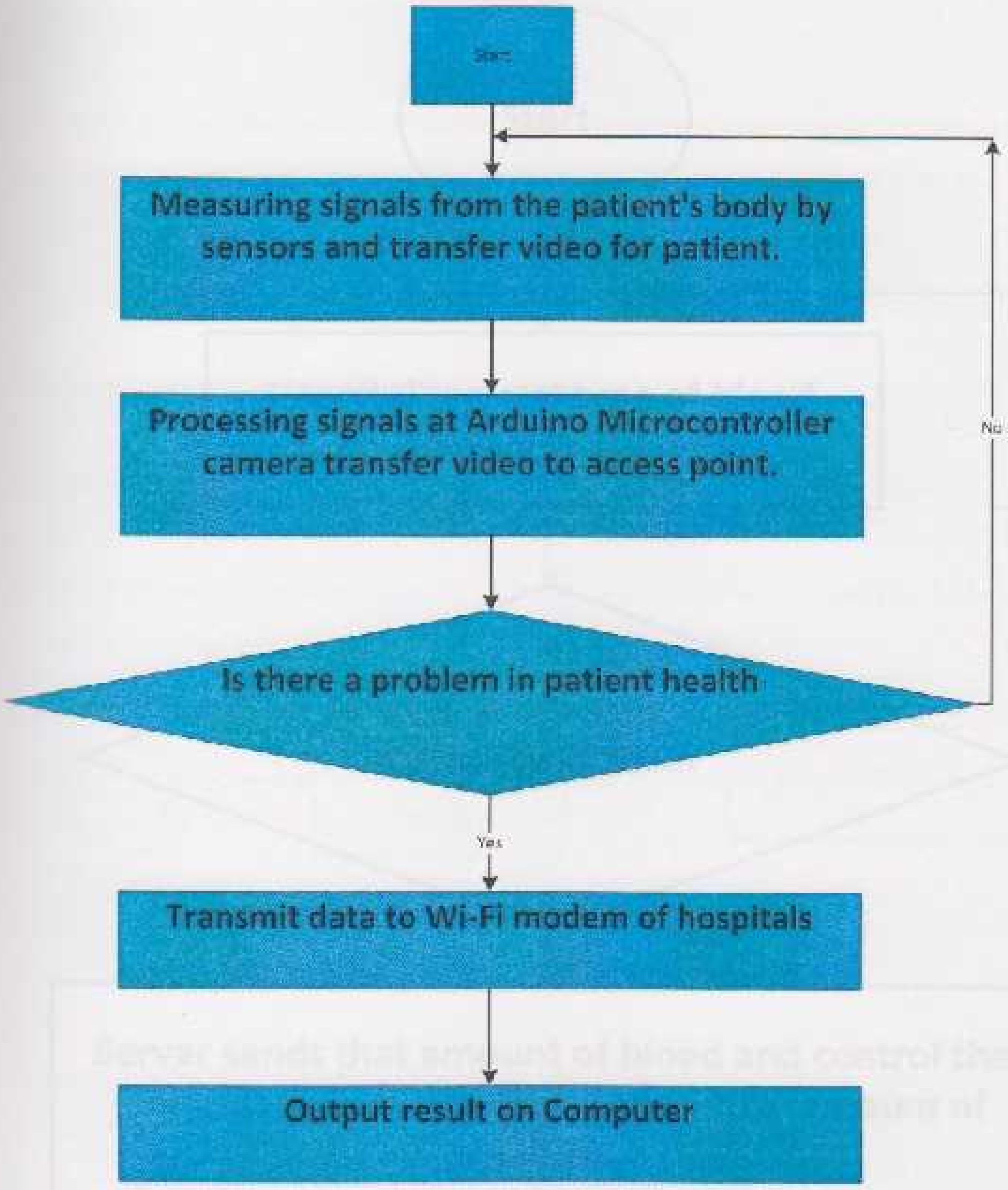

Start
Measuring signals from the patient's body by sensors and transfer video for patient.
Processing signals at Arduino Microcontroller camera transfer video to access point.
Is there a problem in patient health
No
Yes
Transmit data to Wi-Fi modem of hospitals
Output result on Computer

Figure 3.4 :Flow chart of Wi-Fi scenario

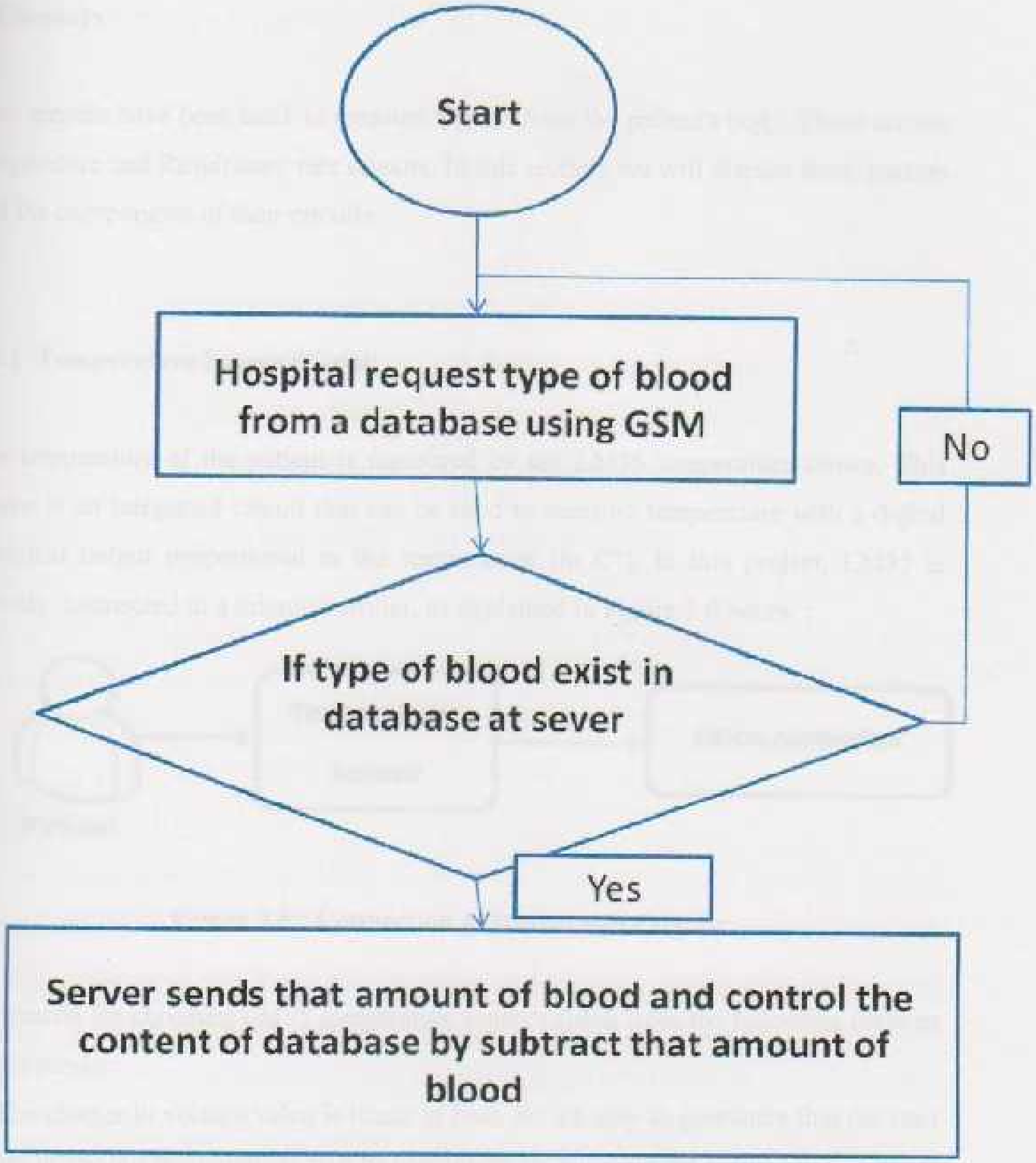

Start
Hospital request type of blood from a database using GSM
No
If type of blood exist in database at sever
Yes
Server sends that amount of blood and control the content of database by subtract that amount of blood

Figure 3.5: The block diagram of GSM scenario for blood bank

3.4 Sensors

Two sensors have been used to measure signals from the patient's body. These are the temperature and Respiratory rate sensors. In this section, we will discuss these sensors and the components of their circuits.

3.4.1 Temperature Sensor Circuit

The temperature of the patient is measured by the LM35 temperature sensor. This sensor is an integrated circuit that can be used to measure temperature with a digital electrical output proportional to the temperature (in C°). In this project, LM35 is directly connected to a microcontroller, as explained in Figure 3.6 below :

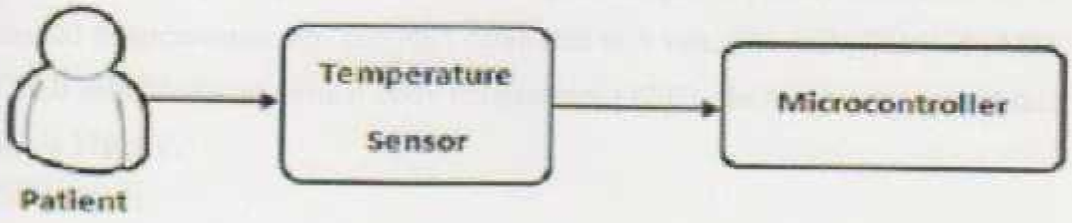


Figure 3.6 : Connection of temperature sensor

The reason for choosing LM35 temperature sensors stems from the following features of this sensor:

1. The change in voltage value is linear to heat. So it's easy to guarantee that the read values at various temperatures will be reasonable by knowing the voltage to heat ratio.
2. Low cost and availability.

Figure 3.7 shows the Components of LM35.

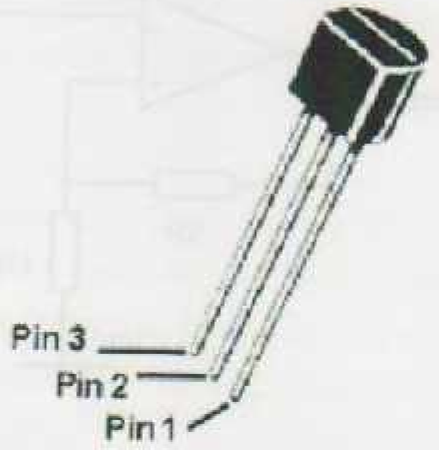


Figure 3.7: Components of LM35

The LM35 has 3 pins: pin1 connected to ground, pin 2 (Vout) connected to be connected to microcontroller, and pin3 connected to 5 volt. The scale factor of LM35 is + 10.0 mV/°C. So, at normal body temperature (37°C), the output voltage of this sensor is 370 mV.

It is necessary to amplify the output signal of the transducer before applying it to the controller. The amplification can be done by using a suitable operational amplifier circuit. Operational amplifiers can be used in two basic configurations to create amplifier circuits. One is the inverting amplifier where the output is the inverse or 180 degrees out of phase with the input, and the other is the non-inverting amplifier where the output is in the same sense or in phase with the input. but in our project we decided to use non-inverting amplifier Because it has good characteristics like input impedance is very high.

The output voltage of LM35 transducer is connected to the input terminal of the non-inverting amplifier circuit shown figure 3.8.

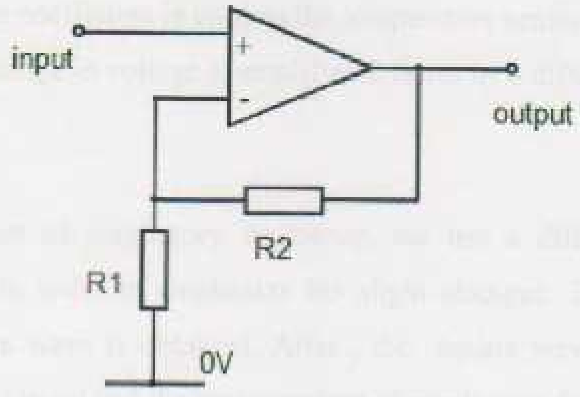


Figure 3.8 Basic non-inverting operational amplifier circuit.

The values of the two resistors R1 & R2 determine the gain of the operational amplifier circuit according to the following equation.

Av = 1 + R2 / R1 ---------- (3.1)

Vout = 5V and Vin = 380mV

Av = Vout / Vin = 5 / 380m = 13 .

So, R2 = 12 KΩ and R1 = 1 kΩ.

3.4.2 : Respiratory sensor circuit

Respiratory ventilation detection

The module for amplification of ambient temperature is a signal conditioner, we can detect the deference between internal and ambient temperature, then complete the measurement of respiratory frequency by this solution as shown in Figure 3.9 :

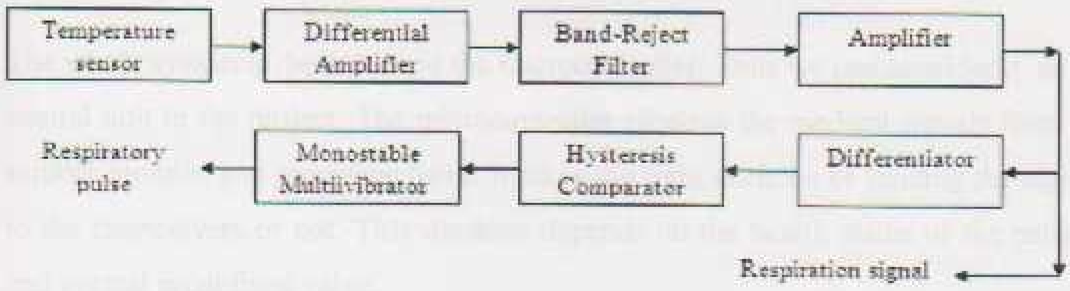


Figure 3.9 Blok diagram of respiration measurement.

Figure 3.9 shows the block diagram of respiratory measurement. A thermistor with negative temperature coefficient is used as the temperature sensor. With a Wheatstone bridge circuit, the change in voltage is amplified 2 times by a differential amplifier.

For the measurement of respiratory frequency, we use a differentiator circuit to differentiate signal in order to emphasize his slight changes. Through a hysteresis comparator, a square wave is obtained. After , the square wave will trigger mono stable multi vibrator circuit and the measurement of respiratory frequency is done.

Figure 3.10 shows Kit of respiratory sensor.

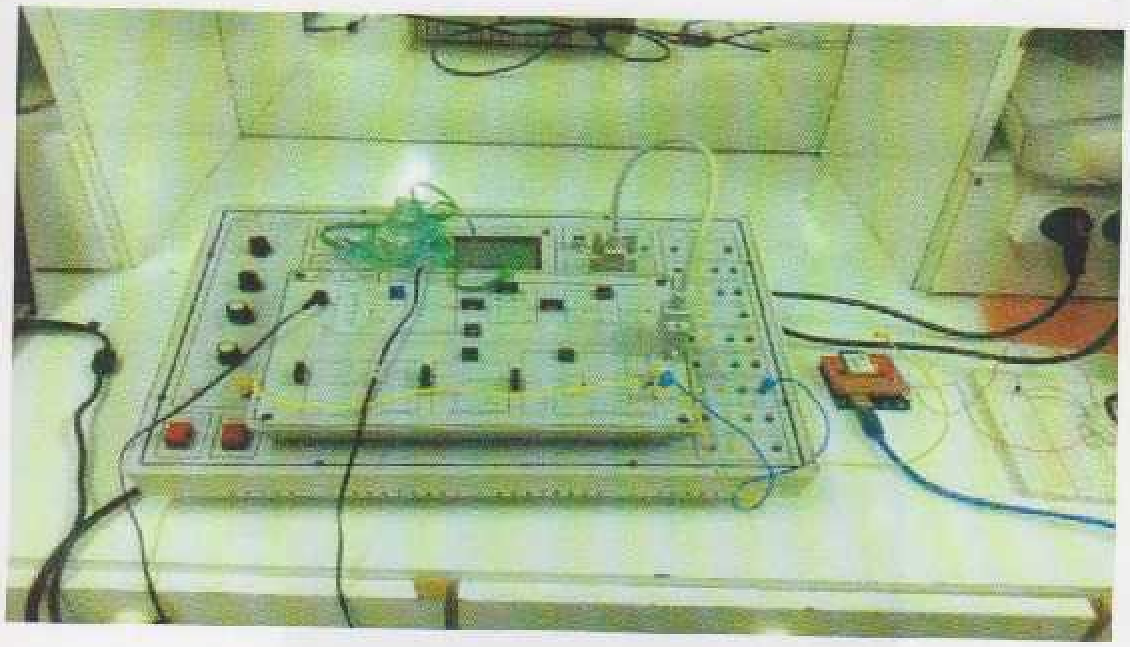


Figure 3.10 Kit of respiration measurement.

3.5 Arduino Microcontroller

The whole system is built around the microcontroller. Thus we can consider it as the central unit in the project. The microcontroller receives the medical signals from the sensors circuits, and processes them. It takes the right decision of sending the signals to the transceivers or not. This decision depends on the health status of the patients and normal predefined value.

We will use two microcontrollers: first is connected to the Wi-Fi transceiver in the first scenario, as shown in Figure 3.11:

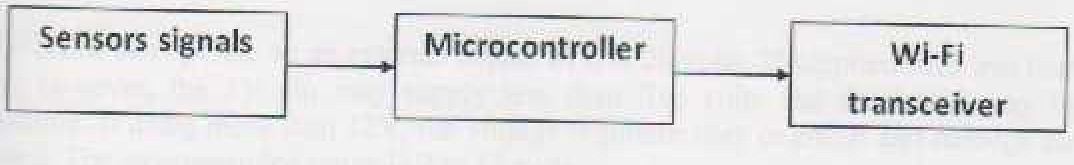


Figure 3.11 : Block Diagram of Microcontroller in Wi-Fi Scenario

Arduino Uno:

it uses 8 bit microcontroller. It is a microcontroller board based on the ATmega328, It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

Shows Figure 3.12 of Arduino Uno:

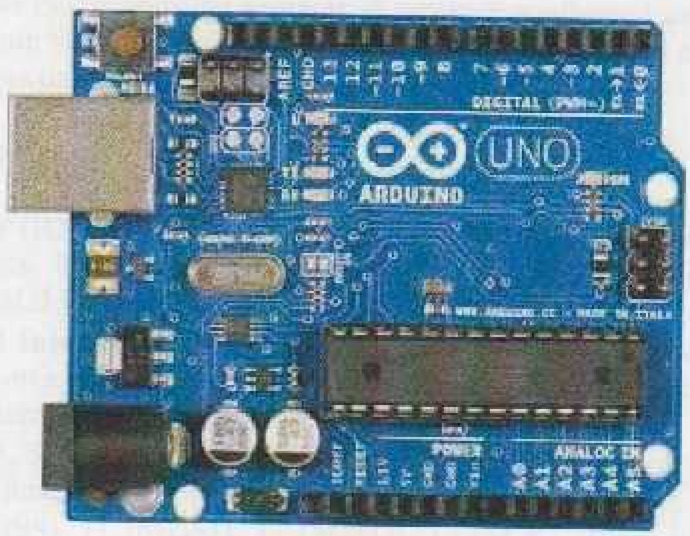


Figure 3.12 : Arduino Uno

Power Pins:

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source),you can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin

- **5V.**This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.

- **3.3V.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.

- **GND.** Ground pins.

- **IOREF.** This pin on the Arduino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs for working with the 5V or 3.3V.

Input and Output pins:

- **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- **External Interrupts: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attach Interrupt ( ) function for details.
- **PWM: 3, 5, 6, 9, 10, and 11.** Provide 8-bit PWM output with the analog Write ( ) function.
- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** These pins support SPI communication using the SPI library.
- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analog Reference ( ) function. Additionally, some pins have specialized functionality:

- **TWI: A4 or SDA pin and A5 or SCL pin.** Support TWI communication using the Wire library.

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with (analog Reference ( )).
- **Reset.** Bring this line LOW to reset the microcontroller, typically used to add a reset button to shields which block the one on the board.

3.6 Wi-Fi transceiver

The wireless transceiver is a device that contains both transmitter and receiver which are combined in a common circuitry. It transmits and receives data wirelessly. In this project, the transceiver is connected to the microcontroller serially. It receives data from the microcontroller and then sends it wirelessly to a PC through an access point, as shown in Figure 3.13 :

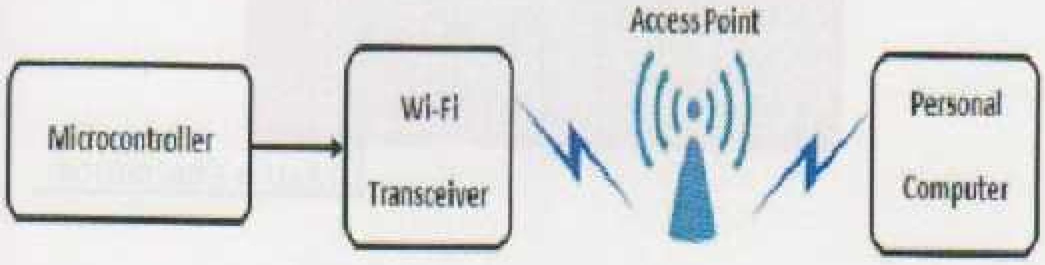


Figure 3.13: Blok Diagram of Wi-Fi Transceiver Connection

The chosen transceiver in this project is WiFly GSX 802.11 b/g Wireless LAN Module,. It enables wireless connections to any embedded serial port and supports bi-directional RS232 signaling at a rate of up to 921Kbps. It is attached with other required components on an integrated board called Wi-Fi shield. This shield is an Arduino shield which enables the Arduino microcontroller to connect to 802.11b/g wireless networks. The featured components of the shield are a Roving Network's RN-131C wireless module and an SC16IS750 SPI-to-UART chip. The SPI-to-UART bridge is used to allow for faster transmission speed and to free up the Arduino's UART. As shown in Figure 3.20, the shield includes the RN-131C, SC16IS750 and their supporting components. It includes a voltage regulator configured to regulate the Arduino's raw voltage to 3.3V. It is also includes all the parts needed to interface the Arduino with the Wireless LAN Module.

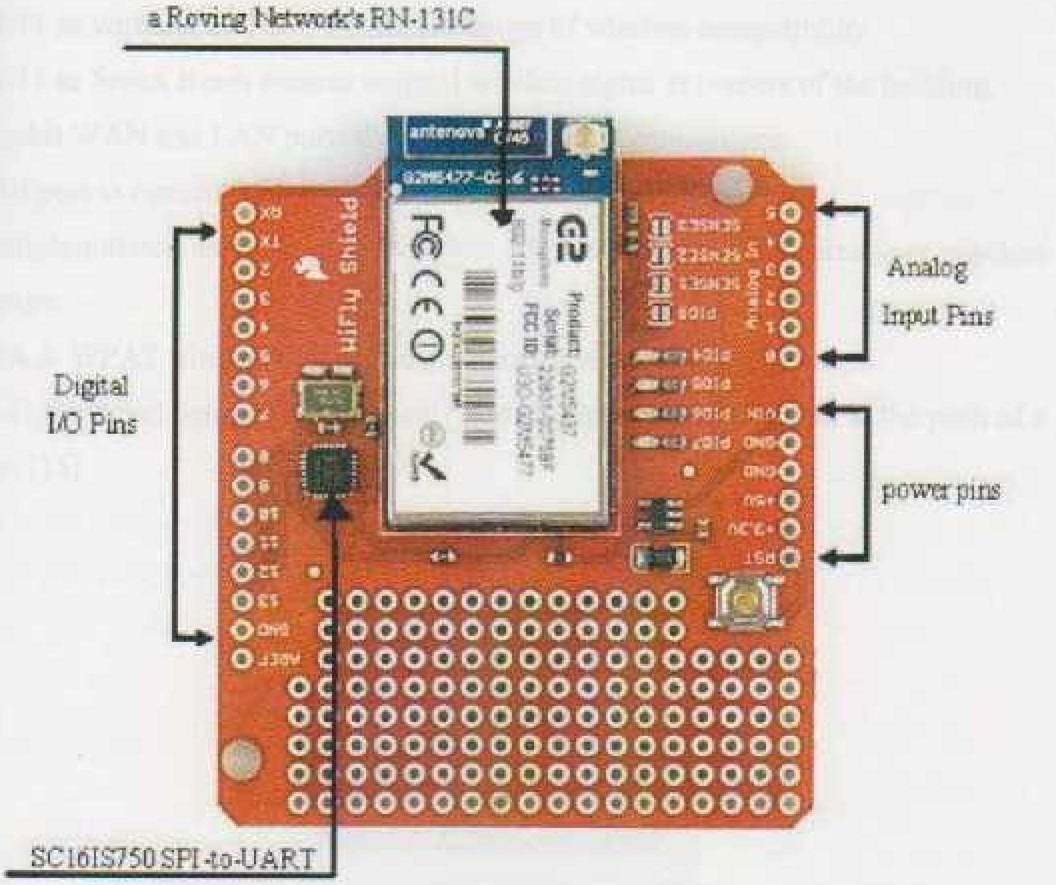


Figure 3.14: Components of WiFly Shield

it is illustrates the schematic diagram of the WiFly shield by showing all of its components. U4 in the figure is the voltage regulator.

Power is taken from the Vin pin of the Arduino, regulated to 3.3V, and provided to both the RN-131C and the SC16IS750. Communication with the WiFly Shield is established over SPI using Arduino digital pins 10-13 (CS, MOSI, MISO, SCLK respectively).[14]

network using the wireless technology and unique mydlink™. By downloading the free mydlink™ app ,it can monitor the network remotely. Plus, with the included SharePort™ Technology, accessing and sharing videos have been so easy.

Features

• 802.11 ac wireless LAN for a complete range of wireless compatibility

• 802.11 ac Smart Beam ensures optimal wireless signal at corners of the building.

• Gigabit WAN and LAN ports for high-speed wired connections.

• USB port to connect storage drives and printers for sharing.

• Multiple internal antennas with Wireless AC technology to optimize your wireless coverage.

• WPA & WPA2 wireless encryption protects the network from intruders.

• Wi-Fi Protected Setup (WPS) securely adds devices to your network at the push of a button.[15]

Figure 3.15 : D Link ac Router

3.8 GSM Modem

A GSM modem is a specialized type of modem which accepts a SIM card, and operates over a subscription to a mobile operator, just like a mobile phone. From the mobile operator perspective, a GSM modem looks just like a mobile phone as shown in figure 3.16

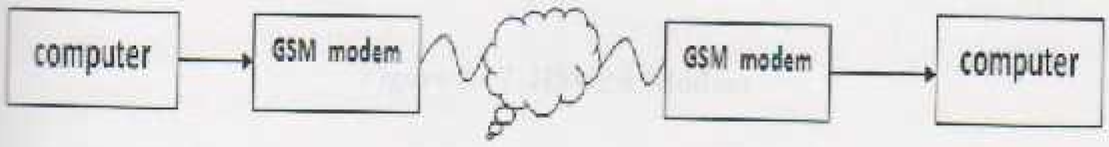


Figure 3.16 GSM block diagram

- For allowing connection of hospitals.

3G modem support or can handle the (Hsupa) standard, and can support high speed or data rate. Also it contain a (USB) interface .figure 3.17 show the Hsupa Modem.

This modem has the following features :

□ USB interface.

□ Based on Sim com module SIM5216E.

□ With TCP/IP stack.

□ WCDMA/HSDPA 900/2100MHz.

□ GPRS class 12.

□ EDGE

□ class12.

Figure 3.17 HSUPA Modem

3.8.1 SIM Card

A Subscriber Identity Module (SIM) card is a portable memory chip used by all GSM devices, including phones and GSM/GPRS modems. These cards hold the personal information of the account holder, including his phone number, address book, text messages, and other data. One of the biggest advantages of SIM cards is that they can easily be removed from one mobile phone and used in any other compatible phone to make a call. This means that, if the user wants to buy a new handset, he can activate it quickly by inserting his old SIM card. The user's phone number and personal information is carried on the card, so there's no need to do anything else to transfer this information.

3.9 Patient – doctor connection

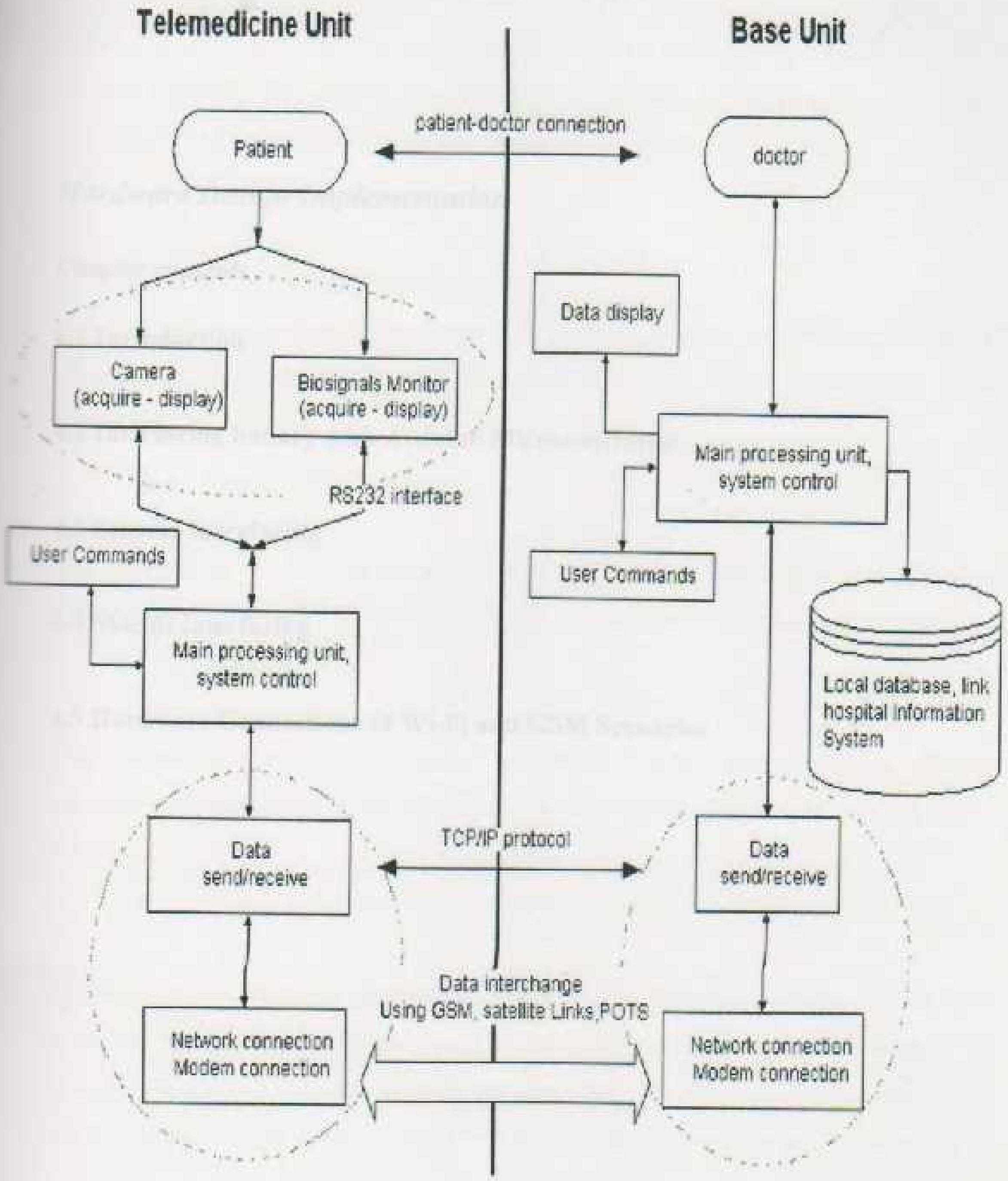

Telemedicine Unit
Base Unit
patient-doctor connection
Patient
doctor
Camera (acquire - display)
Biosignals Monitor (acquire - display)
RS232 interface
User Commands
Main processing unit, system control
Data display
Main processing unit, system control
User Commands
Local database, link hospital information System
TCP/IP protocol
Data send/receive
Data send/receive
Data interchange Using GSM, satellite Links,POTS
Network connection Modem connection
Network connection Modem connection

Figure 3.18 Patient – doctor connection

Chapter Four

Hardware Design Implementation

Chapter contents

4.1 Introduction

After viewing the general system design in the previous chapter, it is now time for presenting the specific details of the connections and interfacing several components in each scenario. This chapter presents hardware details by pointing to each pin and showing how it is combined with other pins in each device.

4.2 Interfacing Arduino Microcontroller

Arduino microcontroller is a central unit in this project, it receives the medical signals from the sensors circuits, processes them and sending the results to the transceivers.

The sensors circuits have been connected to Arduino via digital and analog pins on the Arduino board which has 14 digital input/output pins and 6 analog input pins.

The transceivers have been connected to Arduino via SPI and serial pins. SPI pins are used to connect WiFly shield whereas the Serial pins are used to connect WiFly shield.

As shown in Figure 4.1, a battery has been used to power the Arduino, so we have turned the switch on the Arduino board to the "Batt" position. Then, a 9V battery has been connected to (Vin) and (GND) pins, but there is an internal regulator on the Arduino board to regulate the input power to 5V, because the Arduino operates at 5V.

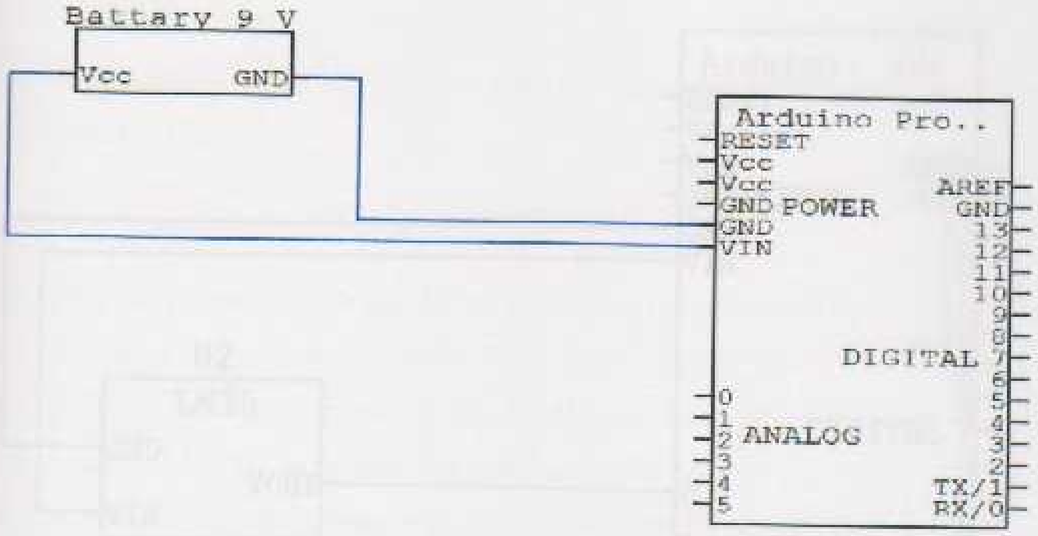


Figure 4.1: Interfacing Arduino Microcontroller with a Battery

4.3 Sensors Interfacing

In this project, we have used two sensors' circuits to measure signals from the patient's body. These are the temperature sensor (LM35), and the respiratory sensor circuit.

