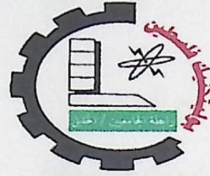


# Palestine Polytechnic University



College of Engineering & Technology  
Electrical & Computer Engineering Department

Graduation Project

Wireless Keyboard

Project Team

Mohammad Abu Sway  
Majdi Hnahen

Project Supervisor

Eng. Amal Dwiek

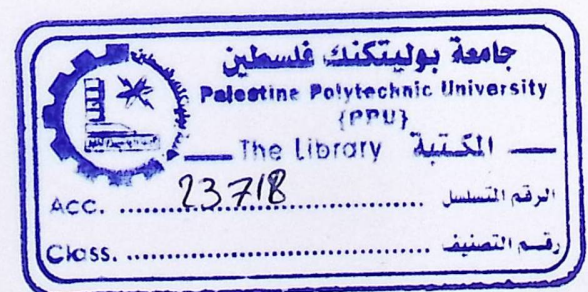
Graduation Project Report

Submitted to the Department of Electrical and Computer Engineering in the College  
of Engineering and Technology

Palestine Polytechnic University

This report is submitted in partial fulfillment of the requirements for the degree of  
B.Sc. in Computer Systems Engineering

Hebron – Palestine  
2008





جامعة بوليتكنيك فلسطين  
الخليل - فلسطين  
كلية الهندسة و التكنولوجيا  
دائرة الهندسة الكهربية والحاسوب

اسم المشروع

لوحة مفاتيح لاسلكية

أسماء الطلبة

مجدي خالد حنيح

محمد احمد ابو صوي

بناء على نظام كلية الهندسة و التكنولوجيا و اشراف و متابعة المشرف المباشر على المشروع و موافقة  
اعضاء اللجنة الممتحنة تم تقديم هذا المشروع الى دائرة الهندسة الكهربية و الحاسوب وذلك للوفاء بمتطلبات درجة  
البكالوريوس في الهندسة تخصص هندسة كهربية فرع هندسة انظمة حاسوب.

توقيع المشرف

توقيع المشرف

توقيع رئيس الدائرة



Palestine Polytechnic University  
Palestine - Hebron  
College of Engineering & Technology  
Electrical & Computer Engineering Department

Project Name

Wireless Keyboard

Project Team

Mohammad Ahmad Abu sway

Majdi Khaled Hneehen

According to Electrical & Computer Engineering Department in Palestine polytechnic university and live follow-up of our supervisor on our project and agreement of examinant commission member we present this project to Electrical & Computer Engineering Department in Palestine polytechnic university to finish the requirement for the degree of Bachelor in computer system engineering

Supervisor signature

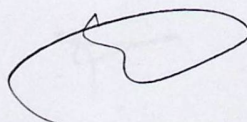
.....

Supervisor signature

.....

Department chief signature

.....





## DEDICATION

To those who give of themselves

So that others may live

To our Families for their patience

To our Colleagues for their support and encouragement

To our Supervisor Eng. Amal Dweik for her supports and advices

Mohammad Abuway  
Majid Hachem  
Date: 23/01/2019

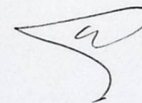


## ACKNOWLEDGMENT

All praise be to Allah The Ultimate Guide and The Cherisher who gave us the courage and ability to complete this project with a satisfying degree of perfection. The most deserving of our acknowledgments is our respected supervisor **Dr. Amal Dweik** who helped us throughout the research of the present project. Next we want to express our deepest gratitude to our teacher **Eng Sami Al Salameen and Eng Rasmi Sayed Ahmad** for taking all the pain to help us in troubleshooting and give us some valuable tips. Thanks are also due to all those who by way of encouragement, prayer or advice might have contributed directly or indirectly towards the actualization of this work.



Mohammad Abusway  
Majdi Hnehen  
Dated: 23/01/2008





## Abstract

The system design wireless keyboard by using a keyboard contains 102 keys to 108 keys, to convert it to wireless keyboard with Radio Frequency signal

The project contains tow main parts

First: Transmitter circuit that will be connected with keyboard to receive data from keyboard and transmit it to computer.

Second: Receiver circuit that will be connected with computer to receive data from transmitter and convert it to computer .

We expect from this project to achieve successful percent about %90 because there is some error that cause from the hardware part.

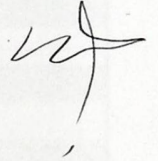
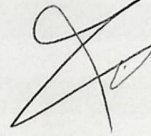


## المخلص

نهدف في هذا المشروع إلى عمل لوحة مفاتيح لاسلكية باستخدام إشارات الراديو لنقل البيانات بدل من الوصلة السلكية ويعتمد المشروع على مبدأ عمل PS2 وبالتالي سوف يتم العمل بالمشروع من قطع وبرمجة في المنطقة ما بين وصلة PS2 في الحاسوب والكيبل الموصول في لوحة المفاتيح ويضم المشروع قسمان أساسيان :

الجزء الأول : قسم الإرسال من لوحة المفاتيح وهذا الجزء يقوم باستقبال البيانات من لوحة المفاتيح ومن ثم يرسلها عبر إشارات الراديو إلى قسم الاستقبال في الجزء الثاني.

الجزء الثاني : قسم الاستقبال ويقوم و يقوم هذا الجزء باستقبال الإشارات اللاسلكية من القسم الأول ونقل البيانات إلى الحاسوب وبالتالي سوف يتم العمل على مبدأ PS2 بجميع جوانبه وطريقة عمله.





# *Content*

Section	Content	Page
<b>Chapter 1: Introduction</b>		<b>1</b>
1.1	Overview	2
1.2	General Idea of the System	2
1.3	Literature Review	3
1.4	Human resources	3
1.5	Estimated Cost	5
1.6	Time Schedule	6
1.7	Risk Management	7
1.8	Project Scope	11
1.9	Report's Contents	11
<b>Chapter 2: Theoretical Background</b>		<b>13</b>
2.1	Overview	14
2.2	System Requirement	14
2.2.1	User Requirement	14
2.2.1.1	Functional Requirement	14
2.2.1.2	Non Functional Requirement	15
2.2.2	System Requirement	15
2.2.2.1	Functional Requirements	16
2.2.2.2	Non Functional Requirements	16
2.3	Theoretical Background	17
2.3.1	Transmitter	17
2.3.2	Microcontroller	23
2.3.3	Serial Port	23
2.4	System Components	25
2.4.1	Hardware Components	25
2.4.1.1	Keyboard	25
2.4.1.2	RT4-433 Transmitter	32
2.4.1.3	RR3-433 Receiver	36
2.4.1.4	ATMega32 Processors	38
2.4.1.5	Why this microcontroller	39
2.4.2	Software Components	39



