الجمهورية الجزائرية الديمقراطية الشعبية
Democratic and Popular Republic of Algeria
وزارة التعليم العالي و البحث العلمي
Ministry of Higher Education and Scientific Research

University of Mohamed Khider Biskra
Faculty of Sciences and Technology
Department of Electric Engineering
Sector : Electronics
Option : Telecommunication

Ref:............

End of studies project
For graduation:

MASTER

Theme

Control Via Internet (TCP/IP)

Presented by :
Brinis Taqiyeddine
Presented the : June 4th 2013

Before the juries are :

Mr KAHOUL Nadir	MAA	President
Mr GUESBAYA Taher	MCA	Supervisor
Mr HEZABRA Adel	MAA	Examiner

Academic year : 2012 / 2013

الجمهورية الجزائرية الديمقراطية الشعبية
Democratic and Popular Republic of Algeria
وزارة التعليم العالي و البحث العلمي
Ministry of Higher Education and Scientific Research

University Mohamed Khider Biskra
Faculty of Sciences and Technology
Department of Electric Engineering
Sector : Electronics
Option : Telecommunication

End of studies project
For graduate :
MASTER

Theme

Control Via Internet

Presented by :

Brinis Taqiyeddine

Favourable opinion of supervisor :

Guesbaya Taher *signature*

Favourable opinion of the president of jury :

Kahoul Nadie *Signature*

Cachet and signature

الجمهورية الجزائرية الديمقراطية الشعبية
Democratic and Popular Republic of Algeria
وزارة التعليم العالي و البحث العلمي
Ministry of Higher Education and Scientific Research

University Mohamed Khider Biskra
Faculty of Sciences and Technology
Department of Electric Engineering
Sector : Electronics
Option : Telecommunication

Theme :

Control Via Internet

Proposed by : Guesbaya Taher
Directed by : Guesbaya Taher

Abstract

The Internet is the largest medium of communication and the fastest growth advanced in the world, which relies protocol (TCP / IP), mainly for its work.

The Internet provides a lot of services through the model client/server, the same philosophy that followed for the development of our project, which aim through Internet transmission of phase services to stage remote control circuit electric is connected to computer (where the server was installed) via the serial port (RS-232), using C Sharp to build and interface the server and developing an Android application to be the client that is connected to internet to control our circuit.

The key words : Internet, TCP/IP protocol, control, Server/Client, Android, serial port (RS-232).

Résumé

L'internet est le plus grand moyen de communication et à la croissance la plus rapide de pointe dans le monde, qui repose protocole (TCP / IP), principalement pour son travail.

L'internet offre beaucoup de services à travers le modèle client / serveur, la même philosophie qui a suivi pour le développement de notre projet, qui vise à travers la transmission sur Internet des services de phase pour mettre en scène la télécommande circuit électrique est connecté à l'ordinateur (où le serveur a été installé) via le port série (RS-232), on utilise C Sharp à construire et interfacer le serveur et développement d'une application Android pour le client qui est connecté à Internet pour contrôler notre circuit.

Les mots clés : Internet, Le protocole TCP/IP, Controller, serveur/client, Android, porte serie (RS-232).

ملخص

تعتبر الأنترنت وسيلة الإتصال الأكبر نموا و الأسرع تطورا في العالم، حيث تعتمد نظام التشفير (TCP/IP) أساسا لعملها.

توفر الأنترنت الكثير من خدماتها عن طريق النموذج ملقم/عميل وهي نفس الفلسفة التي إتبعناها لتطوير مشروعنا الذي نهدف من خلاله نقل الأنترنت من مرحلة الخدمات إلى مرحلة التحكم عن بعد بدارة كهربائية متصلت بجهاز الكمبيوتر (أين تم تثبيت الملقم) عن طريق المنفذ التسلسلي (RS-232) بالإعتماد السي شارب لبناء واجهت الملقم و كذا تطوير تطبيق على نظام الأندرويد ليكون العميل المتصل عن طريق الأنترنت للتحكم في الدارة الكهربائية.

الكلمات الدلالية : أنترنت ، نظام التشفير (TCP/IP)، تحكم ، ملقم/عميل ، أندرويد ، منفذ تسلسلي(RS-232).

Dedications :

For The Memory Of My Father,

And My Wonderful Mother

For my sisters, my brother, and the whole family

Special dedication for my classmates and my teachers

for my friends in good and bad times

Gratitudes :

First of all thanks for Allah who gave us the power and the patient to do this humble project.

all the thanks for Dr : Guesbaya Taher , my teacher and my supervisor

we would thank Mr : Kahoul Nadir , the president of the jury .

and Mr : Hezabra Adel , the examiner .

for all workers in our department for their regardless help

Special thanks for Mr : Hamza and Mr : Abd El Latif , the engineers in the laboratory.

List of tables :

List of figures :

Abstract :

The Internet is the largest medium of communication and the fastest growth advanced in the world, which relies protocol (TCP / IP), mainly for its work.
The Internet provides a lot of services through the model client/server, the same philosophy that followed for the development of our project, which aim through Internet transmission of phase services to stage remote control circuit electric is connected to computer (where the server was installed) via the serial port (RS-232).
Using C Sharp to build and interface the server and developing an Android application to be the client that is connected to internet to control our circuit.

The key words : Internet, TCP/IP protocol, control, Server/Client, Android, serial port.

Résumé :

L'internet est le plus grand moyen de communication et à la croissance la plus rapide de pointe dans le monde, qui repose protocole (TCP / IP), principalement pour son travail.
L'internet offre beaucoup de services à travers le modèle client / serveur, la même philosophie qui a suivi pour le développement de notre projet, qui vise à travers la transmission sur Internet des services de phase pour mettre en scène la télécommande circuit électrique est connecté à l'ordinateur (où le serveur a été installé) via le port série (RS-232).
On utilise C Sharp à construire et interfacer le serveur et développement d'une application Android pour le client qui est connecté à Internet pour contrôler notre circuit.

Les mots clés : Internet, Le protocole TCP/IP, Controller, serveur/client, Android,porte serie.

ملخص:

تعتبر الأنترنت وسيلة الإتصال الأكبر نموا و الأسرع تطورا في العالم، حيث تعتمد نظام التشفير (TCP/IP)أساسا لعملها

توفر الأنترنت الكثير من خدماتها عن طريق النموذج ملقم/عميل وهي نفس الفلسفة التي إتبعناها لتطوير مشروعنا الذي نهدف من خلاله نقل الأنترنت من مرحلة الخدمات إلى مرحلة التحكم عن بعد بدارة كهربائية متصلت بجهاز الكمبيوتر (أين تم تثبيت الملقم) عن طريق المنفذ التسلسلي(RS-232)

بالإعتماد السي شارب لبناء واجهت الملقم و كذا تطوير تطبيق على نظام الأندرويد ليكون العميل المتصل عن طريق الأنترنت للتحكم في الدارة الكهربائية.

الكلمات الدلالية : أنترنت ، نظام التشفير ()، تحكم ، ملقم/عميل ، أندرويد ، منفذ تسلسلي.

Summary :

Introduction :

Introduction :

The evolution of the life style of the humans was constantly a tries to facilitate the work conditions to avoid some of the risks posed by its surroundings. If we go back in time, we will notice that the work accidents were disastrous, and some of them were for simple reasons.

The introduction of technology to the factories had a clear impact on all levels. After all the simple work that the workers done manually, machinery came to cover it quickly and proficiency. However, this resulted new types of risks and injuries. The solution lied in more working away from the work environment to a safer distances. In that phase began the concept of remote control development.

On the other side, the computers development was the milestone in line of history. With their I/O ports feature, they have the possibility to communicate with their environment. As an example, we have the serial port (RS-232). It is one of the most popular port because the simplicity of its protocol and its cheap interface compared to some other ports.

At the beginning of the 90s, the use of computer knew widespread and broke the monopoly of scientists and specialists. And with the integration of the concept of remote control and the computer via one of the ports create a great technological revolution, it provided great accuracy in the following up the work on the machines, as well as it gave some safety to the workers which significantly reduced a lot of work accidents.

At the mid-90s, another technological revolution was occurring in parallel with the previously mentioned (it is always related to the computers). The internet known a quantum leap in various ways, either services or commercial. It has created a new concept of communication and also contributed very significant acheavments in shaping the modern world. In one way or another, all the modern communication technologies are related to the Internet, either directly or indirectly.

Returning to the working conditions specifically for engineers or factory owners and away from the safety side, to stay informed on the state of mechanisms or make changes in particles, prevented the large distances from the work site, it will remain a big obsession. The solution may lie in creating a completely new technology, but it takes exorbitant costs and requires a lot of time.

Introduction

In recent years, other modern technology appeared to the world and known widely in a short time. Smartphones formed a virtuous circle to support the world of communications .They are considered as another technological element can not be ignored if we want to keep pace with the times.

The blending contemporary technologies may be a way to create the new technology that we are looking for. Where it will not cost a lot in addition that it would not need a lot of time. Provide the following items : remote control, Internet, smartphones. By this three modern technologies, we will seek to examine each element separately in an attempt to give a new dimension to the concept of remote control and the creation of a new relationship between human and machin. The real question was : can we communicate between 3items (remote control, Internet and smartphones)! And is there any clear methode should we fallow to reach or goal!

Since we defined our needs, we tried to develop our project methodically. In first chapter, we tried to understand the basics of remote control. We chose the serial port as output for our commands. One of the most interesting books we used, was Serial Port Complete by Jan Axelson that helped us to understand its functionality. We didn't face a real problem in this part.

As we moved farther to chapter two, we wanted to have a better understanding about the internet and its concepts. TCP/IP protocol suite 4th edition by Behrouz A. Forouzan, was the most detailed book could find.

The third chapter was about Android (the operation system of the most popular smartphones). The author Wallace Jackson gave us good start with his book Android Apps For Absolute Beginners.

As a sample, we decided to build simple circuit that contains PIC that we explained its principles in chapter four. The mikroelektronika web site gave us very good start as beginners .

The last chapters (5th and 6th), we gave an idea about the whole programming process. Step by step we talked about the realization of software and circuit. As mentioned the difficulties that faced mostly.

Mainly, building the idea from the base to the end without any example to help was the first problem. After that as we steadied on the last concept, stabilizing the server and client sides were the biggest challenge. As we do not forget the work on the circuit itself.

This kind of projects are wonderful way to login the world of the networking (as we said before, the networking is the new generation of communication) which was good reason for me to choose it. It was a good start to have an overview about the whole networking and good introduction into the depth of the world of internet, web services, and new generation remote control.

Chapter I : Serial Ports fundamentals (RS-232)

Chapter I : Serial Ports Fundamentals (RS-232)

I.1 Introduction:

Nowadays, the serial port communication dominates the world as the most used protocol. It perfectly replaced the parallel communication. In this chapter we will talk about mostly about the the serial communication in generally, as we will focus on the norm RS-232 characteristics since it was the used in our project.

I.2 The RS-232C Standard:

The RS-232C (commonly abbreviated RS-232) standard is also identified as one of the "V" series of specifications from the “Commité Consultatif Internationale de Telegraphie et Telephonie” (CCITT). As a result of this, RS-232 is sometimes referred to as the V.24 standard. The system was originally introduced as a specification for the connection of Data Terminal Equipment (DTE) to a Post Telephone and Telecommunications (PTT) modem. This is shown schematically in Figure I.1.

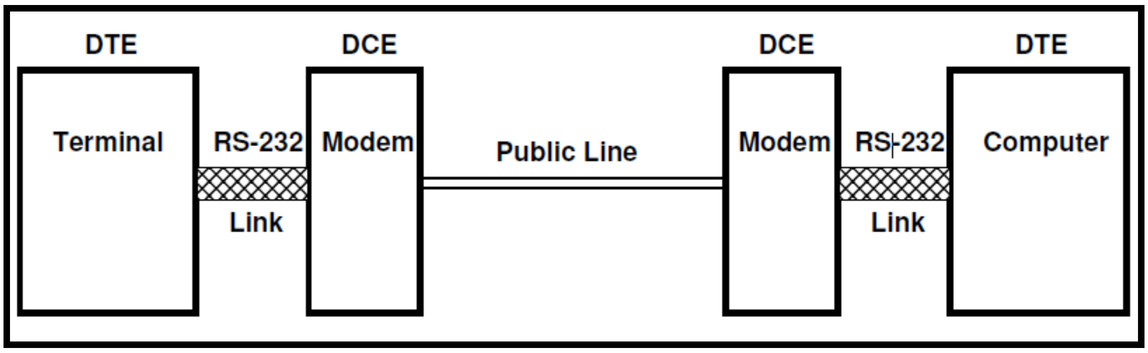


Figure I.1 - Original application of RS-232.C (V.24).

The RS-232 specification was designed to fulfil the needs of the environment shown in Figure I.1. Its primary function application was in short-distance, low-bit-rate links between devices (computers and modems) in clean-room environments. The problems associated with RS-232 have come about largely because we have extended its use to a vast range of different applications, for which it was never intended.

The RS-232 port on each of the devices shown in Figure I.1 is driven by a Universal

Asynchronous Receiver Transmitter (UART). The UART performs conversion of outgoing data from parallel form to serial form and incoming data from serial to parallel form. In addition, it is also equipped with special inputs and outputs, which are used to co-ordinate the flow of data with an external device. These special purpose inputs and outputs are referred to as the hardware hand-shaking lines.

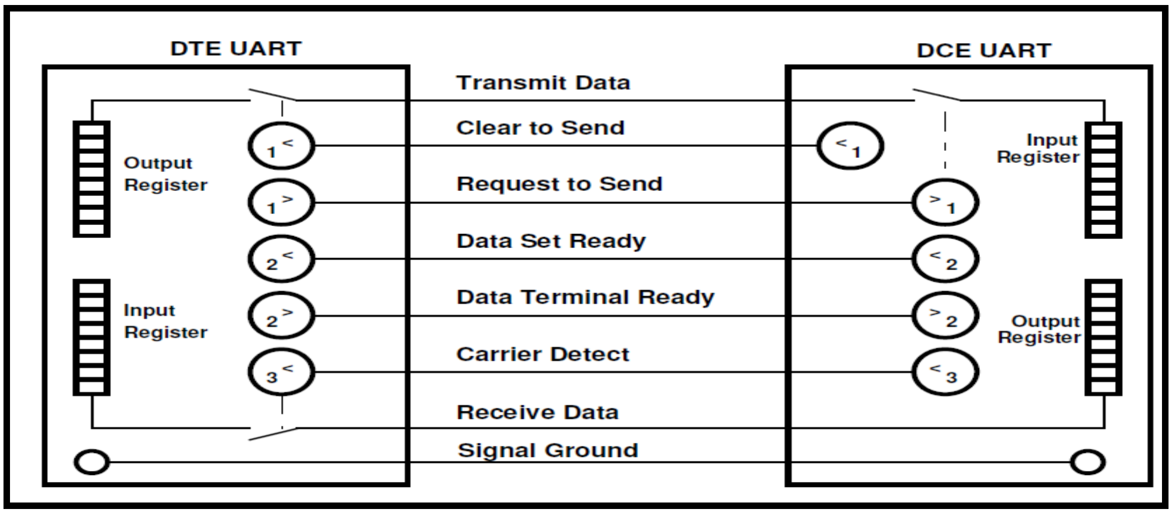


Figure I.2 - Schematic showing DCE and DTE UARTs and Hardware HandShaking on an RS-232C Link.

Figure I.2 shows a schematic of the UARTs in DCE and DTE devices and the way in which data transfer and hand-shaking occurs in the RS-232 system. Hand shaking inputs and outputs in Figure I.2 are signified by a circle containing the input or output number (on the UART). An inequality sign (< or >) inside the circle denotes the normal direction of information flow with respect to the local device.

In Figure I.3, the "Request to Send", "Clear to Send" and "Carrier Detect" lines are directly responsible for switching data transmission on and off. The "Data Set Ready" and "Data Terminal Ready" lines are used so that DCE and DTE devices, respectively, can indicate whether they are powered up and ready for communication. We will look at these lines in more detail as we proceed through this chapter, but for the moment we only need observe the structure of the hand-shaking in relation to the UARTs.

We also need to recognise at this early stage that RS-232 is one of the most important of all the communications specifications associated with manufacturing, if for no other reason than because it is the most widely used and misunderstood system. For all the sophisticated communications networks that are available for manufacturing, few cause as much frustration and irritation as this one standard. Most computer controlled devices in manufacturing have, what are referred to, as "RS-232 compatible" communications ports. As we progress through this chapter, we shall see just how much (or how little) this actually means to the system designer.[1]

I.3 Serial Port Communication Protocols:

The serial port has several communication protocols, we will try to explain in this section of the chapter.

I.3.1 ASCII Code:

Most serial data communications use ASCII code as the basic data unit for communicating among devices. ASCII code was defined by the American National Standards Institute (ANSI), and its full name is American Standard Code for Information Interchange. Basically, all information can be expressed in ASCII by a sequence or combination of 26 English characters, as well as some special characters, and each character can be expressed and transmitted by an associated 7-bit binary code. This representation method is the ASCII method. In serial communication programming, the ASCII code can be expressed in different formats, such as decimal or hexadecimal, and different subroutines can be utilized to translate between the different formats. Appendix A shows a typical ASCII code table that contains the popular characters used in normal data communications.

One point to remember is that in most computer data communications, each data unit is composed of 1 byte, or 8 bits, of binary code. For example, in Figure 1.1, a binary sequence, 01100101 , which is equivalent to a decimal of 101 or a hexadecimal of 65h in the standard ASCII table, is an 8-bit binary code. Only 7 bits are occupied when this code is represented as an ASCII code: 1100101 . The most significant bit (MSB) is not used. It can be concluded that the maximum binary code that can be expressed in ASCII code is 1111111 , which is equivalent to a decimal of 127.

However, in the real world of serial data communications, instead of using 7 bits, a data unit or an ASCII character is actually represented by a byte, or 8 bits, of binary code travelling between two devices. The MSB bit in an 8-bit binary code is often used as a parity bit for error checking, thus making ASCII, in practice, an 8-bit code.

The ASCII table in Appendix A shows the binary sequence represented in Figure 1.1, 01100101 , is equivalent to the ASCII character ‘A’. By using the ASCII code, the data can be successfully expressed and transmitted through a single wire, bit by bit, byte by byte, character by character, from one device to another.

A potential problem exists in using ASCII code during data communications. The incompatibility of different spoken languages can be a challenge; for example, some countries use non- English languages. Special care must be taken to handle such incompatibilities.

Another issue is that most computers employ some human-readable characters as their data communication medium. One exception is IBM, which has developed its own character set, the Extended Binary Coded Decimal Interchange Code (EBCDIC), however, this character set is not compatible with ASCII code.

A solution to these potential problems has been the development of a new character set that extends the ASCII code to meet the requirements of different languages used for data communications in different countries. IBM, Apple, and a number of UNIX vendors have developed a Unicode standard to improve compatibility, it includes 16-bit encoding that encompasses all existing national character code standards. The International Organization for Standardization (ISO) adopted this Unicode standard as a superset of ISO 10646 in June 1992, but the 16-bit encoding set will not be adopted worldwide for quite some time.[2]

I.3.2 DTE And DCE:

The RS-232 standard calls the terminal end of the link the Data Terminal Equipment, or DTE. The modem end of the link is the data circuit-terminating equipment, or DCE. The signals and their functions are named from the perspective of the DTE. For example, TX (transmit data) is an output on a DTE and an input on a DCE, while RX (receive data) is an input on a DTE and an output on a DCE.

The RS-232 ports on PCs are almost always DTEs. It doesn’t matter which device in a link is the DTE and which is the DCE, but every connection between two computers must either have one of each or must emulate the absent interface (typically DCE) with an adapter called a null modem. The null modem swaps the lines so each output connects to its corresponding input.

I.3.3 Serial Data Format In TTL:

In the real world, serial data transmissions between the DTE and DCE can be divided into two categories: synchronous and asynchronous transmissions:

1) **Synchronous data transfer :** The interface includes a clock line typically controlled by one of the computers, and all transmitted bits synchronize to that clock. Each transmitted bit is valid at a defined time after a clock's rising or falling edge, depending on the protocol.
   In program-to-program communication, synchronous communication requires that each end of an exchange of communication respond in turn without initiating a new communication. A typical activity that might use a synchronous protocol would be a transmission of files from one point to another. As each transmission is received, a response is returned indicating success or the need to resend. [3]
2) **Asynchronous data transfer :** The term asynchronous is usually used to describe communications in which data can be transmitted intermittently rather than in a steady stream. the interface doesn't include a clock line. Instead, each computer provides its own clock to use as a timing reference. The computers must agree on a clock frequency, and the actual frequencies at each computer must match within a few percent. A transmitted Start bit synchronizes the transmitter's and receiver's clocks. For example, a telephone conversation is asynchronous because both parties can talk whenever they like. If the communication were synchronous, each party would be required to wait a specified interval before speaking. The difficulty with asynchronous communications is that the receiver must have a way to distinguish between valid data and noise. In computer communications, this is usually accomplished through a special start bit and stop bit at the beginning and end of each piece of data. For this reason, asynchronous communication is sometimes called start-stop transmission.[3]

Almost all actual serial data communications are asynchronous, even when both devices use the same transmission rate (baud rate), because noise and disturbance from external factors and environmental conditions create a situation where the data must travel asynchronously between the DCE and the DTE. To coordinate this asynchronous transmission and make sure the communication between the two devices is correct (in other words, to synchronize the transmitted characters between the two devices), each serial data unit must have a certain format. A control unit called the Universal Asynchronous Receiver Transmitter (UART), is always utilized to synchronize the data flow between the DTE and the DCE. Figure I.3 shows a typical serial data unit format inside the DTE.[4]

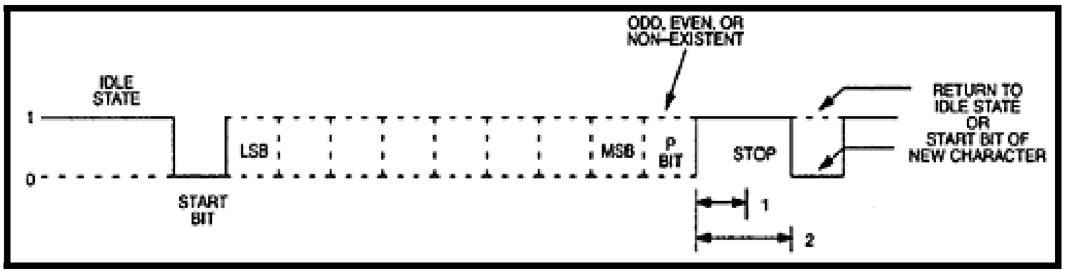

ODD, EVEN, OR NON-EXISTENT
IDLE STATE
1
0
START BIT
LSB
MSB
P BIT
STOP
1
2
RETURN TO IDLE STATE OR START BIT OF NEW CHARACTER

Figure I.3 - A typical serial data format (TTL/CMOS).[5]

I.3.4 Serial Data Format In Transmission Lines:

RS-232 logic levels are defined as positive and negative voltages rather than the positive only voltages of TTL and CMOS logic (Table I.1). At an RS-232 data output (TX), logic 0 is defined as equal to or more positive than +5V, and logic 1 is defined as equal to or more negative than -5V. In other words, the data uses negative logic, where the more positive voltage is logic 0 and the more negative voltage is logic 1.