LM35 has three pins, Vin (input voltage pin), GND (ground pin), and Vout (output voltage pin).

Connecting the LM35 to the Arduino is done through the serial data line from the output voltage pin (Vout) to the ANALOG IN 0 pin on the Arduino board.
LM35 operates at 5V. So as to power the LM35, the input voltage pin (Vin) in LM35 is connected to the voltage pin (Vcc) in Arduino which in turn provides the temperature sensor with the right amount of power which equals 5 volt, as show figure 4.2.

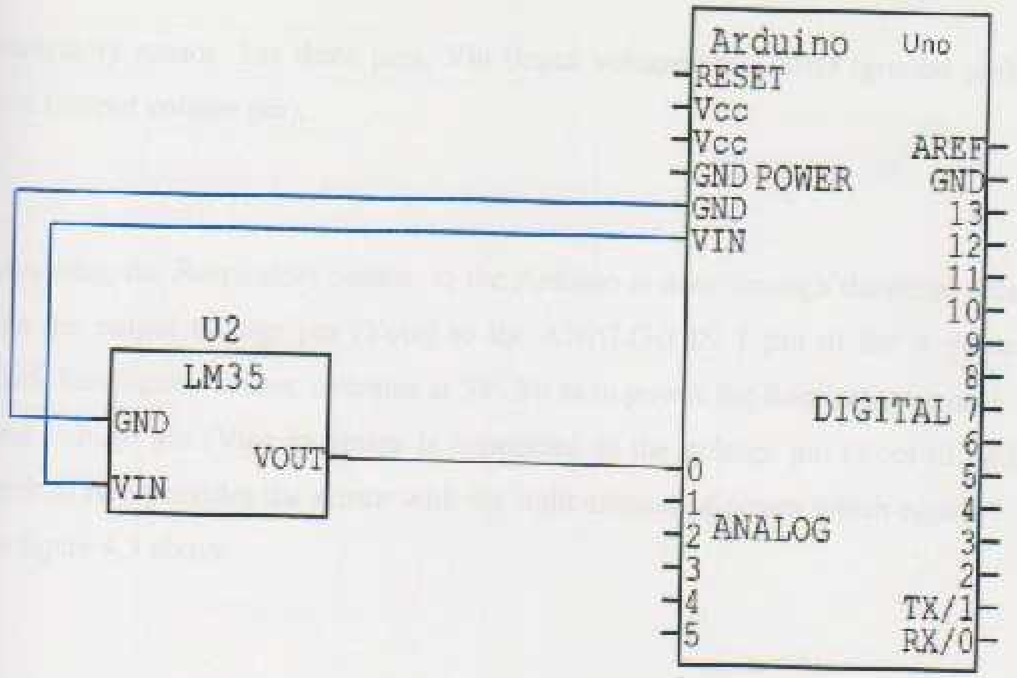


Figure 4.2: LM35 Interfacing

Respiratory sensor interfacing can be achieved by connecting the output pin that provides the analog signal of his heart beats to one of arduino analog input pins 1-5 pins as shown in the figure below :

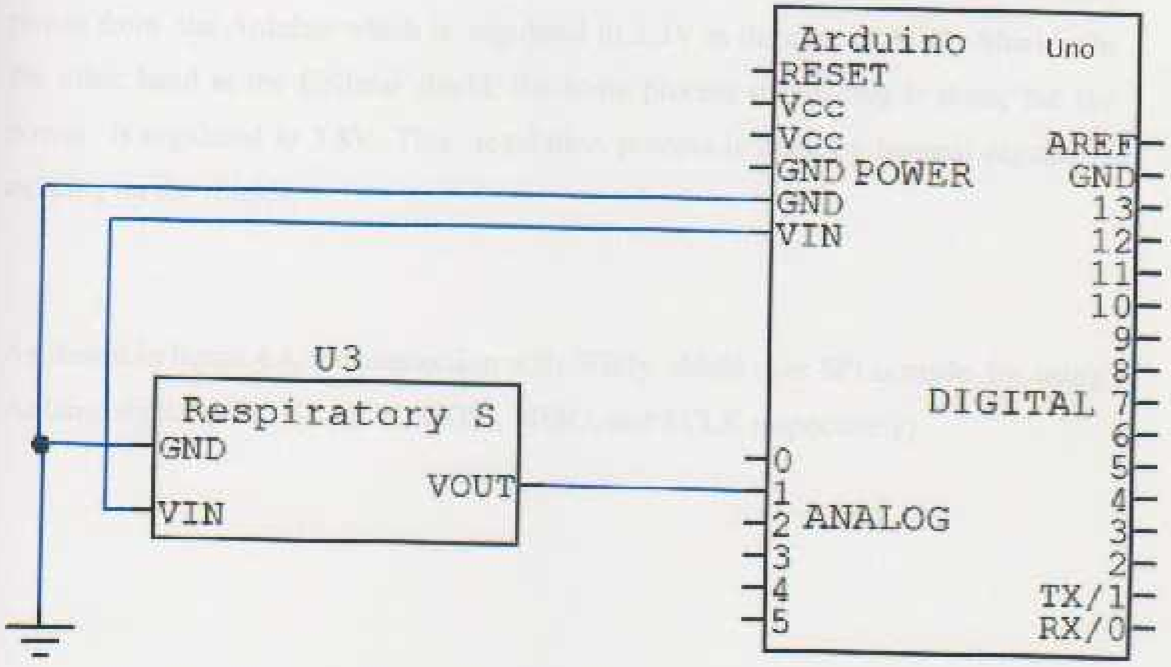


Figure 4.3: Respiratory sensor Interfacing

Respiratory sensor has three pins, Vin (input voltage pin), GND (ground pin), and Vout (output voltage pin).

Connecting the Respiratory sensor to the Arduino is done through the serial data line from the output voltage pin (Vout) to the ANALOG IN 1 pin of the Arduino Pro board. Respiratory sensor operates at 5V. So as to power the Respiratory sensor , the input voltage pin (Vin) in sensor is connected to the voltage pin (Vcc) in Arduino which in turn provides the sensor with the right amount of power which equals 5 volt. See figure 4.3 above.

4.4 Shields Interfacing

Shield can be plugged on the top of Arduino, and then welding can be done on the pins of the shield that make the connection. In this project, two shields have been used. One of them is WiFly shield and the other is Cellular shield.

Welding (Vin) and (GND) pins of Arduino with those of WiFly shield to take power from the Arduino which is regulated to 3.3V in the case of WiFly Shield. On the other hand at the Cellular shield, the same process of welding is done, but the power is regulated to 3.8V. This regulation process is done by internal regulators existing on the shields.

As shown in figure 4.4, the connection with WiFly shield over SPI is made by using Arduino digital pins 10-13 (CS, MOSI, MISO, and SCLK respectively).

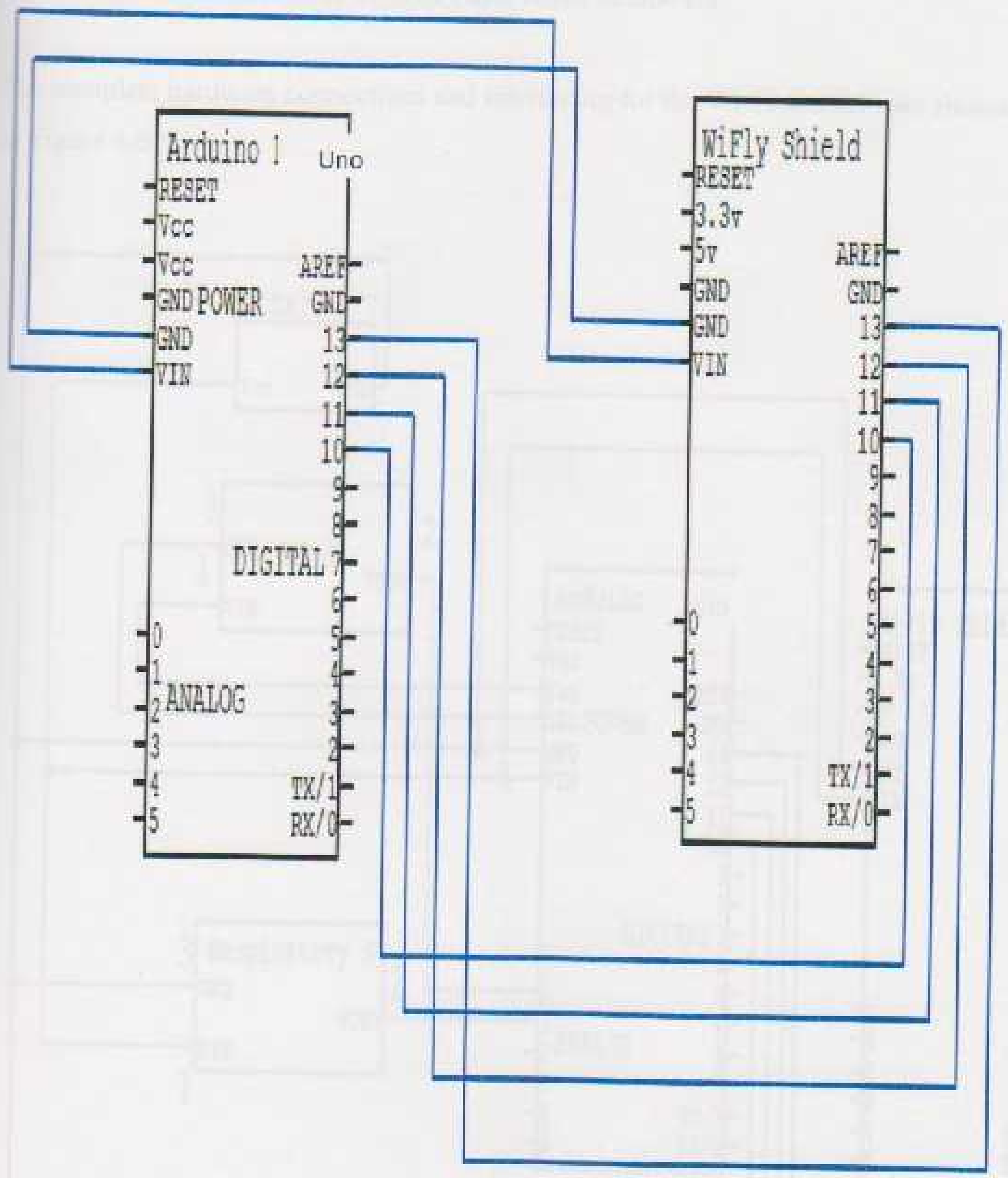


Figure 4.4: Interfacing WiFly Shield

4.5 Hardware Connections of Wi-Fi and GSM Scenarios

The complete hardware connections and interfacing for the Wi-Fi scenario are shown in Figure 4.6.

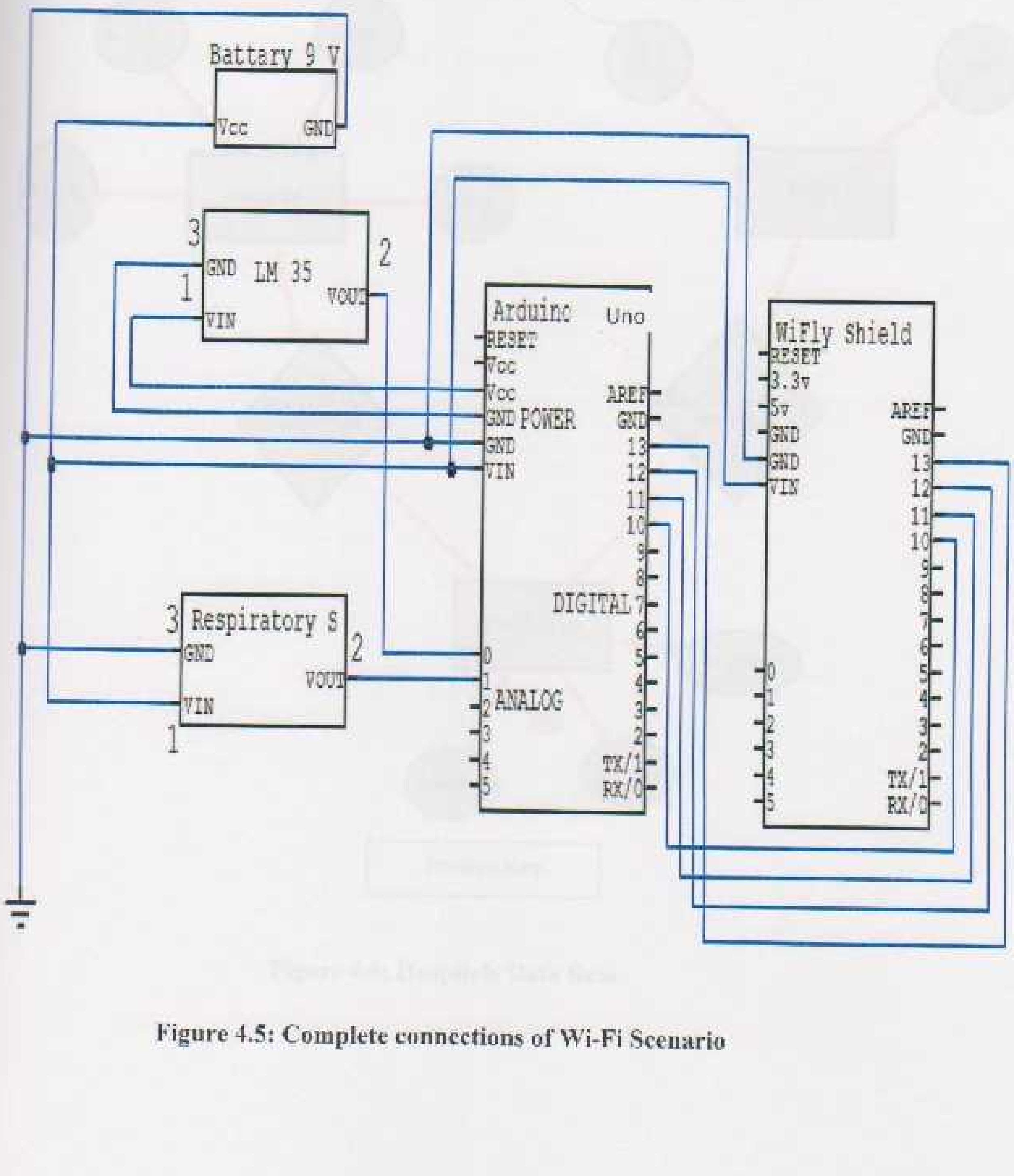


Figure 4.5: Complete connections of Wi-Fi Scenario

4.6 Data Base

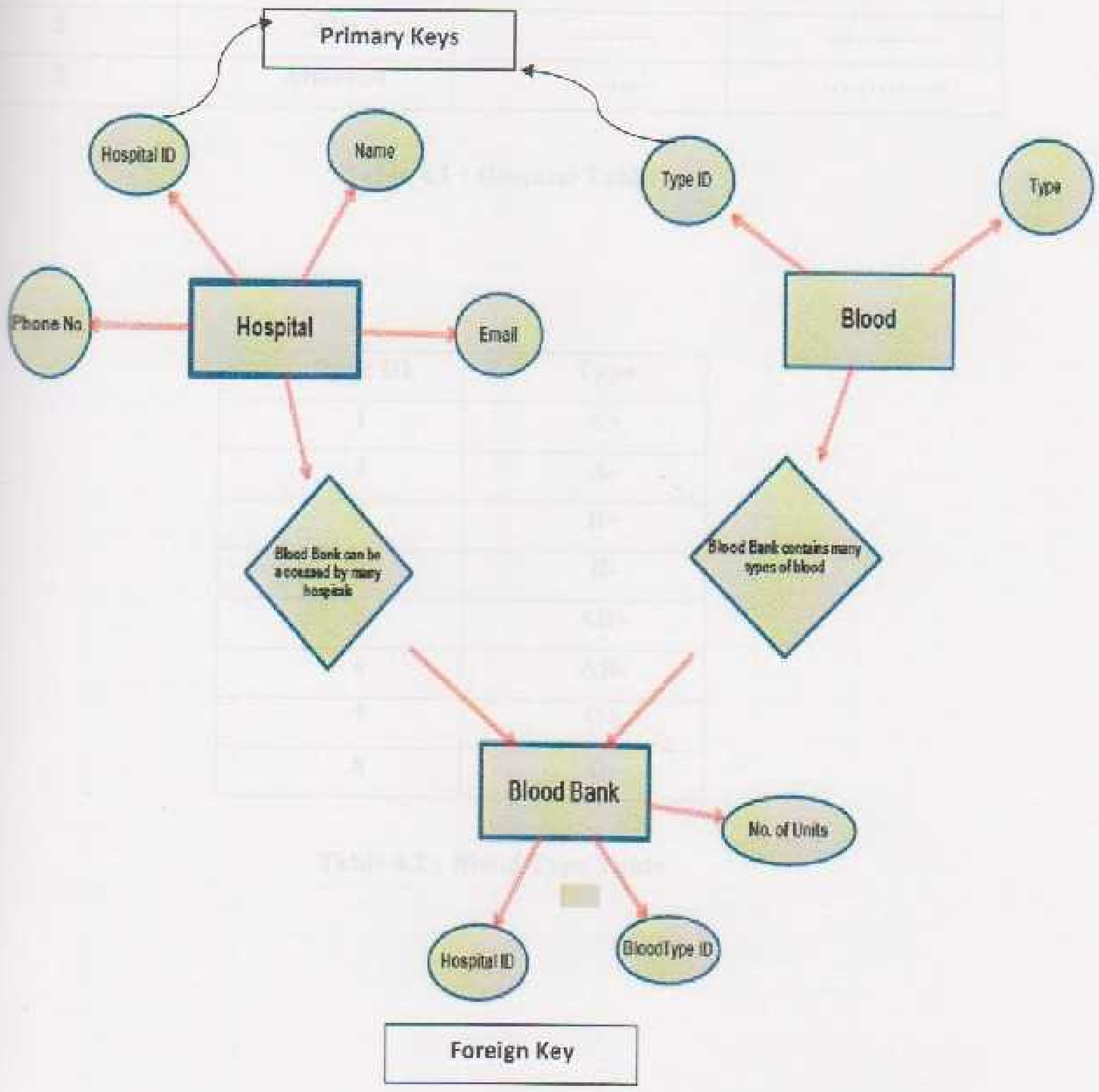


Figure 4.6: Hospitals Data Base

Hospital ID	Name	Phone No.	Email
1	Alahli	....	....
2	Alia	....	....
3	Almezan	....	....

Table 4.1 : Hospital Table

Type ID	Type
1	A+
2	A-
3	B+
4	B-
5	AB+
6	AB-
7	O+
8	O-

Table 4.2 : Blood Type Table

Hospital ID	Type ID	No. of Units
1	1	100
1	2	200
1	3	300
1	4	400
1	5	500
1	6	100
1	7	200
1	8	300
2	1	100
2	2	200
2	3	300
2	4	400
2	5	500
2	6	100
2	7	200
2	8	300
3	1	100
3	2	200
3	3	300
3	4	400
3	5	500
3	6	100
3	7	200
3	8	300

Table 4.3 : Blood Bank Table

Chapter Five

Software Design Implementation

Chapter contents

5.1 Introduction

Software is an important part in any technological system. Operating and controlling any component in this project requires software handling.

This chapter explains the required software with detailed steps and procedures of programming every used component in this project.

It should be noted that most of the designed system has been programmed using the Arduino software which has controlled the microcontrollers and the transceivers.

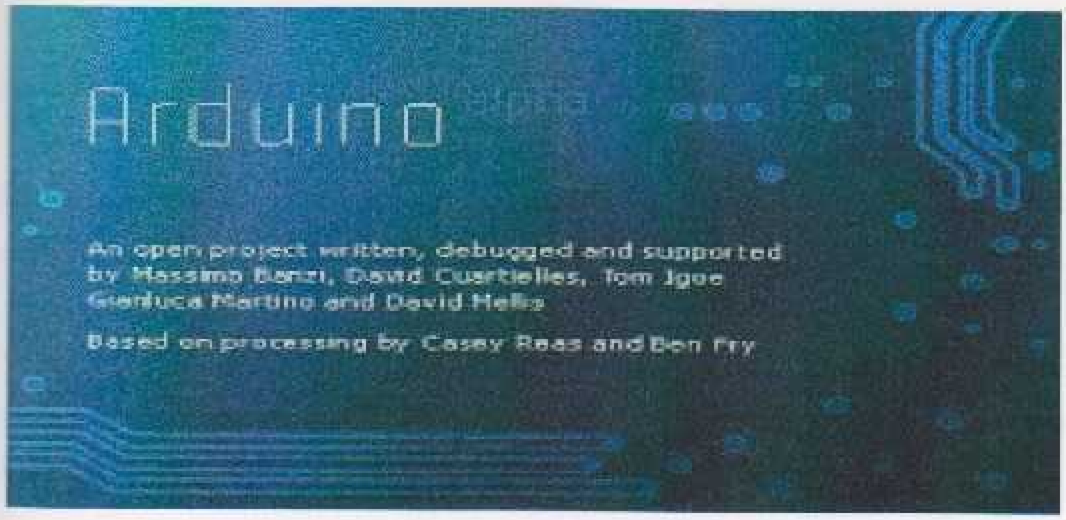
5.2 Arduino Programming

Arduino software is the IDE processor, but can send the Arduino program (or "sketch") to the micro controller in the same board. The Arduino IDE is a cross-platform application written in Java, and is derived from the IDE for the Processing programming language and the Wiring project. It is designed to introduce programming to artists and other newcomers unfamiliar with software development. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and is also capable of compiling and uploading programs to the board. As the Arduino system is open source, anyone can purchase a blank micro controller and put the boot loader on it, or even write their own boot loader.

The Arduino IDE comes with a C/C++ library called "Wiring" (from the project of the same name), which makes many common input/output operations much easier. Arduino programs are written in C/C++, although users only need define two functions to make a runnable program:
Setup ( ) – a function run once at the start of a program that can initialize settings.
Loop ( ) – a function called repeatedly until the board powers off.

Currently, there are 22 IDE versions of software Arduino program, in this project version 21 has been used.

Figure 5.1 shows the interface of Arduino program.



An open project written, debugged and supported by Massimo Banzi, David Cuartielles, Tom Igoe Gianluca Martino and David Mellis

Based on processing by Casey Reas and Ben Fry

Figure 5.1: Arduino Program Interface

Arduino Programming Steps :

- First Step: Selecting Arduino Board

Selecting the entry in the Tools >> Board >> menu that corresponds to the chosen Arduino. We choose the Arduino Uno, as shown in Figure 5.2.

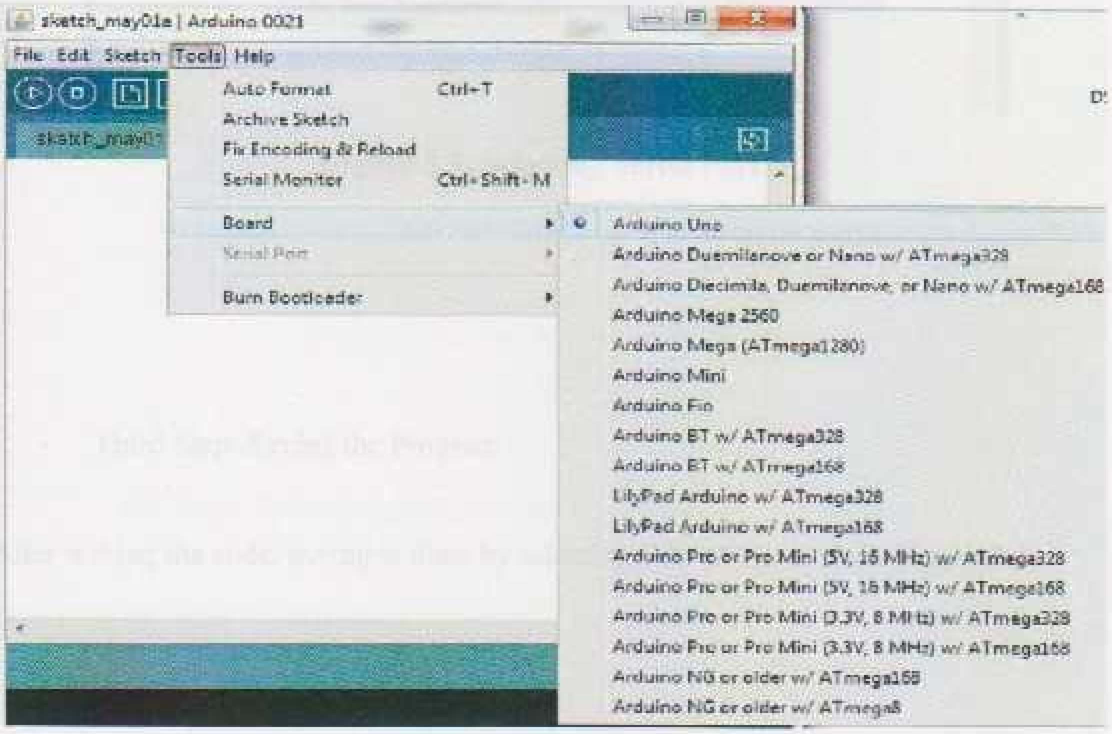

sketch_may01a | Arduino 0021
File Edit Sketch Tools Help
sketch_may01
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Board
Serial Port
Burn Bootloader
● Arduino Uno
Arduino Duemilanove or Nano w/ ATmega328
Arduino Diecimila, Duemilanove, or Nano w/ ATmega168
Arduino Mega 2560
Arduino Mega (ATmega1280)
Arduino Mini
Arduino Fio
Arduino BT w/ ATmega328
Arduino BT w/ ATmega168
LilyPad Arduino w/ ATmega328
LilyPad Arduino w/ ATmega168
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega168
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega168
Arduino NG or older w/ ATmega168
Arduino NG or older w/ ATmega8

Figure 5.2: Select Arduino Uno Board

- Second Step: Selecting the Serial Port

The board communicates with the computer via a USB interface, but with a serial protocol. All this means is that we must select the correct serial port number.

Selecting the serial device of the Arduino board is from Tools >> Serial Port menu. As shown in Figure 5.3, COM8 serial port is selected in this project.

To find out, Arduino board can be disconnected and the menu should be re-opened; the entry that disappears should be the Arduino board. Reconnect the board and select that serial port.

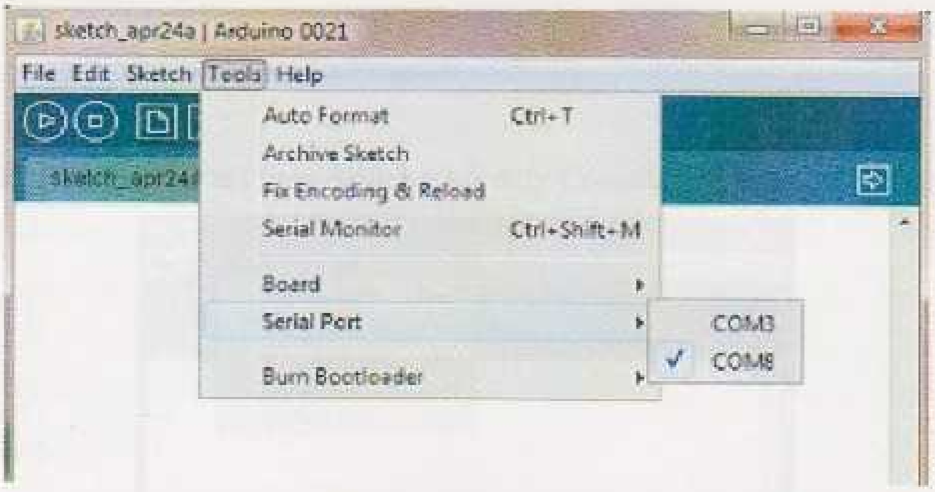


Figure 5.3: Selecting Serial Port

- Third Step: Saving the Program

After writing the code, saving is done by selecting File >> Save, show Figure 5.4.

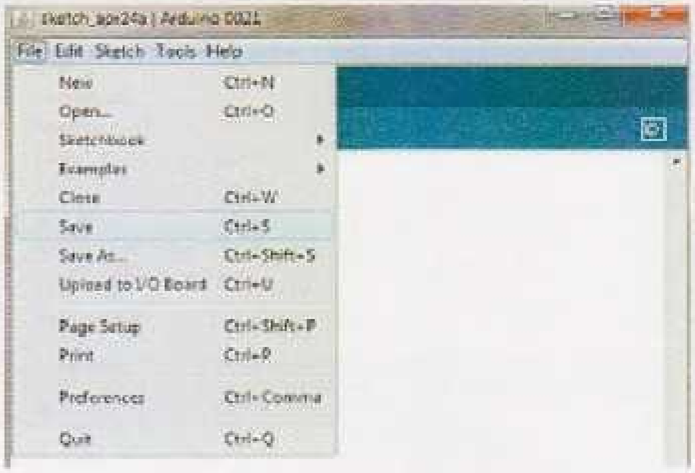


Figure 5.4 Saving the Program

- Forth Step: Compiling the Program

Compiling can be selected from Sketch >> Verify/Compile, as shown in Figure 5.5.

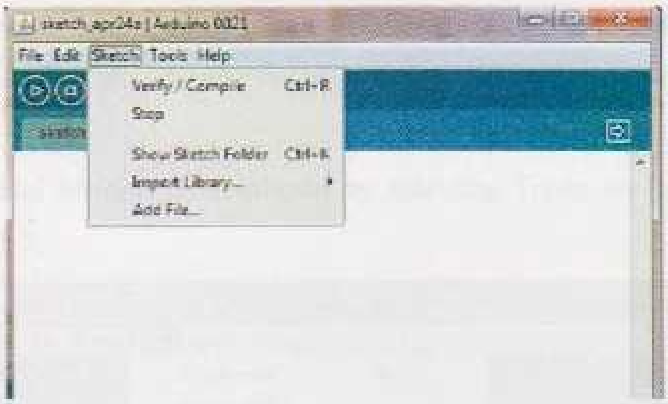

sketch_apr24a | Arduino 0021
File Edit Sketch Tools Help
Verify / Compile Ctrl+R
Stop
Show Sketch Folder Ctrl+K
Import Library...
Add File...

Figure 5.5 Compiling the Program

- Fifth Step: Uploading the Program

Uploading is done by selecting File >> Upload to I/O Board, as shown in Figure 5.6. After waiting a few seconds; we can see the RX and TX leds on the board flashing. If the upload is successful, the message "Done uploading." will appear in the status bar.

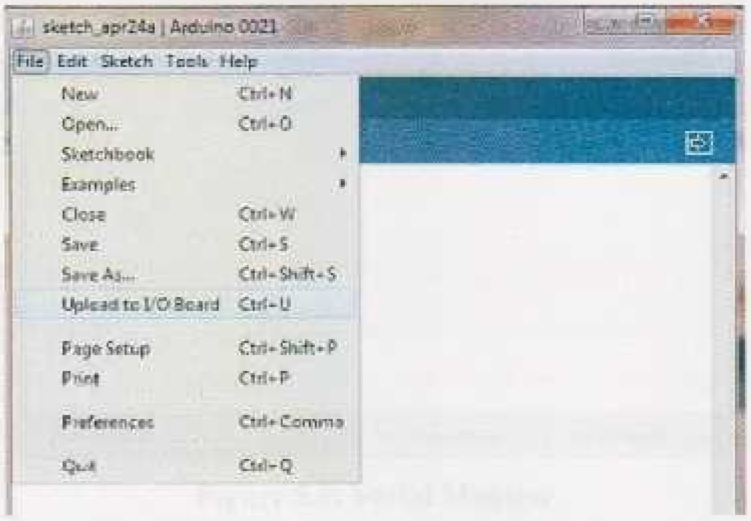


Figure 5.6: Uploading the Program

- Sixth Step: Displaying Result on Serial Monitor

Serial Monitor displays serial data being sent from the Arduino board (USB or serial board). It is also used to send data to the board.

Result on the Serial Monitor is displayed by selecting Tools >> Serial Monitor, as shown in Figure 5.7.

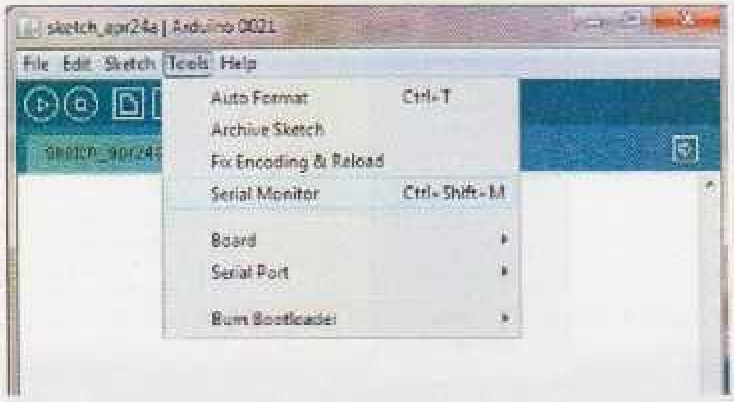


Figure 5.7 Displaying Serial Monitor

Figure 5.8 shows the serial monitor on which the results are displayed to make sure that the code works properly:

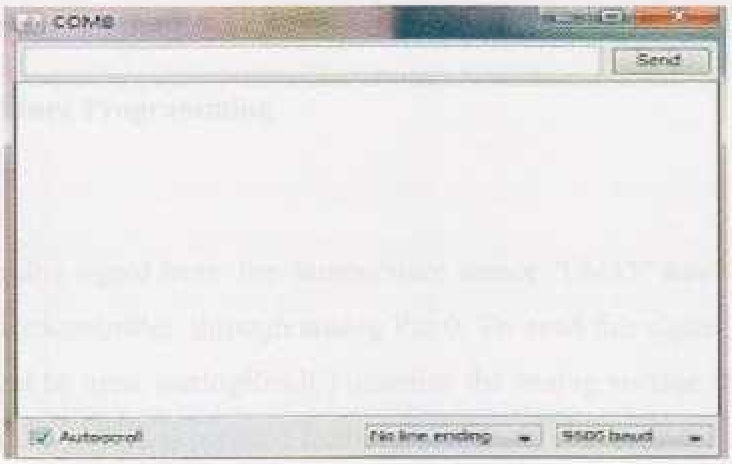


Figure 5.8: Serial Monitor

Arduino Libraries

Libraries are files written in C or C++ (.c, .cpp) which provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, it can be selected from Sketch >> Import Library, as shown in Figure 5.9.