<b>Chapter: Architectural Design</b>		<b>43</b>
3.1	Overview	44
3.2	System Objectives	45
3.3	System Block Diagram	45
3.4	How Does The System Work?	47
3.5	System Modeling	47
3.5.1	Data Flow Model	48
3.5.2	User Case Model	49
3.5.3	Sequence Diagram	50
<b>Chapter 4: Detailed Technical Design</b>		<b>52</b>
4.1	Detailed description of the project phases	53
4.2	Subsystem detailed design	54
4.2.1	Transmitting Subsystem:	54
4.2.2	ATMega32 Processors	56
4.2.3	Receiving Subsystem	59
<b>Chapter 5: Software System Design</b>		<b>62</b>
5.1	Introduction	63
5.2	Software Project	63
5.3	How System Software Works?	64
5.3.1	keyboard Side Software	65
5.3.2	Computer Side Software	66
5.4	Code List	68
<b>Chapter 6: Implementation And Testing</b>		<b>69</b>
6.1	Introduction	70
6.2	Implementation	70
6.3	Testing	71
6.3.1	Hardware Testing	72
6.3.1.1	Microcontroller testing	72
6.3.1.2	Transmitter and Receiver Testing	73
6.3.1.3	keyboard	74



6.3.1.4	Input Circuit Testing	75
6.3.1.5	Output Circuit Testing	76
6.3.2	subsystem Testing	76
6.3.3	Software Testing	78
<b>Chapter 7: Conclusions and Future Works</b>		<b>83</b>
7.1	Conclusions	84
7.2	Problems	85
7.3	Future Work	86

## LIST OF TABLES

<b>Table 1.1:</b> Development plan and team duties for the second semester 2006/2007	4
<b>Table 1.2:</b> Suggested development plan and team duties for first semester 2007/2008.	5
<b>Table 1.3:</b> The cost for each Component	6
<b>Table 1.4:</b> Project scheduling stages and time table for the second semester 2006/2007.	7
<b>Table 1.5:</b> Suggested Project scheduling stages and time table for the first semester 2007/2008	7
<b>Table 2.1:</b> Serial Port Pins	24
<b>Table 4.1:</b> Electrical Characteristics of RT4-XXX	56
<b>Table 4.2:</b> Electrical Characteristics of RR3-XXX	60



## LIST OF FIGURES

Figure (2.1): Electromagnetic Spectrum	22
Figure (2.2): 6 PIN MINI-DIN FEMALE (PS/2 STYLE) at the computer	24
Figure (2.3): The Keyboard	26
Figure (2.4): The microprocessor and controller circuitry of a keyboard	26
Figure (2.5): The key matrix	27
Figure (2.6): Scan code 1	29
Figure (2.7): scan code 2	30
Figure (2.8): One byte transmission from the Keyboard	31
Figure (2.9): One byte transmission to the Keyboard	31
Figure. (2.10): RT4-433.92 Transmitter	32
Figure (2.11) : Basic transmitting unit	33
Figure (2.12): Reciprocal Mixing with Non-Ideal Oscillator	34
Figure (2.13): Output power of amplification	35
Figure. (2.14): RR3-433.92 Receiver	36
Figure (2.15): Basic block for receiver	37
Figure (2.16): Pin layout for ATmega32 Processor	38
Figure ( 3.1 ) general block diagram of system	46
Figure ( 3.2 ) data flow model	48
Figure ( 3.3 ) use case model	49
Figure ( 3.4 ) Sequence Diagram	50
Figure (4.1): Overall System Block Diagram	53
Figure (4.1): Schematic Diagram of Transmitting Unit	55
Figure (4.2): Schematic Diagram of Receiving Unit	59
Figure (5.1): ATmega32 Programming Phases	64
Figure (5.3) : flow chart Keyboard side	66
Figure (5.4): flowchart Computer side	67
Figure (6.1) data input to the transmitter	73
Figure (6.2) output data from receiver	74
figure (6.3) keyboard clock and data	75
Figure (6.4 ) ATmega32 and transmitter testing	76
Figure (6.5) ATmega32 ,transmitter and receiver testing	76
Figure( 6.7 ) baud rate test	79
Figure( 6.7 ): first situation for MC	79
Figure( 6.8 ): Read first bit	80



Figure( 6.9 ): Read second bit	80
Figure( 6.10 ): Read bit7	81
Figure( 6.11 ): Read the last bit and light the led on port B	81
Figure( 6.12 ):Receive 0x55 by the UART	82
Figure( 6.13 ): Send 0x55 to computer on Oscilloscope	82

## Introduction

- 1.1. Overview
- 1.2. General idea of the system
- 1.3. Literature Study
- 1.4. Human resources
- 1.5. Estimated Cost
- 1.6. Time Schedule
- 1.7. Risk Management
- 1.8. Project Name
- 1.9. Report Contents



# *1*

---

## *Introduction*

- 1.1. Overview**
- 1.2. General Idea of the System**
- 1.3. Literature Review**
- 1.4. Human resources**
- 1.5. Estimated Cost**
- 1.6. Time Schedule**
- 1.7. Risk Management**
- 1.8. Project Scope**
- 1.9. Report's Contents**



## **1.1 Overview :**

Wireless keyboard is a technology which enables the work with computer without constraint like cable or distance by using wireless technology.

You can set upright holding the keyboard away from your body, and hold it where you want .

The system design a wireless keyboard that uses Radio Frequency ( RF ) to transmit signals to the computer.

## **1.2 General Idea of the System:**

A keyboard is a primary input device. Using a keyboard, a person can type a document, use keystroke shortcuts, access menus, play games and perform a variety of other tasks.

In our project we will design a wireless keyboard by using a keyboard contains 102 keys. In general, the computer today has several types of input / output devices which are connected to computer by wires . In our system we wish to reduce the number of wires by converting the local keyboard to wireless keyboard .



### **1.3 Literature Review**

Wireless technology is becoming one of the technologies which is used in large area in our life ,like network , and can be designed in different ways .

The system use this technology to build a wireless keyboard, This technology is used in several projects some could be mentioned below :

In Palestine polytechnic university, one of the projects that uses wireless is Portable Security System ,done by the Susan Abu Sharekh and Waheba yaghmoor and, their system uses radio frequency technology to transmit signal between sursor and to the computer to process the signal and provide the user with condition in the dedicated place which we will use in our project .