Parameter	Voltage (V)
Logic 0 or On output	+5 to +15
Logic 1 or Off output	-5 to -15
Logic 0 or On input	+3 to +15V
Logic 1 of Off input	-3 to -15

Table I.1 - RS-232 uses positive and negative voltages.

The status and control signals use the same voltages, but with positive logic. A positive voltage indicates a logic 1 and a function that is On, asserted, or True, and a negative voltage indicates a logic 0 and a function that is Off, not asserted, or False. An RS-232 interface chip typically inverts the signals and converts between TTL/CMOS voltages and RS-232 voltages.

On a UART's output pin, a logic-1 data bit or an Off control signal is a logic high, which results in a negative voltage at the RS-232 output. A logic 0 data bit or asserted control signal is a logic low at the UART and results in a positive voltage at the RS-232 output. Because an RS-232 receiver can be at the end of a long cable, by the time the signal reaches the receiver, the voltage may have attenuated or have noise riding on it. To allow for degraded signals, the receiver accepts smaller voltages as valid. A positive voltage of 3V or greater is a logic 0 at RX or asserted at a control input. A negative voltage of 3V or greater (more negative) is a logic 1 at RX or Off at a control input. The logic level of an input between -3V and +3V is undefined. The noise margin, or voltage margin, is the difference between the output and input voltages. RS-232's large voltage swings result in a much wider noise margin than 5V TTL or CMOS logic.

For example, an RS-232 output of +5V can attenuate or have noise spikes as large as 2V at the receiver and will still be a valid logic 0. Many RS-232 outputs have wider voltage swings that result in even wider noise margins. The maximum allowed voltage swing is ±15V, though receivers must accept voltages as high as ±25V without damage.

Two other terms you might hear in relation to RS-232 are Mark and Space. On the data lines, Space is logic 0 (positive voltage), and Mark is logic 1 (negative voltage) as the Figure I.4 shows. These names have their roots in the physical marks and spaces mechanical recorders made as they logged binary data.[6]

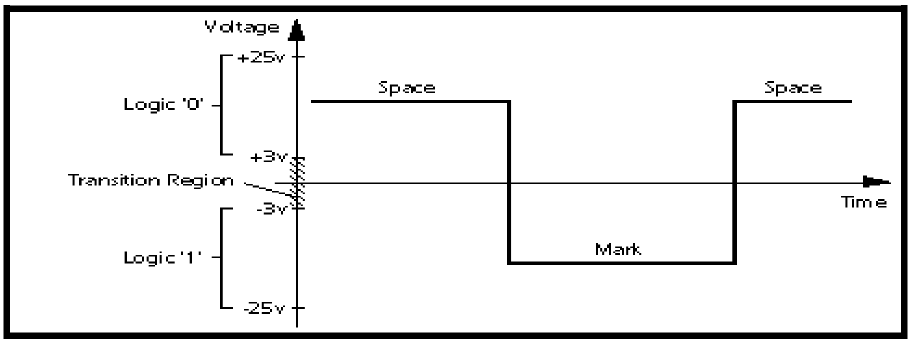


Figure I.4 - The real RS-232 signal (Mark and Space) . [7]

I.4 Essential Settings :

Before we use the RS-232 there are few settings we have to understand and define and they are :

I.4.1 Baud Rate :

Baud rate is defined as the transmission speed of a sequence of binary bits in a transmission line, or in other words, the bps (bit per second). Typical baud rates used in serial data communications are 9600, 19,200, and 38,400. For example, a transmission speed of 300 characters per second is equivalent to a baud rate of 2,400 (300 bytes # 8 bits per second). Some specifically designed I/O serial/parallel interfacing cards have much higher baud rates, and the user can select different baud rates based on the clock frequency generated from the clock generator in the interfacing card. Common values (speeds) are 1200, 2400, 4800, 9600, 19200, and 38400 mostly.[1]

I.4.2 Parity :

A user can select from five parity values: None, Even, Odd, Mark, and Space:

1) **None :** means that the data is transmitted without any parity bits. This is a popular setting when 8 bits of data are transmitted. For this kind of data transmission, no parity bits are available for transmission error checking.
2) **Even :** means that the transmitting serial device will assign a logical 1 to the parity bit if an even number of binary 1's exist in the character's data bits. Otherwise, the parity bit is assigned a logical 0.
3) **Odd :** means that there will always be an odd number of logical 1s in the character's data bits. This setting will indicate an error if an even number of logical 1s is received.
4) **Mark :** means that the DTE will send a logical 1s parity bit before the character's data bits. Space indicates that the transmitting serial device will send a logical 0 parity bit prior to the character's data bits.[1]

I.4.3 Start/Stop Bits :

1) **Start bit :** This is a synchronizing bit added just before each character we are sending.

This is considered a SPACE or negative voltage or a 0.

2) **Stop bit :** This bit tells us that the last character was just sent .This bit can be one of the following values:

1. one(1) stop bit.
2. one and half (1.5) stop bits (one bit with longer period).
3. two(2) stop bits.

This is considered a MARK or positive voltage or a 1. [9]

I.4.4 Data Bits :

Data bits are a measurement of the actual data bits in a transmission. When the computer sends a packet of information, the amount of actual data may not be a full 8 bits. Standard values for the data packets are 5, 7, and 8 bits. Which setting you choose depends on what information you are transferring. For example, standard ASCII has values from 0 to 127 (7 bits). Extended ASCII uses 0 to 255 (8 bits). If the data being transferred is simple text (standard ASCII), then sending 7 bits of data per packet is sufficient for communication. A packet refers to a single byte transfer, including start/stop bits, data bits, and parity. Since the number of actual bits depend on the protocol selected, the term packet is used to cover all instances.[10]

I.5 Connectors :

The RS-232 contains few different connectors, Each was developed for specific reason. And they are :

I.5.1 RS232 on DB9 (9-pin D-type connector) :

There is a standardized pinout for RS-232 on a DB9 connector, as shown below in Table I.2 And Figure I.5 .

Pin No.	Name	Dir	Notes/Description
1	DCD	IN	Data Carrier Detect. Raised by DCE when modem synchronized.
2	RD	IN	Receive Data (a.k.a RxD, Rx). Arriving data from DCE.

3	TD	OUT	Transmit Data (a.k.a TxD, Tx). Sending data from DTE.
4	DTR	OUT	Data Terminal Ready. Raised by DTE when powered on. In auto-answer mode raised only when RI arrives from DCE.
5	SGND	-	Ground
6	DSR	IN	Data Set Ready. Raised by DCE to indicate ready.
7	RTS	OUT	Request To Send. Raised by DTE when it wishes to send. Expects CTS from DCE.
8	CTS	IN	Clear To Send. Raised by DCE in response to RTS from DTE.
9	RI	IN	Ring Indicator. Set when incoming ring detected - used for auto-answer application. DTE raised DTR to answer

Table I.2 - 9-pin D-type connector Pin assignment .[11]

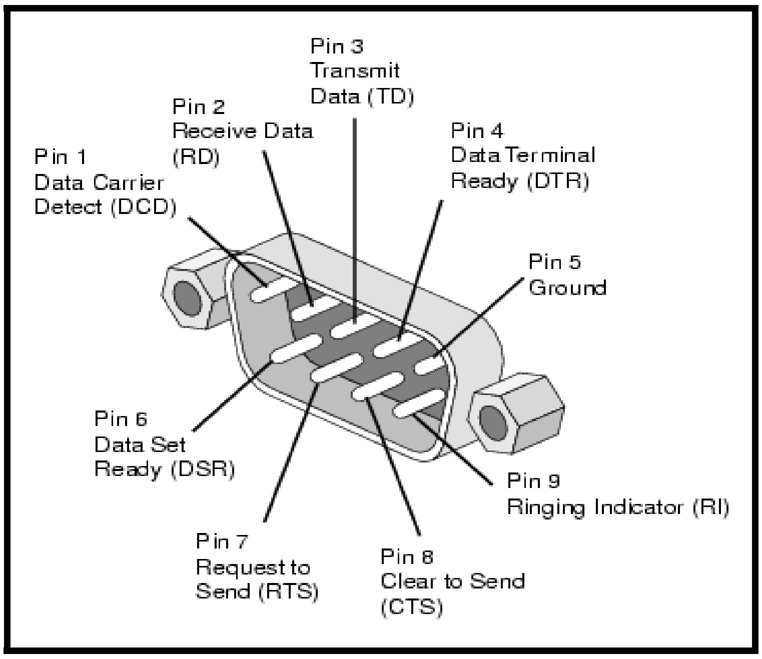


Figure I.5 - RS-232 D-sub connectors 9-pin male.[11]

I.5.2 RS232 on DB25 (25-pin D-type connector) :

In DB-25 connector most of the pins are not needed for normal PC communications, and indeed, most new PCs are equipped with male D type connectors having only 9 pins. Using a

25- pin DB-25 or 9-pin DB-9 connector, its normal cable limitation of 50 feet can be extended to several hundred feet with high-quality cable. RS-232 defines the purpose and signal timing for each of the 25 lines; however, many applications use less than a dozen. There is a standardized pinout for RS-232 on a DB25 connector, as shown below in Table I.3 and Figure I.6. [12]

P.N.	**Name**	**Dir**	Notes/Description
1	-	-	Protective/shielded ground
2	TD	OUT	Transmit Data (a.k.a TxD, Tx) (ASYNC)
3	RD	IN	Receive Data (a.k.a RxD, Rx) (ASYNC)
4	RTS	OUT	Request To Send (ASYNC)
5	CTS	IN	Clear To Send (ASYNC)
6	DSR	IN	Data Set Ready (ASYNC)
7	SGND	-	Signal Ground
8	CD	IN	Carrier Detect (a.k.a DCD).
9	-	-	Reserved for data set testing.
10	-	-	Reserved for data set testing.
11	-	-	Unassigned
12	SDCD	IN	Secondary Carrier Detect. Only needed if second channel being used.
13	SCTS	IN	Secondary Clear to send. Only needed if second channel being used.
14	STD	OUT	Secondary Transmit Data. Only needed if second channel being used.
15	DB	OUT	Transmit Clock (a.k.a TCLK, TxCLK). Synchronous use only.
16	SRD	IN	Secondary Receive Data. Only needed if second channel being used.
17	DD	IN	Receive Clock (a.k.a. RCLK). Synchronous use only.
18	LL	-	Local Loopback
19	SRTS	OUT	Secondary Request to Send. Only needed if second channel being used.
20	DTR	OUT	Data Terminal Ready. (ASYNC)
21	RL/SQ	-	Signal Quality Detector/Remote loopback

22	RI	IN	Ring Indicator. DCE (Modem) raises when incoming call detected used for auto answer applications.
23	CH/CI	OUT	Signal Rate selector.
24	DA	-	Auxiliary Clock (a.k.a. ACLK). Secondary Channel only.
25	-	-	Unassigned

Table I.3 - 25-pin D-type connector Pin assignment. [13]

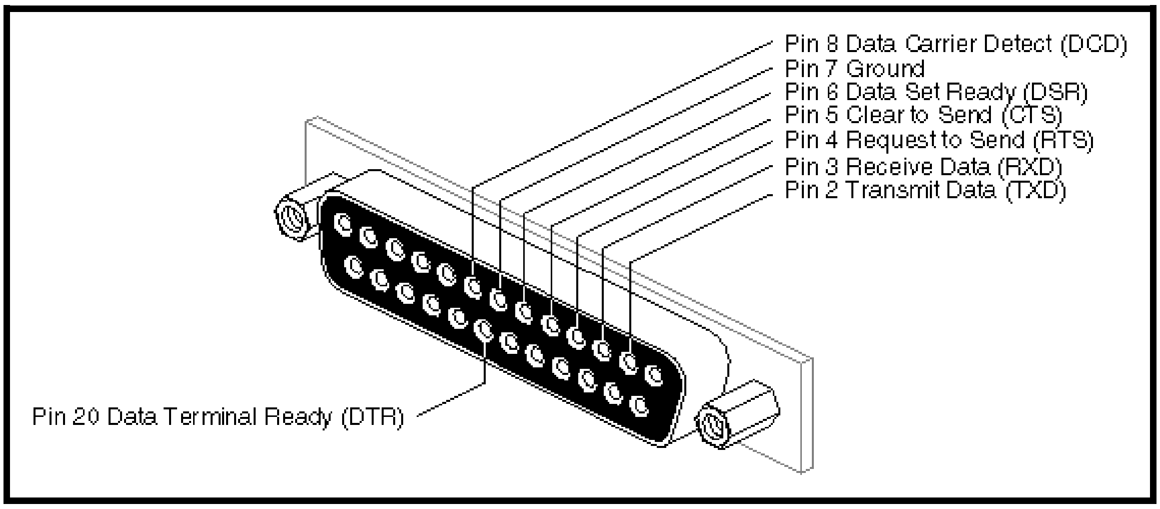

Pin 8 Data Carrier Detect (DCD)
Pin 7 Ground
Pin 6 Data Set Ready (DSR)
Pin 5 Clear to Send (CTS)
Pin 4 Request to Send (RTS)
Pin 3 Receive Data (RXD)
Pin 2 Transmit Data (TXD)
Pin 20 Data Terminal Ready (DTR)

Figure I.6 - RS-232 D-sub connectors 25-pin female. [14]

I.5.3 RS232 on RJ-45 :

RJ-45 (Registered Jack-45) is an eight-wire connector used commonly to connect computers onto local-area networks (LAN), especially Ethernets.

In other words, RJ-45 is a single-line jack for digital transmission over ordinary phone wire, either untwisted or twisted. The interface has eight pins or positions. For faster transmissions in which you're connecting to an Ethernet 10BASET network, you need to use twisted pair wire. RS232D, EIA/TIA-561 standard is applied when connecting to or from a serial port with a 8 position Modular Jack (RJ45) though it is not widely used as such. [12]

Table I.4 and Figure I.7 sill explain more about the wires.

RJ45 Pin No.	Name	DB9 Cross Connect	Notes/Description
1	DSR/RI	6,9	Data set Ready/ring indicator
2	DCD	1	Data Carrier Detect
3	DTR	4	Data Terminal Ready
4	SGND	5	Signal Ground
5	RD	2	Receive Data
6	TD	3	Transmit Data
7	CTS	8	Clear to Send
8	RTS	7	Request to Send

Table I.4 - RJ45 Connector Pin Assignment .

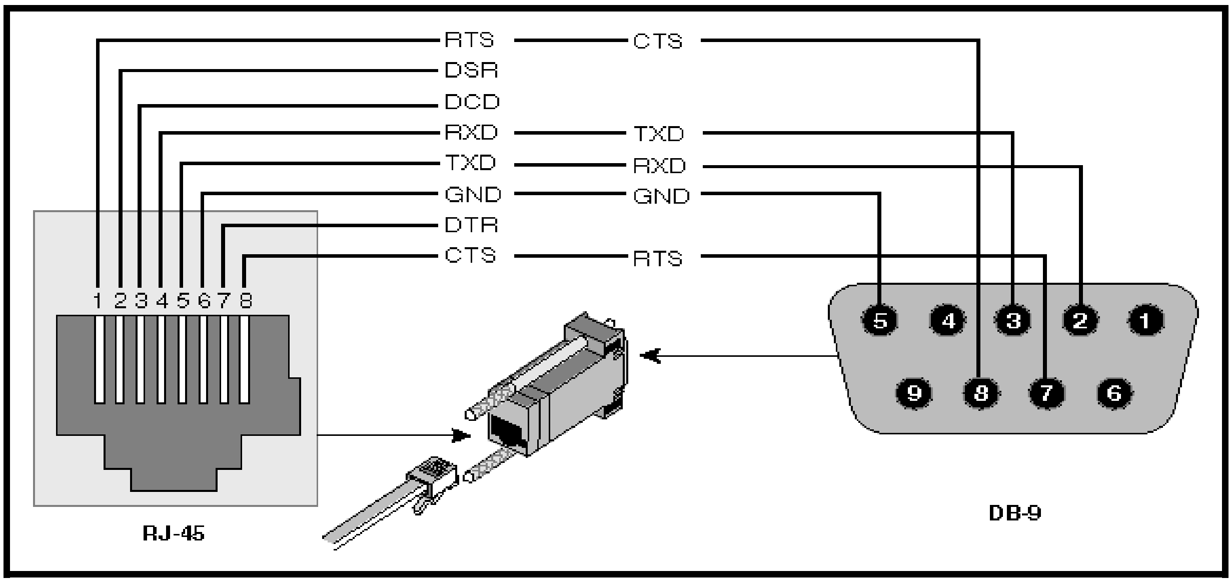


Figure I.7 - RJ45 Connector to DB9 (Flow Control Adaptor). [15]

I.6 Cables:

For RS-232 , there are different vary in length of cables, number of wires and shielding.

I.6.1 Length Limits:

Cable length is one of the most discussed items in RS232 world. The standard has a clear answer, the maximum cable length is 50 feet, or the cable length equal to a capacitance of 2500pF. The latter rule is often forgotten. This means that using a cable with low capacitance allows you to span longer distances without going beyond the limitations of the standard. If for example UTP CAT-5 cable is used with a typical capacitance of 17 pF/ft, the maximum allowed cable length is 147 feet.

The cable length mentioned in the standard allows maximum communication speed to occur. If speed is reduced by a factor 2 or 4, the maximum length increases dramatically. Texas Instruments has done some practical experiments years ago at different baud rates to test the maximum allowed cable lengths. Keep in mind, that the RS232 standard was originally developed for 20 kbps. By halving the maximum communication speed, the allowed cable length increases a factor ten. [16]

The Table I.5 shows data rates and the maximum distances recommended in RS-232.

Data rate (bps)	Distance (ft)
2400	3000
4800	1000
9600	500
19200	50

Table I.5 - Data rates and the maximum distances recommended in RS-232 according to Texas Instruments.

I.6.2 Isolated Lines:

RS-232's generous noise margins help make the interface reliable and immune to data errors caused by external noise coupling into the wires. When a line needs more protection, isolation can keep noise from coupling between a cable's wires and the circuits they connect to . Isolation works by dividing a circuit into sections that have no galvanic connection. Optical and magnetic coupling can transfer data and power between the sections and filter out noise. Isolation can isolate the grounds, the data lines, or both. Ground isolation makes a circuit immune to power surges and noise in the earth ground shared by nearby circuits. With long cables, ground isolation also makes the line immune to differences in ground potential from end to end. Isolating the data lines keeps noise from coupling between the lines and the circuits they connect to.[2]

I.7 Conclusion :

The RS-232 DB-9 is one the most popular norm in the serial ports family. It isn't the fastest in its family, but its electric characteristics and the simplicity of its protocol made it a good choice for the devices developers to build complex or simple circuits.

Therefore we made it as our choice as I/O port to connect it with the circuit we want to control in the end of our project.

Chapter II :

Internet's fundamentals

Chapter II : Internet's Fundamentals

II.1 Introduction :

This chapter can be considered as the most important one in the all project. As the theme title shows, our project is based on understanding the networking in generally (how do we communicate ?, what are the used protocols to communicate?, how do the layers communicate with each other? ...ect). we tried to cover as much as we can, since the networking is huge world by itself, but we focus on the some points more than some others due its importance and how effective it were on our project.

II.2 The OSI model and the TCP/IP protocol suit :

The OSI model and the TCP/IP protocol suit are networking models and a set of communication protocols used internet and similar networks. Both models have some almost same principles with few differences that we will see in this section of chapter.

II.2.1 The OSI model :

Open Systems Interconnection (OSI) model is a reference model developed by ISO (International Organization for Standardization) in 1984, as a conceptual framework of standards for communication in the network across different equipment and applications by different vendors. It is now considered the primary architectural model for inter-computing and internet working communications. Most of the network communication protocols used today have a structure based on the OSI model.

II.2.1.1 Layered Architecture :

The OSI model defines the communications process into 7 layers, which divides the tasks involved with moving information between networked computers into seven smaller, more manageable task groups. A task or group of tasks is then assigned to each of the seven OSI layers. Each layer is reasonably self-contained so that the tasks assigned to each layer can be implemented independently. This enables the solutions offered by one layer to be updated without adversely affecting the other layers.

The OSI 7 layers model has clear characteristics. Layers 7 through 4 deal with end to end communications between data source and destinations. Layers 3 to 1 deal with communications between network devices. On the other hand, the seven layers of the OSI model can be divided into two groups: upper layers (layers 7, 6 & 5) and lower layers (layers 4, 3, 2, 1).

The upper layers of the OSI model deal with application issues and generally are implemented only in software. The highest layer, the application layer, is closest to the end user. The lower layers of the OSI model handle data transport issues. The physical layer and the data link layer are implemented in hardware and software. The lowest layer, the physical layer, is closest to the physical network medium (the wires, for example) and is responsible for placing data on the medium. [15]

II.2.1.2 Communication between layers :

In Figure II.1 , device A sends a message to device B (through intermediate nodes). At the sending site, the message is moved down from layer 7 to layer 1. At layer 1 the entire package is converted to a form that can be transferred to the receiving site. At the receiving site, the message is moved up from layer 1 to layer 7.

The passing of the data and network information down through the layers of the sending device and back up through the layers of the receiving device is made possible by an interface between each pair of adjacent layers. Each interface defines what information and services a layer must provide for the layer above it. Well-defined interfaces and layer functions provide modularity to a network. As long as a layer provides the expected services to the layer above it, the specific implementation of its functions can be modified or replaced without requiring changes to the surrounding layers.