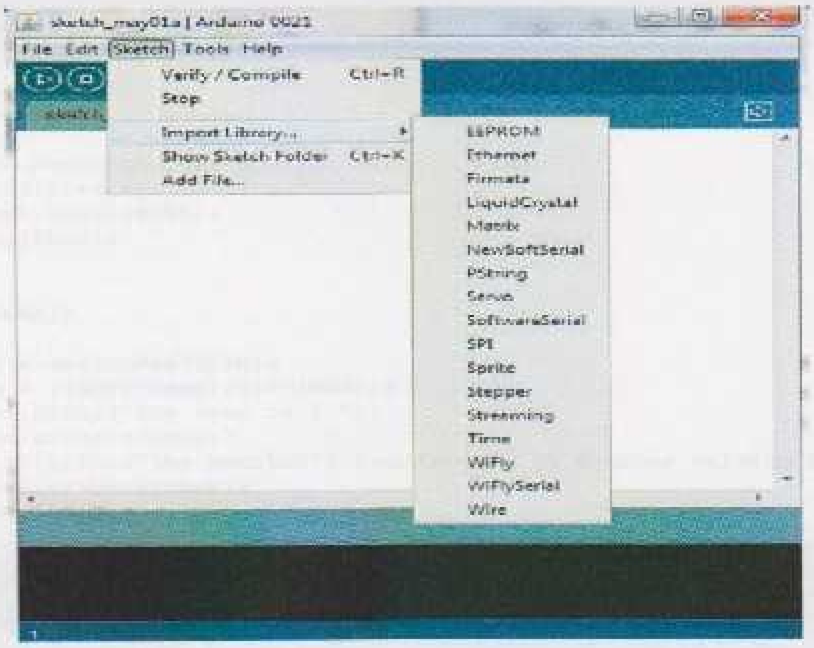


Figure 5.9: Adding Library

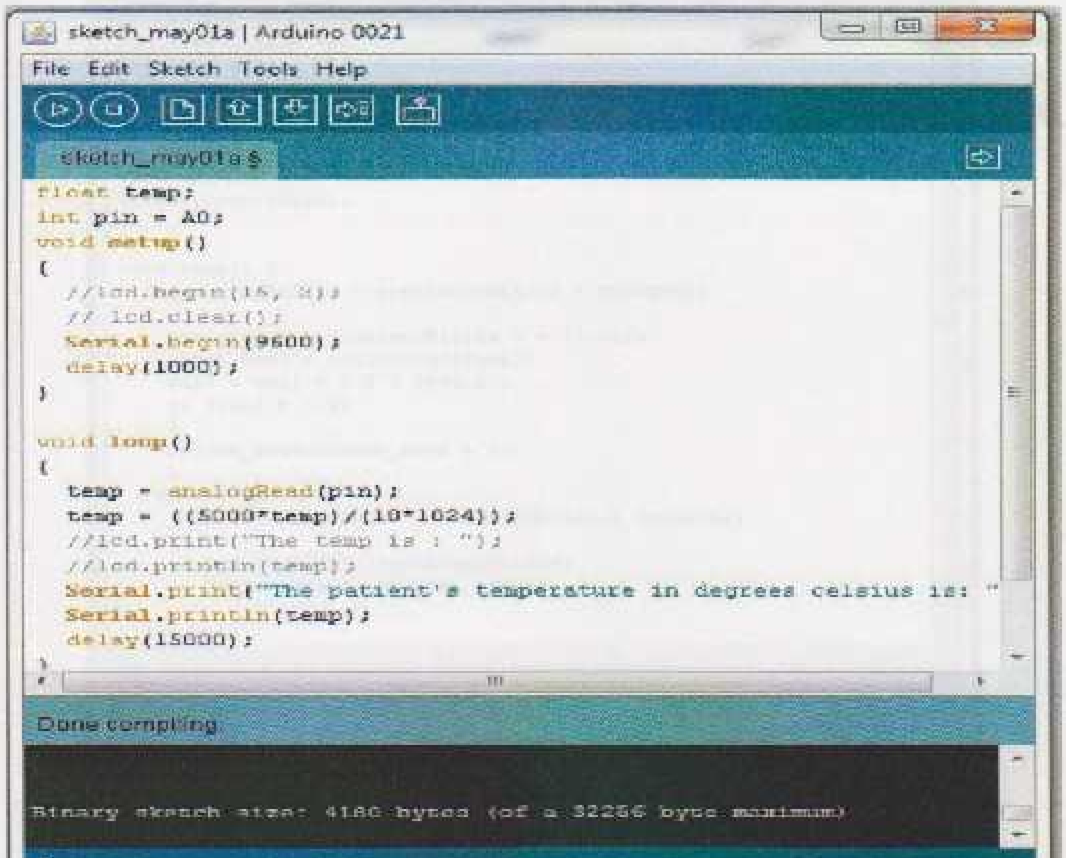
If the library does not exist, it must be installed first. To do so, the library should be downloaded.

5.3 Sensors Programming:

5.3.1 Temperature Programming

The resulted analog signal from the temperature sensor "LM35" has been entered to the Arduino microcontroller through analog Pin 0. To read this signal, analogRead( ) command should be used. analogRead( ) quantize the analog voltage value between 0 and 5V (5000 mV) -which is resulted from the sensor- to convert it to a value between 0 and 1023.

The returned value from the analogRead( ) command must be multiplied by (5000/1024) to return the actual voltage value in millivolts. Then it must be divided by 10 to have a result in degrees Celsius, that is because the output of LM35 sensor is 10 mV per degree Celsius. See Figure 5.10.



```
float temp;
int pin = A0;
void setup()
{
  //lcd.begin(16, 2);
  // lcd.clear();
  Serial.begin(9600);
  delay(1000);
}

void loop()
{
  temp = analogRead(pin);
  temp = ((5000*temp)/(10*1024));
  //lcd.print("The temp is : ");
  //lcd.println(temp);
  Serial.print("The patient's temperature in degrees celsius is: "
  Serial.println(temp);
  delay(15000);
}
```

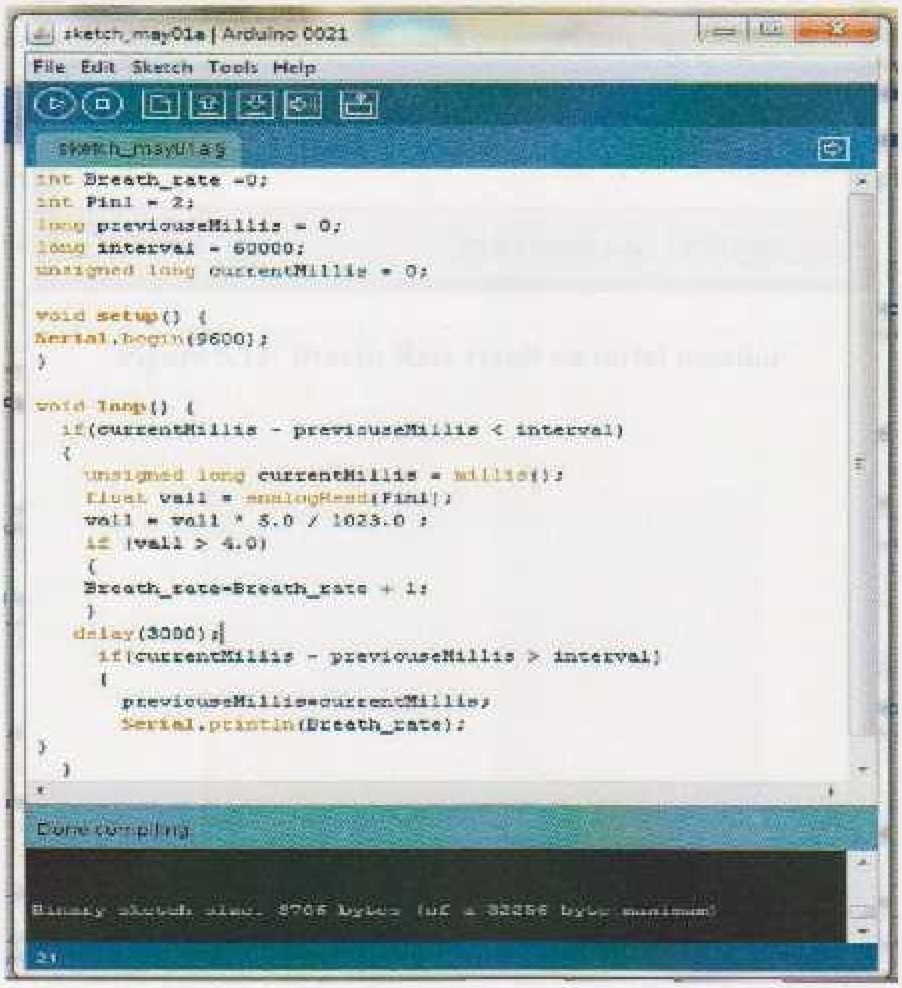
```
Binary sketch size: 4186 bytes (of a 32256 byte maximum)
```

Figure 5.10: Temperature Code

5.3.2 Respiratory Sensor Programming

After getting the analog signal of Respiratory Sensor , it has been processed and analyzed by sending it to Arduino microcontroller through analog pin 1. analogRead( ) command has been used to read the obtained signal from the Arduino microcontroller.

To measure the breath rate, we take values that is larger than 2V and less than 8V; Since the wanted value is the peak of which has a maximum of 8V, and the unwanted value is below 2V, therefore we should take a value that moderate between the upper and the lower values which has been chosen to be 4V (threshold) .



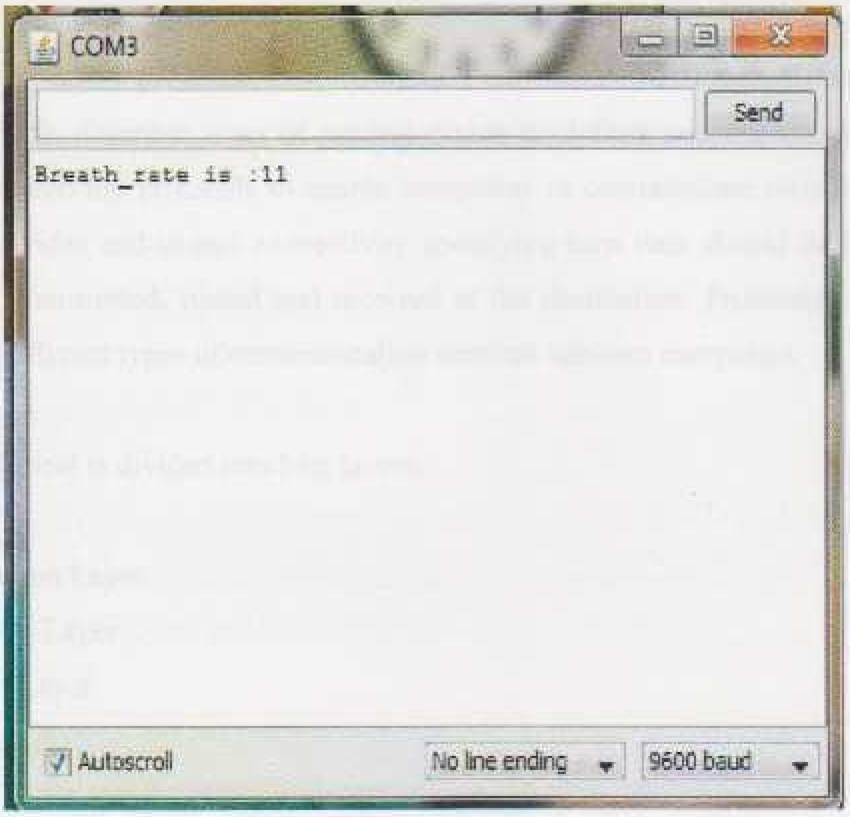
```
int Breath_rate =0;
int Pin1 = 2;
long previouseMillis = 0;
long interval = 60000;
unsigned long currentMillis = 0;

void setup() {
Serial.begin(9600);
}

void loop() {
  if(currentMillis - previouseMillis < interval)
  {
    unsigned long currentMillis = millis();
    float val1 = analogRead(Pin1);
    val1 = val1 * 5.0 / 1023.0 ;
    if (val1 > 4.0)
    {
    Breath_rate=Breath_rate + 1;
    }
   delay(3000);
     if(currentMillis - previouseMillis > interval)
     {
       previouseMillis=currentMillis;
       Serial.println(Breath_rate);
}
  }
```

```
Binary sketch size: 8706 bytes (of a 32256 byte maximum)
```

21

Figure 5.11 Code of Calculating the Breath Rate from Respiration Signal

After that a counter has been built to count the number of breathes during 60 seconds to produce the breath rate and then we got the result. See Figure 5.12.



```
Breath_rate is :11
```

Figure 5.12: Breath Rate result on serial monitor

5.4 WiFly Shield Programming

WiFly shield has been programmed to make a TCP connection with a monitor PC by using an access point as an intermediary between them.

The TCP/IP model (Transmission Control Protocol/Internet Protocol), or Internet Protocol Suite, describes a set of general design guidelines and implementations of specific networking protocols to enable computers to communicate over a network. TCP/IP provides end-to-end connectivity specifying how data should be formatted, addressed, transmitted, routed and received at the destination. Protocols exist for a variety of different types of communication services between computers.

TCP/IP protocol is divided into four layers:

- Application Layer
- Transport Layer
- Internet Layer
- Link layer

Each layer of this model corresponds to a single or multiple layers of the OSI model, where each layer of the TCP/IP is responsible for the activities of the corresponding layer in the OSI model, as shown in Figure 5.16.

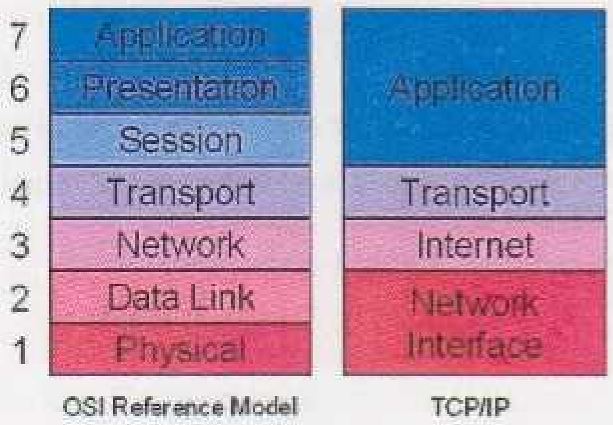


Figure 5.13: Layers of TCP/IP Model Compared with OSI Model

IP protocol is responsible for the transfer of data packets from one computer to another, it sends each packet based on destination address.

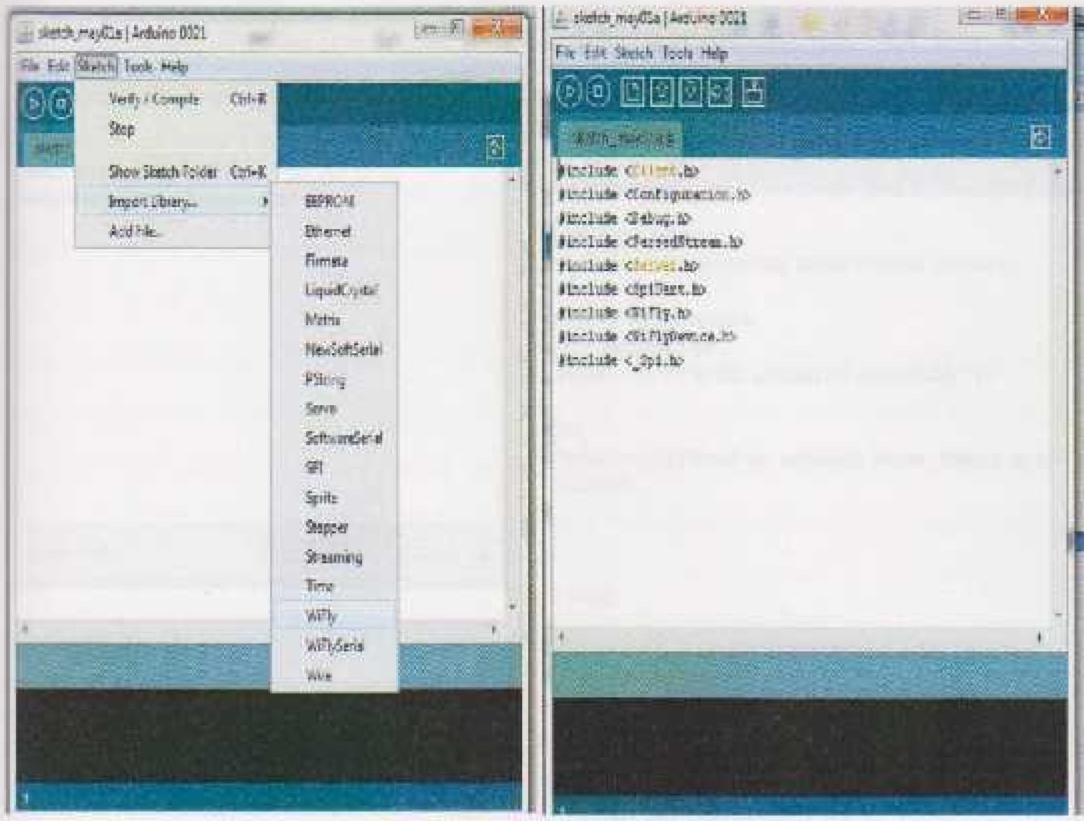
The TCP protocol is responsible for checking the validity of data transfer from a computer to the server computer because of the potential loss of data during transmission.
The TCP perform the checking by revealing the mistakes, and identifying the data lost and then re-transmitting until the arrival of a full data correctly to the final destination.

Steps for Programming WiFly Shield

The WiFly GSX Serial Module from Roving Networks has a very simple serial command interface. To access most of the features of the device, all one needs to connect is TX, RX, VCC, and ground.

The WiFly shield does not actually communicate with the uno using the UART, but has a SC16IS750 SPI-UART bridge that is more complicated to deal with than simple UART communication, but allows for a much higher baud rate if one wants to communicate with the WiFly at higher-than-115200 baud speeds.

The first step of programming is adding the WiFly library in Arduino program by selecting sketch >> import library >>WiFly, as shown in Figure 5.17.



```
#include <Client.h>
#include <Configuration.h>
#include <Debug.h>
#include <ParsedStream.h>
#include <Server.h>
#include <SpiUart.h>
#include <WiFly.h>
#include <WiFlyDevice.h>
#include <_Spi.h>
```

Figure 5.14: Adding WiFly Library

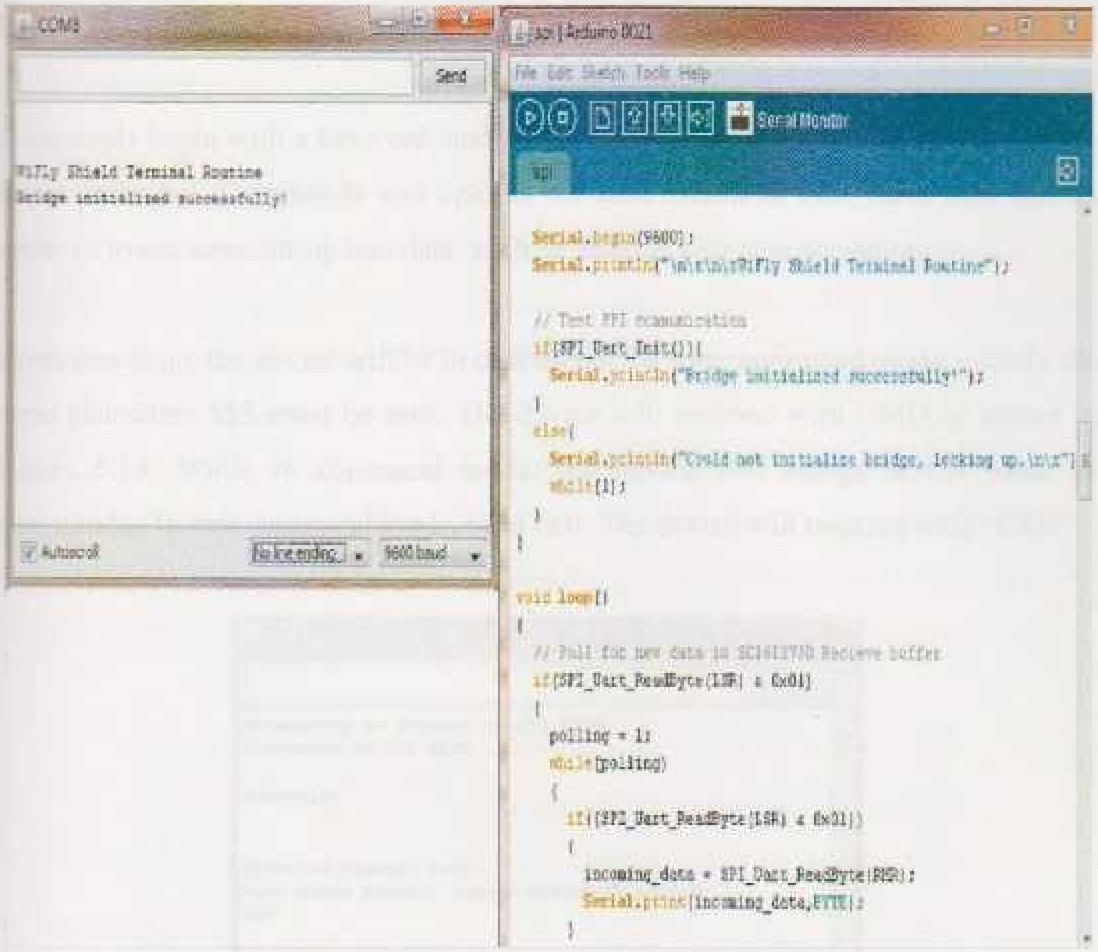
The second step is to initialize the SC16IS750 SPI-UART bridge chip in order to transparently communicate with the WiFly through a terminal. We initialized the SC16IS750 SPI-UART bridge chip by writing a specific code in Arduino software. See Figure 5.18.

This code configures the SC16IS750 to 9600 baud (the default rate of the WiFly), enables the communication FIFO, and then tests the SPI communication by writing to and then reading from the chip's "Scratch-Pad Register" (SPR).

This code is used for a WiFly shield with a default crystal frequency which is 14MHz, but our WiFly shield works on 12MHz, so we had to change the function:

struct SPI_UART_cfg SPI_Uart_config = {0x50,0x00,0x03,0x10};

to struct SPI_UART_cfg SPI_Uart_config = {0x60,0x00,0x03,0x10};



```
WiFly Shield Terminal Routine
Bridge initialized successfully!
```

```
  Serial.begin(9600);
  Serial.println("\n\r\n\rWiFly Shield Terminal Routine");

  // Test SPI communication
  if(SPI_Uart_Init()){
    Serial.println("Bridge initialized successfully!");
  }
  else{
    Serial.println("Could not initialize bridge, locking up.\n\r");
    while(1);
  }
}

void loop()
{
  // Poll for new data in SC16IS750 Recieve buffer
  if(SPI_Uart_ReadByte(LSR) & 0x01)
  {
    polling = 1;
    while(polling)
    {
      if((SPI_Uart_ReadByte(LSR) & 0x01))
      {
        incoming_data = SPI_Uart_ReadByte(RHR);
        Serial.print(incoming_data,BYTE);
      }
```

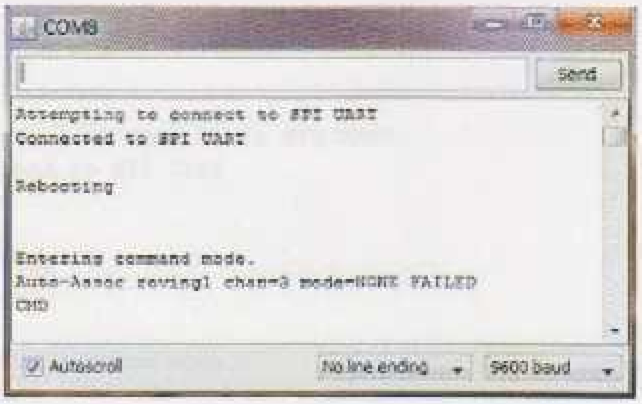
Figure 5.15: Initializing the SC16IS750 SPI-UART Bridge

When adding the WiFly library in Arduino software and initializing it, the SC16IS750 SPI-UART bridge becomes ready without writing the specific code.

Entering Command Mode

Commands begin with a keyword, and have optional additional parameters, generally space delimited .Commands and options are case sensitive. Hex input data can be upper or lower case. String text data, such as SSID is also case sensitive.

Upon power up, the device will be in data mode. To enter command mode, exactly the three characters $$$ must be sent. The device will respond with CMD as shown in Figure 5.19. While in command mode, the device will accept ASCII bytes as commands .To exit command mode, send exit. The device will respond with "EXIT".



```
Attempting to connect to SPI UART
Connected to SPI UART

Rebooting



Entering command mode.
Auto-Assoc roving1 chan=3 mode=NONE FAILED
CMD
```

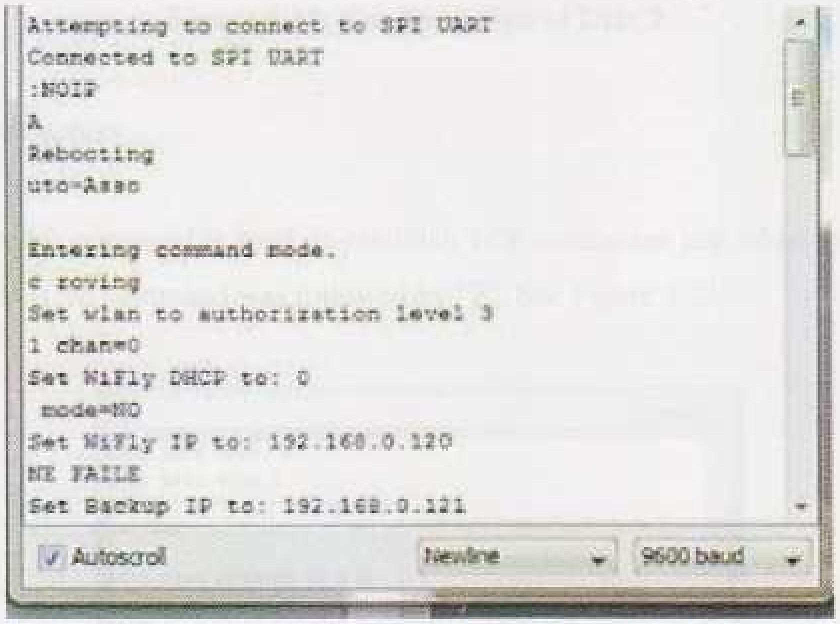
Figure 5.16: Entering Command Mode

Parameters, such as the SSID, channel, IP address, Serial Port settings, and all other settings can be viewed and configured in command mode.
Note that, we can enter command mode in the system code.

IP Parameters

The transceiver must have an IP address to be a part of any network. This IP could be assigned manually using the (set IP address) command followed by the wanted IP. For the WiFly GSX module, the chosen IP address was "192.168.0.101". But for the PC, IP address has been set "192.168.0.100" by (set IP host) command.

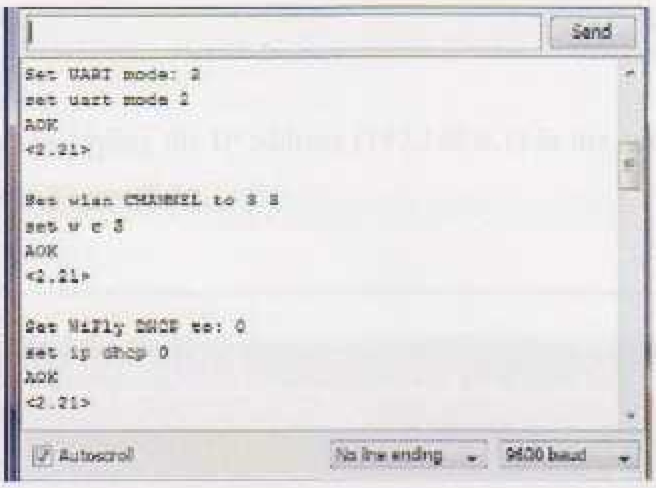
Port number for WiFly GSX module must be reconciled with the port number of the PC.
We have chosen the port number "2000" for both by (set IP local_port) command for WiFly GSX module and (set IP remote_port) command for the PC. See Figure 5.20.



```
Attempting to connect to SPI UART
Connected to SPI UART
:NOIP
A
Rebooting
uto-Asso

Entering command mode.
c roving
Set wlan to authorization level 3
1 chan=0
Set WiFly DHCP to: 0
 mode=NO
Set WiFly IP to: 192.168.0.120
NE FAILE
Set Backup IP to: 192.168.0.121
```

Figure 5.17: Assigning IP & Port Addresses

The Dynamic Host Configuration Protocol (DHCP) is an automatic configuration protocol used on IP networks. DHCP client can be configured using the (set IP DHCP) command. If this command was followed by "0", then the DHCP client is off, and the transceiver will use the stored static IP. If the command was followed by "1", this means that the DHCP client is on and the transceiver will be given an IP address and use the gateway address of the access point. In this project the (set IP DHCP 0) has been used as shown in Figure 5.21.

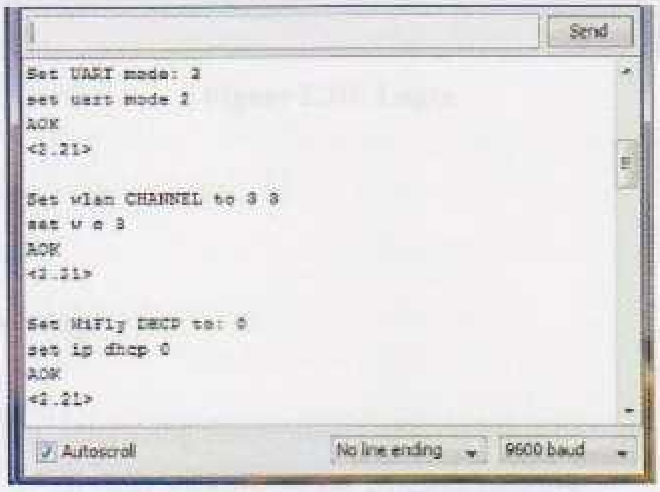


```
Set UART mode: 2
set uart mode 2
AOK
<2.21>

Set wlan CHANNEL to 3 3
set w c 3
AOK
<2.21>

Set WiFly DHCP to: 0
set ip dhcp 0
AOK
<2.21>
```

Figure 5.18: Configuration of DHCP

UART Parameters

(set uart mode) command is used to establish TCP connection just when there is data to send, that if the command was followed by "2". See Figure 5.22.



```
Set UART mode: 2
set uart mode 2
AOK
<2.21>

Set wlan CHANNEL to 3 3
set w c 3
AOK
<2.21>

Set WiFly DHCP to: 0
set ip dhcp 0
AOK
<2.21>
```

Figure 5.19: Set UART Mode

5.5 Configuration of D-LINK router

Steps of router Configuration:

- **Login**

Login is performed by typing the IP address (192.168.0.1) in the address field of web browser, as shown in Figure 5.24.