In the world exist many systems based on wireless technology like wireless keyboard using Infrared, Bluetooth and RF such as students Luke Hejnar and Sean Leventhal designed wireless keyboard using RF .

### **1.4 Human resources:**

The team of the project consists of two students in electrical and computer Engineering .

Project Team:



Mohammad Abu sway

Majdi hneehen

Supervisor:

Eng. Amal Al-Dweik

The following table shows the tasks for the team members for second semester of 2007.

**Table 1.1:** Development plan and team duties for the second semester 2006/2007

		2007												
ID	The Task	2/10	2/17	2/24	3/3	3/10	3/24	3/31	4/7	4/14	4/21	4/29	5/5	5/12
1	Decision of idea	All												
2	Full d Description	All Team												
3	Study	All Team												
4	First Design	Majdi												
5	Initial implementation	Mohammad												
6	Revision	All												
7	Documentation	All Team												

The following table shows the suggested tasks of the team members for the first semester 2007.

**Table 1.2:** Suggested development plan and team duties for first semester 2007/2008.



		2007												
ID	The Task	9/19	9/26	10/3	10/10	10/10	10/17	10/24	10/31	11/7	11/14	11/21	11/28	12/5
1	Hardware circuits	All												
2	Interfacing circuit	All Team												
3	Software implementation	All Team												
4	Testing	ALL												
5	Documentation	All												

### 1.5 Estimated Cost:

The project needs both hardware equipments and software programs that are used to program the Microcontroller (MC) , so we will purchasing all needed electronic components and parts and software programs.

The Hardware Components. There are many electrical Chips and equipments have to be provided Such as:-

**Table 1.3:** The cost for each Component



<b>Component</b>	<b>Cost(\$)</b>
RT4-433	12\$
RR4-433	12\$
Two Microcontrollers	40\$
One Keyboard	3\$
Wires	1\$
Power supply	2\$
Two Prototyping breadboards	4\$
1 STK 500 board	2\$
Miscellaneous electrical components	2\$
<b>Total cost</b>	<b>82\$</b>

### 1.6 Time schedule:

In the following tables we view the stages in designing and building the components of the project and the timing for each stage.

**Table 1.4:** Project scheduling stages and time table for the second semester 2006/2007.



ID	Task Name	Start	Finish	Duration	Sep 2006	Oct 2006					Nov 2006				Dec 2006		
					9/24	10/1	10/8	10/15	10/22	10/29	11/5	11/12	11/19	11/26	12/3	12/10	
1	Idea Decision	09/20/2006	09/26/2006	1w	▽△												
2	Full Description	09/27/2006	10/10/2006	2w	▽△												
3	Study	10/11/2006	11/28/2006	7w	▽△												
4	Primary Design	11/08/2006	11/28/2006	3w	▽△												
5	Primary Implementation	11/21/2006	12/11/2006	3w	▽△												
6	Revision	12/07/2006	12/20/2006	2w	▽△												
7	Documentation	10/19/2006	12/20/2006	9w	▽△												

**Table 1.5:** Suggested Project scheduling stages and time table for the first semester2007/2008.

ID	The Task	Start	Finish	Duration	2007													
					1w	2w	3w	4w	5w	6w	7w	8w	9w	10w	11w	12w	13w	14w
1	Hardware circuits	First W	Tenth W	10W	All													
2	Interfacing circuit	Second W	Fourth W	4W	All Team													
3	Software implementation	Third W	Tenth W	7W	All Team													
4	Testing	Tenth W	Thirteenth W	3W	All													
5	Documentation	First W	Fourteenth W	14W	All													

## 1.7 Risk Management

There are some possible risks that may occur in our project in both hardware and software.



## 1 - Technology Risks

Such risks may occur because of the software or hardware used in the system.

### - Hardware Risks:

- Device failure: the microcontroller may crash because of high voltage supply or other problems.
- tool risk , like breadboard , resistor ,and crystal .
- important pins of microcontroller is failed like interrupt pin .
- power supply give un expected vales .
- Oscilloscope doesn't give accurate signal .
- The bred board used doesn't arrive any signal .

### - Software Risks:

- Compiler give unwanted hex value .
- compiler need to be updated .
- The software on the keyboard side has task , any problem in this task may cause risk .
- computer where program and compiler work may damage .
- oscilloscope in the lab may give wrong signal .
- power supply may give wrong value which cause risk in our plan of project .



## **2 - People Risks**

- Illness of one or more of group members.
- Group meeting difficulties

## **3 - Tools Risks**

Lose of any support software or hardware used to develop the system, like microcontroller programming tools and case tools in software.

## **4 - Requirements Risks**

Risks that might occur if new changes are required in the system requirements that need major changes in the system design.

## **5 - Estimation Risks**

- Risks that may derive from the wrong estimation in the system design, implementation, resources and management.
- Such as read data by MC from the keyboard at inappropriate times will be generate incorrect code.



**Risk Avoidance:**

- Taking care when using hardware components and using them according to their specifications.
- Taking care of the team's member's health during the project development.
- Good estimation and usage of the projects budget and resources.
- Good estimation of system requirements.

**Risk Management:**

- Software development environment risks will be handled by the backup of software.
- Including an extra amount of the hardware components we already have, so when any problem occurs we can find an alternative for the component we lost.
- People risks are handled by using work load balancing on students especially when a member can't perform some of his tasks, then it will be done by other member.



## **1.8 Project scope:**

After completing the main requirements of our project, it can be used with a computer as any wire keyboard with enhanced features as:

- Trade show displays
- Using wireless keyboard for games, printing, control on the computer..etc .
- management your computer from any where in the room .

## **1.9 Report content:**

Report consists of four chapters; the following is a brief description of each chapter.

### **Chapter 1: Introduction**

This chapter presents overview of project, general idea of system, literature review, human resources, estimated cost, time schedule, risk management, project scope, and report content .

### **Chapter 2: Theoretical back ground**

This chapter talks in more details about the basic component used in the project and theoretical back ground.



### **Chapter 3: Architectural Design**

This chapter details the design concepts, introduces project objectives, shows the general block diagram of the system and explains how the system works.

### **Chapter 4: Detailed Technical Design**

This chapter presents out lines formal procedure for design, discusses design options and justifies those chosen for the project.

### **Chapter 5: Software System Design**

This chapter handles the software related to our system, depicts flowcharts about system operation.

### **Chapter 6: System implementation and Testing**

This chapter handles implementation of the project and block testing ,unit testing , integration testing of the system . The testing comprises both software and hardware testing.

### **Chapter 7: Conclusion and Future work**

This chapter provides the conclusions that will be concluded after working the system, and suggestion for future work.