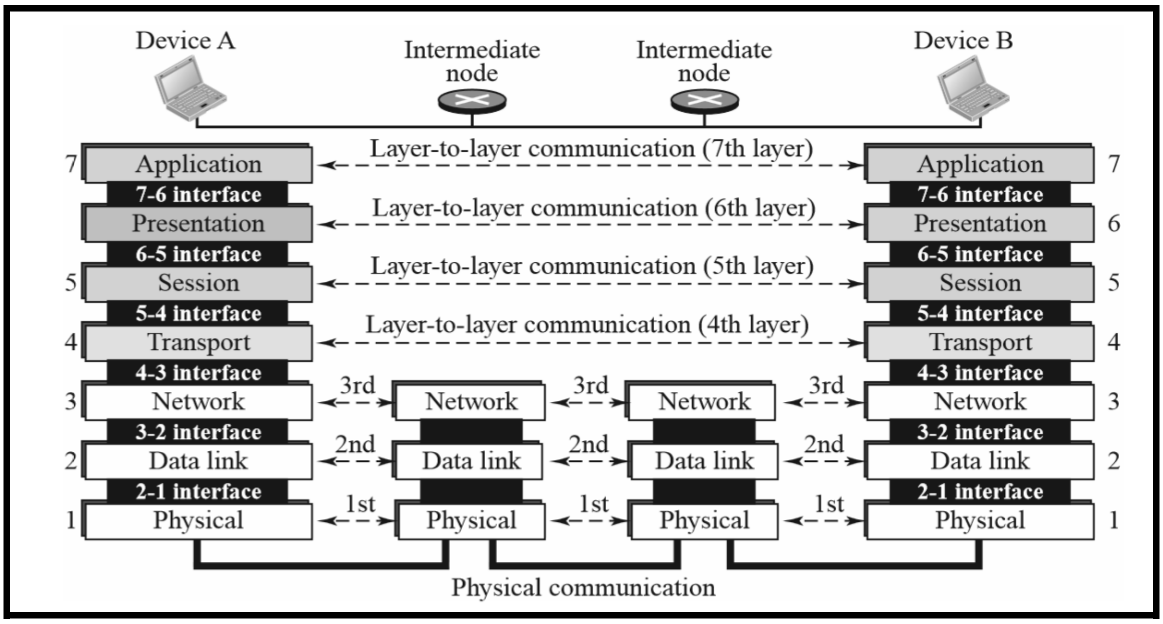


figure II.1 - The OSI Layers.[16]

The seven layers can be thought of as belonging to three subgroups. Layers 1, 2, and 3 (physical, data link, and network) are the network support layers; they deal with the physical aspects of moving data from one device to another (such as electrical specifications, physical connections, physical addressing, and transport timing and reliability). Layers 5, 6, and 7 (session, presentation, and application) can be thought of as the user support layers; they allow interoperability among unrelated software systems. Layer 4, the transport layer, links the two subgroups and ensures that what the lower layers have transmitted is in a form that the upper layers can use.

The upper OSI layers are almost always implemented in software; lower layers are a combination of hardware and software, except for the physical layer, which is mostly hardware. As we will see in the farther details when we explain each layer a side.[15]

II.2.1.3 Layers in the OSI Model :

The OSI Model contains 7 layers and they are :

- **Layer 1 – The Physical Layer – :**

The physical layer coordinates the functions required to carry a bit stream over a physical medium. It deals with the mechanical and electrical specifications of the interface and transmission media. It also defines the procedures and functions that physical devices and interfaces have to perform for transmission to occur. The physical layer is also concerned with the following: [16]

Hardware Specifications Definition The details of operation of cables, connectors, wireless radio transceivers, network interface cards, and other hardware devices are generally a function of the physical layer.

Encoding and Signalling : The physical layer is responsible for various encoding and signalling functions that transform the data from bits that reside within a computer or another device into signals that can be sent over the network.

Data Transmission and Reception : After encoding the data appropriately, the physical layer actually transmits the data, and of course, receives it.

Topology and Physical Network Design : The physical layer is also considered the domain of many hardware related network design issues, such as local area network (LAN) and wide area network (WAN) topology.[17]

- **Layer 2 – The Data Link Layer – :**

The data link layer transforms the physical layer, a raw transmission facility, to a reliable link. It makes the physical layer appear error-free to the upper layer (network layer). Other responsibilities of the data link layer include the following: [16]

Data Framing : The data link layer is responsible for data framing, which is the final encapsulation of higher-level messages into frames that are sent over the net- work at the physical layer.

Addressing : The data link layer is the lowest layer in the OSI model that is concerned with addressing. It labels information with a particular destination location. Each device on a network has a unique number that is used by the data link layer protocol to ensure that data intended for a specific machine gets to it properly. This is usually called a hardware address (since it is intimately related with low-level hardware) or a MAC address.[17]

Error Detection and Handling : The data link layer handles errors that occur at the lower levels of the network stack.

Physical Layer Standards : The physical layer and the data link layer are very closely related.

The requirements for the physical layer of a network are often part of the data link layer standard that describes a particular technology. Certain physical layer hardware and encoding aspects are specified by the data link layer technology being used.[17]

- **Layer 3 – The Network Layer – :**

The network layer is responsible for the source-to-destination delivery of a packet, possibly across multiple networks (links). Whereas the data link layer oversees the delivery of the packet between two systems on the same network (link), the network layer ensures that each packet gets from its point of origin to its final destination.

If two systems are connected to the same link, there is usually no need for a network layer. However, if the two systems are attached to different networks (links) with connecting devices between the networks (links), there is often a need for the network layer to accomplish source-to-destination delivery. Other responsibilities of the network layer include the following: [16]

Routing The defining function of the network layer is routing/moving data across a series of interconnected networks. It is the job of the devices and software routines that function at the network layer to handle incoming packets from various sources, determine their final destination, and then figure out where they need to be sent to get them where they are supposed to go.

Datagram Encapsulation The network layer normally encapsulates messages received from higher layers by placing them into datagrams (also called packets) with a network layer header.

Fragmentation and Reassembly The network layer must send messages down to the data link layer for transmission. Some data link layer technologies limit the length of any message that can be sent. If the packet that the network layer wants to send is too large, the network layer must split the packet up (fragment it), send each piece to the data link layer, and then have the pieces reassembled once they arrive at the network layer on the destination machine. The IP is the best known example of a protocol that performs these functions.

Error Handling and Diagnostics The network layer uses special protocols to allow devices that are logically connected (or that are trying to route traffic) to exchange information about the status of hosts on the network or the devices themselves.[17]

- **Layer 4 – The Transport Layer – :**

The transport layer is responsible for process-to-process delivery of the entire message. A process is an application program running on the host. Whereas the network layer oversees source-to-destination deliveryvof individual packets, it does not recognize any relationship between those packets. It treats each one independently, as though each piece belonged to a separate message, whether or not it does. The transport layer, on the other hand, ensures that the whole message arrives intact and in order, overseeing both error control and flow control at the source-to-destination level. Other responsibilities of the transport layer include the following: [16]

Process-Level Addressing : Addressing at the transport layer is used to differentiate between software programs. This is part of what enables many different software programs to use a network layer protocol simultaneously.

Multiplexing and Demultiplexing : Using the process-level addresses, transport layer protocols on a sending device multiplex the data received from many application programs for transport, combining them into a single stream of data to be sent. The same protocols receive data and then demultiplex it from the incoming stream of datagrams, and direct each one to the appropriate recipient application processes.

Segmentation, Packaging, and Reassembly : The transport layer segments the large amounts of data it sends over the network into smaller pieces on the source machine, and then reassembles them on the destination machine.

Connection Establishment, Management, and Termination : Transport layer connection-oriented protocols are responsible for the series of communications required to establish a connection, maintain it as data is sent over it, and then terminate the connection when it is no longer required.

Flow Control : Transport layer protocols that offer reliable delivery also often implement flow-control features. These features allow one device in a communication to specify to another that it must throttle back the rate at which it is sending data. This will prevent the receiver from being bogged down with data. These features allow mismatches in speed between sender and receiver to be detected and handled.

- **Layer 5 – The Session Layer – :**

The services provided by the first four layers (physical, data link, network and transport) are not sufficient for some processes. The session layer is the network dialogue controller. It establishes, maintains, and synchronizes the interaction between communicating systems. Specific responsibilities of the session layer include the following:

Dialogue control : The session layer allows two systems to enter into a dialogue. It allows the communication between two processes to take place in either half- duplex (one way at a time) or full-duplex (two ways at a time) mode.

Synchronization : The session layer allows a process to add checkpoints (synchronization points) into a stream of data. For example, if a system is sending a file of 2,000 pages, it is advisable to insert checkpoints after every 100 pages to ensure that each 100-page unit is received and acknowledged independently. In this case, if a crash happens during the transmission of page 523, the only pages that need to be resent after system recovery are pages 501 to 523. Pages previous to 501 need not be resent.[16]

- **Layer 6 – The Presentation Layer – :**

The presentation layer is concerned with the syntax and semantics of the information exchanged between two systems. Specific responsibilities of the presentation layer include the following: [16]

Translation : Many different types of computers can exist on the same network, such as PCs, Macs, UNIX systems. Each has many distinct characteristics and represents data in different ways (with different character sets, for example). The presentation layer hides the differences between machines.

Compression Compression (and decompression) may be done at the presentation layer to improve the throughput of data.

Encryption Some types of encryption (and decryption) are performed at the presentation layer to ensure the security of the data as it travels down the protocol stack. [17]

- **Layer 7 – The Application Layer – :**

The application layer enables the user, whether human or software, to access the network. It provides user interfaces and support for services such as electronic mail, remote file access and transfer, shared database management, and other types of distributed information services. Specific services provided by the application layer include the following:

Network virtual terminal : A network virtual terminal is a software version of a physical terminal and allows a user to log on to a remote host. To do so, the application creates a software emulation of a terminal at the remote host. The user’s computer talks to the software terminal, which, in turn, talks to the host, and vice versa. The remote host believes it is communicating with one of its own terminals and allows you to log on.

File transfer, access, and management (FTAM) : This application allows a user to access files in a remote host (to make changes or read data), to retrieve files from a remote computer for use in the local computer, and to manage or control files in a remote computer locally.

E-mail services : This application provides the basis for e-mail forwarding and storage.

Directory services : This application provides distributed database sources and access for global information about various objects and services.[17]

II.2.2 The TCP/IP protocol suit :

The Internet Protocol Suite also known as TCP/IP is the set of communications protocols used for the Internet and other similar networks. It is named from two of the most important protocols in it: the Transmission Control Protocol (TCP) and the Internet Protocol (IP), which were the first two networking protocols defined in this standard. IP networking represents a synthesis of several developments that began to evolve in the 1960s and 1970s, namely the Internet and LANs (Local Area Networks), which emerged in the mid-to late-1980s, together with the advent of the World Wide Web in early 1990s.The Internet Protocol Suite, like many protocol suites, may be viewed as a set of layers. Each layer solves a set of problems involving the transmission of data, and provides a well-defined service to the upper layer protocols based on using services from some lower layers. Upper layers are logically closer to the user and deal with more abstract data, relying on lower layer protocols to translate data into forms that can eventually be physically transmitted.

II.2.2.1 Comparison between OSI and TCP/IP Protocol Suite :

The main differences between the two models are as follows :

1. OSI is a reference model and TCP/IP is an implementation of OSI model.

2. TCP/IP Protocols are considered to be standards around which the internet has developed. The OSI model however is a "generic, protocol-independent standard."
3. TCP/IP combines the presentation and session layer issues into its application layer.
4. TCP/IP combines the OSI data link and physical layers into the network access layer.
5. TCP/IP appears to be a simpler model and this is mainly due to the fact that it has fewer layers.
6. TCP/IP is considered to be a more credible model. This is mainly due to the fact because TCP/IP protocols are the standards around which the internet was developed therefore it mainly gains creditability due to this reason. Where as in contrast networks are not usually built around the OSI model as it is merely used as a guidance tool.
7. The OSI model consists of 7 architectural layers whereas the TCP/IP only has 4 layers .
8. The TCP/IP design generally favours decisions based on simplicity,efficiency and ease of implementation.[18]

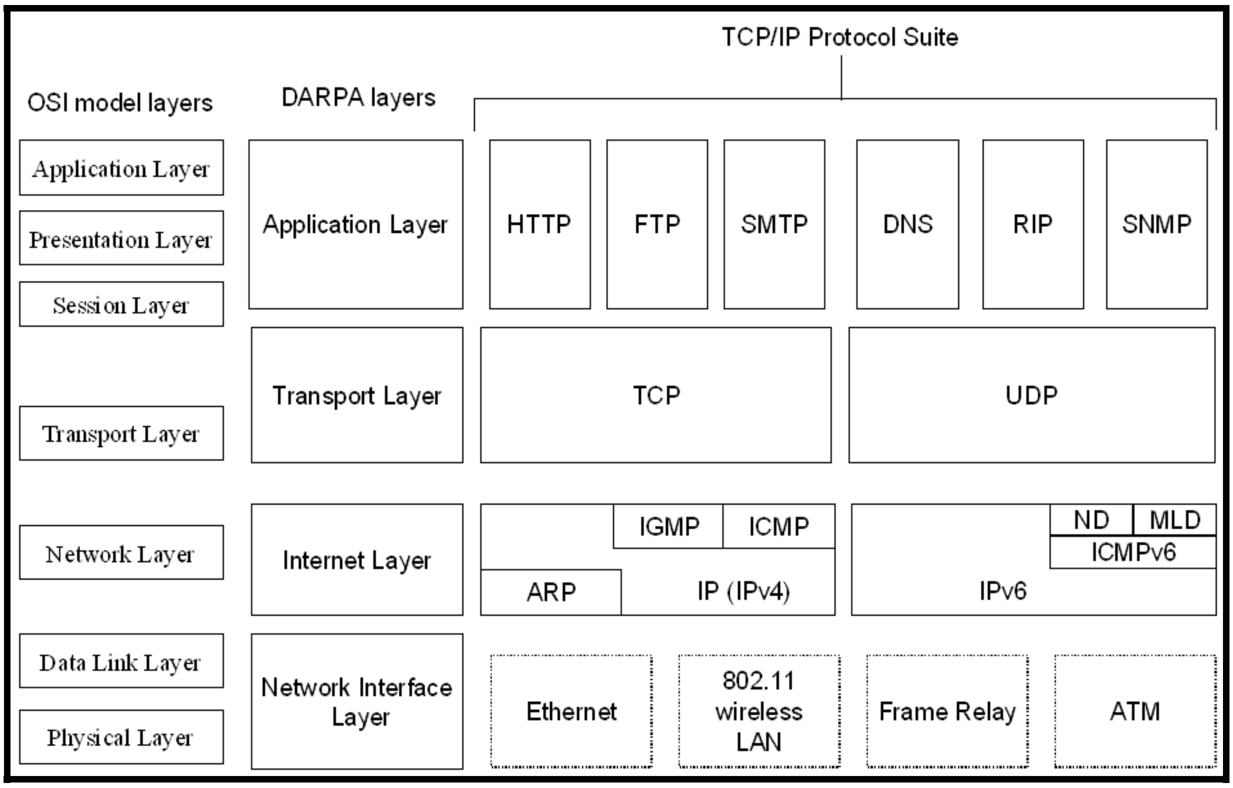


Figure II.2 - The OSI reference model and the architecture of the TCP/IP protocol suite. [19]

II.2.2.2 Layers in the TCP/IP Protocol Suite :

The TCP/IP protocol suit consists 4 architecture layers and they are physical network layer, internet layer, transport layer, and the application layer. We will explain each layer independently in this section.

◆ **Physical Network Layer :**

Network interface layer The network interface layer, also called the link layer or the data link layer, is the interface to the actual network hardware. This interface may or may not provide reliable delivery, and may be packet or stream oriented. In fact, TCP/IP does not specify any protocol here, but can use almost any network interface available, which illustrates the flexibility of the IP layer.

◆ **Internet Layer :**

The internetlayer, also called the internetwor layer or the network layer, provides the "virtual network" image of an internet (this layer shields the higher levels from the physical network architecture below it). Internet Protocol (IP) is the most important protocol in this layer.

It is a connectionless protocol that does not assume reliability from lower layers. IP does not provide reliability, flow control, or error recovery. These functions must be provided at a higher level.[20]

◆ **Transport Layer :**

The transport layer provides the end-to-end data transfer by delivering data from an application to its remote peer. Multiple applications can be supported simultaneously. The most used transport layer protocol is the Transmission Control Protocol (TCP), which provides connection-oriented reliable data delivery, duplicate data suppression, congestion control, and flow control.

Another transport layer protocol is the User Datagram Protocol. It provides connectionless, unreliable, best-effort service. As a result, applications using UDP as the transport protocol have to provide their own end-to-end integrity, flow control, and congestion control, if desired. Usually, UDP is used by applications that need a fast transport mechanism and can tolerate the loss of some data.[20]

◆ **Application Layer :**

The application layer is provided by the program that uses TCP/IP for communication. An application is a user process cooperating with another process usually on a different host (there is also a benefit to application communication within a single host). The interface between the application and transport layers is defined by port numbers and sockets.[20]

II.3 The IP Addresses version 4 :

TCP/IP uses 32-bit binary addresses as universal machine identifiers. Called Internet Protocol addresses or IP addresses, the identifiers are partitioned into two parts: a prefix identifies the network to which the computer attaches and the suffix provides a unique identifier for the computer on that network. The original IP addressing scheme is known as classful, with each prefix assigned to one of three primary classes. Leading bits define the class of an address; the classes are of unequal size. The classful scheme provides for 127 networks with over a million hosts each, thousands of networks with thousands of hosts each, and over a million networks with up to 254 hosts each. To make such addresses easier for humans to understand, they are written in dotted decimal notation, with the values of the four octets written in decimal, separated by decimal points. [21]

II.3.1 Classfull addressing :

The original IP addressing scheme is set up so that the dividing line occurs only in one of a few locations: on octet boundaries. Three main classes of addresses A, B, and C are differentiated based on how many octets are used for the network ID and how many for the host ID. For example, Class C addresses devote 24 bits to the network ID and 8 bits to the host ID. This type of addressing is now often referred to by the made-up word classful to differentiate it from the newer classless scheme. This most basic addressing type uses the simplest method to divide the network and host IDs: The class, and therefore the dividing point, are encoded into the first few bits of each address. Routers can tell from these bits which octets belong to which identifier. [17]

II.3.1.1 Classes and Blocks :

IP addresses were initially divided into several different categories, or classes, based on how many bits of the address were used for the network prefix and how many bits were used for hosts. The classful addressing system defined class A, B, C, D, and E address spaces. Classes A, B, and C were used for general networking. class D was designated for IP Multicast traffic, and class E was set aside for experimental use. The classes A, B, and C are characterized by the following:

1. In class A addresses, Each Class A network address has an 8-bit network prefix, with the highest order bit set to 0 (zero) and a 7-bit network number as the Figure II.3 shows, followed by a 24-bit host number. Today, Class A networks are referred to as “/8s” (pronounced “slash eight” or just “eights”) since they have an 8-bit network prefix. The /8 address space is 50 percent of the total IPv4 unicast address space (check Figure II.6).[22]

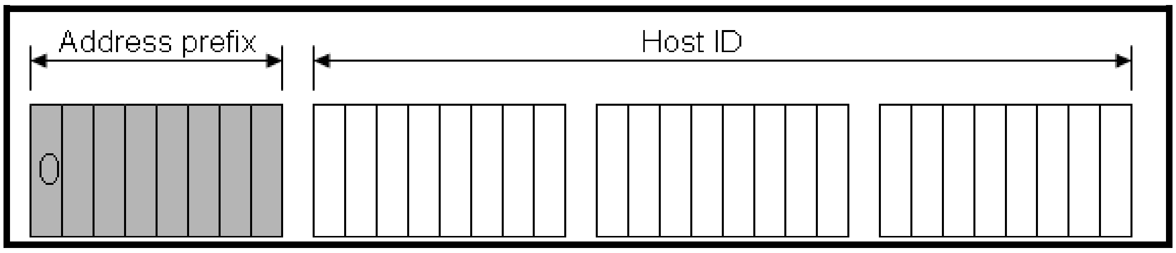


Figure II.3 - Structure of class A addresses. [23]

2. In class B addresses, Each Class B network address has a 16-bit network prefix, with the two highest order bits set to 1-0 and a 14-bit network number (Figure II.4), followed by a 16-bit host number. Class B networks are now referred to as “/16s” since they have a 16-bit network prefix. The entire /16 address represents 25 percent of the total IPv4 unicast address space as we can see in Figure II.6 .[22]

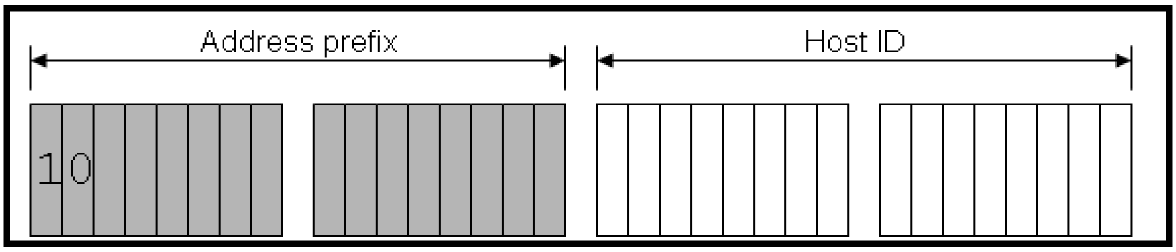


Figure II.4 - Structure of class B addresses.[24]

3. In class C addresses, each Class C network address has a 24-bit network prefix, with the three highest order bits set to 1-1-0 and a 21-bit network number, followed by an 8-bit host number as the Figure II.5 shows. Class C networks are now referred to as “/24s”

since they have a 24-bit network prefix. The entire /24 address represents 12.5 percent (or one eighth) of the total IPv4 unicast address space.[22]

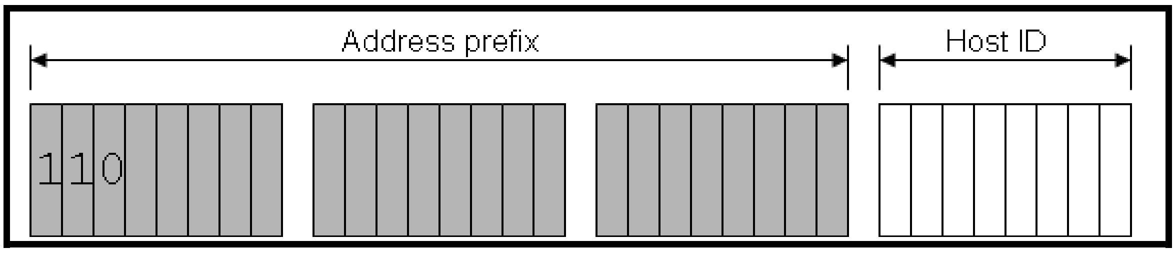


Figure II.5 - Structure of class C addresses. [25]

4. Other Classes, In addition to the three most popular classes, there are two additional classes. Class D addresses have their leading four bits set to 1-1-1-0 and are used to support IP Multi-casting. Class E addresses have their leading four bits set to 1-1-1-1 and are reserved for experimental use.[22]

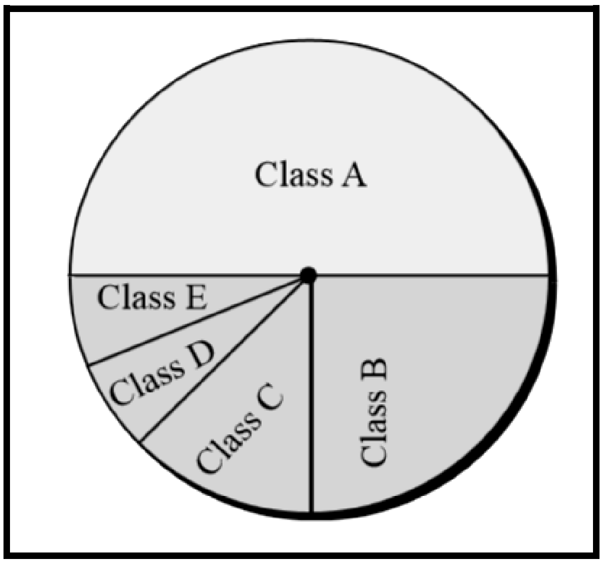


Figure II.6 - Classful addresses.[16]

II.3.1.2 Subnetting and Supernetting :

Subnetting allows you to create multiple logical networks that exist within a single class A, B, or C network. Without subnetting you could create only one network from a class A, B, or C network, which would not be very useful. Each Layer 2 segment on a network must have a unique network prefix, and each device on that segment must have a unique host address. Subnetting allows major networks (class A, B, or C networks) to be divided into smaller subnetworks (subnets). For example, instead of having a single network with 16 million hosts, a class A network could be divided into 65,535 networks, each with 254 host addresses; this is a much more practical network size for most organizations. To subnet a network, bits from the host portion of the address are used to create additional subnets.