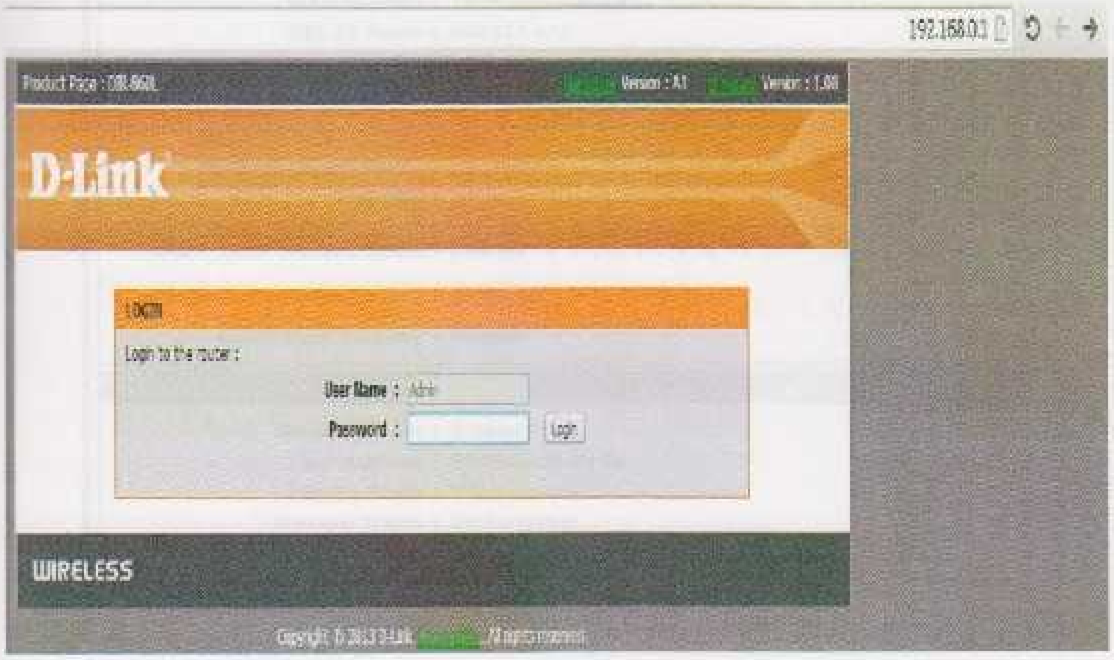


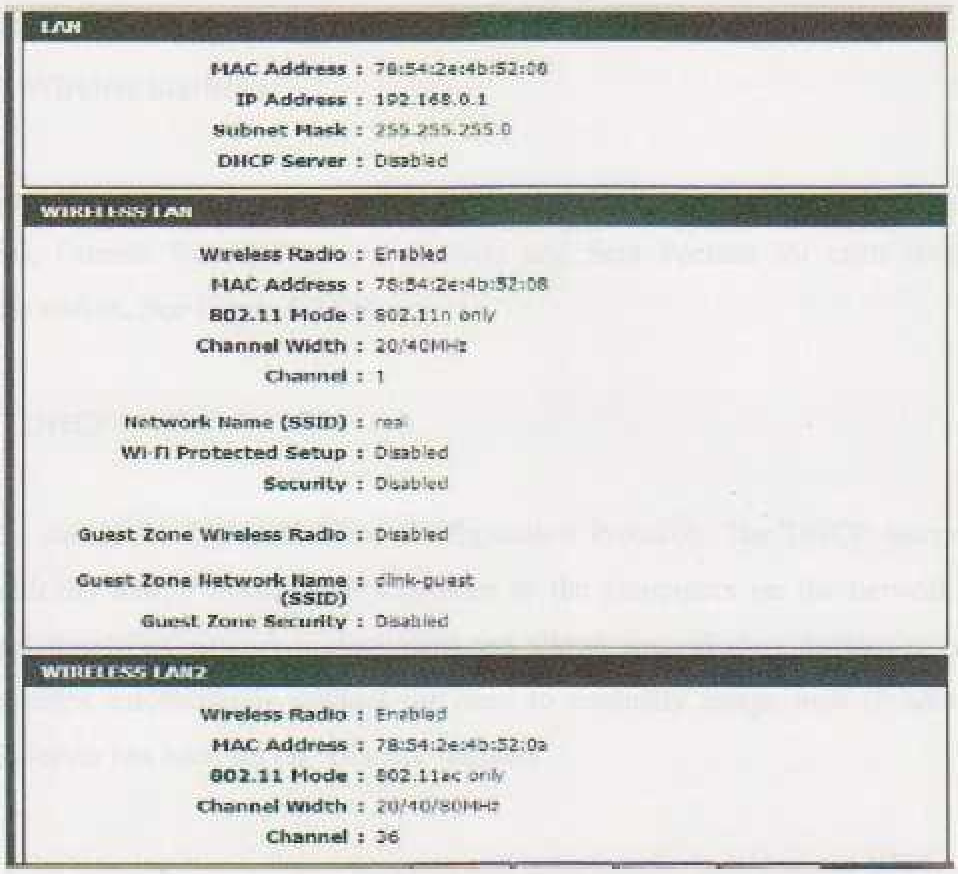
Figure 5.20: Login

- **Wireless settings**

The Wireless option, improving functionality and performance for wireless network, it helps make the AP an ideal solution for the wireless network.

There are six main menus on the leftmost column of the web-based management page: Status, Network, Wireless, DHCP, Wireless Settings and System Tools.

After a successful login by pressing login directly without putting a password, we can configure and manage the router. As shown in Figure 5.21.



```
LAN
                 MAC Address : 78:54:2e:4b:52:08
                  IP Address : 192.168.0.1
                 Subnet Mask : 255.255.255.0
                 DHCP Server : Disabled

WIRELESS LAN
              Wireless Radio : Enabled
                 MAC Address : 78:54:2e:4b:52:08
                802.11 Mode : 802.11n only
               Channel Width : 20/40MHz
                     Channel : 1

        Network Name (SSID) : real
       Wi-Fi Protected Setup : Disabled
                    Security : Disabled

   Guest Zone Wireless Radio : Disabled

    Guest Zone Network Name : dlink-guest
                      (SSID)
         Guest Zone Security : Disabled

WIRELESS LAN2
              Wireless Radio : Enabled
                 MAC Address : 78:54:2e:4b:52:0a
                 802.11 Mode : 802.11ac only
               Channel Width : 20/40/80MHz
                     Channel : 36
```

Figure 5.21: Wireless setting Page

SSID (Set Service Identifier) - Identifies the wireless network name. The created name should be up to 32 characters. We made sure all wireless points in the wireless network with the same SSID. The defined SSID in this project is "real".

Channel - Determines the operating frequency to be used. Channel 1 has been selected in this project. Or sometimes channel 11 is selected.

Mode - Selects the desired wireless mode. The options are: 54 Mbps (802.11g) - Both 802.11g and 802.11n wireless stations can connect to the AP.

300Mbps (802.11n) - Only 802.11b wireless stations can connect to the AP.

We chose mode 300Mbps (802.11n) in our project.

- **Security Settings**

Selecting Network Wireless settings >> Security Settings and disable wireless security for the wireless network . See Figure 5.21 above.

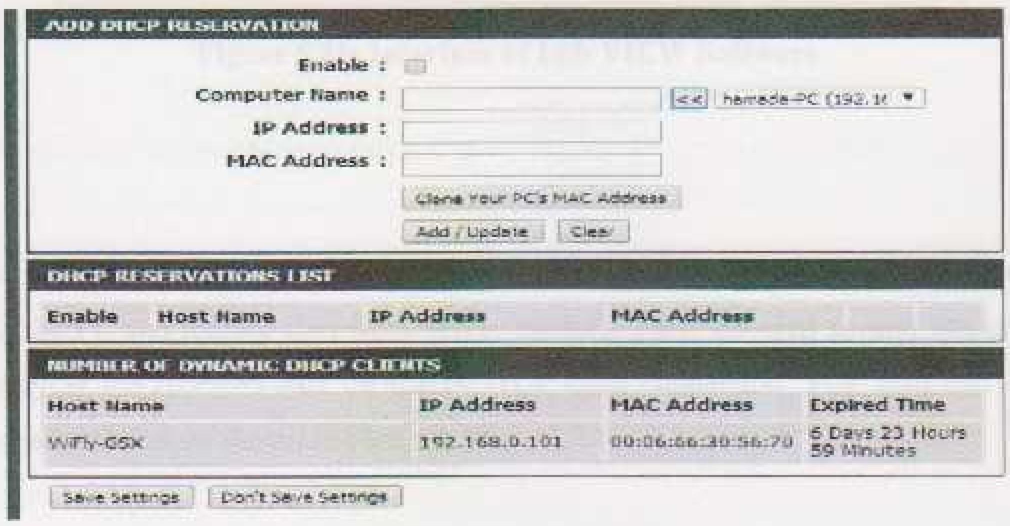
- **Wireless Statistics**

Selecting Wireless >> Network Wireless Settings >> Status will enable view MAC Address, Current Status, Received Packets and Sent Packets for each connected wireless station. See Figure 5.21 above.

- **DHCP**

DHCP stands for Dynamic Host Configuration Protocol. The DHCP Server will automatically assign dynamic IP addresses to the computers on the network. This protocol simplifies network management and allows new wireless devices to receive IP addresses automatically without the need to manually assign new IP addresses. DHCP Server has been set on "Disable" choice.

Now, after running the Arduino program and assign static ip address to WiFly shield it appears as a client connected to router as seen in Figure 5.22 :



ADD DHCP RESERVATION

Enable :	
Computer Name :	hamede-PC (192.1(
IP Address :	
MAC Address :	

DHCP RESERVATIONS LIST

Enable	Host Name	IP Address	MAC Address

NUMBER OF DYNAMIC DHCP CLIENTS

Host Name	IP Address	MAC Address	Expired Time
WiFly-GSX	192.168.0.101	00:06:66:30:56:70	6 Days 23 Hours 59 Minutes

Figure 5.22 : WiFly Shield Appears as a client

5.6 Receiving Data by Lab VIEW Software

Lab VIEW software is a platform and development environment for a visual programming language from National Instruments. The purpose of such programming is automating the usage of processing and measuring equipment in any laboratory setup. The graphical language is named "G". Lab VIEW is commonly used for data acquisition, instrument control, and industrial automation.

Currently, there are many versions of software Lab VIEW program, in this project version 7.1 has been used. Figure 5.23 shows the Interface of Lab VIEW software.

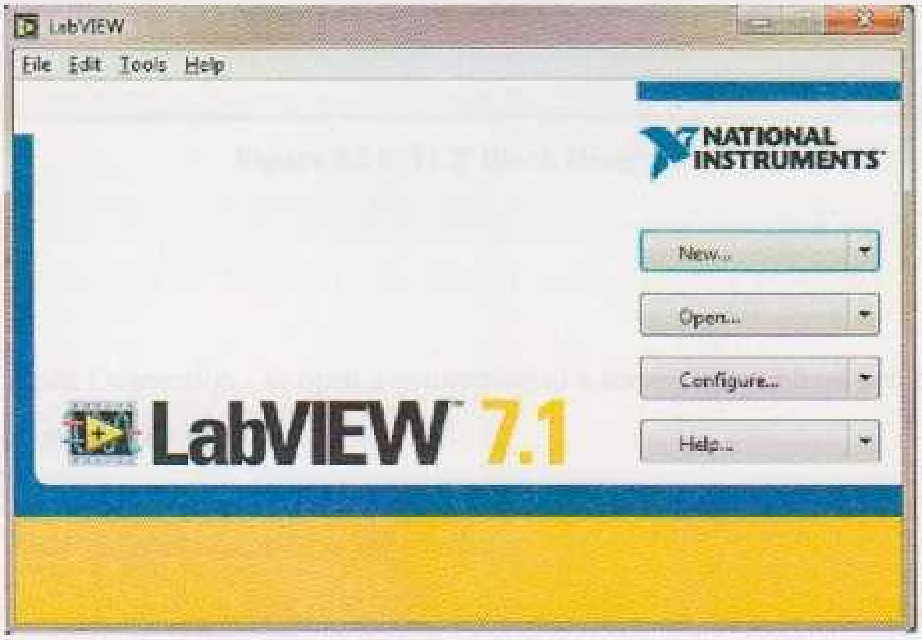


Figure 5.23: Interface of Lab VIEW Software

The purpose of using Lab VIEW in this project is to communicate the Wi-Fi network with the PC and receive data from WiFly module. Then we need several blocks for displaying the received data on the PC TCP open connection, TCP read and TCP close connection, as shown in Figure 5.24.

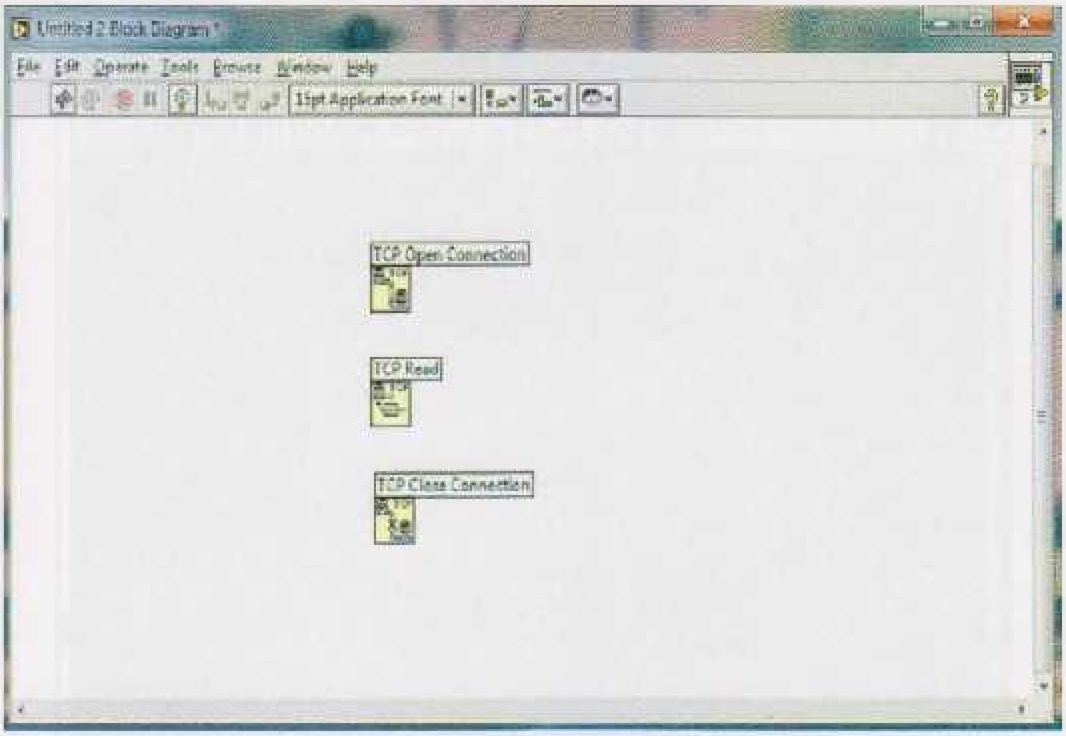

Untitled 2 Block Diagram *
File Edit Operate Tools Browse Window Help
15pt Application Font
TCP Open Connection
TCP Read
TCP Close Connection

Figure 5.24: TCP Block Diagram

* TCP Open Connection - to open a connection to a server. The address and the port of the server must be specified.

* TCP Read - to read results from the server.

* TCP Close Connection - to close the connection to the server.

The building of the previous blocks provides the following program interface – Figure 5.25- in which the results are displayed:

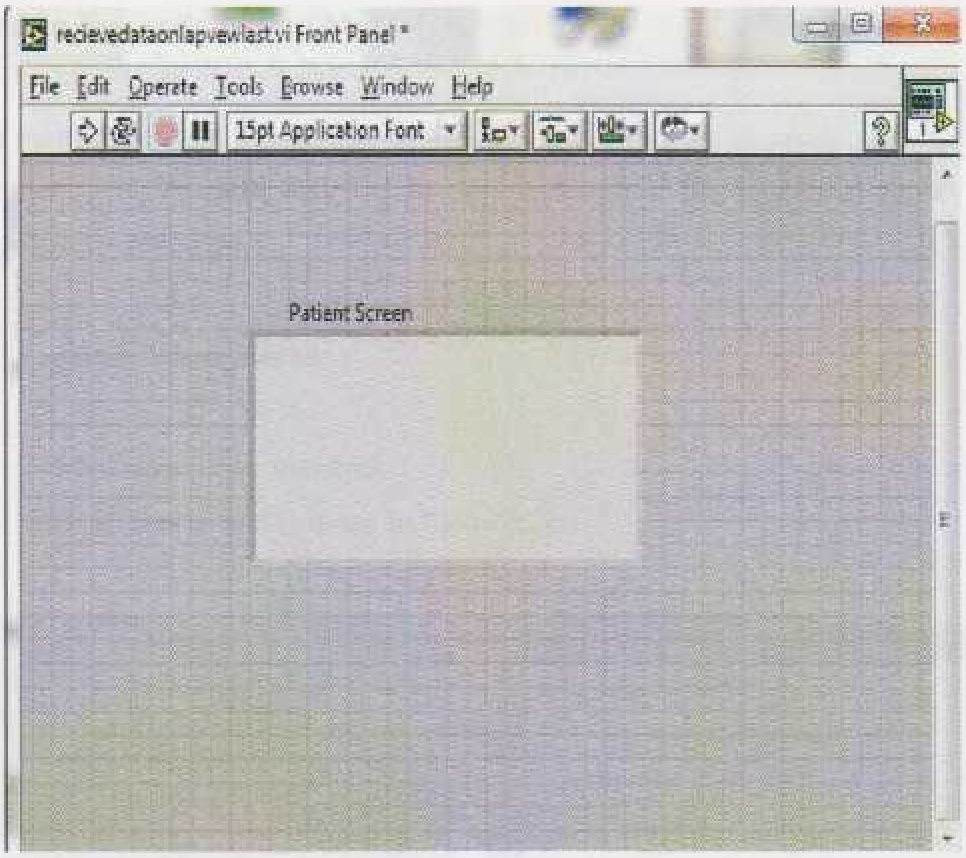

recievedataonlapvewlast.vi Front Panel *
File Edit Operate Tools Browse Window Help
15pt Application Font
Patient Screen

Figure 5.25: Displaying the Results on the Program Interface

5.7 Wamp Server Software (MySQL)

It is a web development platform on Windows allowing us to create dynamic web applications with Apache, PHP and MySQL. Alongside, PhpMyAdmin allows us to manage easily databases.

How to operate Wamp server :

1- Start Wamp server manager.
2- Start all services. As shown in figure 5.26.

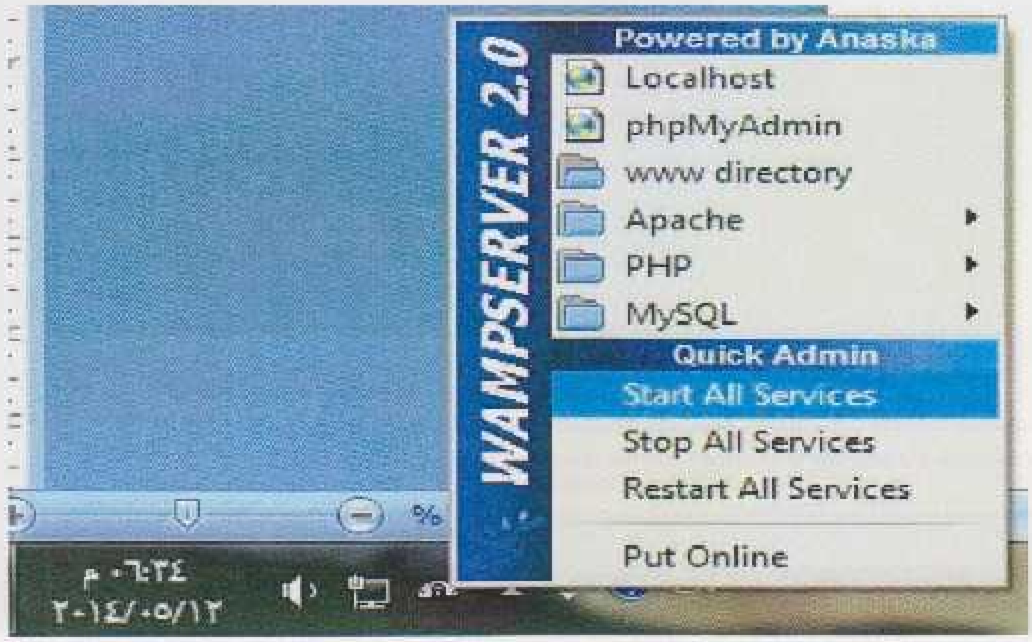
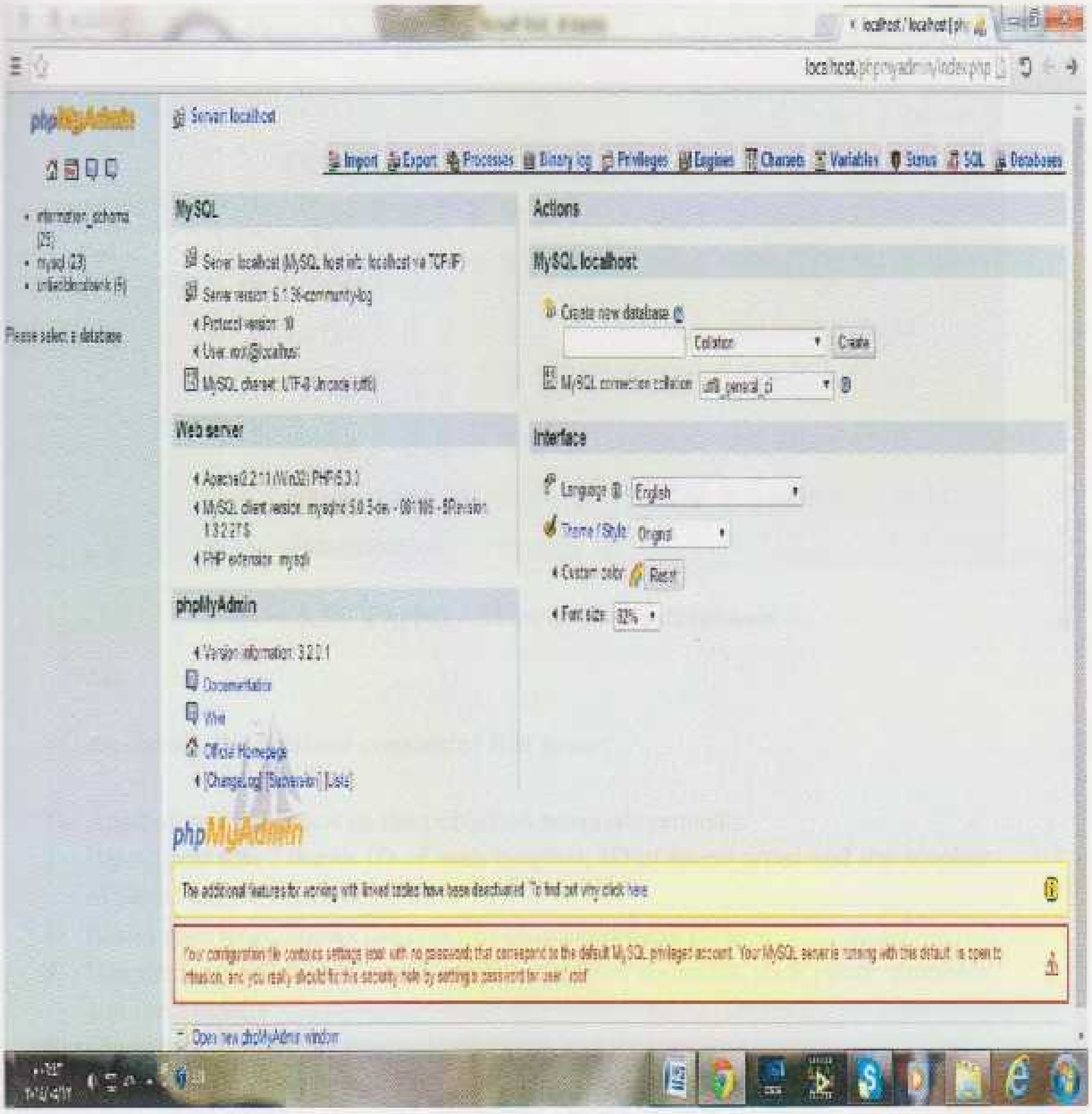


Figure 5.26: Displaying the Wamp server starter page

3- Open web browser then put local host/PhpMyAdmin/ in the search location. As shown in figure 5.27.



Server: localhost

Import | Export | Processes | Binary log | Privileges | Engines | Charsets | Variables | Status | SQL | Databases

MySQL

- Server: localhost (MySQL host info: localhost via TCP/IP)
- Server version: 5.1.36-community-log
- Protocol version: 10
- User: root@localhost
- MySQL charset: UTF-8 Unicode (utf8)

Web server

- Apache/2.2.11 (Win32) PHP/5.3.0
- MySQL client version: mysqlnd 5.0.5-dev - 081106 - $Revision: 1.3.2.27 $
- PHP extension: mysqli

phpMyAdmin

- Version information: 3.2.0.1
- Documentation
- Wiki
- Official Homepage
- [ChangeLog] [Subversion] [Lists]

Actions

MySQL localhost

- Create new database: Collation | Create
- MySQL connection collation: utf8_general_ci

Interface

- Language: English
- Theme / Style: Original
- Custom color: Reset
- Font size: 82%

The additional features for working with linked tables have been deactivated. To find out why click here.

Your configuration file contains settings (root with no password) that correspond to the default MySQL privileged account. Your MySQL server is running with this default, is open to intrusion, and you really should fix this security hole by setting a password for user 'root'.

Open new phpMyAdmin window

Figure 5.27: PhpMyAdmin to create databases

4- Know create database as it needed to be, we create the following data base in order to be programmed. We called it unifiedbloodbank . as shown in figure 2.28.

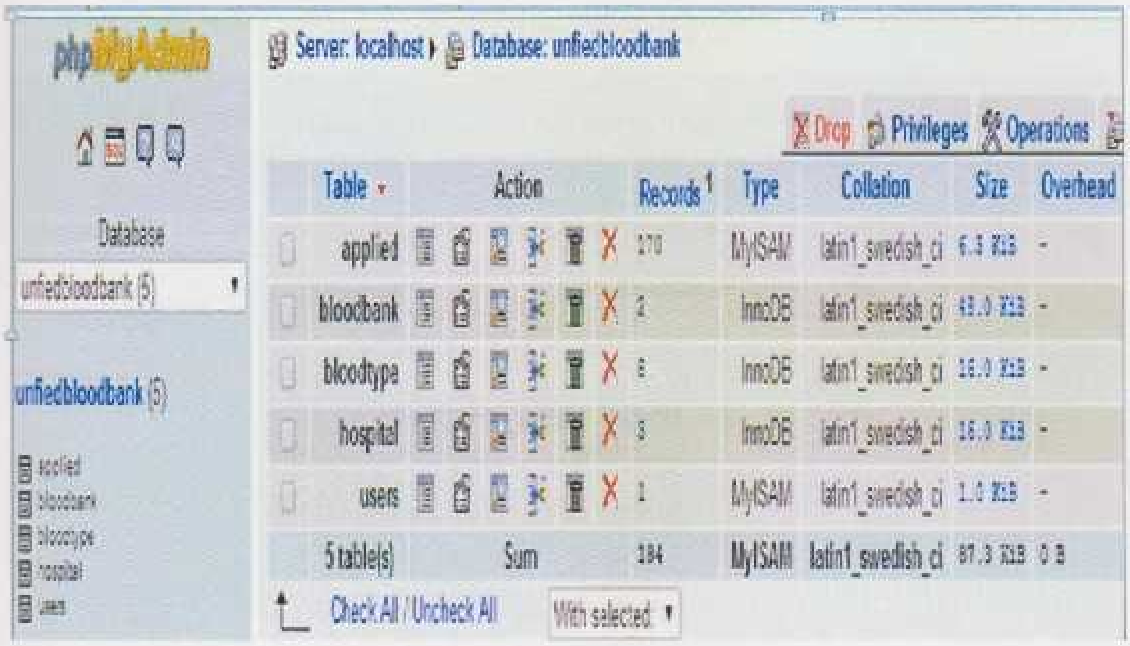


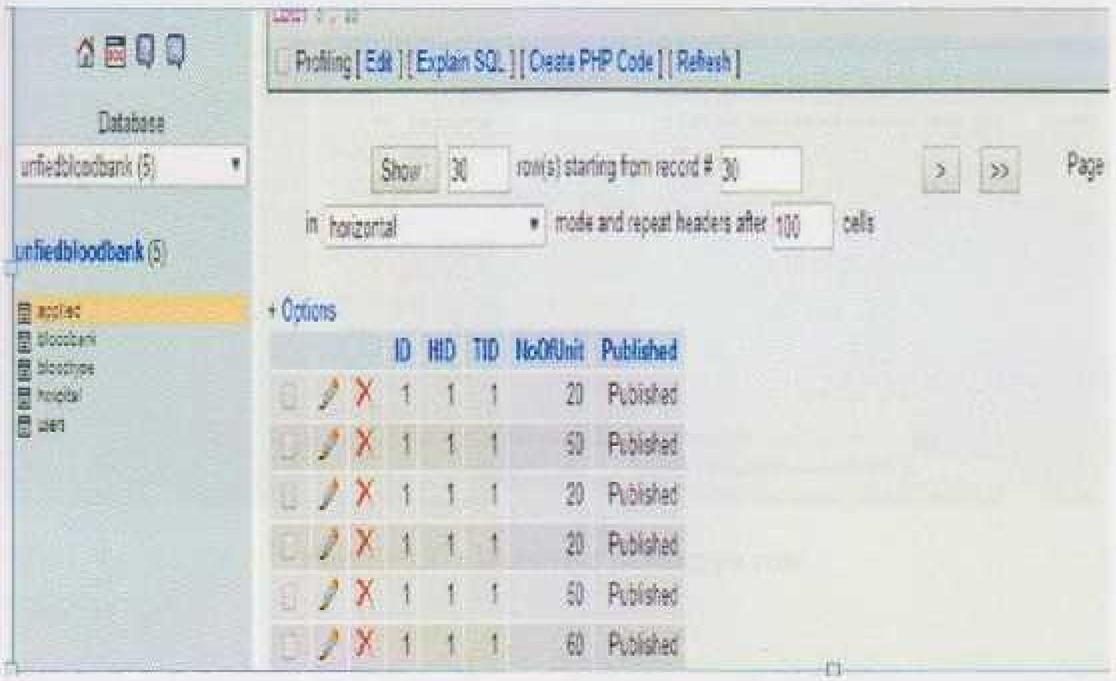
	Table	Action	Records	Type	Collation	Size	Overhead
	applied		270	MyISAM	latin1_swedish_ci	6.3 KiB	-
	bloodbank		2	InnoDB	latin1_swedish_ci	48.0 KiB	-
	bloodtypa		8	InnoDB	latin1_swedish_ci	16.0 KiB	-
	hospital		3	InnoDB	latin1_swedish_ci	16.0 KiB	-
	users		1	MyISAM	latin1_swedish_ci	1.0 KiB	-
	5 table(s)	Sum	184	MyISAM	latin1_swedish_ci	87.3 KiB	0 B

Figure 5.28: PhpMyAdmin to create databases

- As shown, the database consists of five rows :

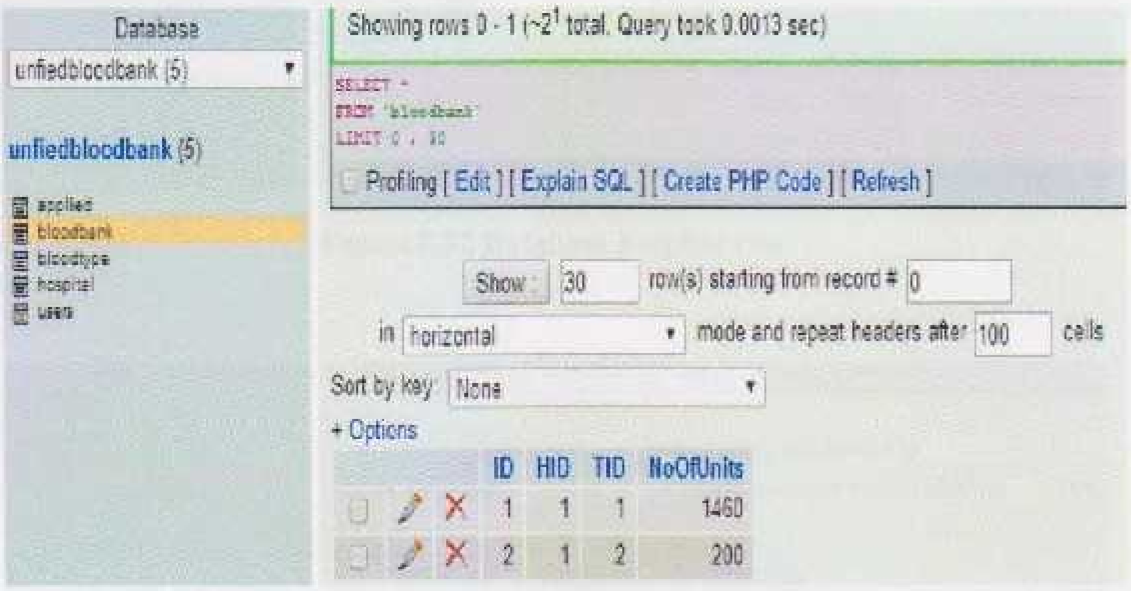
1- Applied row : to show us the published hospitals requests.
2- Bloodbank row : shows ID of each hospital, ID of blood types and the number of blood units of each hospital.
3- Blood type row : shows the types of blood and there ID's.
4- Hospital row : shows ID's of hospitals, name of hospitals, their phone numbers and their emails.
5- Users row : shows the hospitals accounts and their passwords.