# 2

---

## *Theoretical Background*

**2.1. Overview**

**2.2. System Requirement**

**2.3. Theoretical Background**

**2.4. System Components**



## **2.1. Overview:**

This chapter provides an illustrative theoretical background of our project applications in general and for each of its components in particular, and the system requirements .

## **2.2. System Requirements:**

### **2.2.1 User Requirements :**

User requirements contains the following :

#### **2.2.1.1 Functional Requirements**

- 1- when the user presses any key on the keyboard , the keyboard will generate a code that represents the keys.
- 2- the receiver circuit should receives the same code from transmitter circuit.
- 3- wireless keyboard will be implemented under windows or any other operating system .



### **2.2.1.2 Non-Functional Requirements**

Non-Functional Requirements described as follows :

#### **Flexibility**

The wireless keyboard will provide the user a higher flexibility of usage than traditional keyboard.

#### **Efficiency**

The wireless keyboard with radio frequency solves the problems of other wireless keyboards with infrared technology such as the distance between the transmitter and the receiver circuit.

### **2.2.2. System Requirements**

There are a requirement that specify the system details and specification .

#### **2.2.2.1 Functional Requirements**

-The wireless keyboard includes radio frequency transmitter and receiver circuits to transfer the input signal to computer.



### **2.2.1.2 Non-Functional Requirements**

Non-Functional Requirements described as follows :

#### **Flexibility**

The wireless keyboard will provide the user a higher flexibility of usage than traditional keyboard.

#### **Efficiency**

The wireless keyboard with radio frequency solves the problems of other wireless keyboards with infrared technology such as the distance between the transmitter and the receiver circuit.

### **2.2.2. System Requirements**

There are a requirement that specify the system details and specification .

#### **2.2.2.1 Functional Requirements**

-The wireless keyboard includes radio frequency transmitter and receiver circuits to transfer the input signal to computer.



-This circuits will be monitored by two microcontrollers, one of them receive the signal from a keyboard then to radio frequency transmitter. The microcontroller in computer side will receive the signal from RF receiver, then send it to the computer. The C program running on the microcontroller responsible for interfacing signal between the transmitter and the receiver circuits .

### **2.2.2.2 Non Functional Requirements**

1- Reliability: the system should be reliable, meaning it must send the code represent the keys which we press not other code. For example when user press (Ctrl + A) keys, the microcontroller of the key board should send the represent the two, keys we press.

2- Speed: The system must have a fast response to the user's actions.

3- Portability: the user must be able to change the place of the keyboard.

## **2.3 Theoretical Background:**

As we have mentioned previously this project is fully constructed over a communication between the keyboard and PC , we have used RF signal to establish communication.



### 2.3.1 Transmitter

A transmitter is an electronic device which with the aid of an antenna propagates an electromagnetic signal such as radio, television, or other telecommunications.

A transmitter usually has a power supply, an oscillator, a modulator, and amplifiers for audio frequency (AF) and radio frequency (RF). The modulator is the device which modulates the signal information onto the carrier frequency, which is then broadcast. Sometimes a device contains both a transmitter and a radio receiver, with the combined unit referred to as a transceiver.

More generally and in communications and information processing, a transmitter is any object (source) which sends information to an observer (receiver). When used in this more general sense, vocal cords may also be considered an example of a transmitter.

### Types of Transmission Signals

There are many types for signal transmission these types are:

#### **Ultrasound:**



### 2.3.1 Transmitter

A transmitter is an electronic device which with the aid of an antenna propagates an electromagnetic signal such as radio, television, or other telecommunications.

A transmitter usually has a power supply, an oscillator, a modulator, and amplifiers for audio frequency (AF) and radio frequency (RF). The modulator is the device which modulates the signal information onto the carrier frequency, which is then broadcast. Sometimes a device contains both a transmitter and a radio receiver, with the combined unit referred to as a transceiver.

More generally and in communications and information processing, a transmitter is any object (source) which sends information to an observer (receiver). When used in this more general sense, vocal cords may also be considered an example of a transmitter.

### Types of Transmission Signals

There are many types for signal transmission these types are:

#### **Ultrasound:**



Ultrasound is sound of such high frequency, that is, it can not be heard by the human ear. Atypical ultrasound transducer operates at a frequency of 40 KHz but the ear can hear up to 20KHz. Ultrasound transducer is relatively inexpensive and the circuit required is simple to build.

### **Radio Frequency (RF):**

Radio control has the advantage of the operating over a far greater range than ultrasound, also it does not require the physical connection as in the cable control, it is very popular for the control of aircraft's, boat's.....etc.

### **RF Advantages:**

Not line of sight.

Not blocked by common materials: can penetrate most solids and pass through walls.

Longer range.

Not light sensitive.

Not as sensitive to weather/environmental conditions.



### **RF Disadvantages:**

Higher cost than infrared.

Federal Communications Commission (FCC) licenses required for some products.

Lower speed: data rate transmission is lower than wired and infrared transmission.

### **Bluetooth Advantages:**

Its wireless. Eliminate messy and confusing cables at home, the office, or when traveling.

It's inexpensive. Vendors can incorporate it into their products and consumers can upgrade older equipment cheaply.

Uses radio signals so can pass through walls and does not require line of sight.

Lower power consumption. Won't drain your battery.

2.5 GHz radio frequency ensures worldwide operation.

No thinking required. The devices find one another and connect without any user input at all.

### **Bluetooth Disadvantages:**



New type of technology and must be accepted unequivocally by all vendors and manufacturers in order to guarantee compatibility among the array of products.

Designed for only short-range communications certified to no more than 100 meters (with hub broadcaster.)

Data transfer speeds are not as fast as other wireless technologies.

### **Infrared (IR):**

Infrared (IR) is a light-based transmission technology and is not spread spectrum IR devices can achieve a maximum data rate of 4 Mbps at close range, but as a light-based technology, other sources of IR light can interfere with IR transmissions.

The typical data rate of an IR device is about 115 kbps, which is good for exchanging data between handheld devices.

An important advantage of IR networks is that they do not interfere with spread spectrum RF networks. For this reason, the two are complementary and can easily be used together.

### **IR Advantages:**

Low power requirements: therefore ideal for laptops, telephones, personal digital assistants.



Low circuitry costs: \$2-\$5 for the entire coding/decoding circuitry

Simple circuitry: no special or proprietary hardware is required, can be incorporated into the integrated circuit of a product.

Higher security: directionality of the beam helps ensure that data isn't leaked or spilled to nearby devices as it's transmitted.

Portable.