For supernetting, or classless interdomain routing (CIDR), you use bits from the network prefix to create a larger block of addresses. For example you can take several class C blocks and make the equivalent of one class B block that is, 192.168.0.0 can be given a mask of 255.255.0.0 (or /16), effectively killing the old class-based system and turning all IP addresses into a single block that can be divided among organizations much more effectively. The hypothetical organization mentioned above with a few thousand connected users can be given an IP block with a /20 mask, giving them 4096 addresses to work with, which is a much more reasonable number for their size.[26]

II.3.2 Classless addressing :

In the classless system, the classes of the original IP addressing scheme are tossed out the window. The division between the network ID and host ID can occur at an arbitrary point, not just on octet boundaries, as in the classful scheme. The dividing point is indicated by putting the number of bits used for the network ID, called the prefix length, after the address. (Recall that the network ID bits are also sometimes called the network prefix, so the network ID size is the prefix length.) For example, if 227.82.157.177 is part of a network where the first 27 bits are used for the network ID, that network would be specified as 227.82.157.160/27. The /27 is conceptually the same as the 255.255.255.224 subnet mask, since it has 27 one bits followed by 5 zeros. [17]

II.3.2.1 Variable-Length Blocks :

In classful addressing the whole address space was divided into five classes. Although each organization was granted one block in class A, B, or C, the size of the blocks was predefined; the organization needed to choose one of the three block sizes. The only block in class D and the only block in class E were reserved for a special purpose. In classless addressing, the whole address space is divided into variable length blocks. Theoretically, we can have a block of 2^0, 2^1, 2^2, . . . , 2^32 addresses. The only restriction, is that the number of addresses in a block needs to be a power of 2. An organization can be granted one block of addresses. Figure II.7 shows the division of the whole address space into non overlapping blocks.[16]

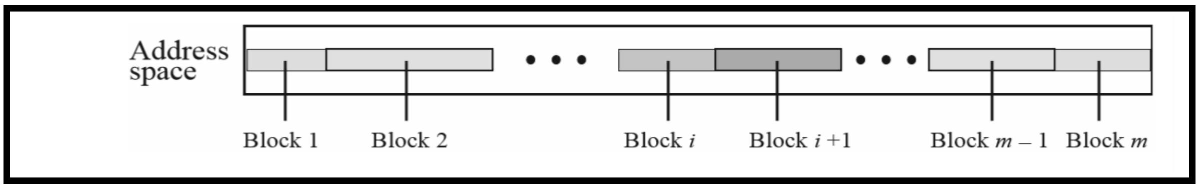


Figure II.7- Variable-length blocks in classless addressing.[16]

II.3.2.2 Two-Level Addressing :

In classful addressing, two-level addressing was provided by dividing an address into net-id and host-id. The net-id defined the network; the host-id defined the host in the network. The same idea can be applied in classless addressing. When an organization is granted a block of addresses, the block is actually divided into two parts, the prefix and the suffix. The prefix plays the same role as the net-id, the suffix plays the same role as the host-id. All addresses in the block have the same prefix; each address has a different suffix. Figure II.8 shows the prefix and suffix in a classless block.

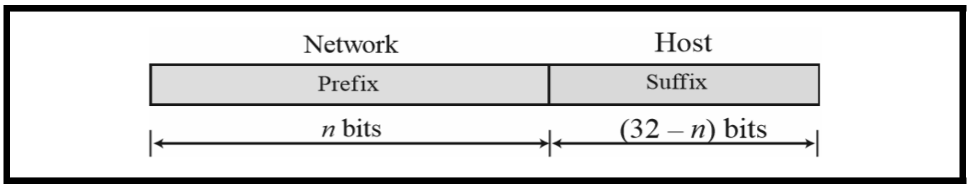


Figure II.8 - Prefix and suffix

In classful addressing, the length of the net-id, n, depends on the class of the address; it can be only 8, 16, or 24. In classless addressing, the length of the prefix, n, depends on the size of the block; it can be 0, 1, 2, 3, . . . , 32. In classless addressing, the value of n is referred to as prefix length; the value of 32 − n is referred to as suffix length.[16]

II.3.2.4 Subnetting :

Three levels of hierarchy can be created using subnetting. An organization (or an ISP) that is granted a range of addresses may divide the range into several subranges and assign each subrange to a subnetwork (or subnet). The concept is the same as we discussed for classful addressing. Note that nothing stops the organization from creating more levels. A subnetwork can be divided into several sub-subnetworks. A sub-subnetwork can be divided into several sub-sub-subnetworks. And so on.

II.3.3 Special addresses :

- **127.0.0.0/8 :** This block is assigned for use as the Internet host loopback address. A datagram sent by a higher-level protocol to an address anywhere within this block loops back inside the host. This is ordinarily implemented using only 127.0.0.1/32 for loopback. Addresses within the entire 127.0.0.0/8 block do not legitimately appear on any network anywhere.
- **169.254.0.0/16 :** This is the "link local" block. It is allocated for communication between hosts on a single link. Hosts obtain these addresses by auto-configuration, such as when a DHCP server cannot be found.
- **172.16.0.0/12** : This block is set aside for use in private networks. Addresses within this block do not legitimately appear on the public Internet. These addresses can be used without a coordination with IANA or an Internet registry.
- **192.0.0.0/24 :** This block is reserved for IETF protocol assignments.
- **192.0.2.0/24 :** This block is assigned as "TEST-NET-1" for use in documentation and example code. It is often used in conjunction with domain names example.com or example.net in vendor and protocol documentation. Addresses within this block do not legitimately appear on the public Internet and can be used without any coordination with IANA or an Internet registry.
- **192.88.99.0/24 :** This block is allocated for use as 6to4 relay anycast addresses. In contrast with previously described blocks, packets destined to addresses from this block do appear in the public Internet.
- **192.168.0.0/16 :** This block is set aside for use in private networks. Addresses within this block do not legitimately appear on the public Internet. These addresses can be used

without any coordination with IANA or an Internet registry.

- **198.18.0.0/15 :** This block has been allocated for use in benchmark tests of network interconnect devices.
- **198.51.100.0/24 :** This block is assigned as "TEST-NET-2" for use in documentation and example code. It is often used in conjunction with domain names example.com or example.net in vendor and protocol documentation. Addresses within this block do not legitimately appear on the public Internet and can be used without any coordination with IANA or an Internet registry.
- **203.0.113.0/24 :** This block is assigned as "TEST-NET-3" for use in documentation and example code. It is often used in conjunction with domain names example.com or example.net in vendor and protocol documentation. Addresses within this block do not legitimately appear on the public Internet and can be used without any coordination with IANA or an Internet registry.
- **224.0.0.0/4 :** This block, formerly known as the Class D address space, is allocated for use in IPv4 multicast address assignments.
- **240.0.0.0/4 :** This block, formerly known as the Class E address space, is reserved for future use. [27]

II.4 The Transmission Control Protocol TCP :

In this section of chapter, we will try to clear more about the Transmission Control Protocol TCP.

II.4.1 Ports and Sockets :

Before we get into explaining the TCP protocol, we have to talk about two main concepts (ports and sockets) that we will need to have the full picture about the transmission control protocol .

II.4.1.1 Ports :

Each process that wants to communicate with another process identifies itself to the TCP/IP protocol suite by one or more ports. A port is a 16-bit number used by the host-to-host protocol to identify to which higher-level protocol or application program (process) it must deliver incoming messages. There are two types of ports:

1. Well-known : Well-known ports belong to standard servers, for example, Telnet uses port 23. Well-known port numbers range between 1 and 1023 (prior to 1992, the range between 256 and 1023 was used for UNIX-specific servers). Well-known port numbers are typically odd, because early systems using the port concept required an odd/even pair of ports for duplex operations. Most servers require only a single port.
2. Ephemeral: Some clients do not need well-known port numbers because they initiate communication with servers, and the port number they are using is contained in the UDP/TCP datagrams sent to the server. Each client process is allocated a port number, for as long as it needs, by the host on which it is running. Ephemeral port numbers have values greater than 1023, normally in the range of 1024 to 65535. Ephemeral ports are not controlled by IANA and can be used by ordinary user-developed programs on most systems. Confusion, due to two different applications trying to use the same port numbers on one host, is avoided by writing those applications to request an available port from TCP/IP. Because this port number is dynamically assigned, it can differ from one invocation of an application to the next. [20]

II.4.1.2 Sockets :

A socket is an abstraction through which an application may send and receive data, in much the same way as an open file handle allows an application to read and write data to stable storage. A socket allows an application to plug in to the network and communicate with other applications that are plugged in to the same network. Information written to the socket by an application on one machine can be read by an application on a different machine and vice versa. Different types of sockets correspond to different underlying protocol suites and different stacks of protocols within a suite. The main types of sockets in TCP/IP today are stream sockets and datagram sockets. Stream sockets use TCP as the end-to-end protocol (with IP underneath) and thus provide a reliable byte-stream service. A TCP/IP stream socket represents one end of a TCP connection. Datagram sockets use UDP (again, with IP underneath) and thus provide a best effort datagram service that applications can use to send individual messages up to about 65,500 bytes in length. Stream and datagram sockets are also supported by other protocol suites. A TCP/IP socket is uniquely identified by an Internet address, an end-to-end protocol (TCP or UDP), and a port number. As you proceed, you will encounter several ways for a socket to become bound to an address.

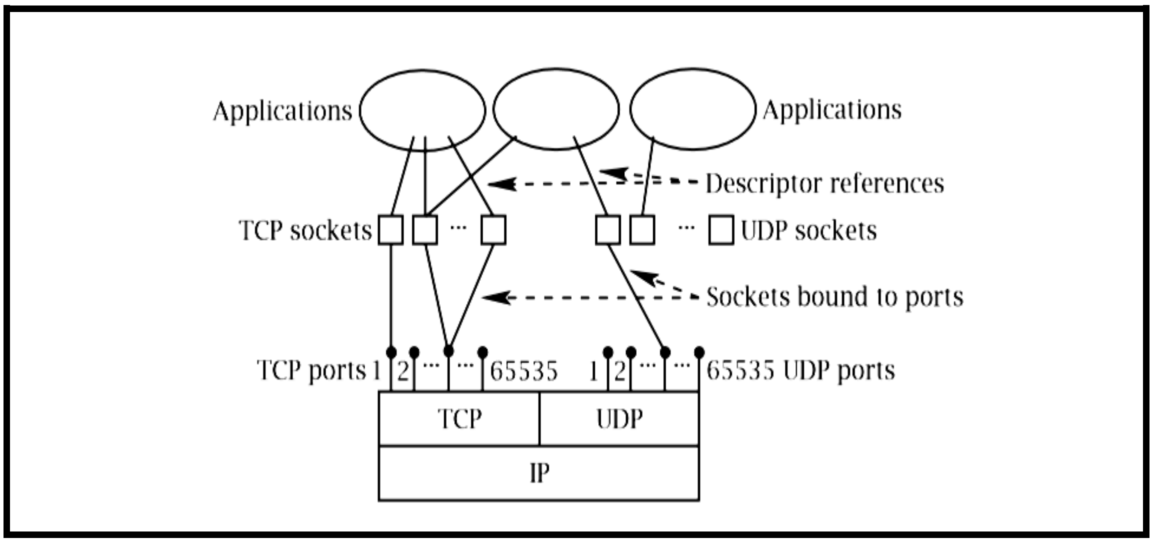


Figure II.9 - Sockets, protocols, and ports.

Figure II.9 depicts the logical relationships among applications, socket abstractions, protocols, and port numbers within a single host. There are several things to note about these relationships. First, a program can have multiple sockets in use at the same time. Second, multiple programs can be using the same socket abstraction at the same time, although this is less common. The figure shows that each socket has an associated local TCP or UDP port, which is used to direct incoming packets to the application that is supposed to receive them. Earlier we said that a port identifies an application on a host. Actually, a port identifies a socket on a host. There is more to it than this, however, because as Figure II.9 shows, more than one socket can be associated with one local port.[28]

II.4.2 What's a TCP protocol ?

Transmission Control Protocol (TCP) is a required TCP/IP standard defined as a highly reliable host-to-host protocol between hosts in packet-switched computer communication networks, and in interconnected systems of such networks.

II.4.3 TCP features :

TCP has several features that are briefly :

1) **<u>Numbering System :</u>** Although the TCP software keeps track of the segments being transmitted or received.
2) **<u>Flow Control :</u>** TCP provides flow control. The sending TCP controls how much data can be accepted from the sending process; the receiving TCP controls how much data can to be sent by the sending TCP. This is done to prevent the receiver from being overwhelmed with data. The numbering system allows TCP to use a byte-oriented flow control.
3) **<u>Error Control :</u>** To provide reliable service, TCP implements an error control mechanism. Although error control considers a segment as the unit of data for error detection (loss or corrupted segments), error control is byte-oriented.
4) **<u>Congestion Control :</u>** TCP, takes into account congestion in the network. The amount of data sent by a sender is not only controlled by the receiver (flow control), but is also determined by the level of congestion, if any, in the network.[16]

II.4.4 TCP connection :

TCP is connection-oriented. A connection-oriented transport protocol establishes a virtual path between the source and destination. All of the segments belonging to a message are then sent over this virtual path. Using a single virtual pathway for the entire message facilitates the acknowledgement process as well as retransmission of damaged or lost frames. You may wonder how TCP, which uses the services of IP, a connectionless protocol, can be connection-oriented. The point is that a TCP connection is virtual, not physical. TCP operates at a higher level. TCP uses the services of IP to deliver individual segments to the receiver, but it controls the connection itself. If a segment is lost or corrupted, it is retransmitted. Unlike TCP, IP is unaware of this retransmission. If a segment arrives out of order, TCP holds it until the missing segments arrive; IP is unaware of this reordering. In TCP, connection-oriented transmission requires three phases: connection establishment, data transfer, and connection termination.

1) **<u>Connection Establishment :</u>** TCP transmits data in full-duplex mode. When two TCPs

in two machines are connected, they are able to send segments to each other simultaneously. This implies that each party must initialize communication and get approval from the other party before any data are transferred.

2) **Data Transfer :** After connection is established, bidirectional data transfer can take place. The client and server can send data and acknowledgements in both directions.
3) **Connection Termination ;** Any of the two parties involved in exchanging data (client or server) can close the connection, although it is usually initiated by the client.
4) **Connection Reset :** TCP at one end may deny a connection request, may abort an existing connection, or may terminate an idle connection.[16]

II.5 Application layer :

In this section of the chapter, we will have a close view about the Client-Server model as we talk about the HTTP protocol.

II.5.1 Client-Server Model :

The purpose of a network, or an internetwork, is to provide services to users: A user at a local site wants to receive a service from a computer at a remote site. One way to achieve this purpose is to run two programs. A local computer runs a program to request a service from a remote computer; the remote computer runs a program to give service to the requesting program. This means that two computers, connected by an internet, must each run a program, one to provide a service and one to request a service.

II.5.1.1 Server :

A server is a program running on the remote machine providing service to the clients. When it starts, it opens the door for incoming requests from clients, but it never initiates a service until it is requested to do so. A server program is an infinite program. When it starts, it runs infinitely unless a problem arises. It waits for incoming requests from clients. When a request arrives, it responds to the request, either iteratively or concurrently.

II.5.2.2 Client :

A client is a program running on the local machine requesting service from a server. A client program is finite , which means it is started by the user (or another application program) and terminates when the service is complete. Normally, a client opens the communication channel using the IP address of the remote host and the well-known port address of the specific server program running on that machine. After a channel of communication is opened, the client sends its request and receives a response. Although the request-response part may be repeated several times, the whole process is finite and eventually comes to an end.

II.5.3 HTTP :

HTTP is a simple protocol. The client establishes a TCP connection to the server, issues a request, and reads back the server's response. The server denotes the end of its response by closing the connection. The file returned by the server normally contains pointers (hypertext links) to other files that can reside on other servers. The simplicity seen by the user is the apparent ease of following these links from server to server.[29]

II.5.3.1 HTTP features :

The protocol used for communication between a browser and a Web server or between intermediate machines and Web servers is known as the HyperText Transfer Protocol (HTTP). HTTP has the following set of characteristics:

1) **Application Level :** H'ITP operates at the application level. It assumes a reliable, connection-oriented transport protocol such as TCP, but does not provide reliability or retransmission itself.
2) **Request/Response :** Once a transport session has been established, one side (usually a browser) must send an HTTP request to which the other side responds.
3) **Stateless :** Each H'ITP request is self contained; the server does not keep a history of previous requests or previous sessions.
4) **Bi-Directional Transfer :** In most cases, a browser requests a Web page, and the server transfers a copy to the browser. HTTP also allows transfer from a browser to a server

(e.g., when a user submits a so-called "form").

5) **Capability Negotiation :** H'ITP allows browsers and servers to negotiate details such as the character set to be used during transfers. A sender can specify the capabilities it offers and a receiver can specify the capabilities it accepts.
6) **Support For Caching :** To improve response time, a browser caches a copy of each Web page it retrieves. If a user requests a page again, HTTP allows the browser to interrogate the server to determine whether the contents of the page has changed since the copy was cached.
7) **Support For Intermediaries :** HTTP allows a machine along the path between a browser and a server to act as a proxy server that caches Web pages and answers a browser's request from its cache.

II.6 New generation :

The internet has changed a lot since the 90s, it has known big jumps on all the levels.

II.6.1 The Ipv6 Addressing :

In the early 1990's the Internet Engineering Task force began an effort to develop a successor to the IPv4 protocol. A prime motivation for this effort was the realization that the 32-bit IP address space was beginning to be used up, with new networks and IP nodes being attached to the Internet (and being allocated unique IP addresses) at a breathtaking rate. To respond to this need of a large IP address space, a new IP protocol, IPv6, was developed. The designers of IPv6 also took this opportunity to tweak and augment other aspects of IPv4, based on the accumulated operational experience with IPv4. An IPv6 address is 128 bits or 16 bytes (octet) long as shown in Figure II.11. The address length in IPv6 is four times of the length address in Ipv4.[30]

II.6.1.1 Design goals of Ipv6 :

The problem of addressing was the main motivation for creating IPv6. Unfortunately, this has caused many people to think that the address space expansion is the only change made in IP, which is definitely not the case. Since making a change to IP is such a big deal, it’s something done rarely. It made sense to correct not just the addressing issue, but also to update the protocol in a number of other respects in order to ensure its viability. In fact, even the addressing changes in IPv6 go far beyond just adding more bits to IP address fields. Some of the most important goals in designing IPv6 include the following:

1) **Larger Address Space :** IPv6 needed to provide more addresses for the growing Internet.
2) **Better Management of Address Space :** Developers wanted IPv6 to include not only more addresses, but also a more capable way of dividing the address space and using the bits in each address.
3) **Easier TCP/IP Administration :** The designers of IPv6 hoped to resolve some of the current labour intensive requirements of IPv4, such as the need to configure IP addresses. Even though tools like DHCP eliminate the need to manually configure many hosts, it only partially solves the problem.
4) **Better Support for Security :** IPv4 was designed at a time when security wasn't much of an issue because there were a relatively small number of networks on the Internet, and those networks' administrators often knew each other. Today, security on the public Internet is a big issue, and the future success of the Internet requires that security concerns be resolved.
5) **Better Support for Mobility :** When IPv4 was created, there really was no concept of mobile IP devices. The problems associated with computers that move between networks led to the need for Mobile IP. IPv6 builds on Mobile IP and provides mobility support within IP itself.[17]

II.6.1.2 Notation :

Increasing the size of IP addresses from 32 bits to 128 bits expands the address space to a gargantuan size, ensuring that we will never again run out of IP addresses, and allowing us flexibility in how they are assigned and used. Unfortunately, there are some drawbacks to this method, and one of them is that 128-bit numbers are very large, which makes them awkward and difficult to use.