The following figures 5.29,5.30,5.31,5.32 describe the upper details:


Database
unfiedbloodbank (5)
unfiedbloodbank (5)
applied
bloodbank
bloodtype
hospital
users
Profiling [Edit] [Explain SQL] [Create PHP Code] [Refresh]
Show : 30 row(s) starting from record # 30
in horizontal mode and repeat headers after 100 cells
Page
+ Options

ID	HID	TID	NoOfUnit	Published
1	1	1	20	Published
1	1	1	50	Published
1	1	1	20	Published
1	1	1	20	Published
1	1	1	50	Published
1	1	1	60	Published

Figure 5.29: Database applied row

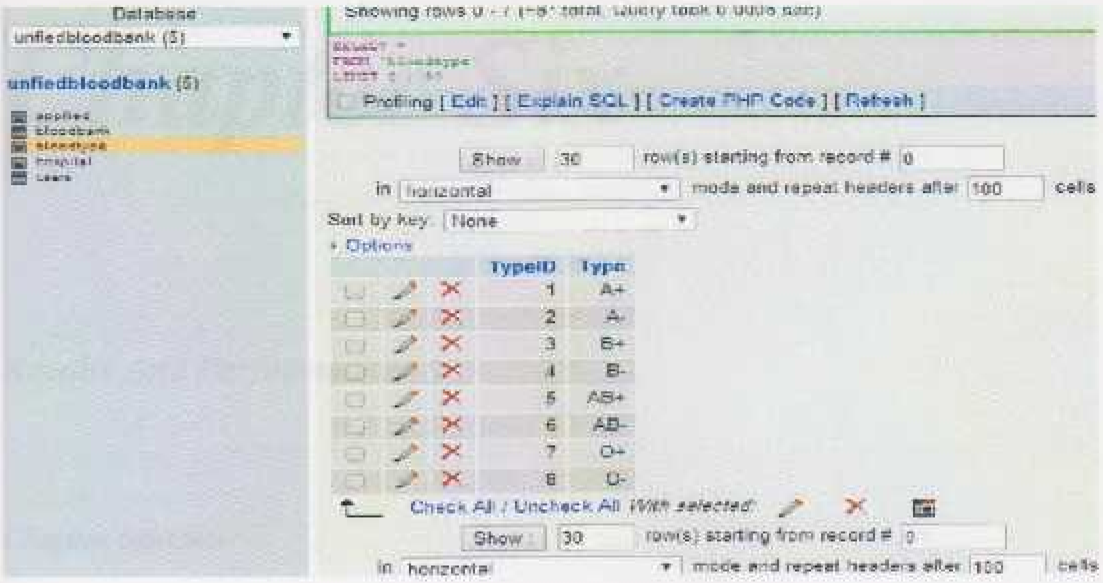

Database
unfiedbloodbank (5)
unfiedbloodbank (5)
applied
bloodbank
bloodtype
hospital
users
Profiling [Edit] [Explain SQL] [Create PHP Code] [Refresh]
Show : 30 row(s) starting from record # 0
in horizontal mode and repeat headers after 100 cells
Sort by key: None
+ Options

Showing rows 0 - 1 (~2 total, Query took 0.0013 sec)

```
SELECT *
FROM `bloodbank`
LIMIT 0 , 30
```

ID	HID	TID	NoOfUnits
1	1	1	1460
2	1	2	200

Figure 5.30: Database Bloodbank row



Database
unfiedbloodbank (5)

unfiedbloodbank (5)

- applied
- bloodbank
- bloodtype
- hospital
- users

Showing rows 0 - 7 (~8¹ total, Query took 0.0005 sec)

```
SELECT *
FROM `bloodtype`
LIMIT 0 , 30
```

Profiling [Edit] [Explain SQL] [Create PHP Code] [Refresh]

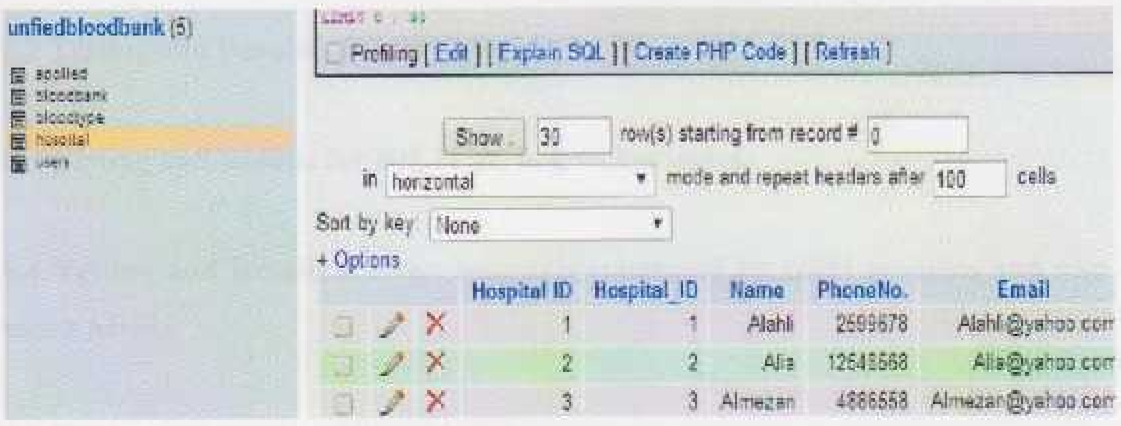
Show : 30 row(s) starting from record # 0
in horizontal mode and repeat headers after 100 cells
Sort by key: None
+ Options

TypeID	Type
1	A+
2	A-
3	B+
4	B-
5	AB+
6	AB-
7	O+
8	O-

Check All / Uncheck All *With selected:*

Show : 30 row(s) starting from record # 0
in horizontal mode and repeat headers after 100 cells

Figure 5.31: Database bloodtype row



unfiedbloodbank (5)

- applied
- bloodbank
- bloodtype
- hospital
- users

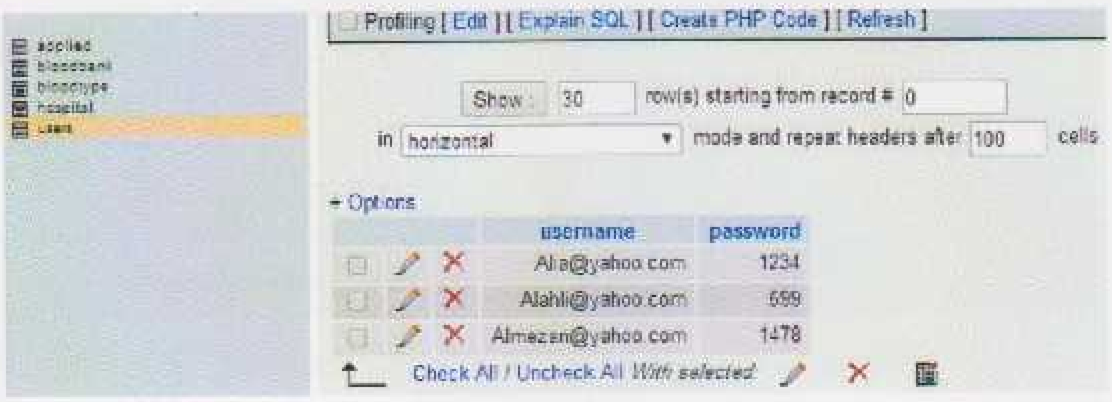
```
LIMIT 0 , 30
```

Profiling [Edit] [Explain SQL] [Create PHP Code] [Refresh]

Show : 30 row(s) starting from record # 0
in horizontal mode and repeat headers after 100 cells
Sort by key: None
+ Options

Hospital ID	Hospital_ID	Name	PhoneNo.	Email
1	1	Alahli	2599678	Alahli@yahoo.com
2	2	Alia	12548568	Alia@yahoo.com
3	3	Almezan	4866558	Almezan@yahoo.com

Figure 5.32 Database hospital row



- applied
- bloodbank
- bloodtype
- hospital
- users

Profiling [Edit] [Explain SQL] [Create PHP Code] [Refresh]

Show : 30 row(s) starting from record # 0
in horizontal mode and repeat headers after 100 cells

+ Options

username	password
Alia@yahoo.com	1234
Alahli@yahoo.com	699
Almezan@yahoo.com	1478

Check All / Uncheck All *With selected:*

Figure 5.33 Database users row

Chapter Six

Results And Performance

Chapter contents

6.1 Introduction.

The final stage to complete the project is to test the systems to get results and measure the performance of our system. This chapter shows all tests and all results.

6.2 Testing and Results of WiFly Shield.

Checking and testing of the sensors and WiFly shield are illustrated in this section in addition to the testing results.

6.2.1 Temperature Sensor Test

Temperature sensor "LM35" has been implemented and tested if it gives the results in volts. After that the connection between LM35 and Arduino microcontroller has been tested. See Figure 6.1.

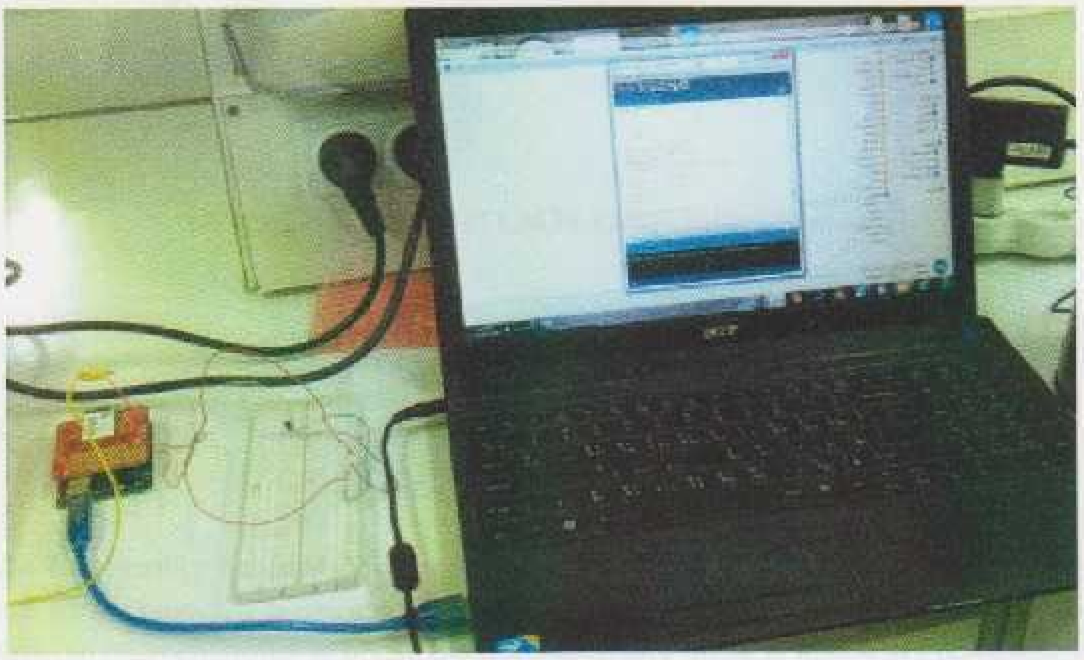
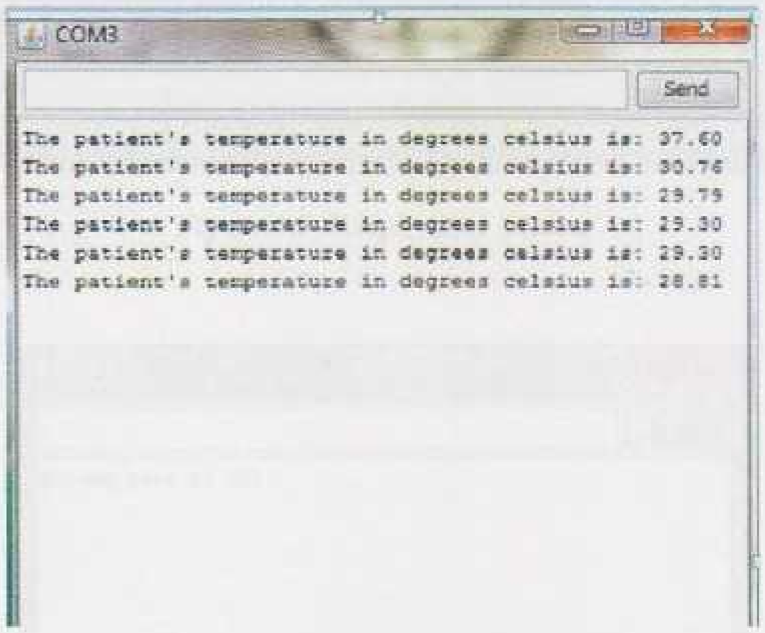


Figure 6.1: Connection between LM35 and Arduino

The resulted voltage value was converted to a Celsius value by a certain process at the Arduino microcontroller.

A test was performed in a room and the result 37.60 C° as it was expected. Figure 6.2 shows the result at the serial monitor of the Arduino program. The result was shown steady after a little time because the thermal sensor takes certain time to give its final output result.



```
The patient's temperature in degrees celsius is: 37.60
The patient's temperature in degrees celsius is: 30.76
The patient's temperature in degrees celsius is: 29.79
The patient's temperature in degrees celsius is: 29.30
The patient's temperature in degrees celsius is: 29.30
The patient's temperature in degrees celsius is: 28.81
```

Figure 6.2: Result of LM35 Temperature Sensor

6.2.2 Respiratory Sensor Test

The **Respiratory Sensor**, shown in Figure 6.3, it is a complete module power supply and Respiration Kit were used and tested, then the output was connected to the Arduino microcontroller to obtain the respiration rate which is shown in Figure 6.4.

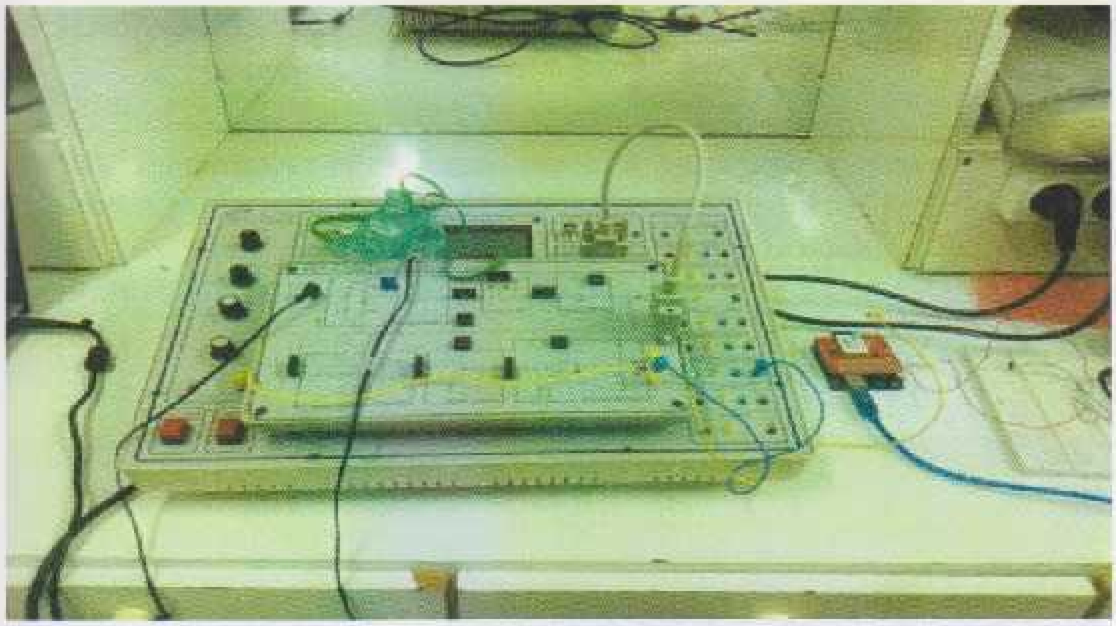
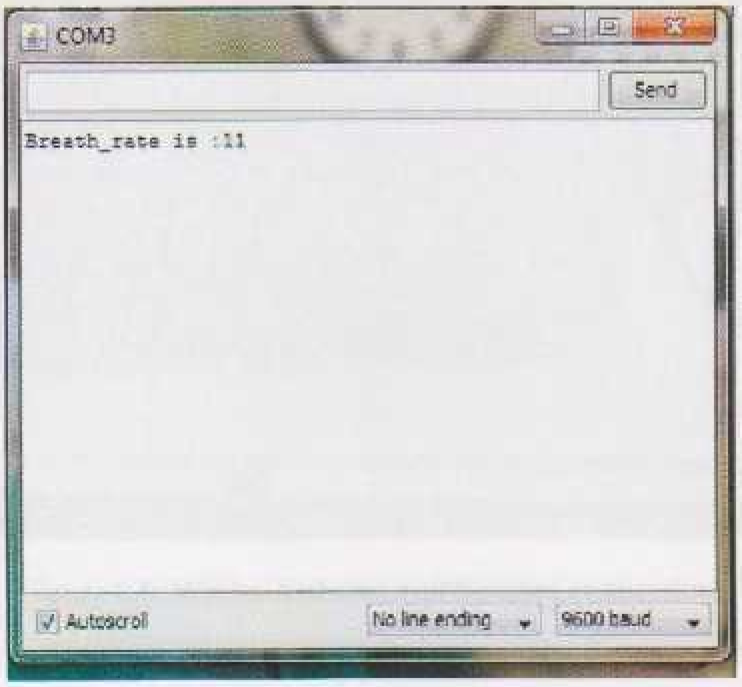


Figure 6.3 Respiratory Sensor



```
Breath_rate is :11
```

Figure 6.4: Breath Rate result

The normal result of respiration rate is between 15-17 breath per minute and this results have been gotten during tests 11,13,16,17and 15. And the above figure shows an example of these tests.

6.2.3 Lab View Testing

The data and the results of the sensors needed to be transferred using wifi networks from access point to laptop, and this was implemented and tested successfully. The figure 6.5 shows the implementation and figure 6.6 shows the result.

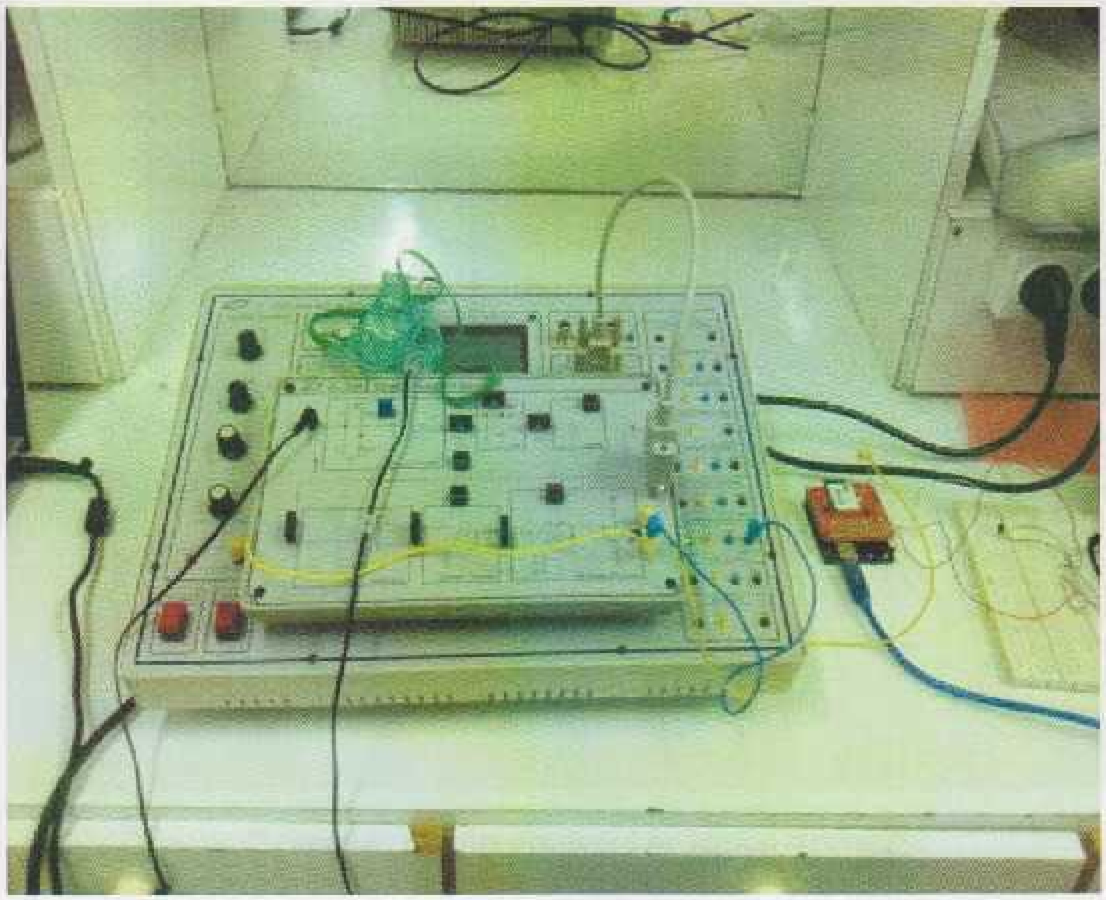
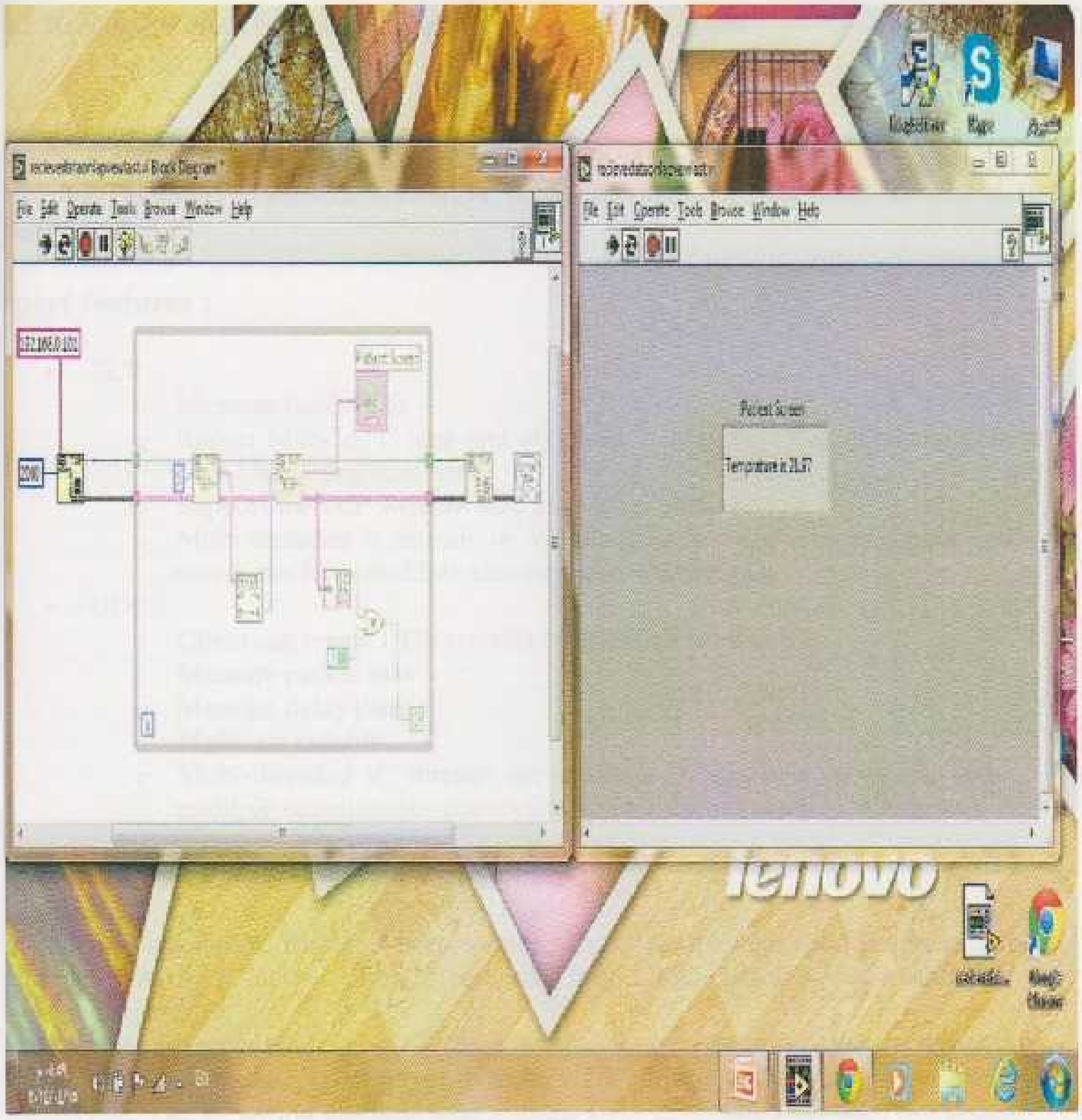


Figure 6.5 WiFly, Arduino and Sensors connections



Patient Screen

Temprature is 26.37

Figure 6.6: Results displayed on laptop screen

6.3 Results for 802.11 ac, n, g and performance comparisons

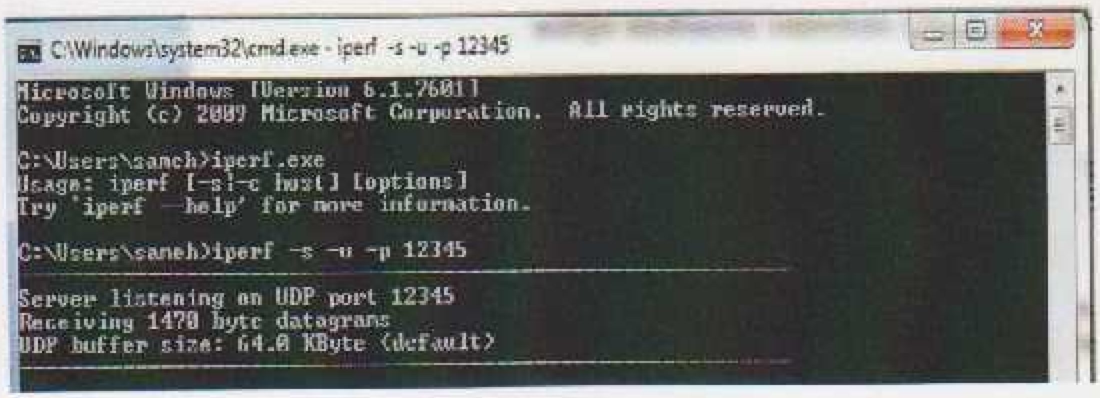
While tools to measure network performance. Iperf was originally developed by NLANR/DAST as a modern alternative for measuring TCP and UDP bandwidth performance.

Iperf features :

- TCP
  - Measure bandwidth
  - Report MSS/MTU size and observed read sizes. (Max. Transmission Unit / Max. Segment Size)
  - Support for TCP window size via socket buffers.
  - Multi-threaded if threads or Win32 threads are available. Client and server can have multiple simultaneous connections.
- UDP
  - Client can create UDP streams of specified bandwidth.
  - Measure packet loss
  - Measure delay jitter
  - Multicast capable
  - Multi-threaded if threads are available. Client and server can have multiple simultaneous connections. (This doesn't work in Windows.)

In order to accomplish the Iperf tests we need firstly to open the Iperf program that must be found in C disk in Users of computer from CMD (Black Screen) using this command iperf.exe.

Then, we need to make a UDP server to make UDP connection between server and a client node. The following example tests a single UDP connection between two nodes on port 12345. As shown in figure 6.7.



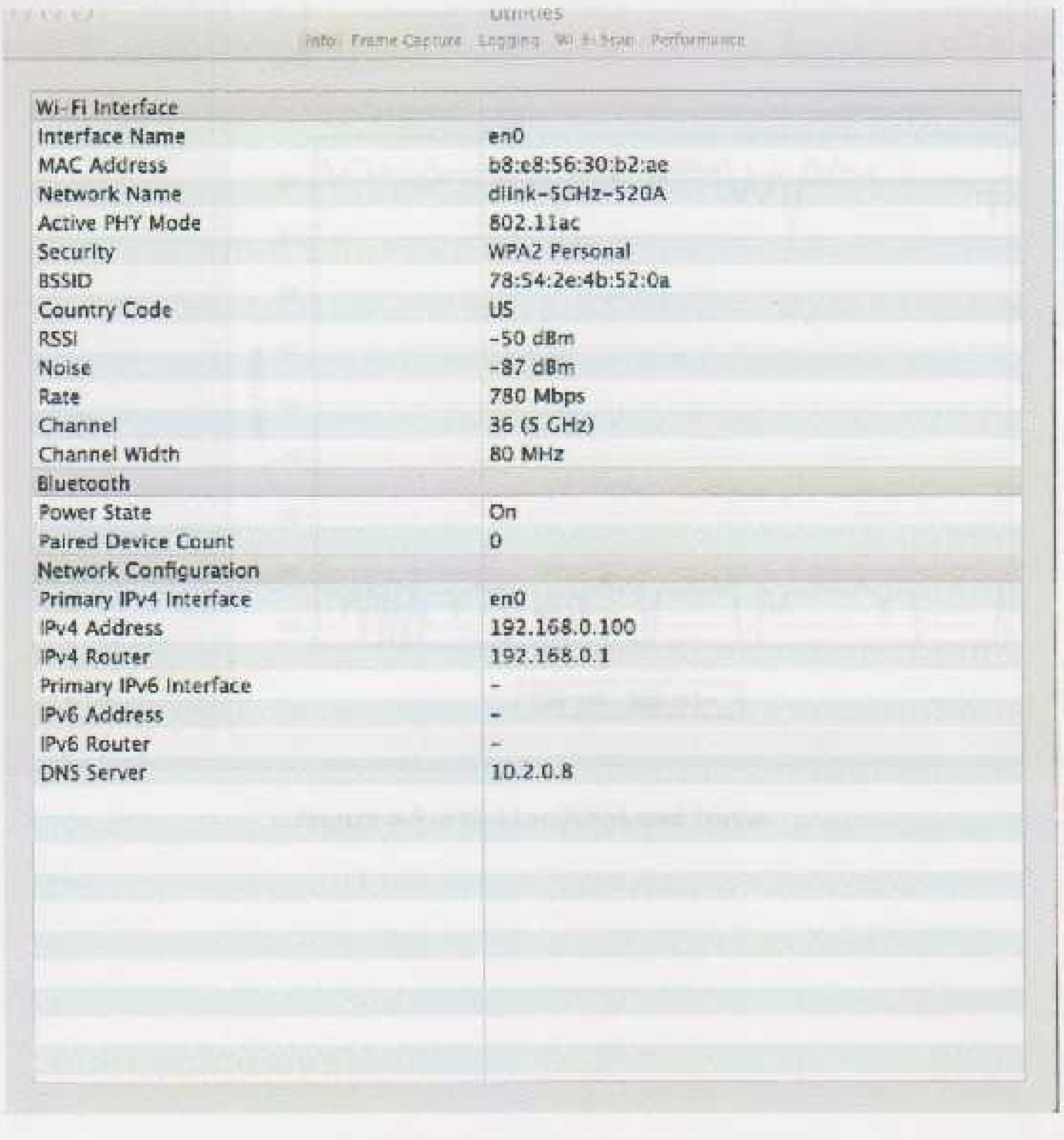
```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\sanch>iperf.exe
Usage: iperf [-s|-c host] [options]
Try `iperf --help' for more information.

C:\Users\saneh>iperf -s -u -p 12345
------------------------------------------------------------
Server listening on UDP port 12345
Receiving 1470 byte datagrams
UDP buffer size: 64.0 KByte (default)
------------------------------------------------------------
```

Figure 6.7 Iperf UDP Server

- This test was performed in two ways and the results was nearly same .

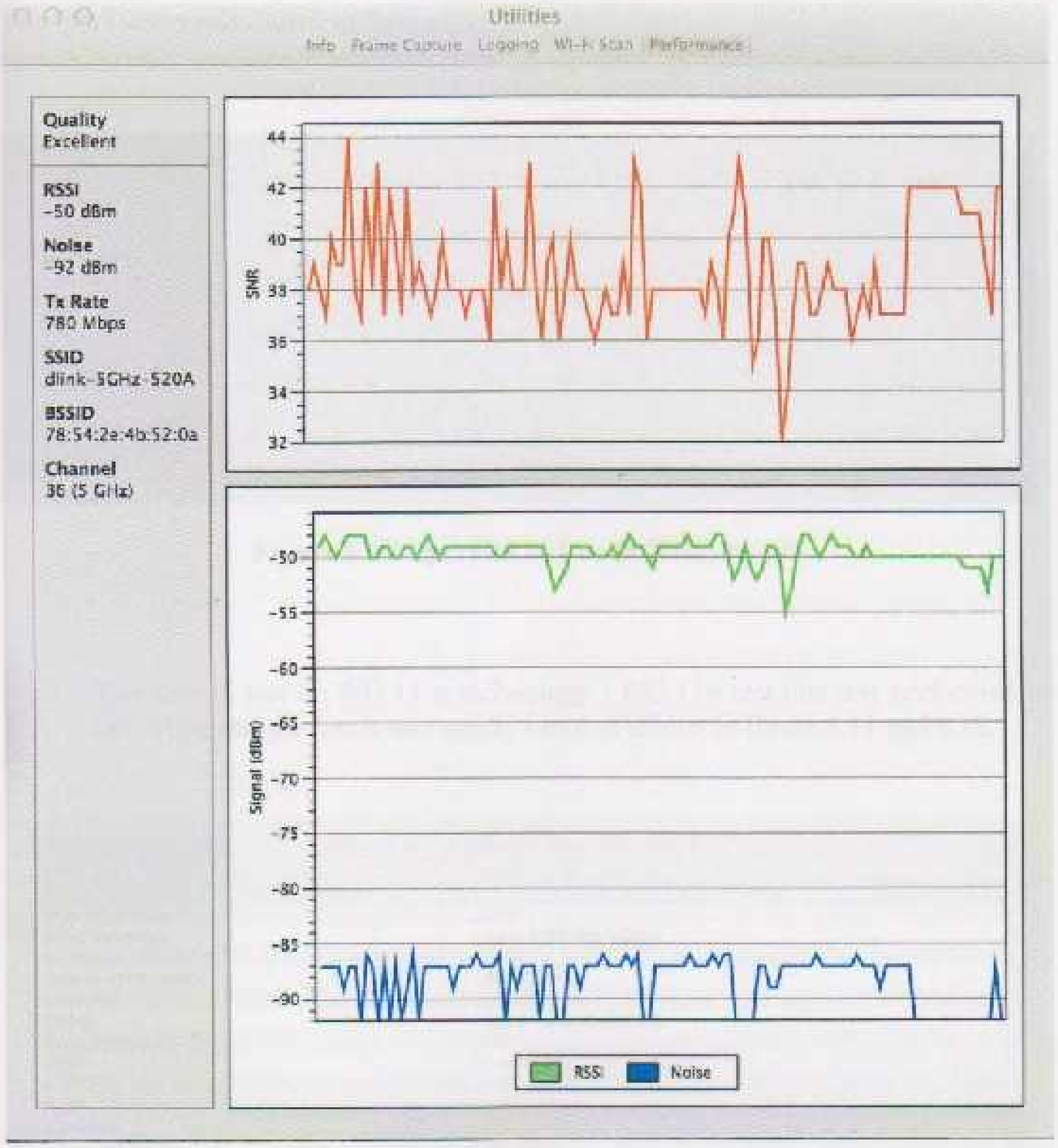
1- The first test on 802.11 ac technology : 802.11ac test this test performed in two ways and the result was nearly same as shown in figure 6.8 and 6.9.