High noise immunity: not as likely to have interference from signals from other devices.

### **IR Disadvantages:**

Line of sight: transmitters and receivers must be almost directly aligned (i.e. able to see each other) to communicate

Blocked by common materials: people, walls, plants, etc. can block transmission

Short range: performance drops off with longer distances

Light, weather sensitive: direct sunlight, rain, fog, dust, pollution can affect transmission

Speed: data rate transmission is lower than typical wired transmission

### **2.3.2 Microcontroller:**



A microcontroller is a cheap single chip microcomputer. Single-chip microcomputer indicates that the complete microcomputer system lies within the confine of the integrated circuit chip. Microcontrollers are capable of storing and running the program that was written, compiled and downloaded into it. The main parts of a microcontroller generally consist of the Central Processing Unit (CPU), Random Access Memory (RAM), Read Only Memory (ROM), input/output lines (I/O lines), serial and parallel ports, timers and other peripherals such as analog to digital (A/D) converter and digital to analog (D/A) converter.

### **Why to use a Microcontroller?**

Microcontrollers are inexpensive microcomputers. The microcontroller's ability to store and run unique programs makes it very flexible.

### **2.3.3 Serial Port:**

It is an I/O device is just a way to get data into and out of a computer. The serial port is much more than just a connector. It converts the data from parallel to serial and changes the electrical representation of the data. Inside the computer, data bits flow in parallel ( using many wires at the same time ). Serial flow is a stream of bits over a single wire (such as transmit or receive pin of the serial connector). For the serial port to create such a flow, it must convert data from parallel (inside the computer) to serial on the transmit pin and conversely.



### Pins and Wires:

The pins and the wires of the serial port; the PC's used 6 pin connectors but only about 4 pins were actually used so today most connectors are only 4-pin. Each of the 4 pins usually connects to a wire. Besides the one wire used for transmitting and receiving data, another pin (wire) is signal clock .and other ground and one to voltage , (see figure 2.2 and Table 2.1).

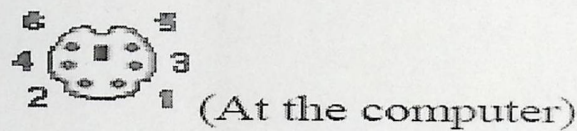


Figure 2.2: 6 PIN MINI-DIN FEMALE (PS/2 STYLE) at the computer.

Table (2.1): Serial Port Pins

Pin	Name	Dir	Description
1	DATA	↔	Key Data
2	n/c	-	Not connected
3	GND	—	Gnd
4	VCC	→	Power , +5 VDC
5	CLK	→	Clock
6	n/c	-	Not connected

## 2.4 System Components



## 2.4.1 Hardware Components:

In this section we provide a full explanation of each component and each part of this project. The system consists of the following components:

Keyboard

Transmitter Unit.

Receiver Unit.

ATMega32 Microcontroller.

### 2.4.1.1 keyboard (KBD) :

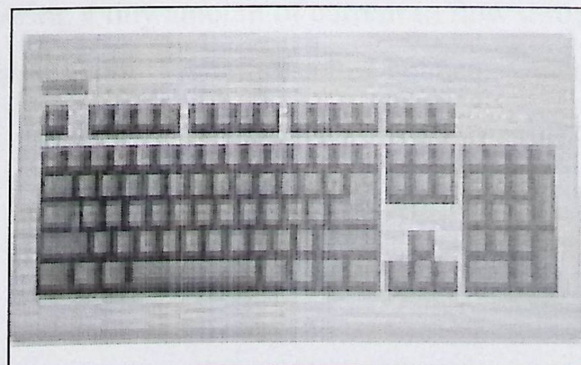
A keyboard's primary function is to act as an input device. Using a keyboard, a person can type a document, use keystroke shortcuts, access menus, play games and perform a variety of other tasks. Keyboards can have different keys depending on the manufacturer. They're also placed at a similar distance from one another in a similar pattern, no matter what language or alphabet the keys represent.

Most keyboards have between 80 and 110 keys, including:

Typing keys

A numeric keypad

Function keys





Control keys

Figure 2.3: The Keyboard

### Inside the Keyboard

A keyboard is a lot like a miniature computer. It has its own processor and circuitry that carries information to and from that processor. A large part of this circuitry makes up the key matrix.

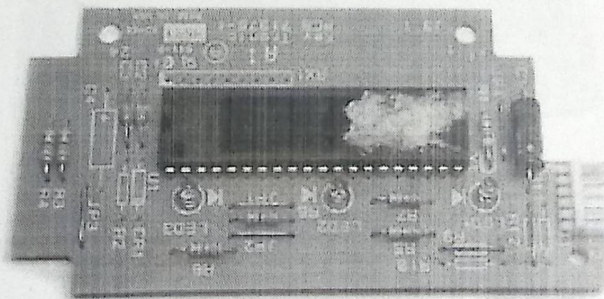


Figure 2.4: The microprocessor and controller circuitry of a keyboard

The key matrix is a grid of circuits underneath the keys. In all keyboards each circuit is broken at a point below each key. When you press a key, it presses a switch, completing the circuit and allowing a tiny amount of current to flow through. The mechanical action of the switch causes some vibration, called bounce, which the



processor filters out. If you press and hold a key, the processor recognizes it as the equivalent of pressing a key repeatedly.

When the processor finds a circuit that is closed, it compares the location of that circuit on the key matrix to the character map in its read-only memory (ROM). A character map is basically a comparison chart or lookup table.

It tells the processor the position of each key in the matrix and what each keystroke or combination of keystrokes represents. For example, the character map lets the processor know that pressing the a key by itself corresponds to a small letter "a" but the Shift and a keys pressed together correspond to a capital "A."

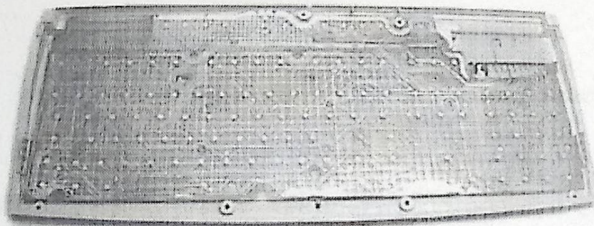


Figure 2.5: The key matrix

A computer can also use separate character maps, overriding the one found in the keyboard.

### PC Keyboard Theory



The keyboard contains processor to scan which key pressed then it will send scan codes to your computer. The scan codes tell your Keyboard Bios, what keys you have pressed or released. Take for example the 'A' Key. The 'A' key has a scan code of 1C (hex). When you press the 'A' key, your keyboard will send 1C down it's serial line. If you are still holding it down, for longer than it's typematic delay, another 1C will be sent. This keeps occurring until another key has been pressed, or if the 'A' key has been released.