- **IPv6 Addresses : Too Long For Dotted Decimal Notation :** Computers work in binary,

and they have no problem dealing with long strings of ones and zeroes, but humans find them confusing. Even the 32-bit addresses of IPv4 are cumbersome for us to deal with, which is why we use dotted decimal notation for them unless we need to work in binary (as with subnetting). However, IPv6 addresses are so much larger than IPv4 addresses that even using dotted decimal notation becomes problematic. To use this notation, we would split the 128 bits into 16 octets and represent each with a decimal number from 0 to 255. However, we would end up not with 4 of these numbers, but 16. A typical IPv6 address in this notation would appear as follows:
128.91.45.157.220.40.0.0.0.0.252.87.212.200.31.255
The binary and dotted decimal representations of this address are shown near the top of Figure II.11. In either case, the word "elegant" doesn't exactly spring to mind.[31]

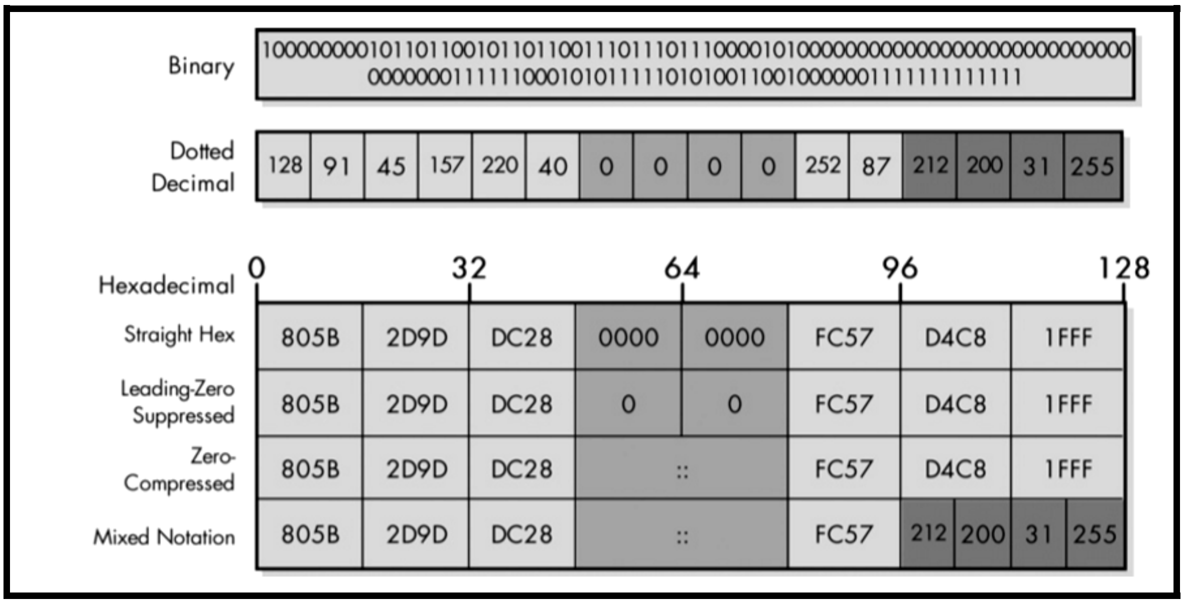


Figure II.10 - Binary, Decimal and Hexadecimal representation of Ipv6 addresses.[17]

The top two rows show binary and dotted decimal representations of an IPv6 address; neither is commonly used (other than by computers themselves!) The top row of the lower table shows the full hexadecimal representation, while the next two rows illustrate zero suppression and compression. The last row shows mixed notation, where the final 32 bits of an IPv6 address are shown in dotted decimal notation. This is most commonly used for embedded IPv4 addresses.[31]

II.6.1.3 IPv6 Address Types :

Like IPv4, IPv6 associates an address with a specific network connection, not with a specific computer. Thus, address assignments are similar to IPv4: an IPv6 router has two or more addresses, and an IPv6 host with one network connection needs only one address. IPv6 also retains (and extends) the IPv4 address hierarchy in which a physical network is assigned a prefix. However, to make address assignment and modification easier, IPv6 permits multiple prefixes to be assigned to a given network, and allows a computer to have multiple, simultaneous addresses assigned to a given interface. In addition to permitting multiple, simultaneous addresses per network connection, IPv6 expands, and in some cases unifies, IPv4 special addresses. In general, a destination address on a datagram falls into one of three categories: [21]

1) **Unicast :** A unicast address defines a single interface (computer or router). The packet sent to a unicast address will be routed to the intended recipient. As we see shortly, IPv6 has designated a large block from which unicast addresses can be assigned to interfaces. [16]
2) **Anycast :** An anycast address defines a group of computers that all share a single address. A packet with an anycast address is delivered to only one member of the group, the most reachable one. An anycast communication is used, for example, when there are several servers that can respond to an inquiry. The request is sent to the one that is most reachable. The hardware and software generate only one copy of the request; the copy reaches only one of the servers. IPv6 does not designate a block for anycasting; the addresses are assigned from the unicast block.[16]
3) **Multicast :** A multicast address also defines a group of computers. However, there is a difference between anycasting and multicasting. In multicasting, each member of the group receives a copy. The IPv6 has designated a block for multicasting from which the same address is assigned to the members of the group.[16]

II.6.2 Android :

Android combines the ubiquity of cell phones, the excitement of open source software, and the corporate backing of Google and other Open Handset Alliance members like Intel, TI, T-Mobile, and NTT DoCoMo.

Android powers hundreds of millions of mobile devices in more than 190 countries around the world. It's the largest installed base of any mobile platform and growing fast every day another million users power up their Android devices for the first time and start looking for apps, games, and other digital content.

Android gives you a world-class platform for creating apps and games for Android users everywhere, as well as an open marketplace for distributing to them instantly. In the next chapter we will talk more about it.

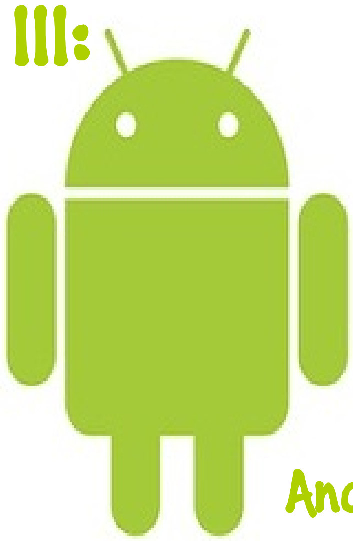
II.7 Conclusion :

The Internet's general concept is basically the server/client philosophy. It depends on our needs to define the type the protocol that can be used.

The HTTP protocol is the most used protocol over internet, that is because its simplicity first and its flexibility. Nowadays, the HTTP protocol isn't only used as request documents protocol, it has been used for different purposes .

The sever is machine by itself in most cases, specially when you need to handle a lot of clients. But in this project, the server will be a software that will be set on ordinary computer since it will only deal with one client.

Chapter III:



Android's Fundamentals

Chapter III : Android's Fundamentals

III.1 Introduction :

Android is a complete operating environment based upon the Linux® V2.6 kernel. Initially, the deployment target for Android was the mobile-phone arena, including smart phones and lower-cost flip-phone devices. However, Android's full range of computing services and rich functional support have the potential to extend beyond the mobile-phone market. Android can be useful for other platforms and applications. In this chapter, we will get an introduction to the Android platform and its architecture as we look to a brief history about it and understand Android's philosophy and the required tools.

III.2 A brief history of Android :

Android was originally created by Andy Rubin, Rich Miner, Nick Sears and Chris White as an operating system for mobile phones, around October,2003. In August 2005, Google acquired Android Inc., and made Andy Rubin the Director of Mobile Platforms for Google.

November 5th,2007, led by Google and a member of 34 technology and mobile companies such as HTC, Sony, Dell, Intel, Motorola, Qualcomm, Texas Instruments, Samsung Electronics, LG Electronics, T-Mobile, Sprint Nextel, Nvidia, and Wind River Systems established together The Open Handset Alliance (OHA). Android, the flagship software of the alliance, is based on an open source license and competes against mobile platforms from Apple, Microsoft, Nokia (Symbian), HP (formerly Palm). They have come together to accelerate innovation in mobile and offer consumers a richer, less expensive, and better mobile experience. Together we have developed Android that is described as "the first complete, open, and free mobile platform".

November 12th,2007 , they released Android Beta SDK . But the first Android device had to wait till September 23rd,2008 with the HTC Dream (G1), featuring Android 1.0. The next couple years knew fast improvement as the Table III-1 shows, each version had new features and better performance.

After the release of the Android 1.0 by HTC, a lot of others mobile-phone companies joined the new wave and so Android knew a lot of changes and improvement over the next years as Table III.1 shows .

Version	Codename	API Level	Market Share	Released Date
1	--	1	--	September 23rd,2008
1.1	Petit Four	2	--	February 9th,2009
1.5	Cupcake	3	0.20%	April 30th, 2009
1.6	Donut	4	0.40%	September 15th, 2009
2.1	Eclair	7	3.70%	January 12nd, 2010
2,2	FroYo	8	14%	May 20th, 2010
2,3,2	Gingerbread	9	0.30%	December 9th,2011
2,3,7	Gingerbread	10	57.20%	September 21st,2011
3,0	Honeycomb	11	--	February 22nd,2011
3.1	Honeycomb	12	0.50%	May 10th,2011
3.2	Honeycomb	13	1.60%	July 15th,2011
4.0.1	Ice Cream Sandwich	14	0.10%	October 19th,2011
4.0.4	Ice Cream Sandwich	15	20.80%	November 28th,2011
4.1	Jelly Bean	16	1.20%	July 9th,2012
4.2	Jelly Bean	17	--	November 13th, 2012
4,2,1	Jelly Bean	17	--	December 27th, 2012
4,2,2	Jelly Bean	17	--	February 11th, 2013

Table III.1 - Android's development.

III.3 Android features :

There are already many mobile platforms on the market today, including Symbian, iOS (iPhoneOS), Windows Mobile, BlackBerry, Java Mobile Edition, Linux Mobile (LiMo), and more. Although some of its features have appeared before, Android is the first environment that combines the following:

- **<u>A truly open free development platform based on Linux and open source:</u>** Handset makers like it because they can use and customize the platform without paying a royalty.

Developers like it because they know that the platform “has legs” and is not locked into any one vendor that may go under or be acquired.

- **A component-based architecture inspired by Internet mashups :** Parts of one application can be used in another in ways not originally envisioned by the developer. You can even replace built-in components with your own improved versions. This will unleash a new round of creativity in the mobile space.
- **Tons of built-in services out of the box:** Location-based services use GPS or cell tower triangulation to let you customize the user experience depending on where you are. A full-powered SQL database lets you harness the power of local storage for occasionally connected computing and synchronization. Browser and map views can be embedded directly in your applications. All these built-in capabilities help raise the bar on functionality while lowering your development costs.
- **Automatic management of the application life cycle:** Programs are isolated from each other by multiple layers of security, which will provide a level of system stability not seen before in smart phones. The end user will no longer have to worry about what applications are active or close some programs so that others can run. Android is optimized for low power, low-memory devices in a fundamental way that no previous platform has attempted.
- **High-quality graphics and sound:** Smooth, antialiased 2D vector graphics and animation inspired by Flash are melded with 3D accelerated OpenGL graphics to enable new kinds of games and business applications. Codecs for the most common industry-standard audio and video formats are built right in, including H.264 (AVC), MP3, and AAC.
- **Portability across a wide range of current and future hardware:** All your programs are written in Java and executed by Android’s Dalvik virtual machine, so your code will be portable across ARM, x86, and other architectures. Support for a variety of input methods is included such as keyboard, touch, and trackball. User interfaces can be customized for any screen resolution and orientation.

Android offers a fresh take on the way mobile applications interact with users, along with the technical underpinnings to make it possible. But the best part of Android is the software that you are going to write for it.[32]

III.3 Android layouts (platform) :

With Android's breadth of capabilities, it would be easy to confuse it with a desktop operating system. Android is a layered environment built upon a foundation of the Linux kernel, and it includes rich functions. The UI subsystem includes:

- Windows
- Views
- Widgets for displaying common elements such as edit boxes, lists, and drop-down lists.

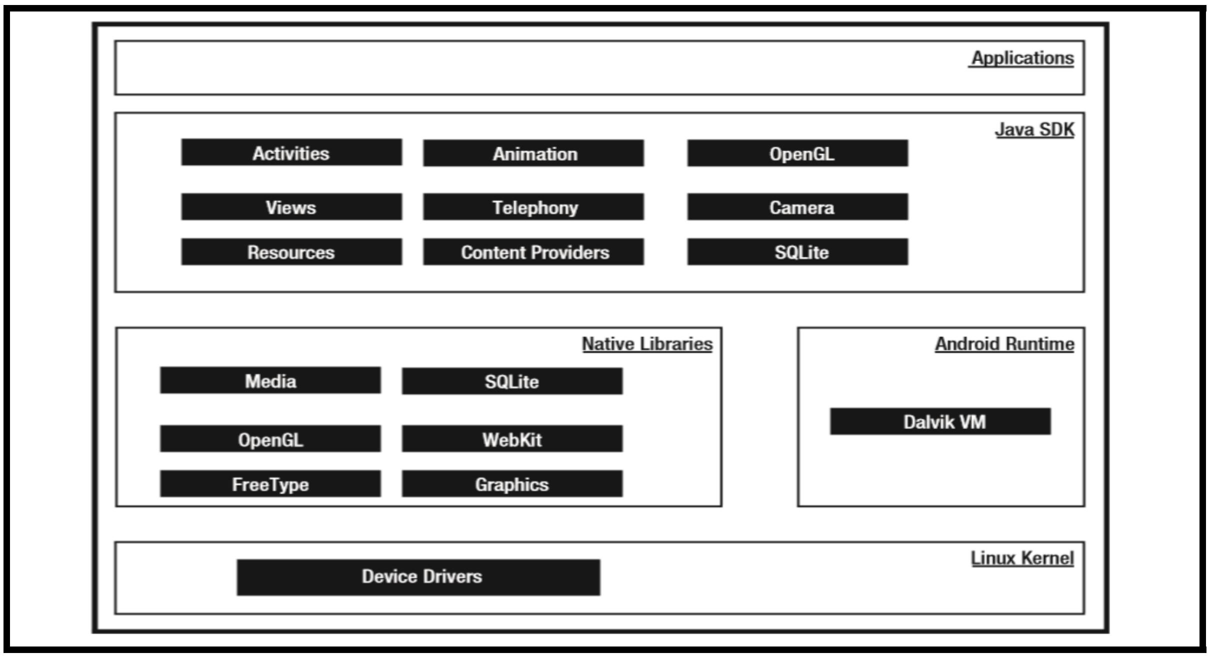


Figure III.1- Android software layers.[33]

Like we mentioned before, at the core of the Android Platform is Linux kernel version 2.6.29, responsible for device drivers, resource access, power management, and other OS duties. The supplied device drivers include Display, Camera, Keypad, WiFi, 3G, Flash Memory, Audio, and IPC (inter-process communication). Although the core is Linux, the majority (if not all) of the applications on an Android device such as the T-Mobile G1 or Motorola Droid are developed in Java and run through the Dalvik VM. Sitting at the next level, on top of the kernel, are a number of C/C++ libraries such as OpenGL, WebKit, FreeType, Secure Sockets Layer (SSL), the C runtime library (libc), SQLite, and Media. The system C library based on Berkeley Software Distribution (BSD) is tuned (to roughly half its original size) for embedded Linux-based devices. The media libraries are based on PacketVideo’s OpenCORE. These libraries are responsible for recording and playback of audio and video formats. A library called Surface Manager controls access to the display system and supports 2D and 3D.

The WebKit library is responsible for browser support; it is the same library that supports Google Chrome and Apple’s Safari. The FreeType library is responsible for font support.

SQLite is a relational database that is available on the device itself. SQLite is also an independent open source effort for relational databases and not directly tied to Android. You can acquire and use tools meant for SQLite for Android databases as well.

Most of the application framework accesses these core libraries through the Dalvik VM, the gateway to the Android Platform. As we indicated in the previous sections, Dalvik is optimized to run multiple instances of VMs. As Java applications access these core libraries, each application gets its own VM instance.

The Android Java API’s main libraries include telephony, resources, locations, UI, content providers (data), and package managers (installation, security, and so on). Programmers develop end-user applications on top of this Java API. Some examples of end-user applications on the device include Home, Contacts, Phone, Browser, and so on.

Android also supports a custom Google 2D graphics library called Skia, which is written in C and C++. Skia also forms the core of the Google Chrome browser. The 3D APIs in Android, however, are based on an implementation of OpenGL ES from the Khronos group .

OpenGL ES contains subsets of OpenGL that are targeted toward embedded systems. From a media perspective, the Android Platform supports the most common formats for audio, video, and images. From a wireless perspective, Android has APIs to support Bluetooth, EDGE, 3G, WiFi, and Global System for Mobile Communication (GSM) telephony, depending on the hardware.[34]

III.4 Android architecture :

Android runs atop a Linux kernel. Android applications are written in the Java programming language, and they run within a virtual machine (VM). It's important to note that the VM is not a JVM as you might expect, but is the Dalvik Virtual Machine, an open source technology. Each Android application runs within an instance of the Dalvik VM, which in turn resides within a Linux-kernel managed process, as shown below in Figure III-2 :[33]

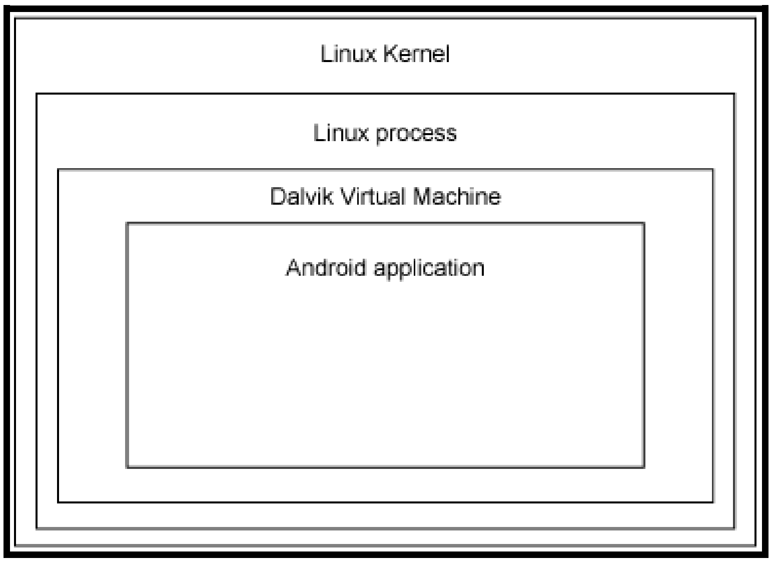


Figure III.2 - Dalvik Virtual Machine.[33]

An Android application is “stratified.” Its functionality is spelled out via Java code, its design via XML markup, and its privileges via the Android Manifest XML file in a way that is truly unique, modular, and powerful. This modularity adds a great deal of extensibility, or development flexibility, to applications. Android makes heavy use of an XML-based markup language to define the basic component design of an application, especially its visual and user interface components. XML “markup” is not technically code, but rather consists of tags, similar to the HTML tags that web developers use to format their online documents. XML is used in Android to define everything from UIs to animation to data access, and even programmatic constructs such as Java object definitions and parameter configurations.[35]

An Android application, along with a file called AndroidManifest.xml, is deployed to a device. AndroidManifest.xml contains the necessary configuration information to properly install it to the device. It includes the required class names and types of events the application is able to process, and the required permissions the application needs to run. For example, if an application requires access to the network (to download a file, for example) this permission must be explicitly stated in the manifest file. Many applications may have this specific permission enabled. Such declarative security helps reduce the likelihood that a rogue application can cause damage on your device.[33]

III.5 Android required tools :

There are three major components of an Android development environment: Java SE , Eclipse, and Android SDK.

- **Eclipse :** Eclipse is an integrated development environment (IDE), which is a piece of software dedicated to allowing you to easily write programming code and run and test that code in an integrated environment. In other words, you write all your code into its text editor, before running and testing that code using commands in Eclipse, without ever needing to switch to another program.[36]
- **Android SDK :** The Android software development kit (SDK) is a collection of files and utilities that work hand-in-hand with the Eclipse IDE to create an Android specific development tool.[36]
- **Java SE :** Java Platform, Standard Edition (Java SE) lets you develop and deploy Java applications on desktops and servers, as well as in today's demanding embedded environments. Java offers the rich user interface, performance, versatility, portability, and security that today’s applications require.[37]

III.6 Conclusion :

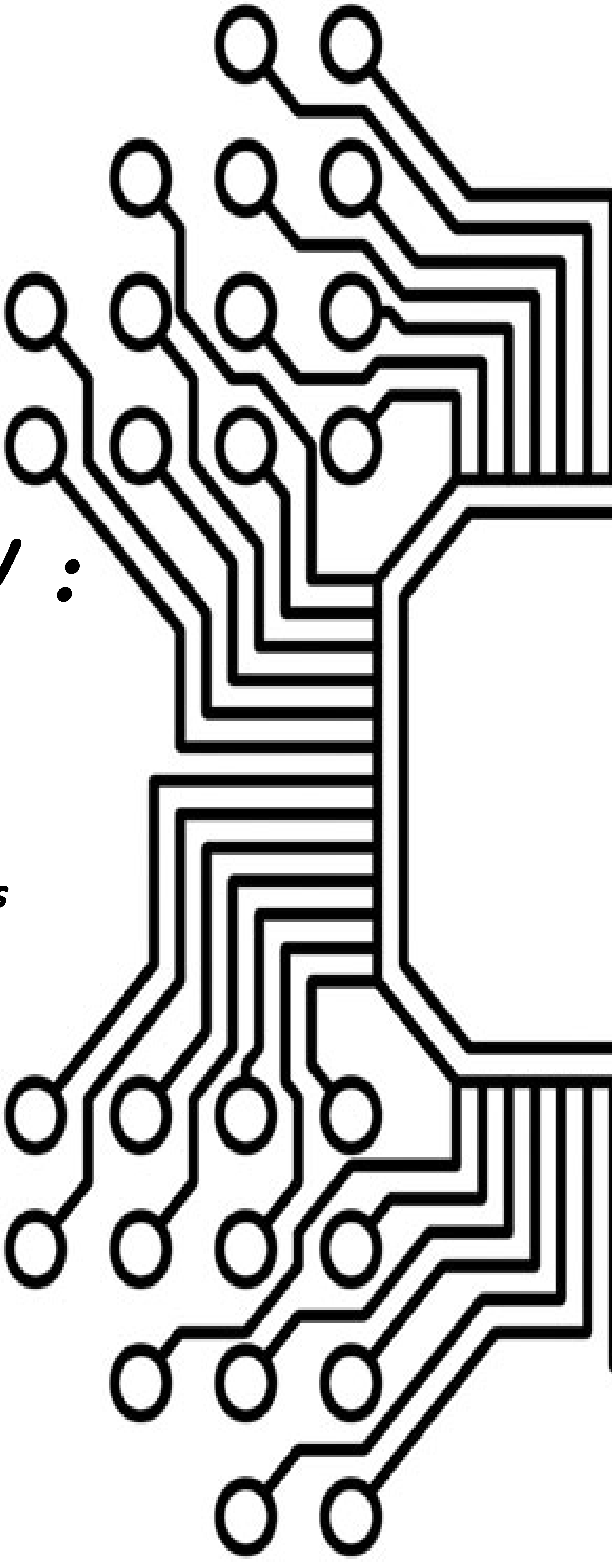
Android is new technology based on Java SE for oriented programming and XML for interface. The idea to build a client for our server as android application was challenge by itself. But we will believe that it has its own benefits.

Since it will be set on the user's smart-phone and we assumes that the 3G technology will be released soon, the user will be able to connect the server whenever he needs to, without any need for WIFI connection.

Chapter IV :

Microcontrollers

Fundamentals



Chapter IV: Microcontrollers Fundamentals

IV.1 Introduction :

To build circuit that can read certain codes, keywords, or commands, answer it with specific keywords as well ,and output electric signals to switch on the LEDs, we need to add an integrated little circuit that is known as "PIC".

In this chapter we will have an overview about PICs in generally, thier different models and thier characteristics.