The result on the first one was as following :



Wi-Fi Interface	
Interface Name	en0
MAC Address	b8:e8:56:30:b2:ae
Network Name	dlink-5GHz-520A
Active PHY Mode	802.11ac
Security	WPA2 Personal
BSSID	78:54:2e:4b:52:0a
Country Code	US
RSSI	-50 dBm
Noise	-87 dBm
Rate	780 Mbps
Channel	36 (5 GHz)
Channel Width	80 MHz
Bluetooth	
Power State	On
Paired Device Count	0
Network Configuration	
Primary IPv4 Interface	en0
IPv4 Address	192.168.0.100
IPv4 Router	192.168.0.1
Primary IPv6 Interface	-
IPv6 Address	-
IPv6 Router	-
DNS Server	10.2.0.8

Figure 6.8 802.11ac numerical results



Quality
Excellent

RSSI
-50 dBm

Noise
-92 dBm

Tx Rate
780 Mbps

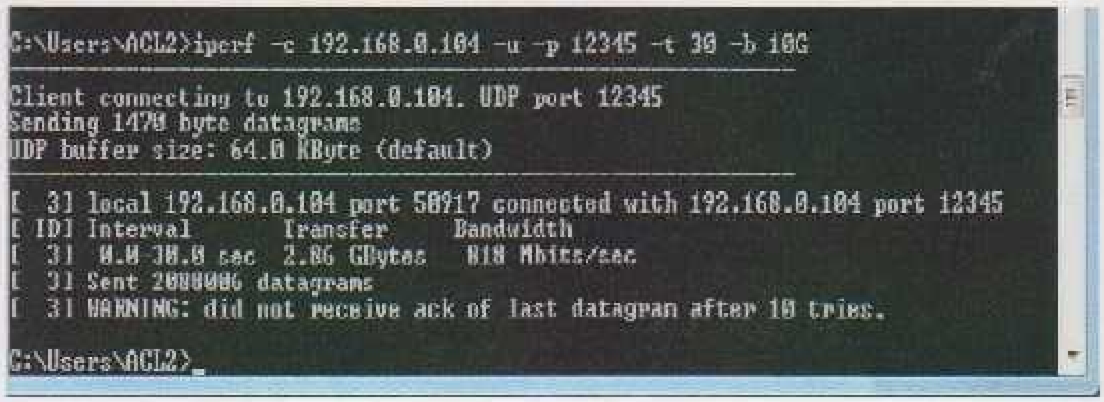
SSID
dlink-5GHz-520A

BSSID
78:54:2e:4b:52:0a

Channel
36 (5 GHz)

Figure 6.9 802.11ac RSSI and Noise

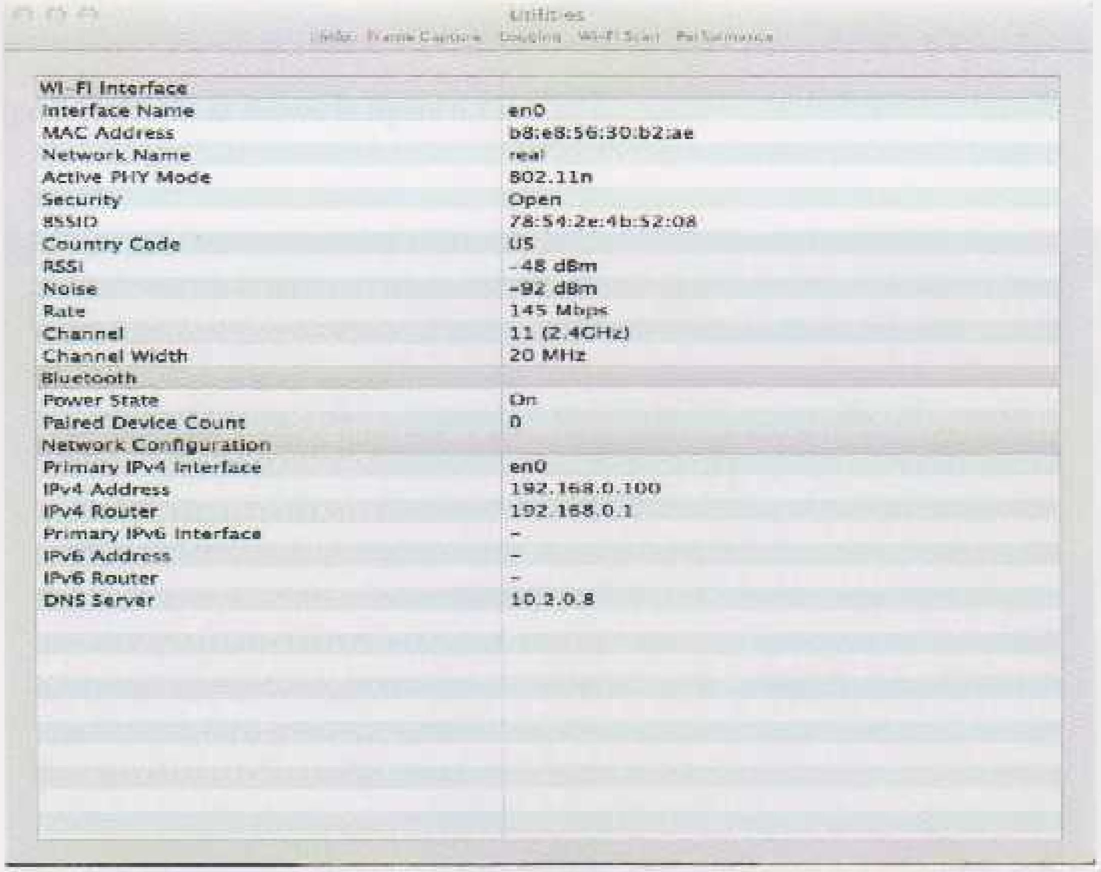
The Iperf test result shown in figure 6.10 :



```
C:\Users\ACL2>iperf -c 192.168.0.104 -u -p 12345 -t 30 -b 10G
------------------------------------------------------------
Client connecting to 192.168.0.104. UDP port 12345
Sending 1470 byte datagrams
UDP buffer size: 64.0 KByte (default)
------------------------------------------------------------
[  3] local 192.168.0.104 port 50917 connected with 192.168.0.104 port 12345
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-30.0 sec  2.86 GBytes   818 Mbits/sec
[  3] Sent 2000006 datagrams
[  3] WARNING: did not receive ack of last datagran after 10 tries.

C:\Users\ACL2>_
```

Figure 6.10 Iperf 802.11 ac (5 GHz) result

2- The second test on 802.11 n technology : 802.11n test this test performed in two ways and the result was nearly same as shown in figure 6.11 and 6.12.



Wi-Fi Interface	
Interface Name	en0
MAC Address	b8:e8:56:30:b2:ae
Network Name	real
Active PHY Mode	802.11n
Security	Open
BSSID	78:54:2e:4b:52:08
Country Code	US
RSSI	-48 dBm
Noise	-92 dBm
Rate	145 Mbps
Channel	11 (2.4GHz)
Channel Width	20 MHz
Bluetooth	
Power State	On
Paired Device Count	0
Network Configuration	
Primary IPv4 Interface	en0
IPv4 Address	192.168.0.100
IPv4 Router	192.168.0.1
Primary IPv6 Interface	-
IPv6 Address	-
IPv6 Router	-
DNS Server	10.2.0.8

Figure 6.11 802.11 n (2.4GHz) numerical results

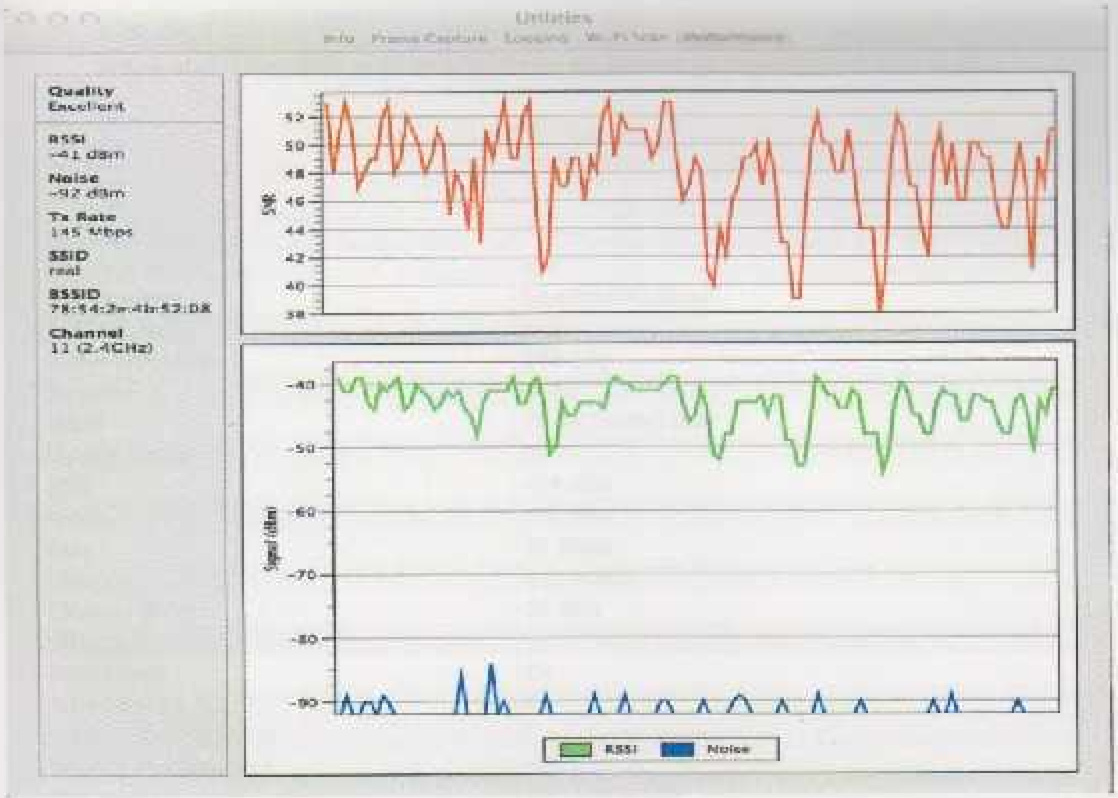
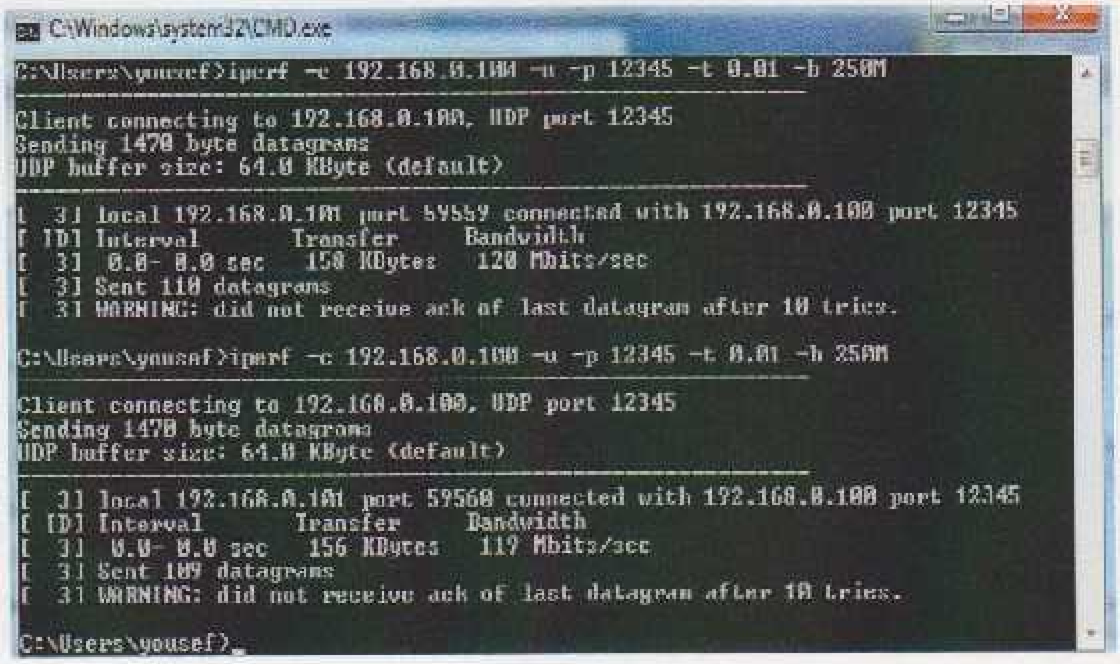


Figure 6.12 802.11 n RSSI and Noise (2.4GHz)

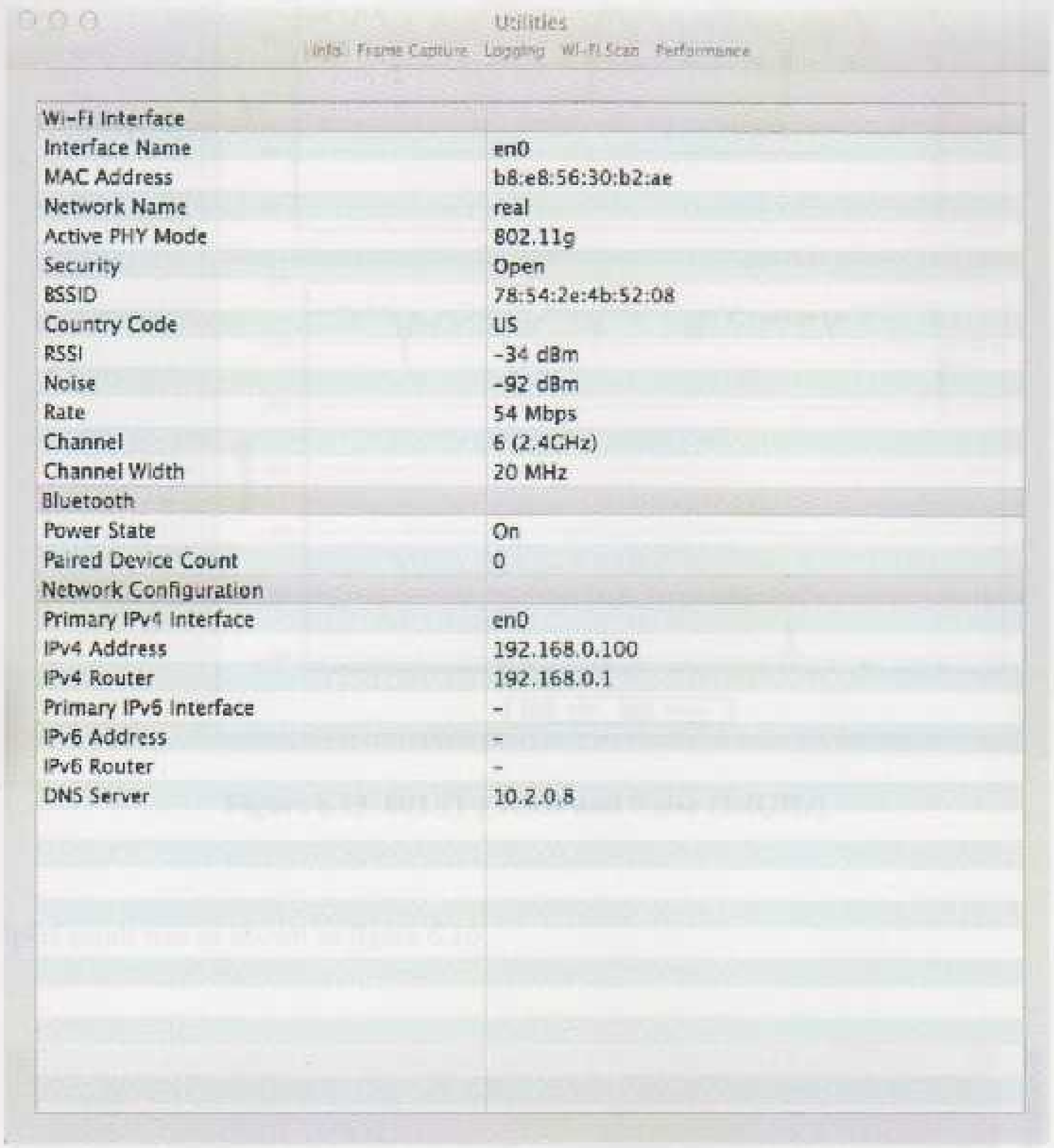
Iperf result was as shown in figure 6.13:



```
C:\Users\yousef>iperf -c 192.168.0.100 -u -p 12345 -t 0.01 -b 250M
------------------------------------------------------------
Client connecting to 192.168.0.100, UDP port 12345
Sending 1470 byte datagrams
UDP buffer size: 64.0 KByte (default)
------------------------------------------------------------
[  3] local 192.168.0.101 port 59559 connected with 192.168.0.100 port 12345
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 0.0 sec   158 KBytes   120 Mbits/sec
[  3] Sent 110 datagrams
[  3] WARNING: did not receive ack of last datagram after 10 tries.

C:\Users\yousef>iperf -c 192.168.0.100 -u -p 12345 -t 0.01 -b 250M
------------------------------------------------------------
Client connecting to 192.168.0.100, UDP port 12345
Sending 1470 byte datagrams
UDP buffer size: 64.0 KByte (default)
------------------------------------------------------------
[  3] local 192.168.0.101 port 59560 connected with 192.168.0.100 port 12345
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 0.0 sec   156 KBytes   119 Mbits/sec
[  3] Sent 109 datagrams
[  3] WARNING: did not receive ack of last datagram after 10 tries.

C:\Users\yousef>_
```

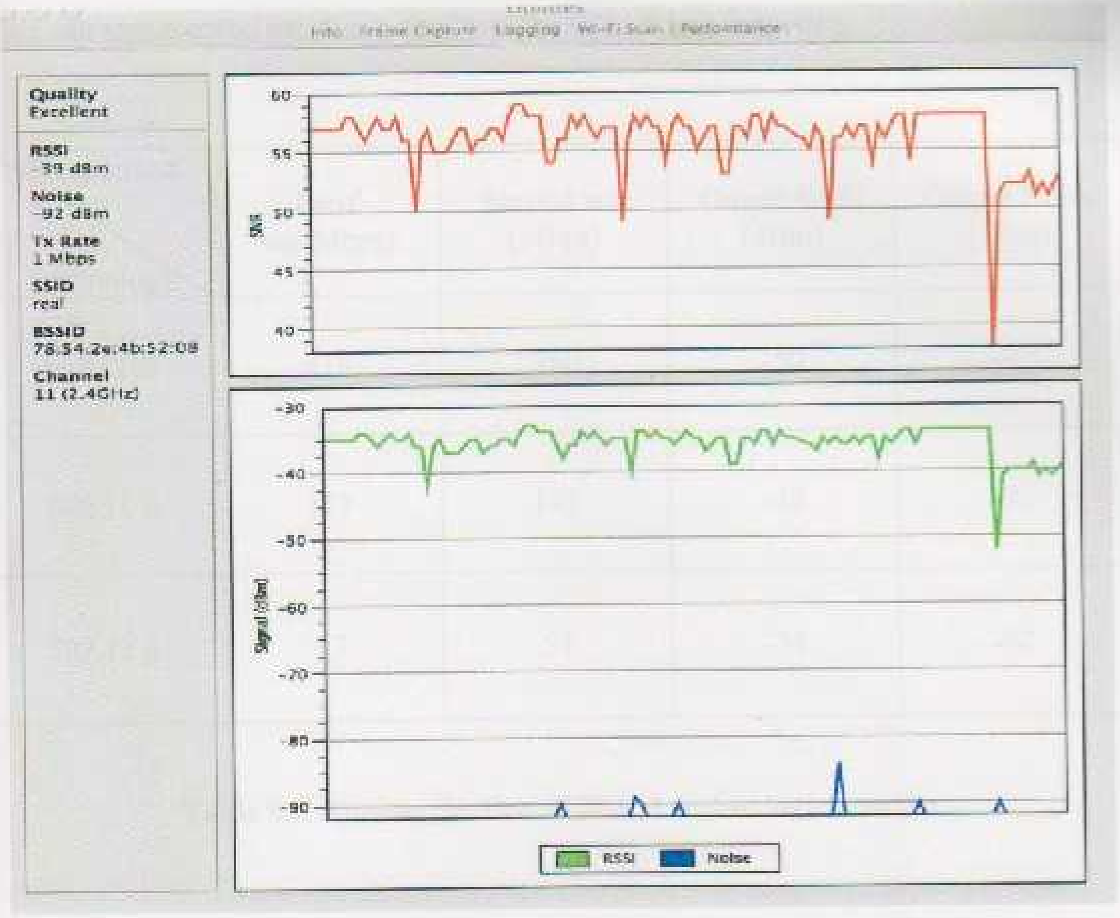
Figure 6.13 Iperf 802.11 n (2.4GHz) result

3- The third test on 802.11 g technology : 802.11 g test this test performed in two ways and the result was nearly same as shown in figure 6.14 and 6.15.



Wi-Fi Interface	
Interface Name	en0
MAC Address	b8:e8:56:30:b2:ae
Network Name	real
Active PHY Mode	802.11g
Security	Open
BSSID	78:54:2e:4b:52:08
Country Code	US
RSSI	-34 dBm
Noise	-92 dBm
Rate	54 Mbps
Channel	6 (2.4GHz)
Channel Width	20 MHz
Bluetooth	
Power State	On
Paired Device Count	0
Network Configuration	
Primary IPv4 Interface	en0
IPv4 Address	192.168.0.100
IPv4 Router	192.168.0.1
Primary IPv6 Interface	-
IPv6 Address	-
IPv6 Router	-
DNS Server	10.2.0.8

Figure 6.14 802.11 g (2.4GHz) numerical results



Quality
Excellent

RSSI
-39 dBm

Noise
-92 dBm

Tx Rate
1 Mbps

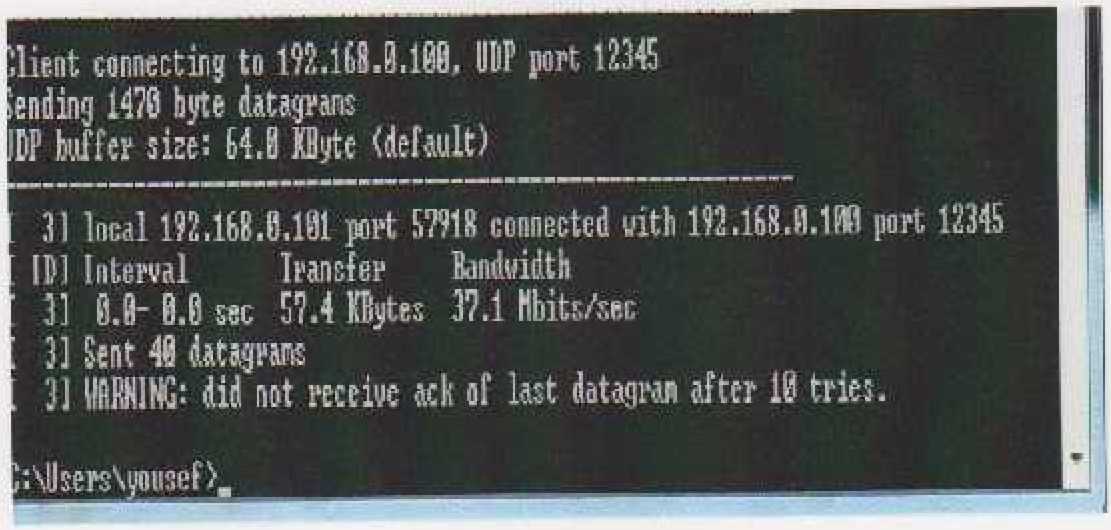
SSID
real

BSSID
78:54:2e:4b:52:08

Channel
11 (2.4GHz)

Figure 6.15 802.11 g RSSI and Noise (2.4GHz)

Iperf result was as shown in figure 6.16:



```
Client connecting to 192.168.0.100, UDP port 12345
Sending 1470 byte datagrams
UDP buffer size: 64.0 KByte (default)
------------------------------------------------------------
[  3] local 192.168.0.101 port 57918 connected with 192.168.0.100 port 12345
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 0.0 sec  57.4 KBytes  37.1 Mbits/sec
[  3] Sent 40 datagrams
[  3] WARNING: did not receive ack of last datagram after 10 tries.

C:\Users\yousef>_
```

Figure 6.16 Iperf 802.11 g (2.4GHz) result

We can summarized the above results in the table 6.1 as following:

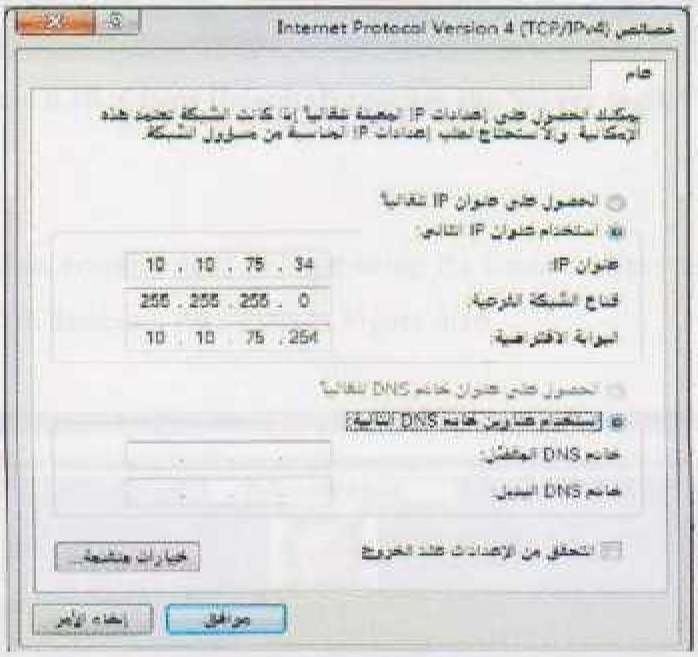
Comparison / Technology	Iperf test(Mbps)	Second test (Mbps)	Output RSSI (dBm)	Output Noise (dBm)
802.11ac	818	780	-50	-92
802.11 n	119	145	-48	-92
802.11 g	37	54	-34	-92

Table 6.1 Summarization of 802.11 technologies results

6.4 Testing and Results for two hospitals connected by GSM modems

We connect the two hospital in this project using GSM modems, in this way we use a public IP address 195.3.191.26 to be assigned to the Server computer and then clients access this server using this public IP address.

First step needed to be done is to assign the public IP address to the server computer, as shown in Figure 6.14.



خصائص Internet Protocol Version 4 (TCP/IPv4)

يمكنك الحصول على إعدادات IP المعينة تلقائياً إذا كانت الشبكة تعتمد هذه الإمكانية. وإلا ستحتاج لطلب إعدادات IP المناسبة من مسؤول الشبكة.

- الحصول على عنوان IP تلقائياً
- [✓ استخدام عنوان IP التالي:]

عنوان IP:	10 . 10 . 75 . 34
قناع الشبكة الفرعية:	255 . 255 . 255 . 0
البوابة الافتراضية:	10 . 10 . 75 . 254

- الحصول على عنوان خادم DNS تلقائياً
- [✓ استخدام عناوين خادم DNS التالية:]

خادم DNS المفضل:	. . .
خادم DNS البديل:	. . .

☐ التحقق من الإعدادات عند الخروج

Figure 6.17 Assigning IP address to server computer

Note : 10.10.75.34 IP address is the private IP address needed to be statically assigned to server computer.

After assigning IP address to need to reach this server by the client computer (hospital) this was tested and Figure 6.15 shows that client reached that server.

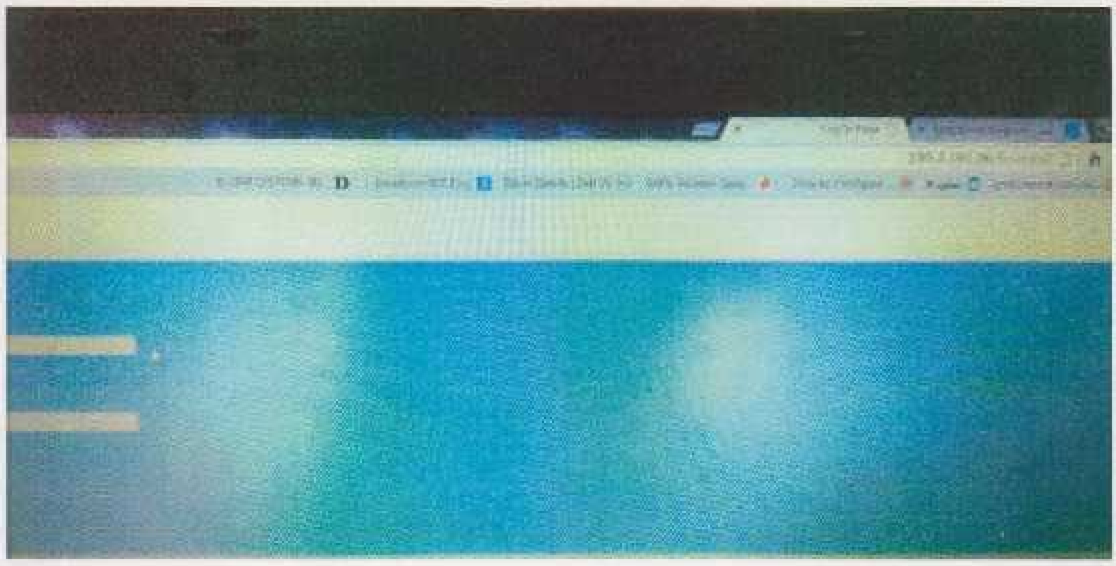


Figure 6.18 Client (hospital) reaches the Server login Page

After that the client hospital need to login using it's Email address and it's Password stored and saved in database. As shown in Figure 6.16.

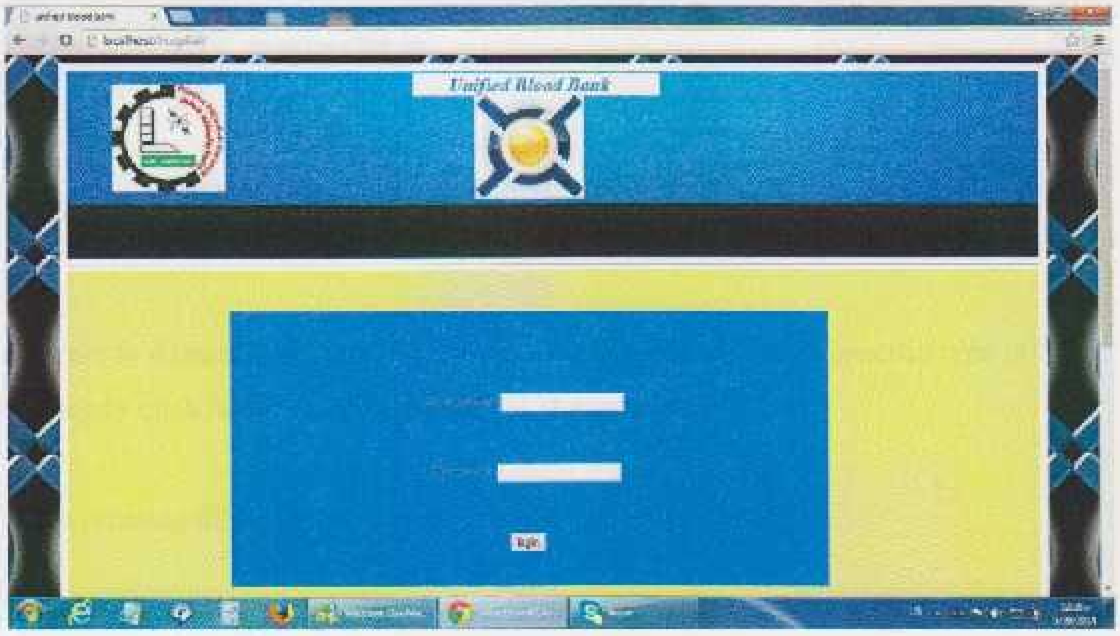
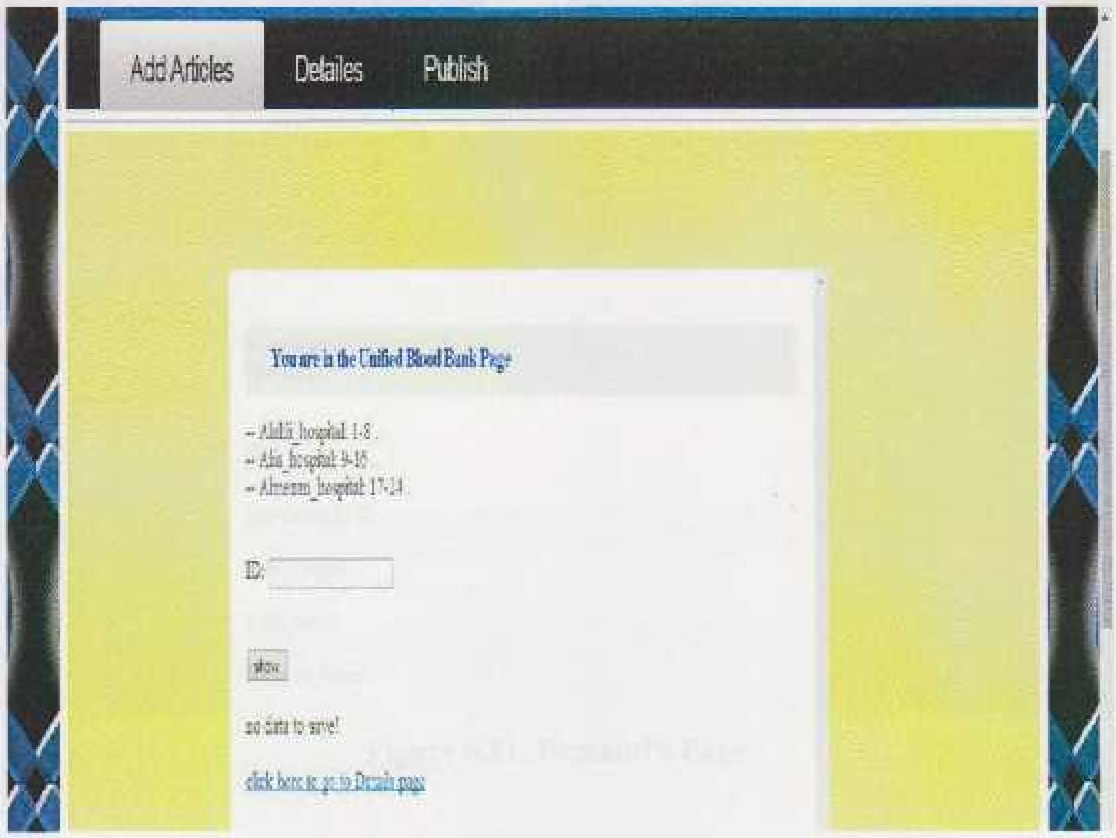


Figure 6.19 Client Login into Login server page

The next page shows ID blank box it's accept numbers between 1 – 24 , each number assigned in this box displays the hospital , type of blood and the number of units of this blood type in that hospital. Figure 6.17 shows that.



You are in the Unified Blood Bank Page

-- Aldik_hospital 1-8 .
-- Ala_hospital 9-15 .
-- Alreman_hospital 17-24 .