However the keyboard will also send another code when the key has been released. Take the example of the 'A' key again, when released, the keyboard will send F0 (hex) to tell you that the key with the proceeding scan code has been released. It will then send 1C, so you know which key has been released.

Your keyboard only has one code for each key. It doesn't care if the shift key has been pressed. It will still send you the same code. It's up to your keyboard BIOS to determine this and take the appropriate action. The keyboard doesn't even process the Num Lock, Caps Lock and Scroll Lock. When you press the Caps Lock for example, the keyboard will send the scan code for the cap locks. It is then up to your keyboard BIOS to send a code to the keyboard to turn on the Caps lock LED.

When an extended key has been released, it would be expect that F0 would be sent to tell you that a key has been released. Then you would expect E0, telling you it was an extended key followed by the scan code for the key pressed. However this is not the case. E0 is sent first, followed by F0, when an extended key has been released.



## Scan Codes

The diagram below shows the Scan Code assigned to the individual keys. The Scan code is shown on the bottom of the key. E.g. The Scan Code for ESC is 76. All the scan codes are shown in Hex.

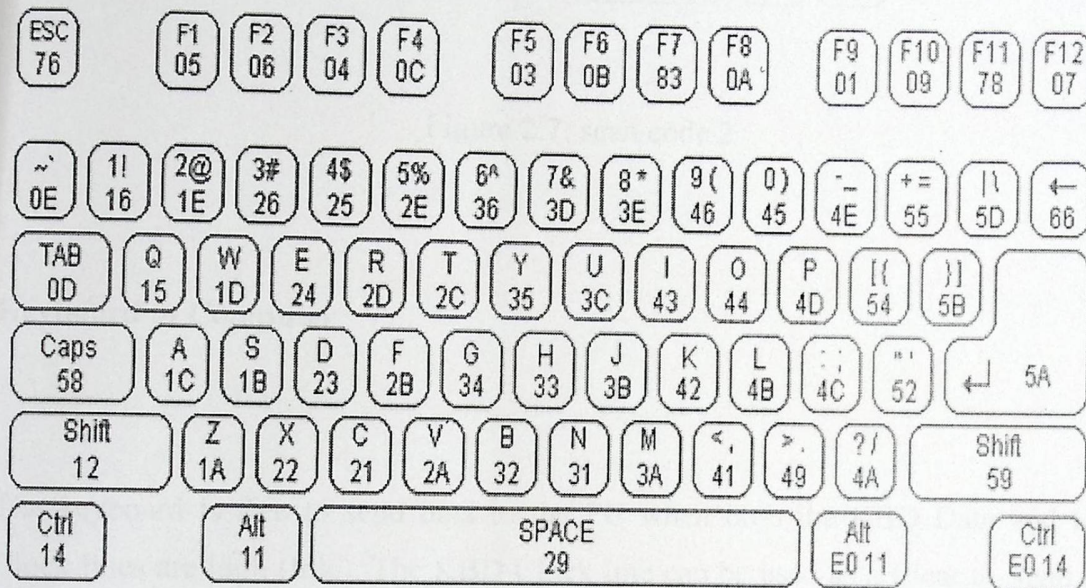


Figure 2.6:Scan code 1

As you can see, the scan code assignments are quite random. In many cases the easiest way to convert the scan code to ASCII would be to use a look up table. Below is the scan codes for the extended keyboard & Numeric keypad.



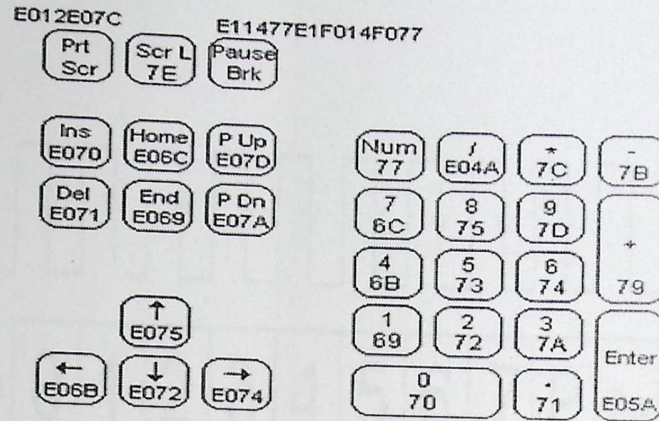


Figure 2.7: scan code 2

### Keyboard to Computer

The keyboard is free to send data to the PC when both the KBD Data and KBD Clock lines are high (Idle). The KBD Clock line can be used as a Clear to Send line. If the PC takes the KBD Clock line low, the keyboard will buffer any data until the KBD Clock is released, ie goes high. Should the Host take the KBD Data line low, then the keyboard will prepare to accept a command from the host.

The transmission of data in the forward direction, ie Keyboard to Host is done with a frame of 11 bits. The first bit is a Start Bit (Logic 0) followed by 8 data bits (LSB First), one Parity Bit (Odd Parity) and a Stop Bit (Logic 1). Each bit should be read on the falling edge of the clock.



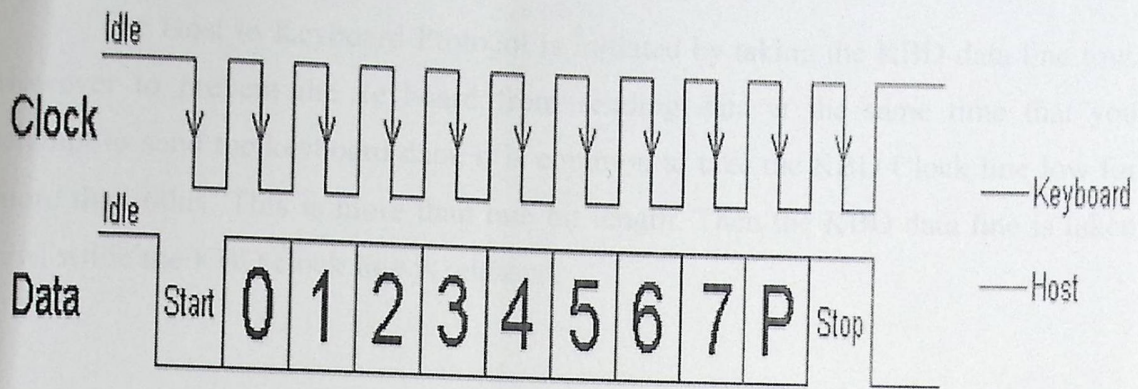


Figure (2.8): One byte transmission from the Keyboard

The above waveform represents a one byte transmission from the Keyboard. The keyboard may not generally change its data line on the rising edge of the clock as shown in the diagram. The data line only has to be valid on the falling edge of the clock. The Keyboard will generate the clock. The frequency of the clock signal typically ranges from 20 to 30 KHz. The Least Significant Bit is always sent first.