IV.2 Generalities :

"PIC" refers to an extensive family of microcontrollers manufactured by Microchip Technology Inc.

A microcontroller is a microprocessor which has I/O circuitry and peripherals built-in, allowing it to interface more or less directly with real-world devices such as lights, switches, sensors and motors. They simplify the design of logic and control systems, allowing complex (or simple!) behaviours to be designed into a piece of electronic or electromechanical equipment. They represent an approach which draws on both electronic design and programming skills; an intersection of what was once two disciplines, and is now called "embedded design".

Modern microcontrollers make it very easy to get started. They are very forgiving and often need little external circuitry.

Among the most accessible are the PIC microcontrollers. The range of PICs available is very broad from tiny 6-pin 8-bit devices with just 16 bytes of data memory which can perform only basic digital I/O, to 100-pin 32-bit devices with 512 kilobytes of memory and many integrated peripherals for communications, data acquisition and control.

One of the more confusing aspects of PIC programming for newcomers is that the low-end devices have entirely separate address and data buses for data and program instructions. When a PIC is described as being 8- or 16-bit, this refers to the amount of data that can processed at once: the width of the data memory (registers in Microchip terminology) and ALU (arithmetic and logic unit). The low-end PICs, which operate on data 8-bits at a time, are divided into four architectural families:

- **Baseline (12-bit instructions) :** These PICs are based on the original PIC architecture, going back to the 1970's and General Instrument's "Peripheral Interface Controller". They are quite limited, but, within their limitations (such as having no interrupts), they are simple to work with. Modern examples include the 6-pin 10F200, the 8-pin 12F509 and the 14-pin 16F506.
- **Mid-Range (14-bit instructions) :** This is an extension of the baseline architecture, adding support for interrupts, more memory and peripherals, including PWM (pulse width modulation) for motor control, support for serial, I2C and SPI interfaces and LCD (liquid crystal display) controllers. Modern examples include the 8-pin 12F629, the 20-pin 16F690 and the 40-pin 16F887.
- **Enhanced Mid-Range (14-bit instructions) :** As the name suggests, this is an enhancement of the mid-range architecture – quite similar, but with additional instructions, simplified memory access (optimist for C compilers), and more memory, peripherals and speed. Examples include the 8-pin 12F1822, the 14-pin 16F1823, and the 64-pin 16F1526.
- **High-end (16-bit instructions) :** otherwise known as the 18F series, this architecture overcomes some of the limitations of the mid-range devices, providing for more memory (up to 128k program memory and almost 4k data memory) and advanced peripherals, including USB, Ethernet and CAN (controller area network) connectivity. Examples include the 18-pin 18F1220, the 28-pin 18F2455, the 40-pin 18F4450 and the 80-pin 18F8520. This can be a little confusing; the PIC18F series has 16-bit program instructions which operate on data 8 bits at a time, and is considered to be an 8-bit chip. [38]

IV.3 Microcontrollers versus microprocessors :

The term microprocessor and microcontroller have always been confused with each other. Both of them have been designed for real time application. They share many common features and at the same time they have significant differences. Both the IC's i.e., the microprocessor and microcontroller cannot be distinguished by looking at them. They are available in different version starting from 6 pin to as high as 80 to 100 pins or even higher depending on the features.

Microprocessor is an IC which has only the CPU inside them i.e. only the processing powers such as Intel's Pentium 1,2,3,4, core 2 duo, i3, i5 etc. These microprocessors don't have RAM, ROM, and other peripheral on the chip. A system designer has to add them externally to make them functional. Application of microprocessor includes Desktop PC's, Laptops, notepads etc.

But this is not the case with Microcontrollers. Microcontroller has a CPU, in addition with a fixed amount of RAM, ROM and other peripherals all embedded on a single chip. At times it is also termed as a mini computer or a computer on a single chip. Today different manufacturers produce microcontrollers with a wide range of features available in different versions. Some manufacturers are ATMEL, Microchip, TI, Freescale, Philips, Motorola etc.

Microcontrollers are designed to perform specific tasks. Specific means applications where the relationship of input and output is defined. Depending on the input, some processing needs to be done and output is delivered. For example, keyboards, mouse, washing machine, digicam, pendrive, remote, microwave, cars, bikes, telephone, mobiles, watches, etc. Since the applications are very specific, they need small resources like RAM, ROM, I/O ports etc and hence can be embedded on a single chip. This in turn reduces the size and the cost.

Microprocessor find applications where tasks are unspecific like developing software, games, websites, photo editing, creating documents etc. In such cases the relationship between input and output is not defined. They need high amount of resources like RAM, ROM, I/O ports etc.

The clock speed of the Microprocessor is quite high as compared to the microcontroller. Whereas the microcontrollers operate from a few MHz to 30 to 50 MHz, today's microprocessor operate above 1GHz as they perform complex tasks. Read more about what is microcontroller.

Comparing microcontroller and microprocessor in terms of cost is not justified. Undoubtedly a microcontroller is far cheaper than a microprocessor. However microcontroller cannot be used in place of microprocessor and using a microprocessor is not advised in place of a microcontroller as it makes the application quite costly. Microprocessor cannot be used stand alone. They need other peripherals like RAM, ROM, buffer, I/O ports etc and hence a system designed around a microprocessor is quite costly. [39]

IV.4 Microcontrollers advances :

Microcontrollers are widely used in today's control systems for the following reasons:

- **Design and Simulation :** Because you are programming with software, detailed simulations may be performed in advance to assure correctness of code and system performance.
- **Flexibility :** Ability to reprogram using Flash, EEPROM or EPROM allows straight forward changes in the control law used.
- **High Integration :** Most microcontrollers are essentially single chip computers with on chip processing, memory, and I/O. Some contain peripherals for serial communication and reading analog signals (with an analog-to-digital converter or ADC). This differentiates a microcontroller from a microprocessor. Microprocessors require that this functionality be provided by added components.
- **Cost :** Cost savings come from several locations. Development costs are greatly decreased because of the design/flexibility advantages mentioned previously. Because so many components are included on one IC, board area and component savings are often evident as well.
- **Easy to Use :** Just program and go, While in the past, programming has often involved tedious assembly code, today C compilers are available for most microcontrollers. Microcontrollers often only require a single 5V supply as well which makes them easier to power and use. [40]

IV.5 Microcontrollers Applications :

In addition to control applications such as the home monitoring system, microcontrollers are frequently found in embedded applications. Among the many uses that you can find one or more microcontrollers: automotive applications, appliances (microwave oven, refrigerators, television and VCRs, stereos), automobiles (engine control, diagnostics, climate control), environmental control (greenhouse, factory, home), instrumentation, aerospace, and thousands of other uses. Microcontrollers are used extensively in robotics. In this application, many specific tasks might be distributed among a large number of microcontrollers in one system. Communications between each microcontroller and a central, more powerful microcontroller (or microcomputer, or even large computer) would enable information to be processed by the central computer, or to be passed around to other microcontrollers in the system.

A special application that microcontrollers are well suited for is data logging. By stick one of these chips out in the middle of a corn field or up in a balloon, one can monitor and record environmental parameters (temperature, humidity, rain, etc). Small size, low power consumption, and flexibility make these devices ideal for unattended data monitoring and recording. [41]

IV.6 Inside a Microcontrollers :

A typical microcontroller as depicted in Figure IV-1 has the following components: a central processing unit (CPU), a random access memory (RAM), a read only memory (ROM), special function registers and peripheral components including serial and/or parallel ports, timers and counters, analogue-to-digital (A/D) converters and digital-to-analogue (D/A) converters.[42]

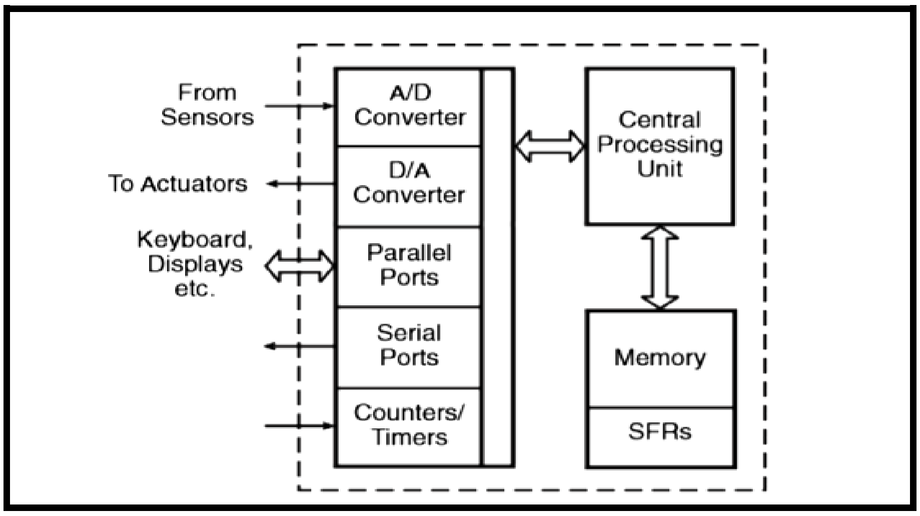


Figure IV-1 : Inside the microcontrollers.

◆ **<u>The Central processing unit (CPU core):</u>**

The CPU is the brain of the microcontroller which is responsible for locating and fetching the correct instruction, decoding it and executing it. This unit connects all parts of the microcontroller through data buses and address buses and determines the transfer of data from one location to another within the microcontroller via the controller. The main components of the CPU are the Program Counter (PC), the Instruction Register (IR) , the Register File and the

Arithmetic Logic Unit (ALU).[43]

◆ **Memory Unit :**

The function of memory in a microcontroller is same as microprocessor. It is used to store data and program. A microcontroller usually has a certain amount of RAM and ROM (EEPROM, EPROM, etc) or flash memories for storing program source codes.

1. RAM : RAM means Random Access Memory. It is general purpose memory that can store data or programs. RAM is 'volatile', which means when the power is shut off, the contents of the memory is lost. Most personal computers have several megabytes of RAM. Most microcontrollers have some RAM built into them, but not very much. 256 bytes is a fairly common amount. Some have more, some have less.[44]
2. ROM : ROM is Read Only Memory. This is typically memory that is programmed at the factory to have certain values. It cannot be changed, but it can be read as many times as you want. ROM is typically used to store programs and data that doesn't change over time. Many Microcontrollers have lots of ROM. Unfortunately, unless you are ordering thousands of parts, the ROM is useless to you, and in fact is wasting address space. Most individuals stay away from controllers with lots of ROM. [44]
3. EPROM : EPROM is Erasable Programmable Read Only Memory. This is ROM that can be field programmed using an EPROM programmer, which is a special device that takes your program information and programs (often called burning) the EPROM. To erase an EPROM, you need an EPROM eraser. This is usually a small box that has a strong UV light bulb. You place the EPROM part into the box for several minutes. The UV light causes the program to be erased. Most EPROM chips have a small clear window into the chip. This is for the UV light to pass through. EPROM devices range from 8kb to 32kb. Bigger versions are available, but not usually used on Microcontrollers.

   There are EPROM chips that are not 'windowed'. A more correct term for this is PROM or OTP (One Time Programmable) packages. You will typically find that windowed parts usually cost more. OTP parts are great if your design is done, and you never intend to change the program again. [44]

4. EEPROM : EEPROM is Electrically Erasable Programmable Read Only Memory. EEPROM is extremely useful for a variety of uses. For example, when you save configuration information on many household devices, the information is commonly written into EEPROM. EEPROM can also be 'programmed' from software, so you typically do not need to remove the part from your circuit to reprogram it. This is a big advantage. It typically takes about 10 milliseconds to write each byte of EEPROM.
5. Flash EEPROM : Yet another version of the EEPROM, Flash memory programs much faster. A byte may only take a few hundred microseconds to program. That is an advantage over EEPROM. However, Flash memory typically requires you to erase the entire contents before you can program it. Flash EEPROM usually comes in large quanities. [44]

◆ **Input-Output Unit :**

Ports refer to a group of pins on a microcontroller which can be accessed simultaneously, or on which we set the desired combination of zeros and ones, or read from them an existing status. They represent the connection of the CPU and the outside world. The ports are basically I/O registers which can be read and written under program control. [46]

◆ **Converters :**

Analogue-to-digital and digital-to-analogue converters provide an interface with analogue devices. For example, the analogue-to-digital converter provides an interface between the microcontroller and the sensors that produce analogue electrical equivalents of the actual physical parameters to be controlled. [42]

1. Analog to Digital Converter (ADC) : ADC converters are used for converting the analog signal to digital form. The input signal in this converter should be in analog form (e.g. sensor output) and the output from this unit is in digital form. The digital output can be use for various digital applications (e.g. measurement devices).[45]
2. Digital to Analog Converter (DAC) : DAC perform reversal operation of ADC conversion.DAC convert the digital signal into analog format. It usually used for controlling analog devices like DC motors, various drives, etc. [45]

◆ **<u>Serial communication :</u>**

Parallel connections between the microcontroller and peripherals via input/output ports is the ideal solution for shorter distances- up to several meters. However, in other cases - when it is necessary to establish communication between two devices on longer distances it is not possible to use a parallel connection - such a simple solution is out of question. In these situations, serial communication is the best solution.

Today, most microcontrollers have built in several different systems for serial communication as a standard equipment. Which of these systems will be used depends on many factors of which the most important are:

- How many devices the microcontroller has to exchange data with?
- How fast the data exchange has to be?
- What is the distance between devices?
- Is it necessary to send and receive data simultaneously?

One of the most important things concerning serial communication is the Protocol which should be strictly observed. It is a set of rules which must be applied in order that the devices can correctly interpret data they mutually exchange. Fortunately, the microcontrollers automatically take care of this, so the work of the programmer/user is reduced to simple write (data to be sent) and read (received data).[46]

◆ **<u>Timer Unit :</u>**

Since we have the serial communication explained, we can receive, send and process data. However, in order to utilize it in industry we need a few additionally blocks. One of those is the timer block which is significant to us because it can give us information about time, duration, protocol etc. The basic unit of the timer is a free-run counter which is in fact a register whose numeric value increments by one in even intervals, so that by taking its value during periods T1 and T2 and on the basis of their difference we can determine how much time has elapsed. This is a very important part of the microcontroller whose understanding requires most of our time. [47]

◆ **<u>Watchdog:</u>**

One more thing is requiring our attention is a flawless functioning of the microcontroller during its run-time. Suppose that as a result of some interference (which often does occur in industry) our microcontroller stops executing the program, or worse, it starts working incorrectly.

Of course, when this happens with a computer, we simply reset it and it will keep working. However, there is no reset button we can push on the microcontroller and thus solve our problem. To overcome this obstacle, we need to introduce one more block called watchdog. This block is in fact another free-run counter where our program needs to write a zero in every time it executes correctly. In case that program gets "stuck", zero will not be written in, and counter alone will reset the microcontroller upon achieving its maximum value. This will result in executing the program again, and correctly this time around. That is an important element of every program to be reliable without man's supervision.[47}

- **Bus :**

Physically, the bus consists of 8, 16 or more wires. There are two types of buses: the address bus and the data bus. The address bus consists of as many lines as necessary for memory addressing. It is used to transmit the address from the CPU to the memory. The data bus is as wide as the data, in our case it is 8 bits or wires wide. It is used to connect all circuits inside the microcontroller.[46]

IV.7 Conclusion :

The microcontrollers are new generation to design “intelligent circuits”. It might be slower compared to microprocessor but its feathers make it the best solution for the circuits thar don't require high speeds.

In our project, a simple microcontrller that has UART interface, will preserve us a lot of implementing with microprocessor or logical circuits, but we chose to use PIC18f4550 for futre developments.

<Chapter V>

SOFTWARE

</Chapter V>

Chapter V : Software

V.1 Introduction :

As the theme of our project shows, the depths of it was about network programming. In this chapter we will talk about the work's steps that we fallowed in our tries to realize our goal. After that we will explain the functionality of our programs and the languages that we used in each one.

V.2 The Work's steps :

Our basic idea was to build a web page that allows a communication between two sides, those two sides are the known as the client and sever sides.

The server side is a piece of software that will do all of the operations, between receiving the commands and sending the statuses of our circuit that is connected to our computer through our serial port RS-232.

For the client side it will be our browser that allows us to open the web page on the other node (computer). Using the HTTP protocol, the communication will allow us to send and receive commands and statuses respectively.

After some consolations, some said that the basic idea isn't real challenge as it is already realized before. We decided to look farther for new available technologies. Therefore we saw in Android good solution for few reasons:

1. Totally new technology in the world (first release was in 2008).
2. Large variety devices (different sizes and trademarks with different technologies).
3. It's becoming one of the most used mobile operating system.
4. Most of the Android devices are considered lower-cost compared to others mobile operating system devices and available for everyone.
5. It was personal challenge to build an application for Android.

As we stabilised on final idea, we tried to be methodical in our work. As first step we had to go back to concentrate on building a controller. The Controller is part of the server, or it can be considered as 2nd mode for server that give the user the ability for local control on the circuit.

We will talk more about the details of the Controller in “ The software development ”section.

The second step was working in parallel on the server and and client. They were developed for for our specific needs, in perspective as we imagined how the software preform in certain cases. As we will explain later in this chapter.

The last step was programming the PIC, though it was not as complicated as the first two steps but it had its specialities as we will show later in this chapter of our project.

V.3 Obstacles and difficulties :

Over the whole processes of developing our software, we faced many difficulties and obstacles that we give in fallowing points:

1. As beginning, the project had “general” title, so it gave us ultimate choices. it can be considered as advantage but in same time, too many options led us to loss the concentration on one path for long times.
2. We barely knew some programming languages that we used. We needed some time to master it on many levels.
3. The ceaseless changes of IP address with every disconnect and the unstable of network service (such projects require fix IP address while the network service are using dynamic IP address system).
4. To handle the errors and stabilize the server and the client on certain regime to work on.
5. As we looked for expert advices, we couldn't find any of them.
6. At some point, I lost all the data and the source code of the programs after the hard driver damaged , and I had to start all over again from the first point.

V.4 The software development:

The software have three main parts and each one is developed in different programming language. With Microsoft Visual C# 2010 express we designed the first part which is the Server/Controller . Secondly using Eclipse, we developed the Android application. Finally, we looked up to build our circuit and with MikroC pro we programmed our PIC18f4550.

V.4.1 Server/Controller :

The Server/Controller is our software that will be setted on our computer where the circuit is relied. This software was built using the Microsoft Visual C# 2010 express. It contains two main modes that will allow the user to choose between them.

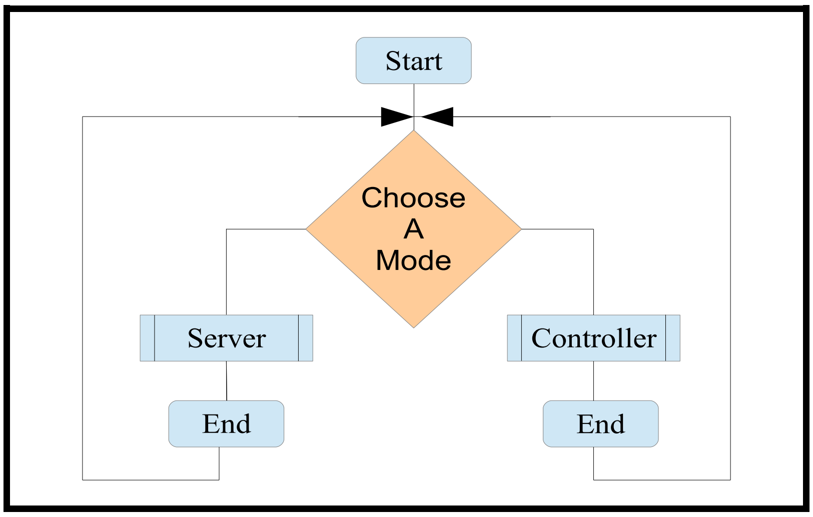


Figure V.1 – The flowchart of Servers/Controller.

In Figure V.1, we see the flowchart of Servers/Controller as it runs. By clicking on the button “Choose the Mode”, that we see in figure V.2, after Choosing one of the two modes (Controller or Server), you will run another program and with another interface. The two programs are independent from each other simply because each one has its functionality. And here we will explain each mode in the next titles.

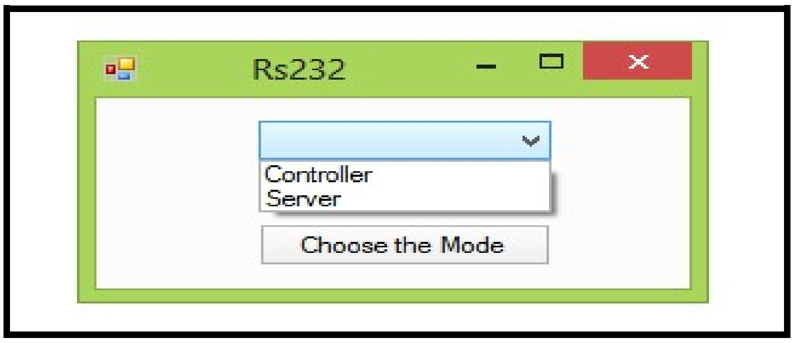


Figure V.2- The interface of Server/Controller.

V.4.1.1 Controller :

The Controller was the first part of program that we built. The main idea behind it is to give the user the right to control the circuit that's relied to computer through the serial port (RS-232).