ID:

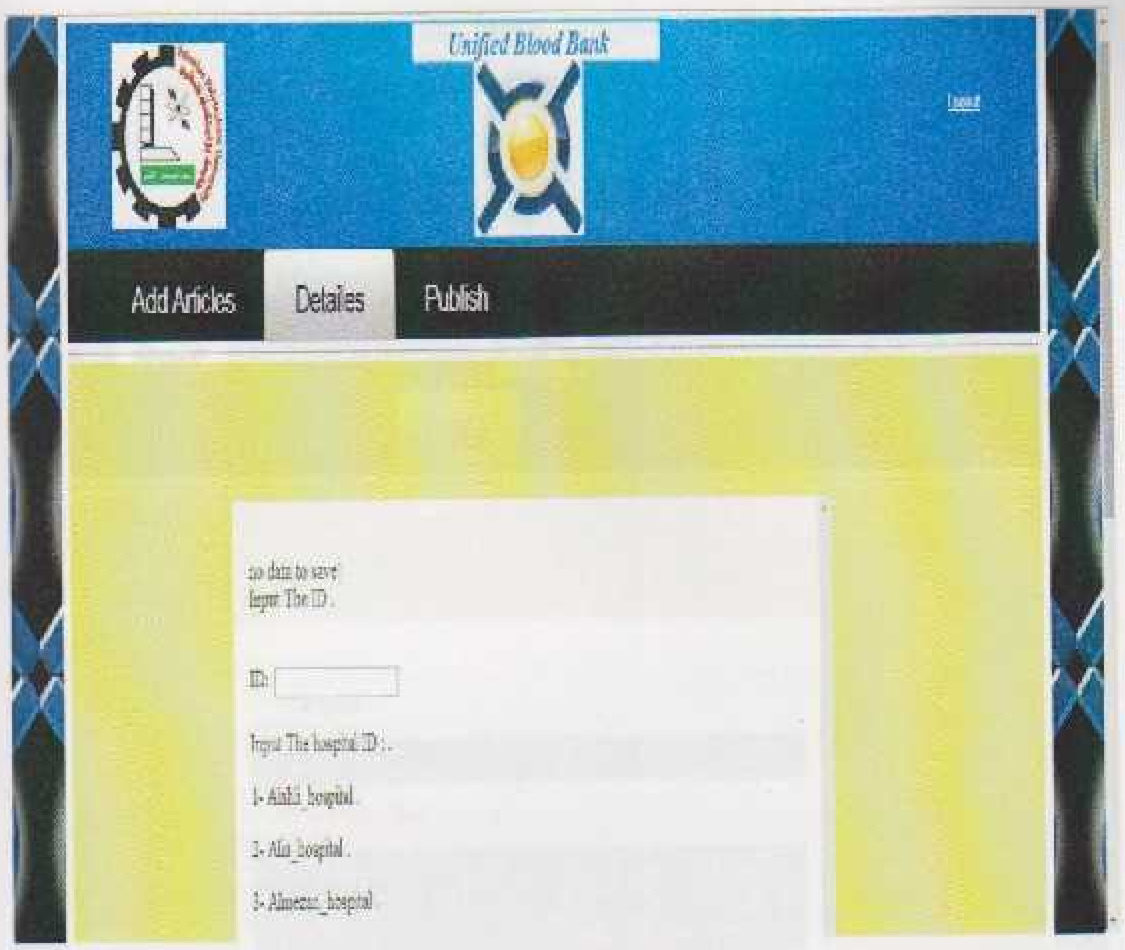
no data to save!

click here to go to Details page

Figure 6.20 General Page displays general information's

In order to demand from other hospitals a certain amount and a specific type of blood " to apply click here " must be pressed as shown above.

After pressing this link the page shown in Figure 6.18 appears.



no data to save
Input The ID .

ID:

Input The hospital ID : .

1- Alahli_hospital .

2- Alia_hospital .

3- Almezan_hospital .

Figure 6.21 Demand's Page

ID : represents the serial number of the request.

Hospital ID : represents the target hospital needed to request from.

Type of Blood ID : represents the type of blood needed.

NoOfUnits : represents the requested number of units.

After sending this request to the target hospital a notification appears at target hospital, if the hospital approve this request Published button is selected. As shown in Figure 6.19.

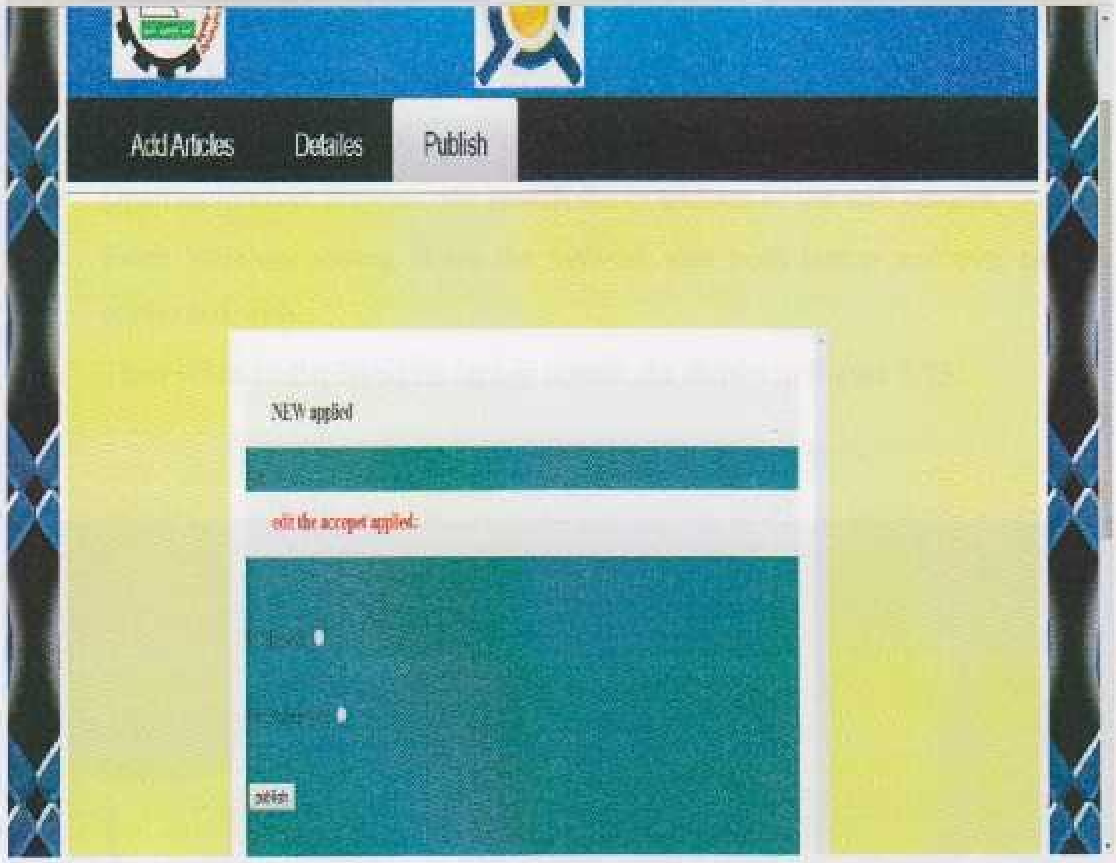


Figure 6.22 Target hospital received a notification

Target hospital received a notification contain the number of units needed, after selecting Published and pressing publish button the number of units of a specific type of blood approved by hospital will be subtracted in the database from number of blood units of that hospital.

6.5 Testing and results of live video streaming.

In this section of this chapter, we show the steps that have been down in order to display live video on laptop via wifi camera.

- First step we connect the wifi camera with the laptop to know the ip address that has been granted or given by access point .
- Then, connect both wifi camera and laptop on the same local network.
- Open an internet browser, the search the ip address of the wifi camera.
- From Wireless setting chose the network that both laptop and wifi camera connected with.
- Then video is displayed on laptop screen. As shown in Figure 6.23.

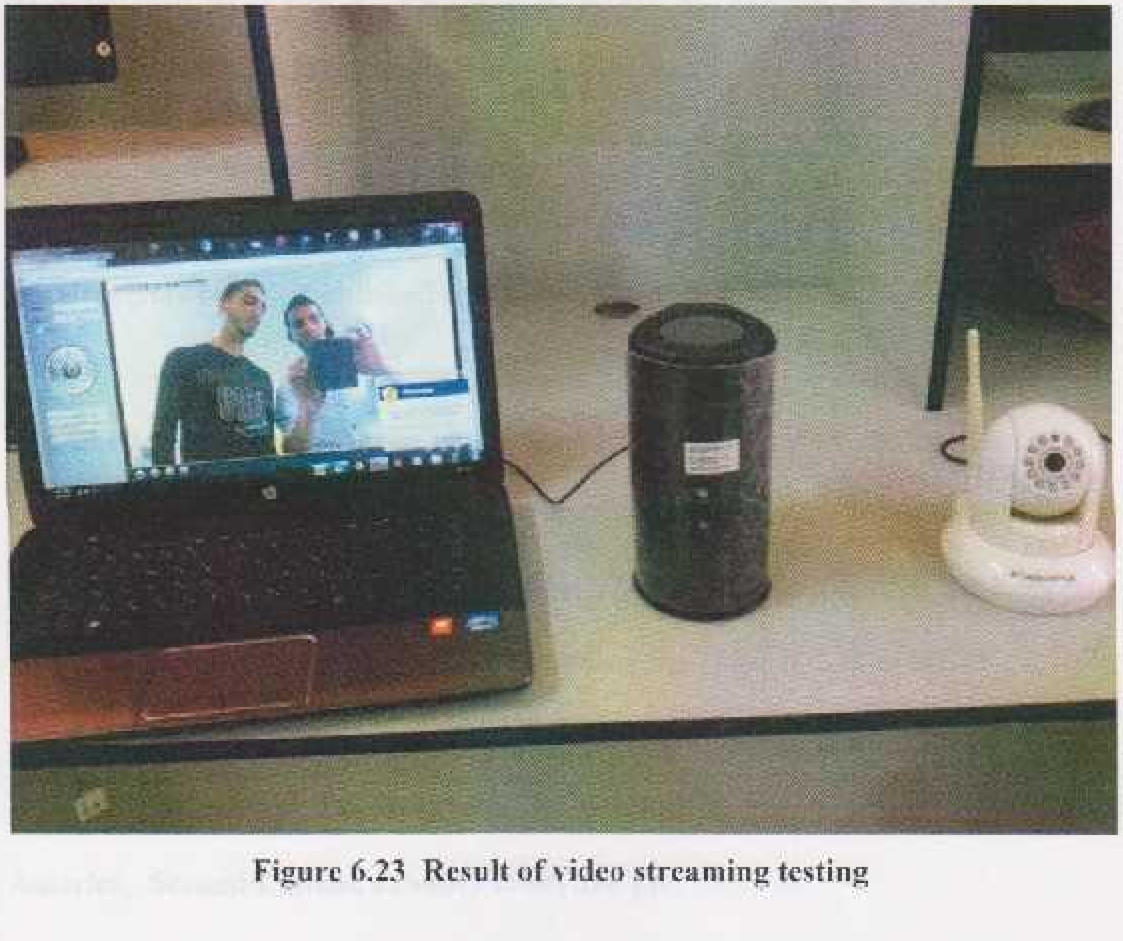


Figure 6.23 Result of video streaming testing

REFERENCES

[1] Ackerman, M., Craft, R., Ferrante, F., Kratz, M., Mandil, S., & Sapci, H., "Telemedicine & Mobile Telemedicine System An Overview " , Published in IEEE Antennas & Propagation Magazine, Volume 8, No. 1, 2002.

[2] Sodhro Ali Hassan, Ye Li , E-Health Telecommunication Systems and Networks, Published Online March 2013. Available at (http://www.scirp.org/journal/etsn).

[3] Vladimir Oleshchuk . Rune Fensli, Remote Patient Monitoring Within a Future 5G Infrastructure, published in Springer Science and Business Media, 2010.

[4] Ackerman, M., Craft, R., Ferrante, F., Kratz, M., Mandil, S., & Sapci, H., "Telemedicine & Mobile Telemedicine System An Overview " , Published in IEEE Antennas & Propagation Magazine, Volume 8, No. 1, 2002.

[5] Cheu, Y., & Ganz, A., (2005), "Mobile Telemedicine System Using 3g Wireless Networks", published in Business Briefings: US Healthcare Strategies, 2005.

[6] C.S. Pattichis , E. Kyriacou , S. Voskarides , M.S. Pattichis , R. Istepanian , C.N. Schizas , "Wireless Telemedicine Systems: An Overview", Published in IEEE Antennas & Propagation Magazine, Vol.44, No.2, pp 143-153, 2002.

[7] Rich Watson, whitepaper, "Understanding the IEEE 802.11ac Wi-Fi standard", September 2012.

[8] and [9] John Ross, The Book of Wireless, published in united state of America, Second Edition, January 2008, 336 pp.

[10] Lajos L. Hanzo, Yosef Akhtman, Li Wang, Ming Jiang, " MIMO-OFDM for LTE, WiFi and WiMAX", published in United Kingdom Jone Wiley & son's Ltd, Fist Edition, 2011 .

[11] Min Zhu, "Electrical Engineering and control", published in Nanchang, China, volume 2 ,June 2011.

[12] Subhas Chandra , Mukhopadhyay, " Sensors ", published in verlag Berline Heidelberg, 2008.

[13] Steven F. Barrett, "Arduino Microcontroller", published in morgan and clypool, 2010.

[14] Dieter Uckelmann, Bernd Scholz-Reiter, Ingrid Rügge, Bonghee Hong, Antonio Rizzi, "WiFly Shield", published in verlag Berline Heidelberg, 2012.

[15] Abbey Road, Park Royal, D-Link (Europe) Ltd., D-Link House, published in D-Link Corporation, London, 2013

CHAPTER SEVEN

RECOMMENDATIONS

These are some developmental recommendations for moving forward :

1. Connecting Wi-Fi network with the internet to monitor the patient s health fromhis home through Wi- Fi.

2. To implement more wireless communication technologies other than GSM or Wi-Fi, such as GPRS and satellite communication network.

3. Expanding the system to measure other parameters such as bloodpressure, EEG,ECG,the level of sugar in the blood and other vital signals.

4. Communicate with the patient and direct instructions by doctor from his home by using GSM modem .
5. Displaying data and video on the same displaydevice on LabView in the monitoring room.
6. Using alarm device and connect it with the microcontroller, it work when patientexposed for risk.
7. Employ some programs such as Skype to communicate with the patient.

APPENDIX A

- "SPI-UART" Initializing Code
- Code of "Wi-Fi"
- "Block Diagram of TCP/IP Receiver Program on LabView"

“SPI-UART” Initializing Code

```
#//include "WiFly.h"
#//include <SPI.h<
#include <string.h> // Required for strlen ()

//SCI16IS750 Registers
#define THR        0x00 << 3
#define RHR        0x00 << 3
#define IER        0x01 << 3
#define FCR        0x02 << 3
#define IIR        0x02 << 3
#define LCR        0x03 << 3
#define MCR        0x04 << 3
#define LSR        0x05 << 3
#define MSR        0x06 << 3
#define SPR        0x07 << 3
#define TXFIFO     0x08 << 3
#define RXFIFO     0x09 << 3
#define DLAB       0x80 << 3
#define IODIR      0x0A << 3
#define IOSTATE    0x0B << 3
#define IOINTMSK   0x0C << 3
#define IOCTRL     0x0E << 3
#define EFCR       0x0F << 3
#define DLL        0x00 << 3
#define DLM        0x01 << 3
#define EFR        0x02 << 3
#define XON1       0x04 << 3
```

```
#define XON2    0x05 << 3
#define XOFF1    0x06 << 3
#define XOFF2     0x07 << 3
//Arduino SPI pins
#define CS   10
#define MOSI      11
#define MISO      12
#define SCK     13
//Communication flags and variables
char incoming_data ؛
char TX_Fifo_Address = THR؛
char clr = 0 ؛
char polling = 0 ؛
//SC16IS750 Configuration Parameters
structSPI_UART_cfg
}
char DivL,DivM,DataFormat,Flow ؛
؛{
structSPI_UART_cfgSPI_Uart_config } =
· x60,0x00,0x03,0x10 ؛{
void setup ()
}
// Initialize SPI pins
pinMode(MOSI, OUTPUT ؛(
pinMode(MISO, INPUT ؛(
pinMode(SCK,OUTPUT ؛(
pinMode(CS,OUTPUT ؛(
digitalWrite(CS,HIGH); //disable device
SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR1)|(1<<SPR0 ؛(
clr=SPSR ؛
clr=SPDR ؛
delay(10 ؛(
Serial.begin(9600 ؛(
Serial.println("\n\r\n\rWiFly Shield Terminal Routine ؛("
 // Test SPI communication
if(SPI_Uart_Init }(()
Serial.println("Bridge initialized successfully ؛("!
 {
else }
Serial.println("Could not initialize bridge, locking up.\n\r ؛("
while(1 ؛(
{
{

void loop ()
}
// Poll for new data in SC16IS750 Recieve buffer
if(SPI_Uart_ReadByte(LSR) & 0x01 (
 }
polling = 1 ؛
while(polling (
}
if((SPI_Uart_ReadByte(LSR) & 0x01 ((
}
incoming_data = SPI_Uart_ReadByte(RHR ؛(
Serial.print(incoming_data,BYTE ؛(
  {
else
 }
polling = 0 ؛
 {
 {
 {
 // Otherwise, send chars from terminal to SC16IS750
else if(Serial.available (()
 }
incoming_data = Serial.read ؛()
```

```
select ;()
spi_transfer(0x00); // Transmit command
spi_transfer(incoming_data ;(
deselect ;()
{
{

char SPI_Uart_Init(void  (
//Initialize SC16IS750
}
char data = 0 ;
SPI_Uart_WriteByte(LCR,0x80); // 0x80 to program baudrate
SPI_Uart_WriteByte(DLL,SPI_Uart_config.DivL); //0x60 = 9600
with Xtal = 12.288MHz
SPI_Uart_WriteByte(DLM,SPI_Uart_config.DivM  ;(
SPI_Uart_WriteByte(LCR, 0xBF); // access EFR register
SPI_Uart_WriteByte(EFR, SPI_Uart_config.Flow); // enable
enhanced registers
SPI_Uart_WriteByte(LCR, SPI_Uart_config.DataFormat); // 8 data
bit, 1 stop bit, no parity
SPI_Uart_WriteByte(FCR, 0x06); // reset TXFIFO, reset RXFIFO,
non FIFO mode
SPI_Uart_WriteByte(FCR, 0x01); // enable FIFO mode
// Perform read/write test to check if UART is working
SPI_Uart_WriteByte(SPR,'H ;('
data = SPI_Uart_ReadByte(SPR ;(
if(data == 'H  }('
return 1  ;
{
else  }
return 0  ;
{
}
```

SPI_Uart_WriteByte(char address, char data (
Write byte to register address on SC16IS750

```
 }
long int length ؛
char senddata[2 ؛[
senddata[0] = address ؛
senddata[1] = data ؛
select ؛()
length = SPI_Write(senddata, 2 ؛(
deselect ؛()
 {
long intSPI_Write(char* srcptr, long int length (
 //Send entire string to SC16IS750
 }
for(long int i = 0; i < length; i (++
 }
spi_transfer(srcptr[i ؛([
 {
return length ؛
{
void SPI_Uart_WriteArray(char *data, long intNumBytes (
 //Send entire string to THR of SC16IS750
 }
long int length ؛
select ؛()
length = SPI_Write(&TX_Fifo_Address,1 ؛(
while(NumBytes> 16) // Split array into 16 character chunks
  }
length = SPI_Write(data,16 ؛(
NumBytes -= 16 ؛
data += 16 ؛
 {
length = SPI_Write(data,NumBytes ؛(
deselect ؛()
 {
char SPI_Uart_ReadByte(char address (
 //Read from SC16IS750 register
```

```
}
char data ؛
address = (address | 0x80 ؛(
select ؛()
spi_transfer(address ؛(
data = spi_transfer(0xFF ؛(
deselect ؛()
return data   ؛
 {
void SPI_Uart_println(char *data (
 //Write string to SC16IS750 followed by a carriage return
 }
SPI_Uart_WriteArray(data,strlen(data ؛((
SPI_Uart_WriteByte(THR, 0x0d ؛(
 {
void SPI_Uart_print(char *data  (
 //Write string to SC16IS750, no carriage return
 }
SPI_Uart_WriteArray(data,strlen(data ؛((
 {
char spi_transfer(volatile char data (
 }
SPDR = data;                    // Start the transmission
while (!(SPSR & (1<<SPIF)))     // Wait for the end of the
transmission
 }
 ؛{
return SPDR;                    // return the received byte
{

void select(void  (
 }
digitalWrite(CS,LOW ؛(
 {
void deselect(void (
 }
```

```
digitalWrite(CS,HIGH ؛(
{
```

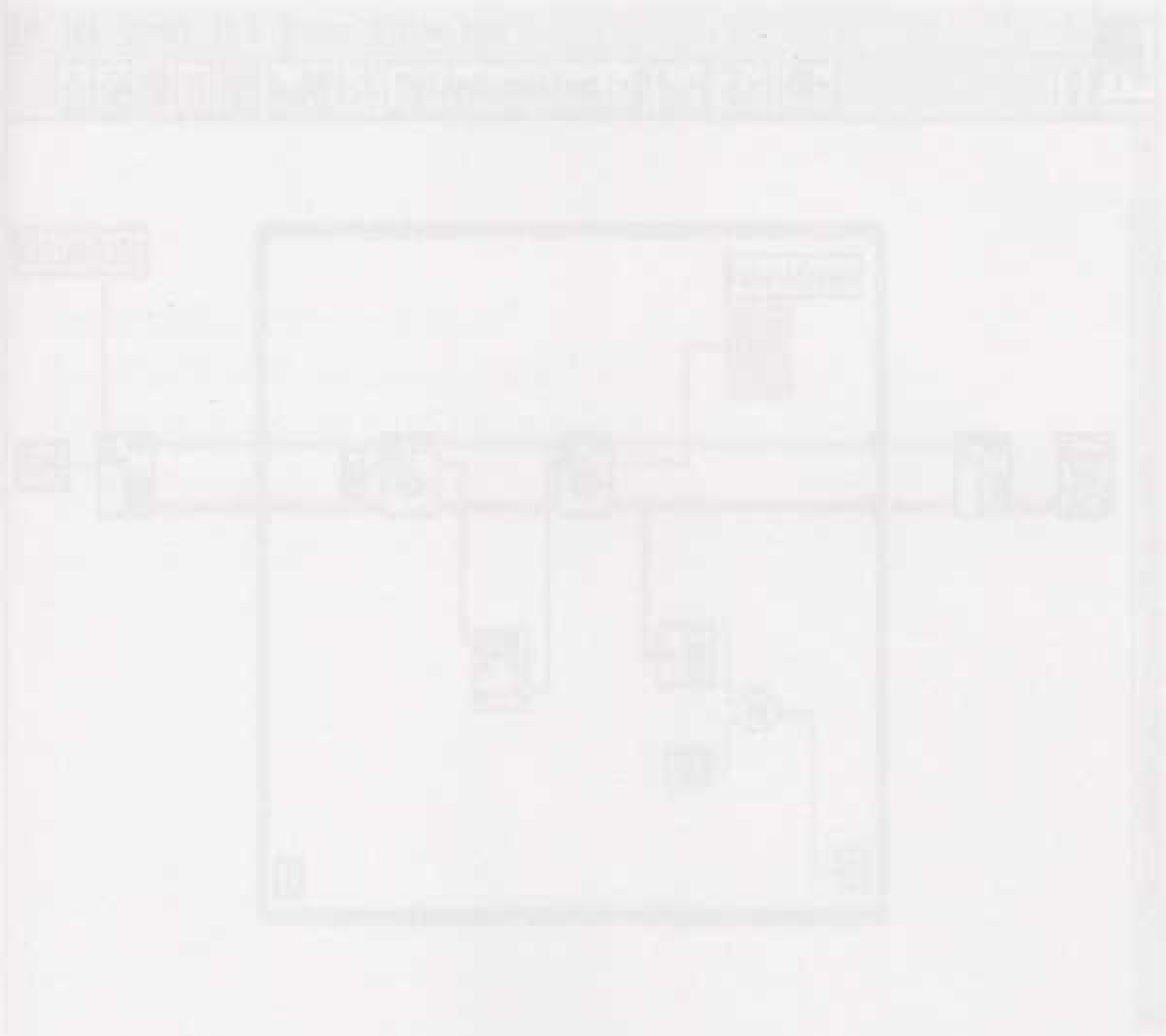
Code of “Wi-Fi”

```
//include the WiFly code to communicate with SpiSerial:
#include <Client.h<
#include <ParsedStream.h<
#include <Server.h<
#include <SpiUart.h<
#include <WiFly.h<
#include <WiFlyDevice.h<
#include <_Spi.h<

long previousMillis = 0;        // will store last time
long interval = 60000;          // interval  (milliseconds (
unsigned long currentMillis = 0 ؛
float temp ؛
float val=0 ؛
intBreath_rate ؛
int Pin0 = 0 ؛
int Pin1=2 ؛
const char AUTH_LEVEL [] = "0؛"
//const char auth_phrase[] = "mohammad؛"
const char SSID [] = "real ؛"
const char DHCP [] = "0"; //0=static (DHCP disabled  (
const char WIFLY_IP [] = "192.168.0.101 ؛"
const char UART_MODE [] = "2  ؛"
const char WIFLY_PORT [] = "2000 ؛"
const char PC_IP [] = "192.168.0.100 ؛"
const char PC_PORT [] = "2000 ؛"
const char NETMASK [] = "255.255.255.0 ؛"
const char CHANNEL []="11 ؛"

//displays on terminal everything in SPI Serial buffer
void readFromWifly }()
delay(100  ؛(
while(SpiSerial.available() > 0 } (
Serial.print(SpiSerial.read(), BYTE ؛(
 {
Serial.print("\n ؛("
delay(100 ؛(
 {

void setup } ()
Serial.begin(9600 ؛(
SpiSerial.begin؛()

 //Exit command mode if we haven't already
SpiSerial.println  ؛("")
SpiSerial.println("exit ؛("
readFromWifly ؛()

  //Enter command mode
SpiSerial.print ؛("$$$")
delay(100 ؛(

 //Reboot to get device into known state
SpiSerial.println("reboot ؛("
readFromWifly ؛()
delay(3000 ؛(

 //Enter command mode
Serial.println("\nEntering command mode ؛(".
SpiSerial.print ؛("$$$")
readFromWifly ؛()
delay(500 ؛(
 //Set authorization level to <auth_level<
SpiSerial.print("set wlanauth ؛("
SpiSerial.println(AUTH_LEVEL  ؛(
readFromWifly ؛()
delay(500؛(
//SpiSerial.print("set wlan phrase؛("
//SpiSerial.println(auth_phrase؛(
//readFromWifly؛()

//set UART mode to make TCP/IP connection
SpiSerial.print("set uart mode ؛("
SpiSerial.println(UART_MODE ؛(
readFromWifly ؛()
delay(500 ؛(

 //Set wlan channel to 11
SpiSerial.print("set wlan channel ؛("
SpiSerial.println(CHANNEL  ؛(
readFromWifly ؛()
delay(500 ؛(

//disable DHCP
SpiSerial.print("set ipdhcp ؛("
SpiSerial.println(DHCP ؛(
readFromWifly ؛()
delay(500 ؛(

//set WiFly IP Address
SpiSerial.print("set ip address ؛("
SpiSerial.println(WIFLY_IP ؛(
readFromWifly ؛()
delay(500؛(

//set WiFly remote port
SpiSerial.print("set iplocal_port ؛("
SpiSerial.println(WIFLY_PORT  ؛(
readFromWifly ؛()
delay(500 ؛(

//set PC IP Address
SpiSerial.print("set ip host ؛("
SpiSerial.println(PC_IP ؛(
readFromWifly ؛()
delay(500 ؛(

//Set PC remote port
SpiSerial.print("set ipremote_port ؛("
SpiSerial.println(PC_PORT ؛(
readFromWifly ؛()
delay(500 ؛(

//set Netmask Address
SpiSerial.print("set ipnetmask ؛("
SpiSerial.println(NETMASK ؛(
readFromWifly ؛()
delay(500 ؛(

 //Join wifi network <ssid<
SpiSerial.flush ؛()
SpiSerial.print("join ؛("
SpiSerial.println(SSID  ؛(
delay(5000  ؛(
readFromWifly ؛()

//save config
SpiSerial.println("save ؛("

 //exit from command mode
SpiSerial.println("exit ؛("
readFromWifly ؛()
 {
```

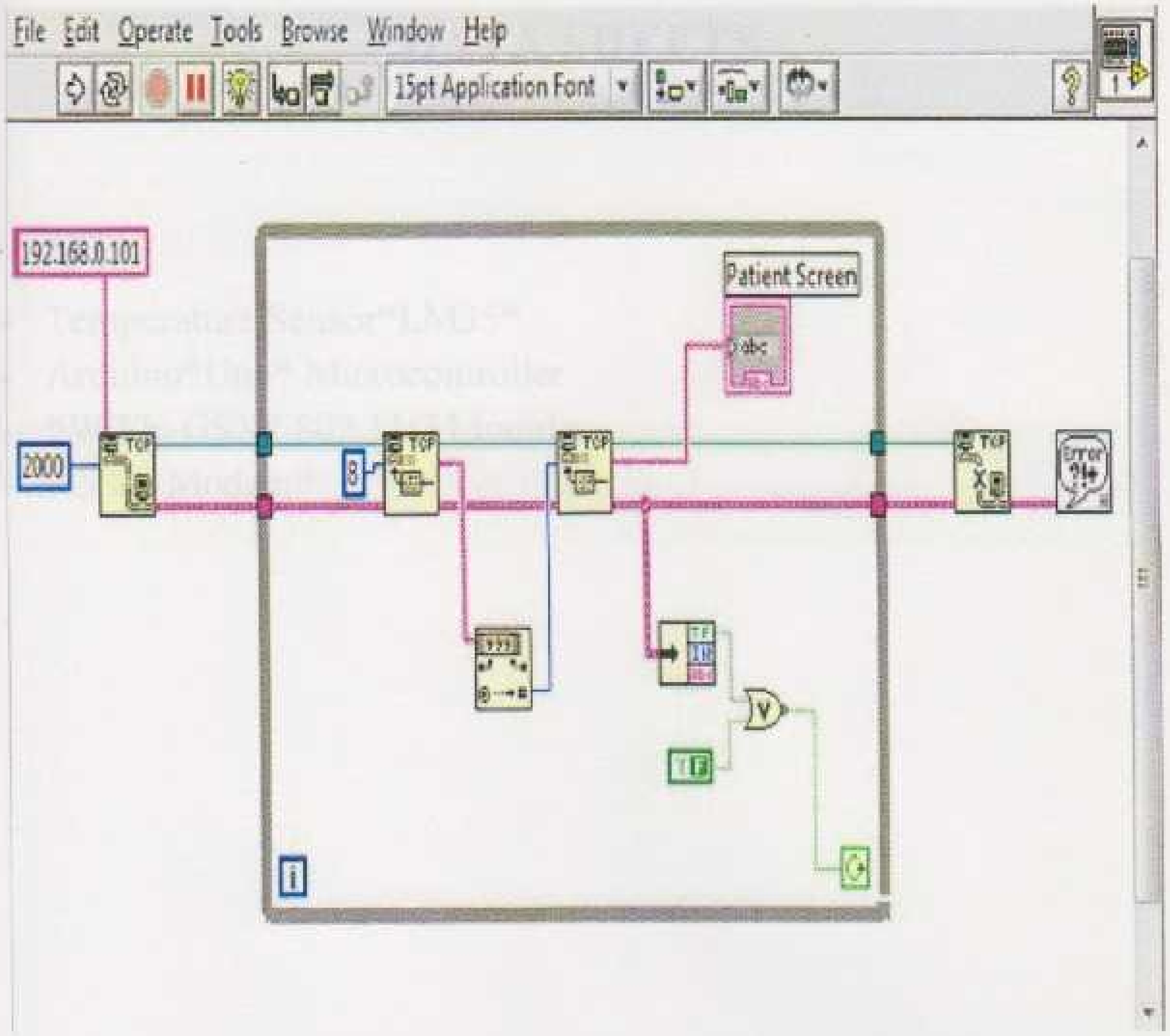
```
void loop ()
  }
temp = analogRead(Pin0          ؛(
temp = (500.0 * temp)/1024.0 ؛
Breath_rate=0 ؛
if(currentMillis - previousMillis< interval (
 }
unsigned long currentMillis = millis ؛()
float val1 = analogRead(Pin1  ؛(
val1=val1*5.0/1023.0  ؛
if(val1 > 4.0 (
}
Breath_rate=Breath_rate+1 ؛
{
delay(3000؛(
if(currentMillis - previousMillis> interval (
 }
previousMillis = currentMillis    ؛
Serial.println(Breath_rate ؛(
 {
 {
// if(temp < 36.4|temp > 37.8 )//)&Breath_rate<15|Breath_rate>18 (
  }//
SpiSerial.print("temprature is ؛("
SpiSerial.println(temp؛(
 //SpiSerial.println("deg C ؛("
SpiSerial.print("Breath rate is ؛(":
SpiSerial.print(Breath_rate ؛(
 {//
// else if(temp<36.4|temp>37.8 (
  }//
// SpiSerial.print("Temprature is ؛(" :
// SpiSerial.print(temp ؛(
  {//
```

```
//else if(Breath_rate>15|Breath_rate>18 (
  }//
// SpiSerial.print("Breath rate is :(":
// SpiSerial.print(Breath_rate :(
 {//
delay(450000 :(
interval=510000:
{
```

"Block Diagram of TCP/IP Receiver Program (LabView)"



"Block Diagram of TCP/IP Receiver Program (LabView) "



APPENDIX B

DATA SHEETS

- Temperature Sensor“LM35“
- Arduino“Uno“ Microcontroller
- “WiFly GSX“ 802.11G Module
- “GSM Modem“

"LM35" Temperature Sensor Datasheet

National Semiconductor

November 2000

LM35
Precision Centigrade Temperature Sensors

General Description

The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 thus has an advantage over linear temperature sensors calibrated in ° Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling. The LM35 does not require any external calibration or trimming to provide typical accuracies of ±¼°C at room temperature and ±¾°C over a full −55 to +150°C temperature range. Low cost is assured by trimming and calibration at the wafer level. The LM35's low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies, or with plus and minus supplies. As it draws only 60 μA from its supply, it has very low self-heating, less than 0.1°C in still air. The LM35 is rated to operate over a −55° to +150°C temperature range, while the LM35C is rated for a −40° to +110°C range (−10° with improved accuracy). The LM35 series is available packaged in hermetic TO-46 transistor packages, while the LM35C, LM35CA, and LM35D are also available in the plastic TO-92 transistor package. The LM35D is also available in an 8-lead surface mount small outline package and a plastic TO-220 package.