### Computer to Keyboard

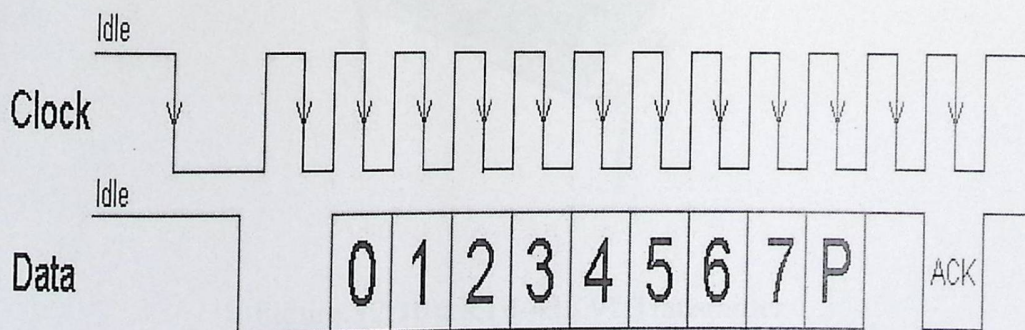


Figure (2.9): One byte transmission to the Keyboard



The Host to Keyboard Protocol is initiated by taking the KBD data line low. However to prevent the keyboard from sending data at the same time that you attempt to send the keyboard data, it is common to take the KBD Clock line low for more than 60us. This is more than one bit length. Then the KBD data line is taken low, while the KBD clock line is released.

The keyboard will start generating a clock signal on it's KBD clock line. This process can take up to 10mS. After the first falling edge has been detected, you can load the first data bit on the KBD Data line. This bit will be read into the keyboard on the next falling edge, after which you can place the next bit of data. This process is repeated for the 8 data bits. After the data bits come an Odd Parity Bit.

#### 2.4.1.2 RT4-433 Transmitter:

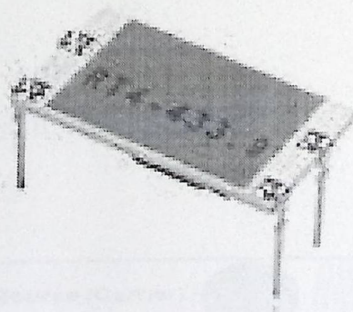


Figure. (2.10): RT4-433.92 Transmitter

We use a RT4-XXX transmitter that is a hybrid circuit that allows realizing a



complete radio transmitter. It is a wireless (AM) transmitter with range up to 75m; it can be used in car alarm system or gate remote controlling. It is a complete RF transmitter with very stable frequency.

The transmitter unit consists of one microcontroller ,and the wireless transmitter. When a person press any key on keyboard special signal will be generated , a signal will be send to the MCU on the transmitter which is programmed so that it can handle with all keyboard keys .

The transmitter is responsible for sending the appropriate message from the transmitter MCU to the receiver circuit next to computer .

### How Does Transmitter Work?

We will explain all the levels of transmission process, as shown in the Figure(2.11).

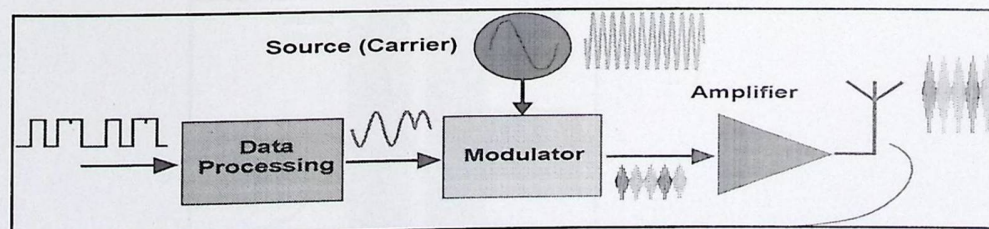


Figure 2.11: Basic transmitting unit



### Frequency Source (Carrier):

In order to make the information signal pass through the air, it must be modulated on to a carrier signal which is sufficiently stable with temperature and other factors to allow detection by a tuned receiver in the presence of interference.

The two most important factors affecting the design of a carrier frequency source are: Frequency Stability and Phase Noise.

The stability of the oscillator with temperature determines the channel spacing required to contain the modulated carrier signal. Conversely, for a given regulated channel spacing, the frequency stability determines the maximum data rate that can be supported without violating the channel boundary.

The oscillator phase noise results in a broadband component to the carrier signal which will extend into adjacent channels. If the phase noise is too high, this can corrupt the modulation source itself, and limit adjacent channel selectivity due to reciprocal mixing, as shown in Figure (2.12).

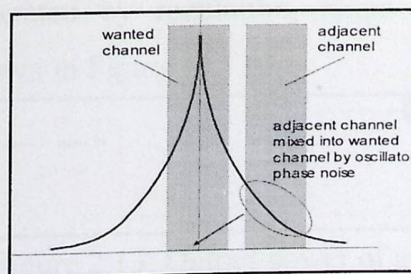


Figure 2.12: Reciprocal Mixing with Non-Ideal Oscillator



### Modulation:

The method of imposing the information signal onto the carrier signal is termed *modulation* which can be AM modulation or FM modulation and must be accomplished cost effectively and accurately for maximum range and minimum interference .

Amplitude modulation (AM) is a method of impressing data onto an alternating-voltage (AC) carrier waveform. The highest frequency of the modulating data is normally less than 10 percent of the carrier frequency. The instantaneous amplitude varies depending on the instantaneous amplitude of the modulating data.

### Amplification:

The amplifier is a key part of the transceiver , and must be efficient which means low cost and possibly linear.

Output power is dictated by regulation, range requirement, cost and linearity considerations, as shown in Figure (2.13).

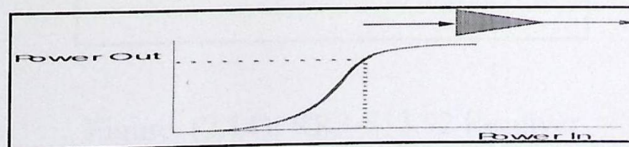


Figure 2.13: Output power of amplification



## Antenna:

The antenna is often the most poorly engineered part of a communication system. Good design will ensure maximum range, high amplifier efficiency (good matching), good selectivity, minimal pollution, good interference rejection, good sensitivity.