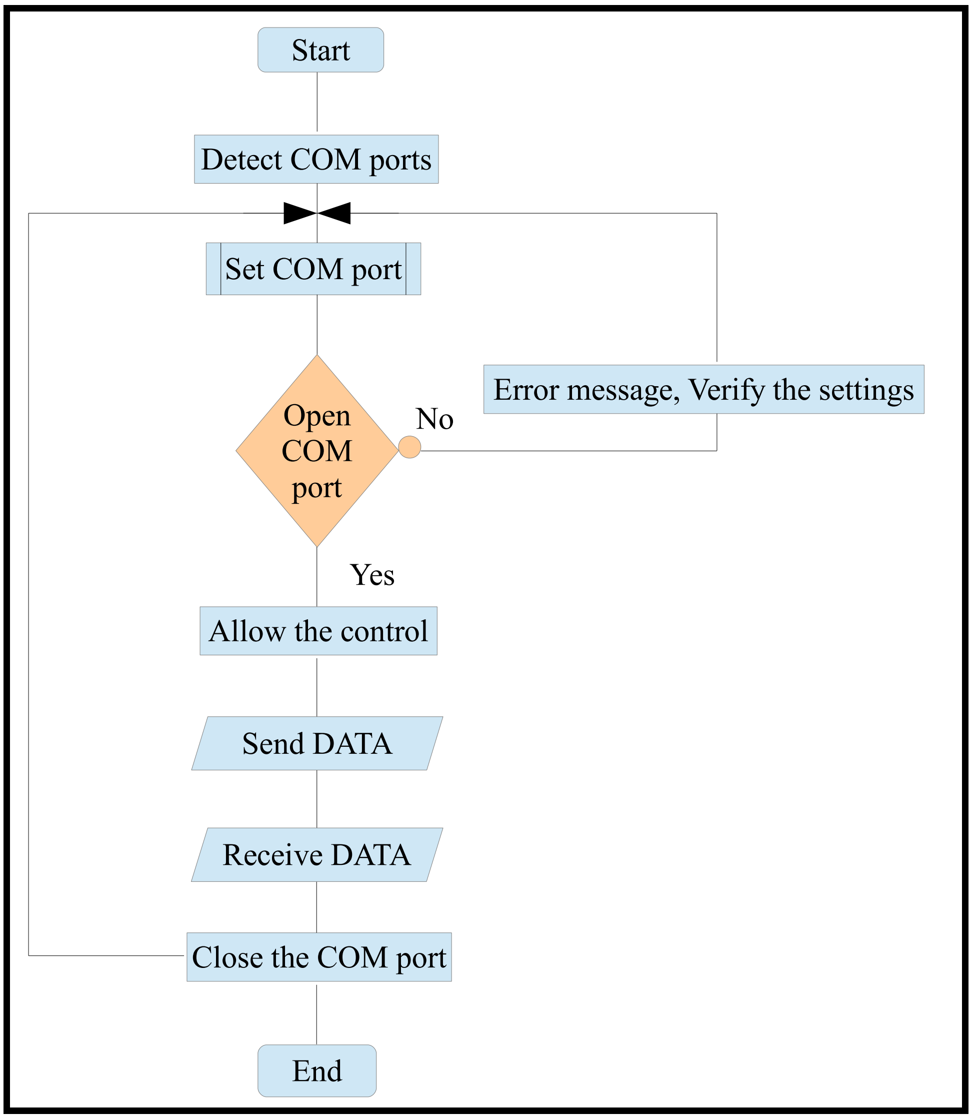


Figure V.3 - The flowchart of Controller.

The flowchart in Figure V.3 shows how our program will run. First of all the program will detect the available COM ports in our computer as it starts and saves it as string array before it posts it in combo-box that is designed for it as we see it in Figure V.4 (the text next to each combo-box explains its contain). This process is too fast we can't feel the delay as the interface shows up.

After choosing COM port (expecting that the user knows which one is relied by circuit) and setting the band rate, parity, stop bit and the data bits that depend on the characteristics of his circuit, the user will click on the “open COM port” button.

Clicking the “open COM port” button will apply all the settings on and unlock the Control area (we will explain it we move forward). There are couple cases we have to expect :

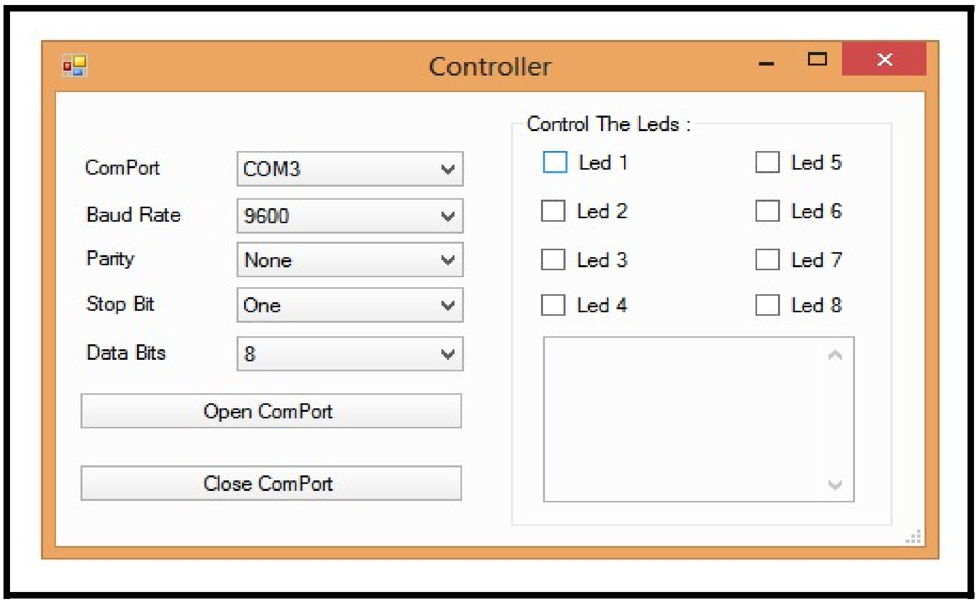
1. the user might forget to set one of the options, that will gives him an error to check his settings.
2. The COM port might be already open/used . A error message will appear and the user has to use/choose another one COM port.

When the COM port is open, the “Control the Leds” area (that's shown in Figure V.4) will be unlocked. The reason for locking it in the beginning, is to show the user that he can't control anything before setting the COM port, and even trying to control the LEDs will be useless.

The “Control the Leds” area was designed for our specific needs. It contains 8 check-boxes for the 8 LEDs that we have to control, and text-box (multi-lines) where we will post what we receive from our circuit (the answer for our commands).

As we start commanding, we will check the check-box for the intended LED or LEDs. When the user checks it, a certain command will be send to the circuit to turn on the intended LED and the circuit will “answer” us with the status of the LED that will be posted on text-box. But when we un-check it, another command will be send to turn off the LED and the circuit will update us with the status of the LED that will be posted on text-box as well.

There is another button in the interface as we can see in the Figure V.4 that's called “Close ComPort”. This button is designed to close the COM port in case the user finished the control or needed to change to COM port for some reasons. Clicking it will unlock the “Control the LEDs” again as well.



ComPort: COM3
Baud Rate: 9600
Parity: None
Stop Bit: One
Data Bits: 8

Figure V.4 - The interface of Controller.

V.4.1.2 Server :

The Server is all what our project about. It was developed in parallel while developing the Android application as we mentioned before. This piece of software is communicate through the internet to the our client, that will be our application on an Android machine (smart-phone or tablet in our case). Using the TCP/IP protocol it will send some information about the local computer that's running on and receive some other informations as well (we will see later the type of information that it will send).

As this part of program starts it will be listening to specific port and waiting for the client to connect, and as the client connects, and using the socket that will be received carrying the first command to send the ports name, the server will send the available ports on the computer one by one. After it finishes sending the ports names it will close the socket and start waiting for new connection from the client. The Figure V.5 shows the flowchart oh how this process runs.

And as the Figure V.5 shows, after the user receives the COM ports names and set up the configurations and send it, the server will receive it and set the configurations on the intended port and close this socket to wait for new connection.

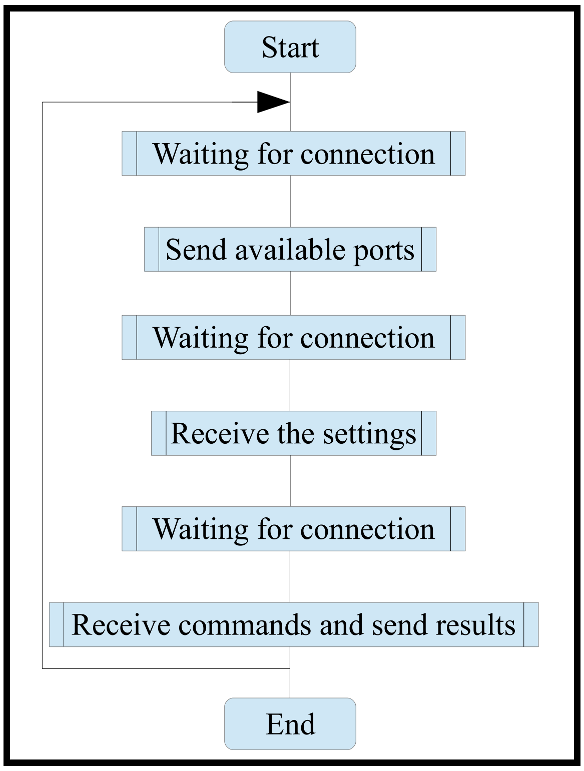


Figure V.5 - The flowchart of Servers .

The final part will be almost endless loop, between receiving commands to turn ON/OFF the LEDs, applying the commands (sent it through the serial port RS-232 and receives the circuit's responds) and sending the circuit's responds to client side. Till the client closes the connection it will restart the whole process all over again.

Over all those steps, on computer side, we can just watch the received commends from Android applications and received responds from the circuit. As the Figure V.6 shows, the interface has two text-boxes. Each will post the received respectively.

The upper text-box as its title shows, is the one that will post the server's status. Meaning : in the text-box we will post which part of the code the server is actually running (is it sending ports names! Or is it receiving the settings!, what commands is it receiving!).

For the lower text-box, it will be directed to post the port status in generally. It will only post the responds for our circuit (from our PIC to be more specific).

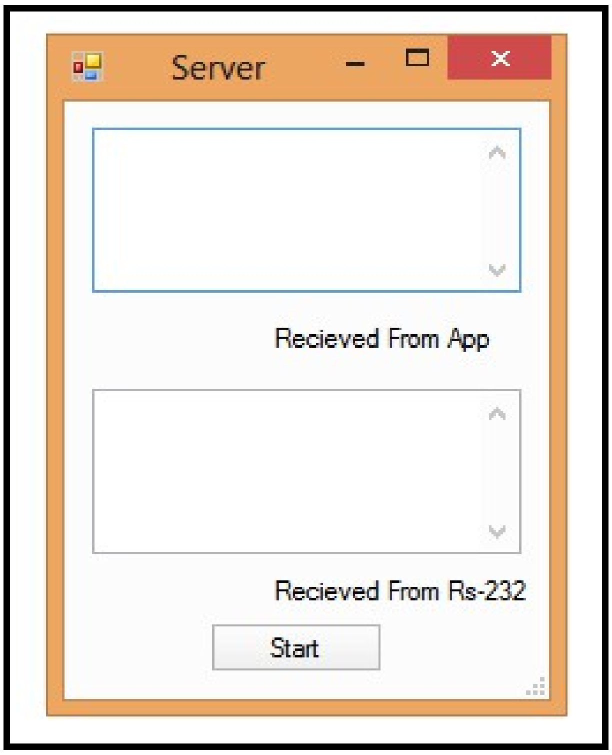


Figure V.6 - The interface of Server.

V.4.2 Android Application :

The Android application presents the client side what is connected via internet to our server. This connection will allow the user to send all the commands and receive the circuit status. It was developed in same time as the server.

As we run the application, we will go through two steps before starting the control on our circuit. Each step present a program by itself that has its own interface, the program is wrote in Java programming language while the interface uses the XML.

In the Android programming each step is called an Activity or intent, so in the end we had to build 3 activities (each activity has its Java program and XML interface). The Eclipse software eases the organisation of project as it packs the project in “.APK” file for installation on the Android device.

After we explain what activity means. Now, we are going to explain the tree activities that we built respectively by it functionality.

V.4.2.1 The first activity :

In the first activity, the use will entre the IP address of our Server. As we mentioned in the obstacles and difficulties, such projects require fix IP address to make the connection automatically.

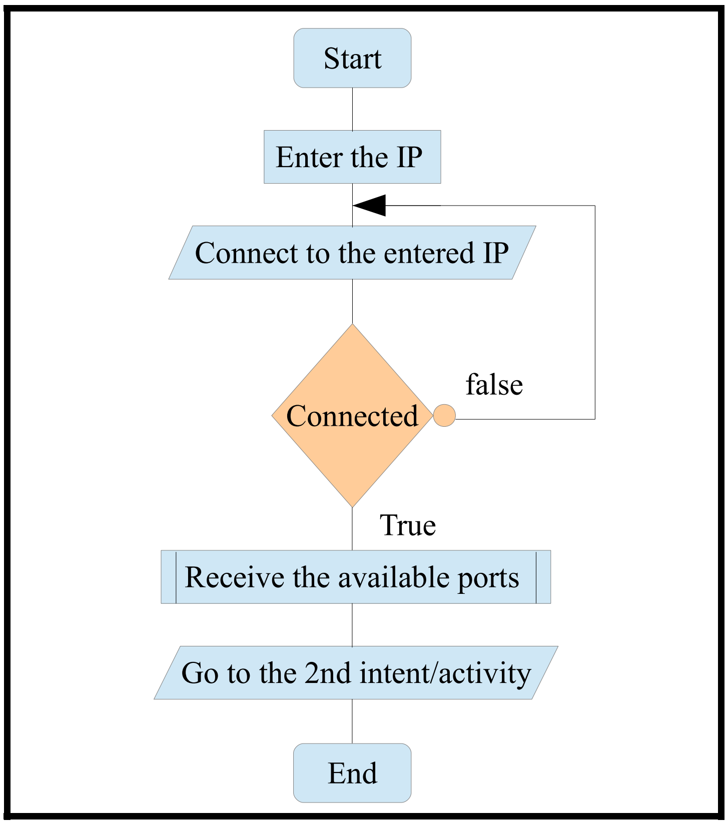

Start
Enter the IP
Connect to the entered IP
Connected
false
True
Receive the available ports
Go to the 2nd intent/activity
End

Figure V.7 - The flowchart of the first activity.

As the Figure V.7 shows, as the application starts, it will wait for the user to enter the IP address of the server. The user will write the IP address in a special field that is showing the Figure V.8. To start the connection the user will need to click on “Connect” button. The Java program will try to connect to entered IP address through defined port that both of Server and the client agreed on (in our program we chose port “6677”).

the first activity wasn't build only to enter the IP address, this activity will receive the available ports on the Server computer.

When the activity is connected to the Server, it will send a command to ask for the ports names. It will store it to be used in the second activity.

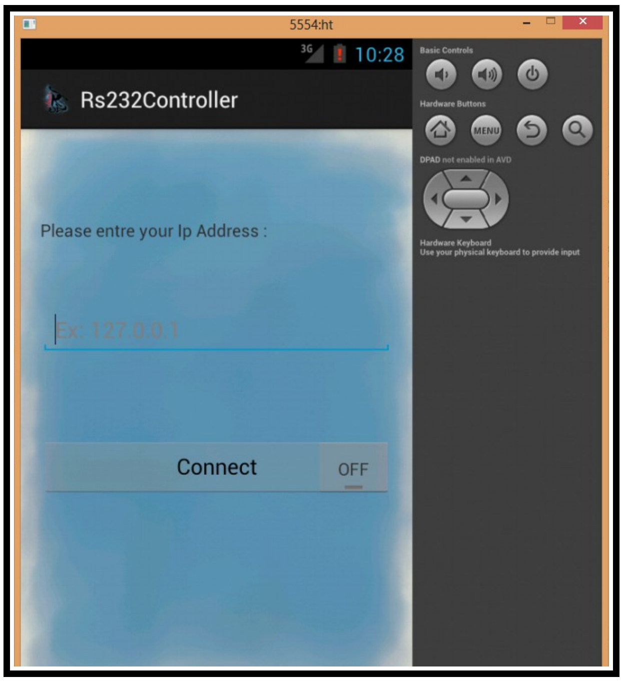


Figure V.8 - The interface of the first activity.

V.4.2.2 The second activity :

the second activity was designed to send settings, in certain scenario the user might have two or more circuits that are connected to the server through different ports (even it will be hard to find a computer with more than one COM port nowadays) . As there is a possibility that every circuit might has its own settings and configuration, as part of this scenario we gave the user a the right to set it COM port as he needs to.

Figure V.9 shows that the program isn't that different from the one from the first activity. As it starts running the user can choose the right parameters as we see in Figure V.10, and he clicks the “Apply” button to send those settings to the Server side.

Clicking on the “Apply” button will actually recall the first IP address and try connect it again through the same port. When the 2nd activity is connected to Server, it will start sending the settings one by one till it finishes them all. The Server side knows that it will receive 5 settings, so it will close the connection and the socket automatically. As the connection is closed the application will move to the last activity.

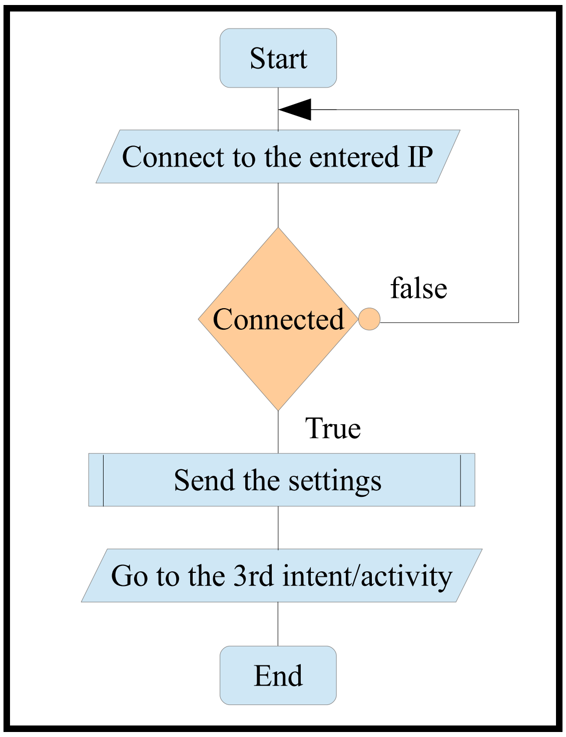
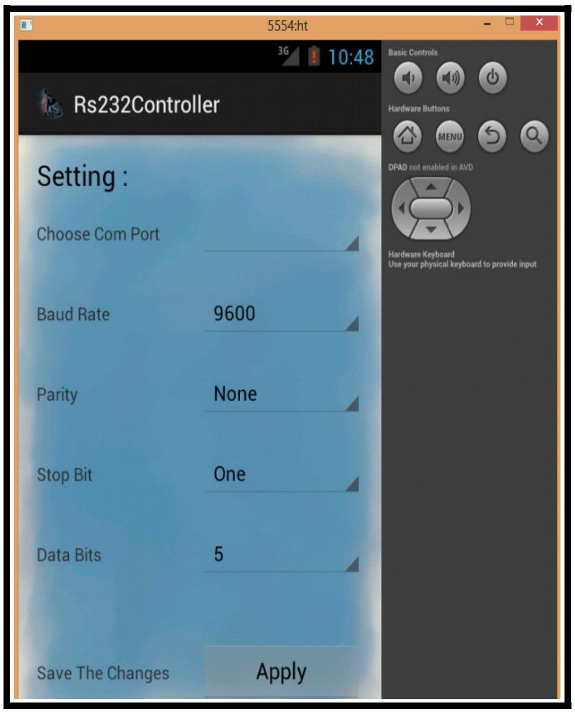


Figure V.9 - The flowchart of second activity.



Setting :

Choose Com Port

Baud Rate 9600

Parity None

Stop Bit One

Data Bits 5

Save The Changes Apply

Figure V.10 - The interface of the second activity.

V.4.2.3 The third activity :

the second activity is closed after it sent the settings of our COM port, now we get to the third activity where the real control happens.

The flowchart in Figure V.11 shows that this activity isn't different from the from the programming perspective. It uses the same concept with some changes to fit for our needs. Like the previous activity, it will need to recall the IP address that the user used in the very beginning of the application, and through the same port, it will try to connect the server.

As they are connected, it will wait for the user to start controlling by click on one of the LEDs button or more. Like we can see in Figure V.12, the interface has actually 9 buttons, 8 are to switch on and off the LEDs and one is called “Disconnect”.

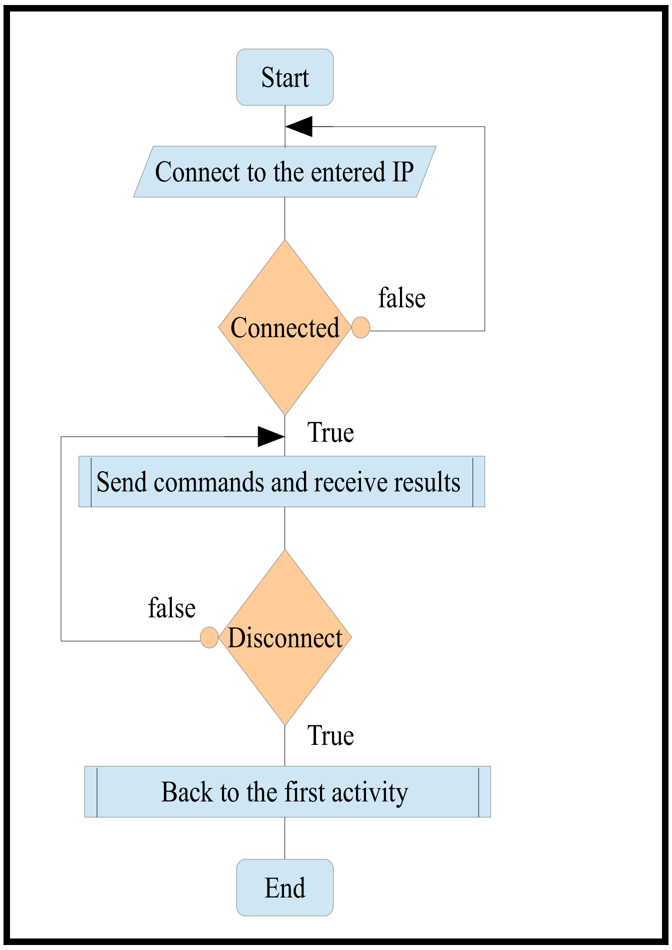

Start
Connect to the entered IP
Connected
false
True
Send commands and receive results
false
Disconnect
True
Back to the first activity
End

Figure V.11 - The flowchart of third activity.

The 8 LEDs buttons are all “OFF” as basic status . Clicking on it will turn it to “ON” status and it will send command to the Server. The Server will “translate” the command, know the intended LED, and send a specific command for that LED through the COM port to the PIC. The PIC will “respond” the Server as it receive it , then the server will read again that respond to answer the application about the LED new status. It will show up as comment in text area under the “Disconnect” button (we can't see that text area in Figure V.12 because the Emulator (Virtual Machine) hasn't the same screen size as the device that the application is designed for).

The same process will be looped over and over every time we click on LEDs button to switch off the LED, or turn on/off the others LEDs.