Features

- Calibrated directly in ° Celsius (Centigrade)
- Linear + 10.0 mV/°C scale factor
- 0.5°C accuracy guaranteeable (at +25°C)
- Rated for full −55° to +150°C range
- Suitable for remote applications
- Low cost due to wafer-level trimming
- Operates from 4 to 30 volts
- Less than 60 μA current drain
- Low self-heating, 0.08°C in still air
- Nonlinearity only ±¼°C typical
- Low impedance output, 0.1 Ω for 1 mA load

Typical Applications

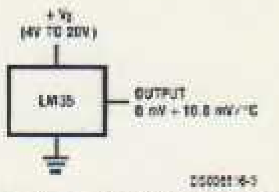
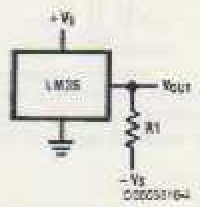


FIGURE 1. Basic Centigrade Temperature Sensor (+2°C to +150°C)



Choose $R_1 = -V_S/50\ \mu A$

V_{OUT} = +1,500 mV at +150°C

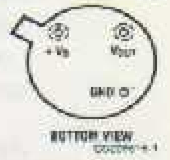
= +250 mV at +25°C

= −550 mV at −55°C

FIGURE 2. Full-Range Centigrade Temperature Sensor

tion Diagrams

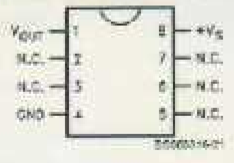
TO-46
Metal Can Package*



ed to negative pin (GND)

er LM35H, LM35AH, LM35CH, LM35CAH or LM35DH
See NS Package Number H03H

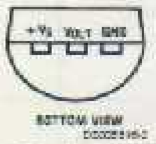
SO-8
Small Outline Molded Package



N.C. = No Connection

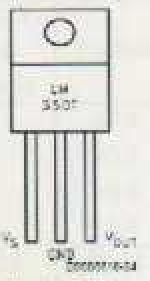
Top View
Order Number LM35DM
See NS Package Number M08A

TO-92
Plastic Package



Order Number LM35CZ, LM35CAZ or LM35DZ
See NS Package Number Z03A

TO-220
Plastic Package*



*Tab is connected to the negative pin (GND).
Note: The LM35DT pinout is different than the discontinued LM35DP

Order Number LM35DT
See NS Package Number TA03F

Absolute Maximum Ratings (Note 10)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

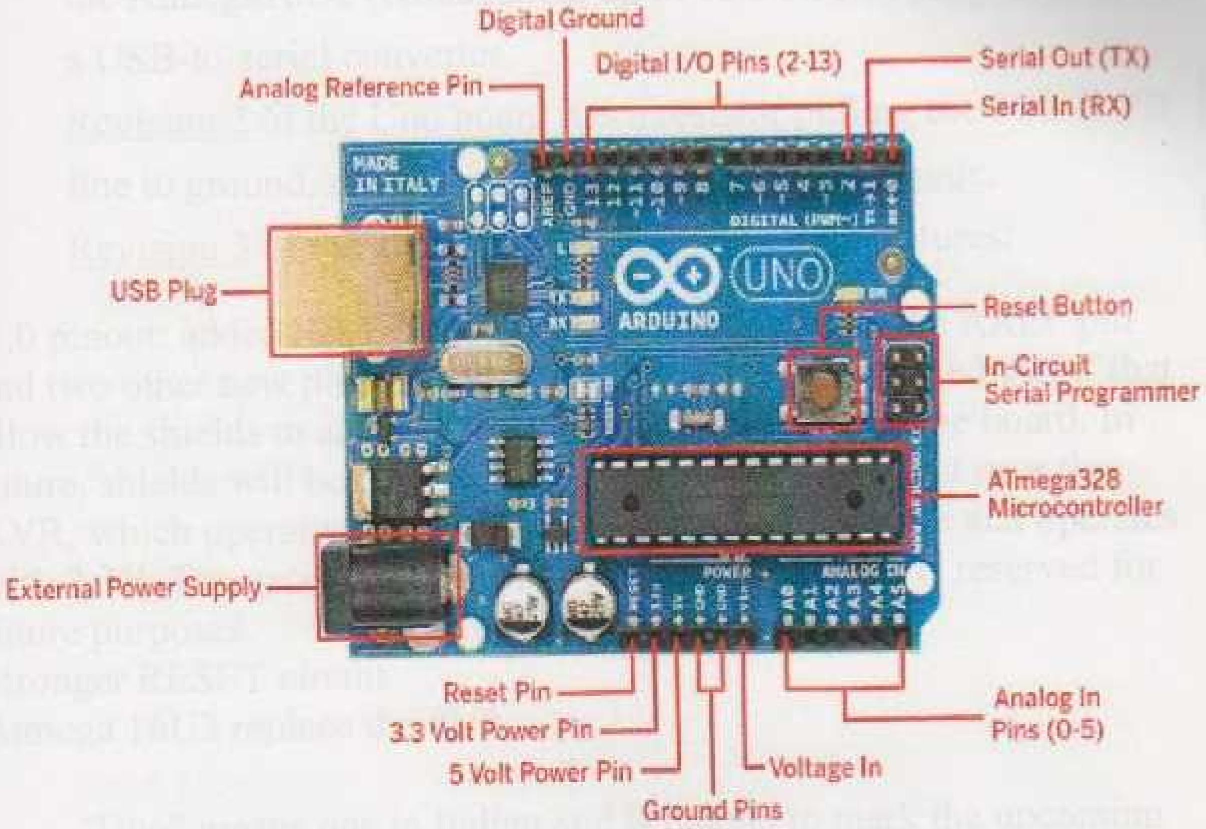
Supply Voltage	+35V to −0.2V
Output Voltage	+6V to −1.0V
Output Current	10 mA
Storage Temp.;	
TO-46 Package,	−60°C to +180°C
TO-92 Package,	−60°C to +150°C
SO-8 Package,	−65°C to +150°C
TO-220 Package,	−65°C to +150°C
Lead Temp.:	
TO-46 Package, (Soldering, 10 seconds)	300°C
TO-92 and TO-220 Package, (Soldering, 10 seconds)	260°C
SO Package (Note 12)	
Vapor Phase (60 seconds)	215°C
Infrared (15 seconds)	220°C
ESD Susceptibility (Note 11)	2500V
Specified Operating Temperature Range: T_{MIN} to T_{MAX} (Note 2)	
LM35, LM35A	−55°C to +150°C
LM35C, LM35CA	−40°C to +110°C
LM35D	0°C to +100°C

Electrical Characteristics

(Notes 1, 6)

Parameter	Conditions	LM35A			LM35CA			Units (Max.)
		Typical	Tested Limit (Note 4)	Design Limit (Note 5)	Typical	Tested Limit (Note 4)	Design Limit (Note 5)	
Accuracy (Note 7)	T_A=+25°C	±0.2	±0.5		±0.2	±0.5		°C
	T_A=−10°C	±0.3			±0.3		±1.0	°C
	T_A=T_{MAX}	±0.4	±1.0		±0.4	±1.0		°C
	T_A=T_{MIN}	±0.4	±1.0		±0.4		±1.5	°C
Nonlinearity (Note 8)	T_{MIN}≤T_A≤T_{MAX}	±0.18		±0.35	±0.15		±0.3	°C
Sensor Gain (Average Slope)	T_{MIN}≤T_A≤T_{MAX}	+10.0	+9.9, +10.1		+10.0		+9.9, +10.1	mV/°C
Load Regulation (Note 3) 0≤I_L≤1 mA	T_A=+25°C	±0.4	±1.0		±0.4	±1.0		mV/mA
	T_{MIN}≤T_A≤T_{MAX}	±0.5		±3.0	±0.5		±3.0	mV/mA
Line Regulation (Note 3)	T_A=+25°C	±0.01	±0.05		±0.01	±0.05		mV/V
	4V≤V_S≤30V	±0.02		±0.1	±0.02		±0.1	mV/V
Quiescent Current (Note 9)	V_S=+5V, +25°C	56	67		56	67		μA
	V_S=+5V	105		131	91		114	μA
	V_S=+30V, +25°C	56.2	68		56.2	68		μA
	V_S=+30V	105.5		133	91.5		116	μA
Change of Quiescent Current (Note 3)	4V≤V_S≤30V, +25°C	0.2	1.0		0.2	1.0		μA
	4V≤V_S≤30V	0.5		2.0	0.5		2.0	μA
Temperature Coefficient of Quiescent Current		+0.39		+0.5	+0.39		+0.5	μA/°C
Minimum Temperature for Rated Accuracy	In circuit of *Figure 1*, I_L=0	+1.5		+2.0	+1.5		+2.0	°C
Long Term Stability	T_J=T_{MAX}, for 1000 hours	±0.08			±0.08			°C

Arduino"Uno" Microcontroller Data sheet



Overview

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

Revision 2 of the Uno board has a resistor pulling the 8U2 HWB line to ground, making it easier to put into DFU mode.

Revision 3 of the board has the following new features:

- 1.0 pinout: added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible with both the board that uses the AVR, which operates with 5V and with the Arduino Due that operates with 3.3V. The second one is a not connected pin, that is reserved for future purposes.
- Stronger RESET circuit.
- Atmega 16U2 replace the 8U2.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards.

Summary

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Schematic & Reference Design

EAGLE files: arduino-uno-Rev3-reference-design.zip (NOTE: works with Eagle 6.0 and newer)

Schematic: arduino-uno-Rev3-schematic.pdf

Note: The Arduino reference design can use an Atmega8, 168, or 328, Current models use an ATmega328, but an Atmega8 is shown in the schematic for reference. The pin configuration is identical on all three processors.

Power

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- VIN. The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- 5V.This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.

- 3V3. A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- GND. Ground pins.
- IOREF. This pin on the Arduino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs for working with the 5V or 3.3V.

Memory

The ATmega328 has 32 KB (with 0.5 KB used for the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), anddigitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function.
- SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.

- LED: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. Additionally, some pins have specialized functionality:

- TWI: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.

There are a couple of other pins on the board:

- AREF. Reference voltage for the analog inputs. Used with analogReference().
- Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

See also the mapping between Arduino pins and ATmega328 ports. The mapping for the Atmega8, 168, and 328 is identical.

Communication

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed.

However, on Windows, a .inf file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows for serial communication on any of the Uno's digital pins.

The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details. For SPI communication, use the SPI library.

Programming

The Arduino Uno can be programmed with the Arduino software (download). Select "Arduino Uno from the Tools > Board menu (according to the microcontroller on your board). For details, see the reference and tutorials.

The ATmega328 on the Arduino Uno comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.

The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available . The ATmega16U2/8U2 is loaded with a DFU bootloader, which can be activated by:

- On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2.
- On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode.

You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader). See this user-contributed tutorial for more information.

Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of theATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the

first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.
The Uno contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

USB Overcurrent Protection

The Arduino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics

The maximum length and width of the Uno PCB are 2.7 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Four screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

"WiFly GSX" 802.11G Module Data sheet

ROVING NETWORKS
WIRELESS FOR LESS

WiFly GSX

WiFly GSX

802.11 b/g wireless LAN Modules

User Manual and Command Reference

RN-131G, RN-131C, RN-134, RN-121, RN-123 & RN-125

Version 2.21
July 11th, 2010

2. Hardware Interface

See the specific module data sheets on the Roving Networks website for hardware specifications, and layo information.

2.1. Power

There are two options for powering the RN-131G module directly.

DC SUPPLY:

Apply 3.3 VDC power to VBATT (pin 20), and V3.3IN (pin 21).
Tie 3.3VREG-IN (pin 18) to GROUND.
Leave 3.3V-REG-OUT (Pin 17) floating/no connect.

BATTERY:

Apply battery = 2.0 to 3.3VDC to VBATT (pin 20).
Leave V3.3IN pin 21 floating/no connect.
Tie pin 17 to pin 18. (This enables the on board battery boost 3.3V switcher).

There is a built-in brownout monitor which will shut down the chip when the voltage drops below 2.0 VD

WARNING: Do NOT exceed the voltage ratings damage to the module will result.

#1: The Sensor inputs SENS0-7 are extremely sensitive to over voltage. Under no conditions shou these pins be driven above 1.2VDC. Placing any voltage above this will permanently damage the radio module and render it useless.

#2: Placing 5VDC or any voltage above 3.3Vdc into the VDD pins of the module will permanently damage the radio module.

#3: Placing 3.3Vdc into the PIO's while they are set as outputs will permanently damage the module. The failure mode is a short across GND and VCC.

2.2. Reset

Reset is active LOW and is optional/does not need to be connected. The reset pin is 3.3V tolerant and ha an internal pull up of 100K to the VBATT.

2.3. UART

Connect a common ground when using the external TX, RX inputs.

For a 3 wire DB-9 interface (connect TX, RX, GND only)

Factory default is hardware flow control disabled; CTS and RTS are not required.

PIO's are not 5.0 VDC tolerant. If using a 5.0 VDC circuit, input, PIO and UART input pins require a resistor divider. A suggestion is to use a 10K resistor in series with 20k resistor to ground.

2.4. Status Indictors

PIO 4, 5 and 6 are active high and can be connected to external LEDs to provide network, connection and data status.

Condition	PIO6=Red LED	PIO5=Yellow LED	PIO4=Green LED
ON solid			Connected over TCP
Fast blink	Not Associated	Rx/Tx data transfer	No IP address
Slow blink			IP address OK
OFF	Associated		

3. Configuration

3.1. Entering Command Mode

Upon power up, the device will be in data mode. To enter command mode, exactly the three characters $$$ must be sent. The device will respond with CMD.

While in command mode, the device will accept ASCII bytes as commands.
To exit command mode, send **exit**<cr>. The device will respond with "EXIT".

Parameters, such as the SSID, channel, IP address, Serial Port settings, and all other settings can be viewed and configured in command mode.

ASCII characters can be sent through a terminal emulator connected to the UART or via Telnet. When using the UART communications settings should match the settings used when RN-131g connects, for example: the default is 9600 baudrate, 8 bits, No Parity, 1 stop bit, and hardware flow control disabled.

Use TeraTerm as your terminal emulator. Please **DO NOT** use HyperTerminal as it is known to have issues with our products. TeraTerm can be downloaded from our website:
http://www.rovingnetworks.com/support/teraterm.zip

Type **$$$** on in the terminal emulator. You should see "CMD" returned to you. This will verify that your cable and comm. settings are correct. Most valid commands will return an "**AOK**", response, and invalid ones will return an "**ERR**" description.

To exit command mode, type "exit"<cr>.

NOTE: You can enter command mode locally over the UART interface at any time when not connected, and also when connected if the appropriate settings are enabled.

Remote configuration using ADHOC mode

Using adhoc mode to configure the device eliminates the need for the module to be associated with a network access point. In adhoc mode the module creates it own "on demand" network that you can connect to via your computer like you would to any other network.

To enable adhoc mode via hardware set *PIO9* high (3.3V) at power up. On the RN-134 PIO9 is on the J1 jumper block. When the module powers up with PIO9 set high, the WiFly module creates an adhoc network with the following

SSID:	WiFly-GSX-XX where XX is the final two bytes of the devices MAC address
Channel:	1
DHCP:	OFF
IP address:	169.254.1.1
Netmask:	255.255.0.0

With the adhoc jumper in place the above settings override the current saved configuration settings.

From your computer, connect to the WiFly-GSX-XX network. This is an open network which does not require a pass phrase or pass key. Note: currently the WiFly only supports OPEN mode for creating adhoc networks.

NOTE: It may take a couple of minutes for Auto IP in Windows to assign an IP address and connect to the network. You can check IP address of your Windows computer by running the *ipconfig* command in the command window. If connected, this command will show you the IP address and net mask for your computer.

The IP address assigned by Auto IP must be on the subnet 169.254.x.y otherwise the WiFly GSX module will not be accessible.

NOTE: If your machine has both a wireless and wired interface hardware you will need to disable the wired LAN interface hardware before connecting to the adhoc network. If the wired LAN is enabled the computer will assign an IP address that is not on the same subnet as the WiFly module.

Once connected and you have a good IP address, telnet into the WiFly module on port 2000

telnet 169.254.1.1 2000

You should see the response "*HELLO*"

You can now enter command mode and configure the module.

4. WiFly Command Reference

4.1. Command Syntax

Commands begin with a keyword, and have optional additional parameters, generally space delimited. Commands and options are case sensitive. Hex input data can be upper or lower case. String text data, such as SSID is also case sensitive.

The first command is fully decoded and must be complete. Other command parameters can be shorted by using only the first character.

For example,

set uart baudrate 115200 is valid,

set uart b 115200 is also valid,

set u b 115200 is also valid, however,

s uart baudrate 115200 is <u>NOT</u> valid.

Numbers can be entered as either decimal, (like 115200 above) or HEX. To enter HEX, use **0x<value>**. For example, the HEX value FF would be entered as 0xFF.

4.2. Command Organization

Commands fall into 5 general categories:

SET COMMANDS	Take effect immediately, permanently (save command issued).
GET COMMANDS	Retrieve the permanently stored information for display to user.
STATUS COMMANDS	See what is going on with the interface, IP status, etc.
ACTION COMMANDS	Perform action such as scan, connect, disconnect, etc.
FILE IO COMMANDS	Upgrade, load and save configuration, delete files, etc.

NOTE: You must save any changes made or the module will load the previous settings upon reboot or power up.

When the system boots, all configuration data is loaded into RAM variables from the file called "config". The set commands actually only modify the RAM copy of variables in the system. In general, the IP, WLAN and UART settings need a save and reboot to take effect, since they operate at boot up time. For example you only associate, set the channel and get your ip address once at power up.

Most of the other commands take effect immediately like the COMM settings and timers. This allows temporary change of parameters "on the fly" to test features, minimizes power usage and saves on flash re-write cycles.

Once all configuration is complete, the user must save the settings using the **save** command to store the configuration data, otherwise it will not take effect upon reboot or reset. Multiple configurations can be stored by using the **save <filename>** command, and these configurations can be loaded using the **load <filename>** command.

5. SET Commands

These commands begin with "set". There are 6 major categories.

Adhoc	controls the adhoc parameters
Broadcast	controls the broadcast hello/heartbeat UDP message
COMM	communication and data transfer, timers, matching characters
DNS	DNS host and domain
FTP	FTP host address and login information
IP	IP settings
Option	optional and not frequently used parameters
Sys	system settings such as sleep and wake timers
Time	timer server settings
UART	serial port settings such as baudrate and parity
WLAN	wireless interface settings, such as ssid, chan, and security options

5.1. Adhoc Parameters

set adhoc beacon <ms>	sets the adhoc beacon interval in milliseconds. Default is 100.
set adhoc probe <num>	sets the adhoc probe timeout in seconds. Default is 60. This is the number of seconds waiting for probe responses before declaring "ADHOC is lost" and disabling the network interface.

5.2. Broadcast Parameters

set broadcast address <addr>	sets the address to which the UDP hello/heartbeat message is sent. The default address is 255.255.255.255

set broadcast interval <value>	sets the interval at which the hello/heartbeat UDP message is sent. Interval is specified in seconds. The value is a mask that is compared to a free running seconds counter. For example if interval = 0x7, a packet will be sent every 8 seconds. The minimum interval value is 1 (every 2 seconds) and max value is 0xff (every 256 seconds). Setting the interval value to zero disables sending UDP broadcast messages. The default interval is 7.
set broadcast port <port>	sets the port number to which the UDP hello/heartbeat message is sent. The default port is 55555.

5.3. COMM Parameters

set comm $ <char>	sets character used to enter command mode. Typically used when "$$$" is a possible data string. Default is '$'. Care should be taken when setting this to note the new character as once this setting is saved every subsequent reboot will ignore "$$$" and look for "<char><char><char>".
set comm close <string>	sets the ASCI string that is sent to the local UART when the TCP port is closed. If no string is desired, use 0 as the <string> parameter. Max string length is 32 characters. Default is *CLOS*
set comm open <string>	sets the string that is sent to the local UART when the TCP port is opened. If no string is desired, use 0 as the <string> parameter. Max string length is 32 characters. Default is *OPEN*
set comm remote <string>	sets the string that is sent to the remote TCP client when the TCP port is opened. If no string is desired, use 0 as the <string> parameter. Max string length is 32 characters. Default is *HELLO*
set comm idle <secs>	sets the Idle Timer Value. This is the number of seconds with no transmit or receive data before the connection is closed automatically. Default is 0, never disconnect on idle.
set comm match <value>	sets match character. An IP packet will be sent each time the match character appears in the data. Value is entered as the decimal (13) or hex (0xd) of the of the ASCII character. Default is 0, disabled. The match character is one of three ways to control TCP/IP packet forwarding. The others are size and timer. For more information see section 10.1 on System Timers and Auto Connect Timers and section 10.4 on UART Receiver.
set comm size <value>	sets the flush size. An IP packet will be sent each time "value" bytes are received. Default is 64 bytes. You should set this value to the largest

	possible setting to maximize TCP/IP performance. Maximum value = 1420 (at 9600) bytes.
	NOTE: This value is set automatically when the baudrate is set, in an attempt to optimize the link. It is assumed that higher baudrates equates to more data and hence the flush size is increased.
	Flush size is one of three ways to control TCP/IP packet forwarding. The others are match character and timer. For more information see section 10.4 on UART Receiver.
set comm time <num>	sets the flush timer. An IP packet will be sent if no additional bytes are received for "num" milliseconds. Num is one milliseconds intervals. 1 is the minimum value. Default is 10 (10 milliseconds). Setting this value to 0 will disable forwarding based on the flush timer.
	Flush timer is one of three ways to control TCP/IP packet forwarding. The others are match character and size. For more information see section 10.1 on System Timers and Auto Connect Timers

5.4. DNS Parameters

set dns address <addr>	sets the IP address of the DNS sever. This is auto-set when using DHCP, and needs to be set in STATIC IP or Auto-IP modes.
set dns name <string>	sets the name of the host for TCP/IP connections.
set dns backup <string>	sets the name of the backup host for TCP/IP connections.

5.5. FTP Parameters

set ftp filename <file>	sets the name of the file transferred when issuing the "ftp u" or "ftp g" commands.
set ftp addr <addr>	sets the ftp server IP address.
set ftp remote <port>	sets the ftp server remote port number (default is 21).
set ftp user <name>	sets the ftp user name for accessing the FTP server.
set ftp pass <pass>	sets the ftp password for accessing the FTP server.

5.6. IP Parameters

set ip address <addr> sets the IP address of the WiFly GSX module. If DHCP is turned on, the IP address is assigned and overwritten during association with the access point. IP addresses are "." delimited. Note this is different from the RN-111b module which is space delimited!

Example: "set ip a 10.20.20.1"

set ip backup <addr> sets a secondary host IP address. If the primary host IP is not reachable the module will try the secondary IP address if set.

set ip dchp <value> enable/disable DHCP mode. If enabled, the IP address, gateway, netmask, and DNS server are requested and set upon association with access point. Any current IP values are overwritten.

DHCP Cache mode can reduce the time it takes the module to wake from deep sleep thus saving power. In cache mode, the lease time is checked and if not expired, the module uses the previous IP settings. If the lease has expired the module will attempt to associated and use DHCP to get the IP settings. DHCP cached IP address does not survive a power cycle or reset.

Mode	Protocol
0	DHCP OFF, use stored static IP address
1	DHCP ON, get IP address and gateway from AP
2	Auto-IP, generally used with Adhoc networks
3	DHCP cache mode, Uses previous IP address if lease is not expired (lease survives reboot)
4	Reserved for future use

set ip flags <value> Set TCP/IP functions. Value is a bit mapped register. Default = 0x7.

Bit	Function
0	TCP connection status. See note below
1	Bypass Nagle algorithm and use TCP_NODELAY
2	TCP retry enabled (42 total)
3	UDP RETRY (attempts retry if no ACK from UDP)
4	DNS host address caching enabled
5	ARP table caching enabled
6	UDP auto pairing enabled
7	Add 8 byte timestamp to UDP or TCP packets

NOTE: When the link to an associated to an access point is lost while a TCP connection is active, the TCP connection can be left in hung/ inconsistent state. In some cases, the TCP connection will not recover. In version 2.20 and later, if the link to the access point is regained within 60 seconds, the TCP connection will survive.

With version 2.20 we have changed the operation of bit0 in the "ip flags" register. Previously this bit specified the TCP copy function, but controls the TCP socket function while associated on a network.

- If bit 0 is set (default) TCP connections are kept open when the connection to the access point is lost.
- If bit 0 is cleared (by setting "set ip flags 0x6" for example) then when the connection to the access point is lost and TCP is connected, the connection will be closed.

set ip gateway <addr> sets the gateway IP address, If DHCP is turned on, the gateway IP address is assign and overwritten during association with the access point.

set ip host <addr> sets the remote host IP address. This command is used for making connections from the WiFly module to a TCP/IP server at the IP address <addr>.

set ip localport <num> sets the local port number.

set ip netmask <value> sets the network mask. If DHCP is turned on, the net mask is assign and overwritten during association with the access point.

set ip protocol <value> sets the IP protocol. Value is a bit mapped setting. To connect to the WiFly GSX module over TCP/IP such as Telnet the device must have the use the TCP Server protocol / bit 2 set. To accept both TCP and UDP use value = 3 (bit 1 and bit 2 set)

Bit Position	Protocol
0	UDP
1	TCP Server & Client (Default)
2	Secure (only receive packets with IP address matches the store host IP)
3	TCP Client only
4	HTTP client mode

set ip remote <value> sets the remote host port number.

NOTE: Due to an issue in the UART hardware, the UART does not support even or odd parity.

5.11. WLAN Parameters

set wlan auth <value>

Sets the authentication mode. Not needed unless using auto join mode 2. i.e. ***set wlan join 2***

Note: During association the WiFly module interrogates the Access Point and automatically selects the authentication mode.

The current release of Wifly firmware supports these security modes:
- WEP-128 (open mode only, NOT shared mode)
- WPA2-PSK (AES only)
- WPA1-PSK (TKIP only)
- WPA-PSK mixed mode (some APs, not all are supported)

Value	Authentication Mode
0	Open (Default)
1	WEP-128
2	WPA1
3	Mixed WPA1 & WPA2-PSK
4	WPA2-PSK
5	Not Used
6	Adhoc, Join any Adhoc network

set wlan channel <value>

sets the wlan channel, 1-13 is the valid range for a fixed channel. If 0 is set, then scan is performed, using the ssid, for all the channels set in the channel mask.

set wlan ext_antenna <0, 1>

determines which antenna is active, use 0 for chip antenna, 1 for UF.L connector. Default = 0. Only one antenna is active at a time and the module must be power cycled after switching the antenna.

set wlan join <value>

sets the policy for automatically joining/associating with network access points. This policy is used when the module powers up, including wake up from the sleep timer.

Value	Policy
0	Manual, do not try to join automatically
1	Try to join the access point that matches the stored SSID, passkey and channel. Channel can be set to 0 for scanning. (Default)

2	Join ANY access point with security matching the stored authentication mode. This ignores the stored SSID and searches for the access point with the strongest signal. The channels searched can be limited by setting the channel mask.
3	Reserved – Not used
4	Create an Adhoc network, using stored SSID, IP address and netmask. Channel MUST be set. DHCP should be 0 (static IP) or set to Auto-IP with this policy. (unless another Adhoc device can act as DHCP server) This policy is often used instead of the hardware jumper to creat a custom Adhoc network

set wlan hide <0, 1> Hides the WEP key and WPA passphrase. When set, displaying the wlan settings shows ****** for these fields. To unhide the passphrase or passkey, re-enter the key or passphrase using the set wlan key or set wlan passphrase command. Default = 0, don't hide.

set wlan key <value> sets the 128 bit WEP key. If you are using WPA or WPA2 you should enter a pass phrase with the set wlan passphase command. Key must be EXACTLY 13 bytes (26 ASCII chars). Data is expected in HEX format, "0x" should NOT be used here.

Example : "set w k 112233445566778899AABBCCDD"

Hex digits > 9 can be either upper or lower case.

The Wifly GSX only supports "open" key mode, 128 bit keys for WEP. WEP-128, shared mode is not supported as it is known to be easily compromised and has been deprecated from the WiFi standards.

set wlan linkmon <value> sets the link monitor timeout threshold. If set to 1 or more, WiFly will scan once per second for the AP it is associated with. The value is the threshold of failed scans before the WiFly declares "AP is Lost", de-authenticates. The WiFly will retry the association based on the join policy variable. A value of 5 is recommended, as some APs will not always respond to probes. Default is 0 (disabled). Without this feature,

	there is no way to detect an AP is no longer present until it becomes available again (if ever).
set wlan mask <value>	sets the wlan channel mask used for scanning channels with the auto-join policy 1 or 2, used when the channel is set to 0. Value is a bit-map where bit 0 = channel 1. Input for this command can be entered in decimal or hex if prefixed with 0x. Default value is 0x1FFF (all channels)
set wlan num <value>	sets the default WEP key to use. 1-4 is the valid range. Example : "set w n 2" sets the default key to 2.
set wlan phrase <string>	sets the passphrase for WPA and WPA2 security modes. 1-64 chars. The passphrase can be alpha and numeric, and is used along with the SSID to generate a unique 32 byte Pre-shared key (PSK), which is then hashed into a 256 bit number. Changing either the SSID or this value re-calculates and stores the PSK. If exactly 64 chars are entered, it is assumed that this entry is already an ASCII HEX representation of the 32 byte PSK and the value is simply stored. For passphrases that contain spaces use the replacement character $ instead of spaces. For example "my pass word" would be entered "my$pass$word". The replacement character can be changed using the optional command **set opt replace <char>.** Example : "set w p password" sets the phrase.
set wlan rate <value>	sets the wireless data rate. Lowering the rate increases the effective range of the WiFly-GSX module. The value entered is mapped according to the following table

Value	Wireless Data Rate
0	1 Mbit/sec
1	2 Mbit/sec
2	5.5 Mbit/sec
3	11 Mbit/sec
4 - 7	Invalid
8	6 Mbit/sec
9	9 Mbit/sec
10	12 Mbit/sec
11	18 Mbit/sec
12	24 Mbit/sec (default)

get time	display the time server UDP address and port number.
get wlan	display the ssid, chan, and other wlan settings.
get uart	display the UART settings.
ver	return the software release version

7. Status Commands

These commands begin with "show", and they return the current values of variables in the system. In s cases, for example IP addresses, the current values are received from the network, and may not match stored values.

show battery Displays current battery voltage, (only valid for Roving battery powered product like RN-370 and temperature sensors)

show connection Displays connection status in this HEX format: 8XYZ

Bit location	13-16	9-12	7	6	5	4	0-3
Function	fixed	channel	DNS found	DNS server	Authen	Assoc	TCP status
Value	8	1-13	1=resolved	1= contacted	1= OK	1=OK	0= Idle, 1=Connected 3= NOIP 4= Connecting

show io	Displays IO pin levels status in this HEX format: 8ABC Example: **show i** returns 8103 indicates pins 0, 1 and 9 high level.
show net <n>	Displays current network status, association, authentication, etc. Optional parameter displays only the MAC address of the AP currently associated.
show rssi	Displays current last received signal strength.
show stats	Displays current statistics, packet rx/tx counters, etc.
show time	Displays number of seconds since last powerup or reboot
show q <0-7>	Display the value of the an analog interface pin from 0 to 7. The value returned will the format 8xxxxx where xxxxx is voltage in microvolts sampled on the channel yo request with the 8 in front as a start marker.

	ping h pings the stored host IP address, the host IP address can be set with the *set ip host <addr>* command
	ping i pings a known Internet server at www.neelum.com by first resolving the URL (proves that DNS is working and proves the device has internet connectivity).
	ping 0 terminates a ping command
reboot	forces a reboot of the device (similar to power cycle)
scan <time> <P>	Performs an active probe scan of access points on all 13 channels. Returns MAC address, signal strength, SSID name, security mode. Default scan time is 200ms / channel = about 3 seconds. **time** is an optional parameter, this is the time in ms per channel. For example, "scan 30" reduces the total scan time down to about 1 second. This command also works in Adhoc mode. If the optional P parameter is entered, the module will perform a passive scan, and list all APs that are seen in passive mode.
time	Sets the Real time clock by synchronizing with the time server specified with the time server parameters (see section 5.9) This command sends a UDP time server request packet.

9. File IO Commands

del <name> <num>	Deletes a file. Optional <num> will override the name and use the sector number shown in the "ls" command.
load <name>	Reads in a new config file.
ls	Displays the files in the system
save	Saves the configuration to "config" (the default file).
save <name>	Saves the configuration data to a new file name
boot image <num>	Makes file <num> the new boot image.
ftp get <name>	Retrieves a file from the remote FTP server. If <name> not specified, the stored ftp filename is used.
ftp update <name>	Deletes the backup image, retrieves new image and updates the boot image.

"GSM Modem" Data Sheet

Introduction

The ZTE USB modem is a multi-mode USB modem, working in HSUPA/HSDPA/WCDMA/EDGE/ GPRS/GSM networks. With USB interface connecting to a laptop or a desktop PC, it integrates the functionality of a modem and a mobile phone (SMS), and combines mobile communication with Internet perfectly. It supports data and SMS services through the mobile phone network, enabling you to communicate anytime and anywhere. Specification The following table shows the specification of your device.

Specification

>HSDPA/UMTS 850, 2100MHz
>GSM/GPRS/EDGE 850/900/1800/1900 MHz
>HSDPA DL 7.2Mbps
>UMTS DL/UL 384Kbps
>USB 2.0 Interface
>Microsoft Windows and MAC OS compatible
>Compact size 75 x 26 x 10 mm
>Weight 21g
>Idle current approx 100mA, Max 450mA
>Maximum Emitted Power 250mW in 3G, 2W in GSM
>12 month warranty

* Actual speeds will be less. Speeds may vary due to congestion, distance from the cell, local conditions, hardware, software and other factors.

Hardware Installation

1. Put finger on the bottom finger grip, and then lift the front cover of
the modem to release and remove.
Notes:
Do not open the front cover rudely so as to avoid the damage of the front cover.

2. Insert the SIM/USIM card into the SIM/USIM card slot.
Insert the SIM/USIM card with the metal contact area facing downwards into the slot, and then
push SIM/USIM card as far as possible.

3. Insert the microSD card into the microSD card slot.
Notes:This modem only supports microSD card.
Insert your microSD card into the microSD card slot.

4. Put the front cover of the modem flatly upon the body of modem. Direct
the front cover towards the locking catches on the sides of the modem
and push the front cover of the modem to lock the front cover into its
place.

5. Connect the Modem to your laptop or desktop PC.
Notes: microSD is a trademark of SD Card Association.

□ Plug the USB connector into your computer's USB port and make sure that it is tightly inserted.

□ The OS automatically detects and recognizes new hardware and starts the installation wizard.