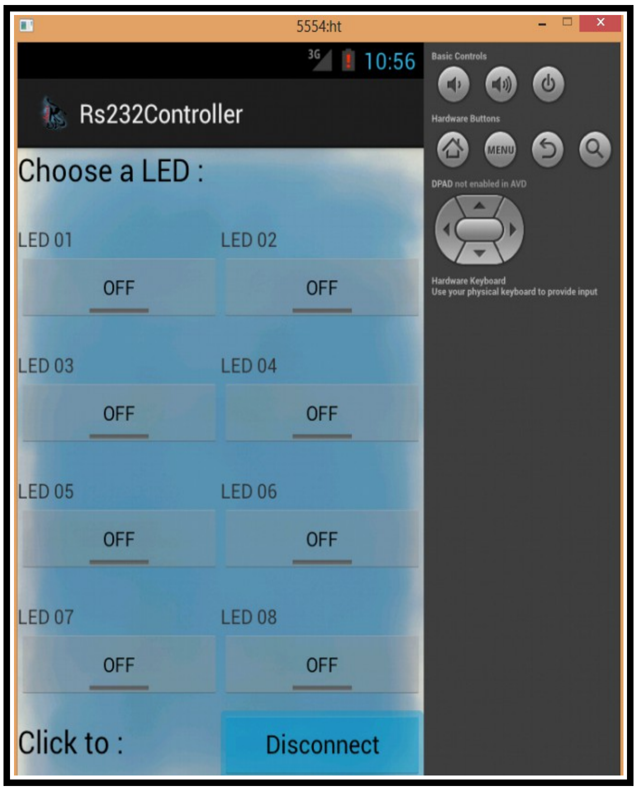


Figure V.12 - The interface of the third activity.

The “Disconnect” button is as its name says, it is the button that will end the connection with the Serve. Clicking it will end the connection and return to the first activity. We designed its functionality in cases the user has another circuit connected to another Server that is the same as the first one, only running on different machine with different IP address.

V.4.3 Programming the PIC :

The last part of the programming, was to build a program that will read the received commands from the serial port (RS-232) and translate it to electric signal to switch ON/OFF the LEDs.

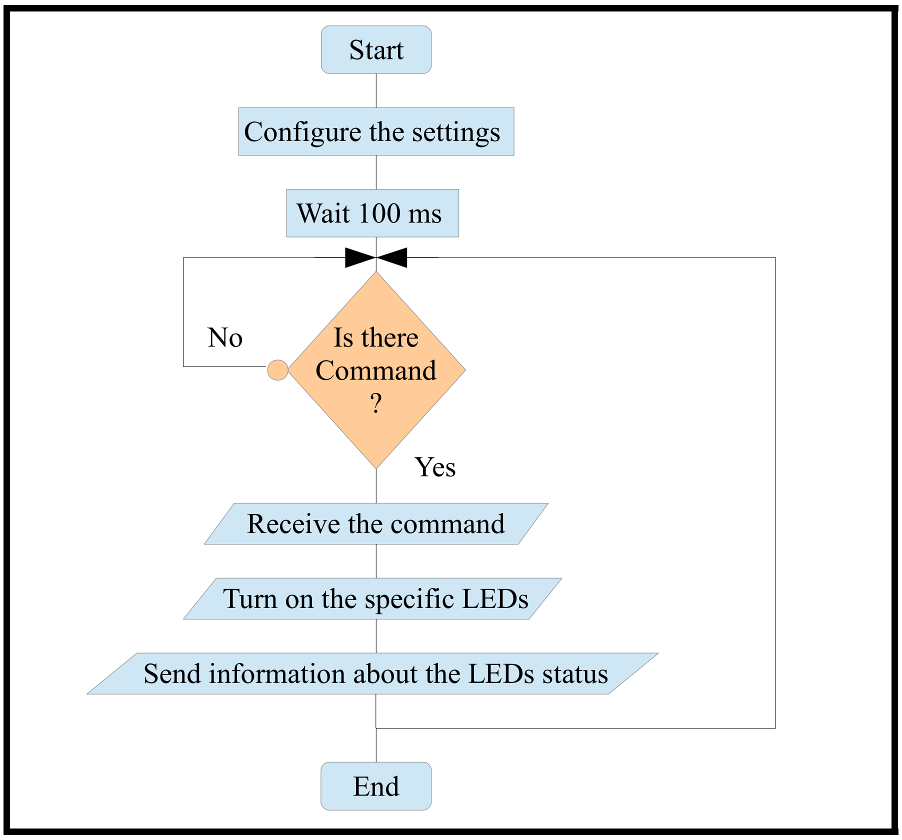


Figure V.13 - The flowchart of PIC program.

Figure V.13 explains the running steps of our program. As it starts the PIC will delay for 100 millisecond as the PIC stabilises after it configures the settings. When the PIC is stable, I will start testing for any interruptions. The interruptions are the incoming commands.

When it receives a command, it will save it in specific byte to translate it right after that to electric signal. This electric signal will present the 8bits, each bit present a LED. The electric signal will go out through output port to switch ON/OFF the LEDs. We chose the PORT B as our output port.

After it receives and translates the commands, the PIC will send back a message of the received to the Server or the Controller (it doesn't matter which program on the computer).

V.5 Results:

On the period we were developing our software, we ended with the fallowing results :

1. <u>For the Controller :</u>

we couldn't find any circuit that we might test our Controller on as first test. So we figured out simple solution for that. By connecting the TX and RX pins, we created short circuit that received what we send and It worked actually.

After while we were able to develop the PIC program and we had better vision about our circuit that we simulated it with ISIS Proteus professional and created a virtual COM port using VSPE (Virtual Serial Port Emulator) we made sure that this part of program actually works as we wished.

2. <u>For the Server and Android application :</u>

Using LAN (Local Area Network), we tried to communicate the Server and Android application. But since we didn't build our circuit yet, we made some change in the Server side. We simplified it to receive the commands and resend it again just to make sure that we created a real communication between the two sides. And so it gave us the expected results, it worked well.

The second test was to test the application on real Android device. Using Crius Tablet model Q7A that runs Android Ice Cream Sandwich (version 4.0.4), it gave us the same results. It worked as well.

3. <u>For the PIC program :</u>

In the begging, we checked for any errors on the MicroC pro. As it was clear, we used ISIS Proteus professional to simulate our PIC and our circuit, and using Virtual Serial Port Emulator software (it generates virtual COM port) with the Controller that we built, It worked well like we mentioned earlier.

Chapter VI : Hardware

Chapter VI: Hardware

VI.1 Introduction :

The last step in our project was the design circuit that will be a sample for how useful our project can be. In this chapter we'll have an overview about the circuit that we made, as we explain the work steps that we fallowed one by one.

VI.2 The work steps :

After we solved the main problem of our project (to send and receive a command over the internet), we moved to hardware part.

In this part we started by designing the circuit that we had a picture on, so we used ISIS Proteus pro to simulate it after we finished the PIC18f4550 programming. Though we wanted to use smaller PIC for the circuit but that was the best choice we got. After all, it will be great for future development.

When we made sure that it works fine in simulation, we tried to realize as fast as we can since the time was running out of us.

VI.3 The circuit development :

In this section of the chapter, we will explain more about the previous mentioned steps which were respectively the design and simulation as beginning, then the realization.

VI.3.1 The design and simulation :

In previous chapter, we talked about how we built our program to control circuit that contains 8 LEDs that is relied to our computer through serial port RS-232.

Using the PIC18f4550 that will be the “brain” of the circuit, it'll receive the commends from the COM port and respond the Server/Controller about the LEDs statuses.

The Figure VI.1 presents the schematic of the circuit, it was exported from ISIS Proteus pro. As you can see, in the centre, a block (1) presents the PIC18f4550 in basic shape with only essential pins.

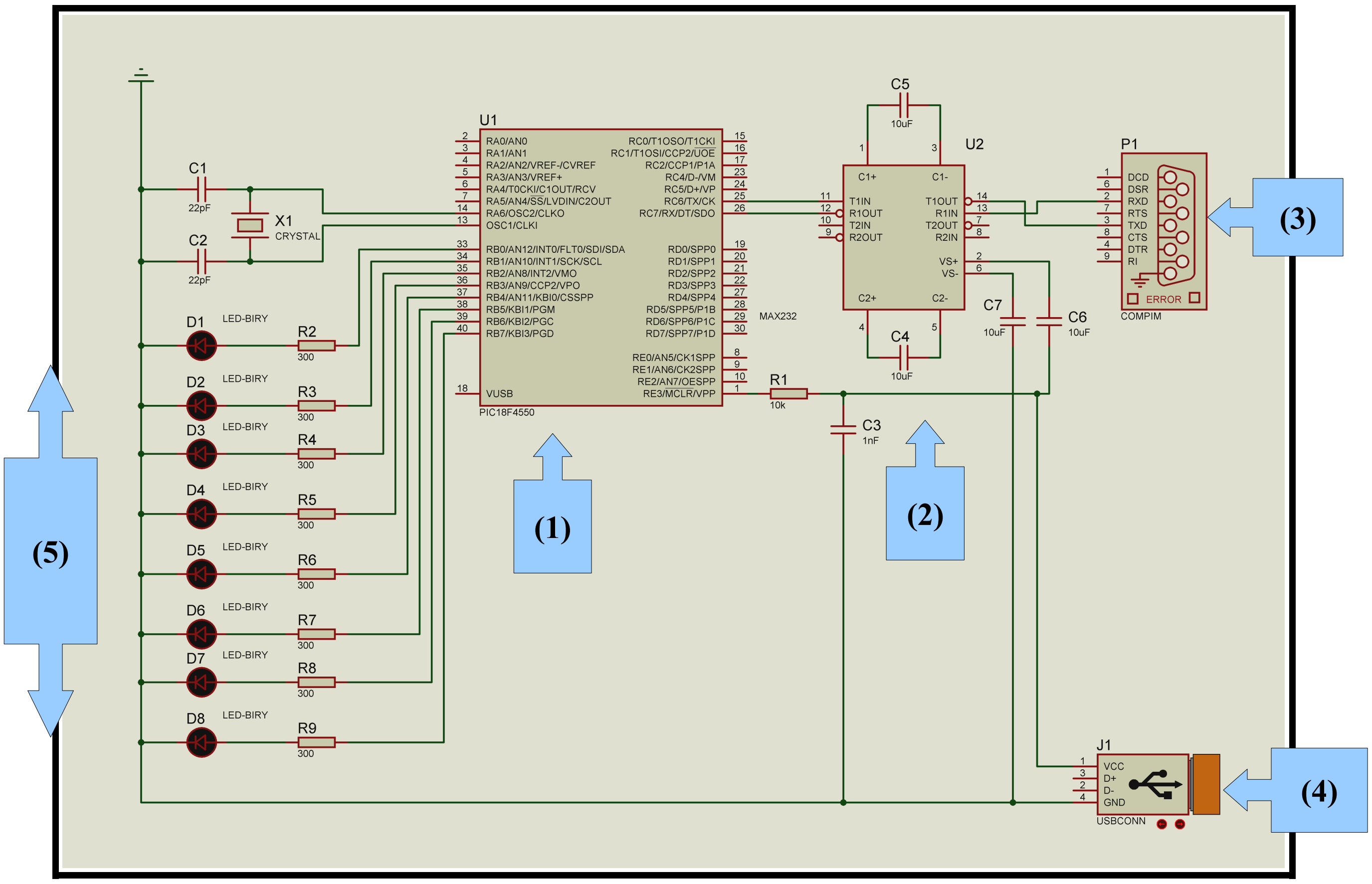


Figure VI.1 -The circuit schema.

The block (3) is virtual connector for serial port RS-232, it's used to connect the PIC18f4550 with Server/Controller though VSPE (Virtual Serial Port Emulator).

As we saw the characteristics of COM port in Chapter I, we knew that we can't connect directly the RS-232 to the PIC18f4550 because of the electric features of each. The block (2) in figure VI.1 solved that problem, it's MAX-232 can convert RS-232's signals to suitable signals that can be used for TTL/CMOS digital logic circuit in generally.

The PIC18f4550 has few ports that can be configured as input or output port. We chose the port B as our output for the LEDs (block (5)).

we still have a 4th block that we will explain in the next point this chapter.

VI.3.2 The realization :

The Figure VI.2 below shows the circuit that we realised based on figure VI.1.

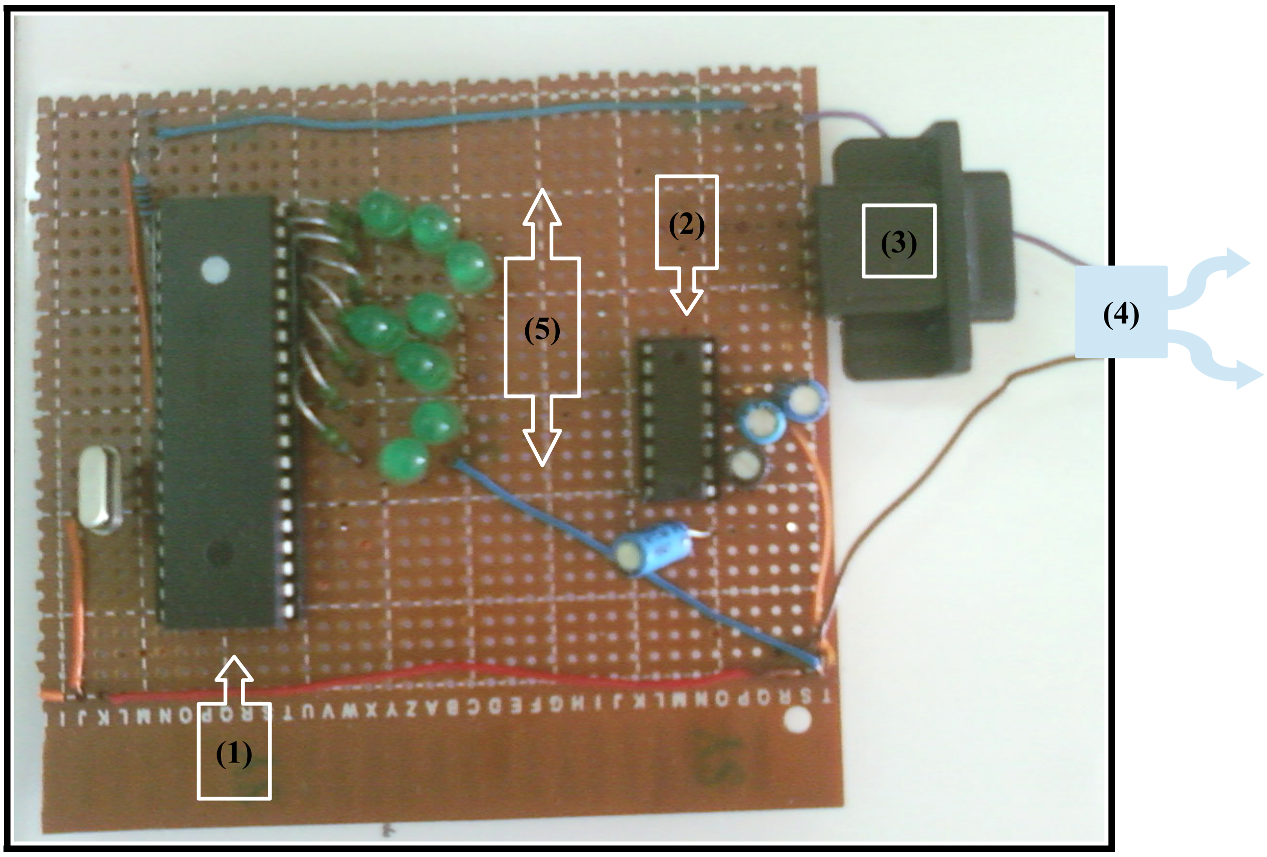


Figure VI.2 – The realized circuit.

The blocks (1), (2), (3), (4), and block (5) are the same in Figure VI.1. For block (4), we used two wires from the USB port which are Vcc and GND to aliment the circuit with power. The circuit will not need external power, all it needs, is to be linked to computer with RS-232 to receive and send commands and with USB cable for the power.

VI.4 The results:

Till the moment we wrote those lines, the circuit didn't work as we wished. Though it worked virtually, we couldn't figure out the problem.

The hardware and the implementation are away too hard than what we imagined and the problems are endless compared to the software side that we could manage it with the time.

General conclusion and perspectives

General conclusion and perspectives :

This kind of projects are wonderful experience to login the world of the networking and the remote control development. To realize it we have to understand several points :

1. The remote control demands a study to define the needs of our circuit. The principles are almost the same, but each circuit has its characteristics and its goals that's built for.
2. The networking (Internet in generally) requires good experience the programming and a good read for its concepts.
3. The Android is totally new technology that will need more time master. Since it didn't stabilize on finale version/release (and probably it will not because there will be always improvements), there will be always something new to learn.

The networking isn't that easy as it seems. The main problem is that it's hard to stabilize connection server-client. And the biggest fear is if the network collapses, the whole system will fail.

The serial port norm RS-232 has medium speed and the circuit didn’t need high speed to transfer the data (to receive the commands and send the response) otherwise the USB port will be more useful if we needed faster transformations.

The PIC18f4550 has large number of pin (40 pin), though we wanted to use smaller one such as PIC18f2550 (28 pin)because our circuit didn't need all that number of pins. But it was the best choose that we could find.

This project is an opening for farther goals. We prove that we can mirage the 3 technologies. And now we are looking to develop the hardware and software by :

1. improve the security of the communication between the server and the client.
2. Communicate the client side straight to the circuit. For example, adding WIFI captor to the circuit that will give it an independent IP address so we can connect it directly.
3. Adding the WIFI captor will need new PIC that can handle this type interruptions

finally as an advice, a team work that is composed of informations who are expert in networking and electricians to develop the hardward will cover all the missing pieces in this project.

Bibliography :

[1] Dario J. Toncich, Data Communications and Networking for Manufacturing Industries Second Edition, Chrystobel Engineering, ISBN: 0-646-10522-1, 1993, pp [143,144]

[2] Jan Axelson, Serial Port Complete: COM Ports, USB Virtual COM Ports, and Ports for Embedded Systems Second Edition, Lakeview Research LLC, ISBN: 978-1931448-07-9, 2007, pp [43..70]

[3] http://www.sena.com/download/tutorial/tech_Serial_v1r0c0.pdf

[4] Ying Bai, The Windows Serial Port Programming Handbook, CRC Press LLC, ISBN 0-8493-2213-8, 2005, pp [10..15]

[5] http://www.maximintegrated.com/images/appnotes/75/547Fig01.gif

[6] http://www.zytrax.com/tech/layer_1/cables/tech_rs232.htm

[7] http://www.camiresearch.com/Data_Com_Basics/image28.gif

[8] DALAS semiconductors, Applications note83, fundamentals of RS-232 serial communications

[9] http://www.plcmanual.com/bits-software-handshaking

[10] http://digital.ni.com/public.nsf/allkb/2AD81B9060162E708625678C006DFC62

[11] http://www.db9-pinout.com/

[12] http://www.brainlubeonline.com/serial.pdf

[13] http://www.zytrax.com/tech/layer_1/cables/tech_rs232.htm

[14] http://techpubs.sgi.com/library/dynaweb_docs/0630/SGI_Admin/books/MUX_IG/sgi_html/figures/3-1.25.pin.connector.gif

[15] http://www.javvin.com/osimodel.html

[16] Behrouz A. Forouzan, tcp/ip protocol suite 4th edition, mcgraw hill, ISBN 978-0-07-337604-2, 2010, pp[25..797]

[17] Charlos M.Kosierok, The TCP/IP Guide, No Strach press, ISBN: 987-1-59327-047-6, 2005, pp[102..378]

[18] http://www.scribd.com/doc/14206202/Difference-Between-OSI-and-TCPIP-Models

[19] http://technet.microsoft.com/en-us/library/Bb726993.caop0201_big(l=en-us).gif

[20] Lydia Parziale, David T. Britt, Chuck Davis, Jason Forrester, Wei Liu, Carolyn Matthews, Nicolas Rosselot, TCP/IP Tutorial and Technical Overview (redbooks), IBM, 2006, pp[7..145]

[21] Douglas E.Comer, Internetworking with TCP-IP (Principles, protocols, and architectures) 4th Edition, Prentice hall, ISBN: 0-13-018380-6, 2000, pp[75..612]

[22] http://www.apnic.net/__data/assets/pdf_file/0020/8147/501302.pdf

[23] **http://technet.microsoft.com/en-us/library/Bb726995.tcch0304_big(l=en-us).gif**

[24] **http://technet.microsoft.com/en-us/library/Bb726995.tcch0305_big(l=en-us).gif**

[25] **http://technet.microsoft.com/en-us/library/Bb726995.tcch0306_big(l=en-us).gif**

[26] IPv4 Addressing Guide, Cisco, February 2012 Series, pp[4]

[27] http://tools.ietf.org/html/rfc5735#section-3

[28] Michael J. Donahoo and Kenneth L. Calvert , TCP IP Sockets in C, Second Edition Practical Guide for Programmers, MORGAN KAUFMANN PUBLISHERS, ISBN: 1-55860-826-5, 2001,pp [8-9]

[29] W. Richard stevens, tcp ip illustrated volume 3, ADDISON WESLEY, ISBN: 0-321-63495-3 , 1996, pp[162]

[30] James F. Kurose and Keith W. Ross, Computer Networking A Top down Approach Featuring the Internet 3rdEdition, ADDISON WESLEY, 2000, pp[320]

[31]
http://www.tcpipguide.com/free/t_IPv6AddressandAddressNotationandPrefixRepresentati.htm

[32] Ed Brunetter, Hello Android : Introducing Google's Mobile Development Platform, The Pragmatic Bookshelf, ISBN: 978-1-934356-17-3, July 2009, pp[14,15]

[33] http://www.ibm.com/developerworks/library/os-android-devel/

[34] Sayed Y. Hashimi, Satya Komatineni, and Dave MacLean, Pro Android 2, Apress, ISBN 978-1-4302-2659-8, 2010, pp[10,11]

[35] Wallace Jackson, Android Apps For Absolute Beginners (2nd ed.), Apress, ISBN: 978-1-4302-4788-3, 2012, pp [15]

[36] Wallace Jackson, Android Apps For Absolute Beginners, Apress, ISBN: 978-1-4302-3446-3, 2011, pp [7.8]

[37] http://www.oracle.com/technetwork/java/javase/overview/index.html

[38] http://www.gooligum.com.au/tutorials/PIC_Intro_0.pdf

[39]http://www.engineersgarage.com/tutorials/difference-between-microprocessor-and-microcontroller

[40] http://inst.eecs.berkeley.edu/~ee128/fa04/labs/lab7-intro.pdf

[41] http://www.newagepublishers.com/samplechapter/001599.pdf , pp[10,11]

[42] Anil K. Maini, Digital Electronics Principles Devices and Applications, John Wiley & Sons Ltd, ISBN: 978-0-470-03214-5, 2007, pp[567..570]

[43] http://lgjohn.ecen.ceat.okstate.edu/4213/lectures/microarch.pdf , pp[5.8]

[44] http://www.seattlerobotics.org/encoder/sep97/basics.html

[45] http://www.circuitstoday.com/basics-of-microcontrollers

[46] http://www.mikroe.com/chapters/view/1/

[47] Nebojsa Matic, PIC Microcontrollers for Beginners, http://www.mikroelektronika.co.yu/english/product/books/PICbook/0_Uvod.htm , pp[6.7]