### 2.4.1.3 RR3-433 Receiver

Because we are making a wireless system, we need to transmit the signals from the door to the receiver module which is designed such that it can handle the message from transmitter.

Here we used the RR3-433 receiver, which is compatible with the used transmitter.

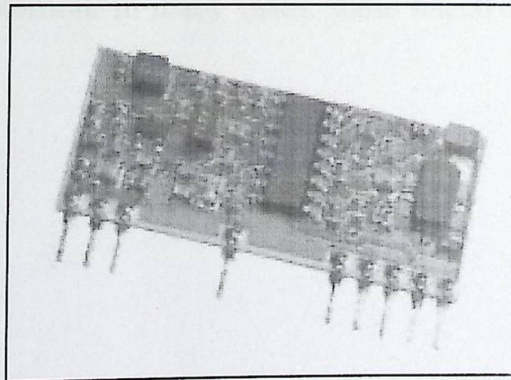


Figure. (2.14): RR3-433.92 Receiver



## How Does Receiver Work?

We will explain all the levels of transmission process, as shown in the Figure (2.15).

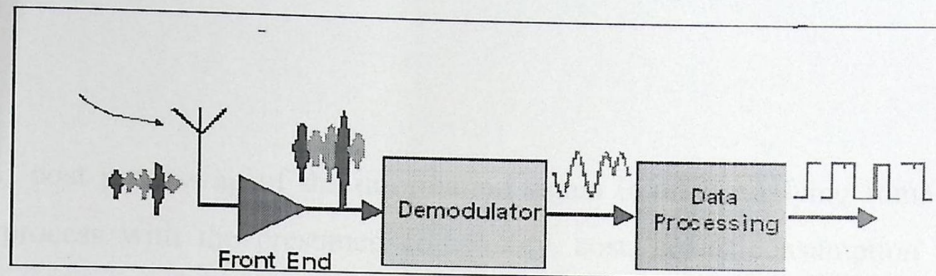


Figure 2.15: Basic block for receiver.

## Reception

Key to the sensitivity is the receiver 'front end'. The main task is to boost weak wanted signals, often in the presence of strong unwanted signals which introduce minimal noise and distortion. In many cases, some selective filtering is required to assist this task.

## Demodulation

The process of removing the information signal from the carrier is termed *demodulation*. The aim is to design a circuit (or algorithm) that will achieve this task



optimally in the presence of noise, interference and varying signal strength, frequency and phase, in addition of being power efficient and cheap.

## Data Processing

Pre and post processing of the information signal is often implying some form of micro process with the presumed complexity, cost, power consumption and size penalties. The benefits of matched filtering, error detection and correction (coding).

### 2.4.1.4 ATmega32 Processors :

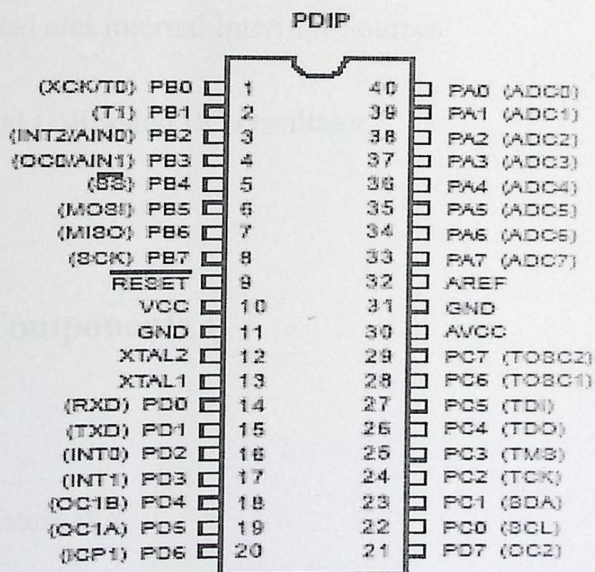


Figure 2.16: Pin layout for ATmega32 Processor .



### 2.4.1.5 Why this microcontroller:

High – performance , low-power

Advance RISC Architecture

Nonvolatile Program and Data Memories

1 - 32K Bytes of In-System Self-Programmable Flash

2 - 1024 Bytes EEPROM Endurance: 100,000 Write/Erase Cycles

3 - 2K Byte Internal SRAM

4 - Programming Lock for Software Security

Special Microcontroller Features

1 - External and Internal Interrupt Sources

2 - Internal Calibrated RC Oscillator

### 2.4.2 Software Components :

The software of the system is :

1- First, it listen for incoming data from the UART .

2- Second, it echoes these codes to the computer.



The microcontroller decodes the sequence of bytes it receives using the state machine shown. As bytes are read from the UART they are added to a queue to be sent on the PS/2 bus.

When the connection is idle, and there is data to send, the microcontroller sets several state variables. These state variables cause an interrupt to generate a clock signal on the PS/2 port and to place data on the PS/2 data line. This interrupt occurs four times for every clock cycle, writing the data, stop, start and parity bits to the PS/2 connection.

The software on the keyboard side has a tasks:

It listens to the data coming from the keyboard and it sends it out to the transmitter through the UART.

The keyboard generates a clock on the Clock line and sends its data on the Data line.

Whenever the clock is goes low, 1 bit of data is present on the Data line and the microcontroller reads it.

To read the data line at the appropriate times., When the interrupt executes, one data bit is shifted into a register.

Since the ps/2 protocol sends a start bit, followed by 8 data bits, then a parity and a stop bit, we only shift in the data bits into a register. Once we have the 8 data bits, we place the byte into a queue.



The ps2 protocol also allows the computer to talk to the keyboard. To do this, we also use the external edge triggered interrupt to place a data bit on the DATA line at the appropriate times so that the keyboard can get it.

All of this operation we will program the MC to do this task in good way .



## Summary

This chapter provides an illustrative theoretical background for our project applications in general and for its each component in particularly like types of Transmission Signals as RF,IR and Bluetooth, and how the wire keyboard works and description fro all chips we will use in our project such as MC , transmitter circuit and receiver circuit and other chips. and the system requirements for user and system .

## Architectural Design

- 3.1 Introduction
- 3.2 Project Objectives
- 3.3 General Block Diagram
- 3.4 How System Works
- 3.5 System modeling



# 3

---

## *Architectural Design*

- 3.1 Introduction**
- 3.2 Project Objectives**
- 3.3 General Block Diagram**
- 3.4 How System Works**
- 3.5 System modeling**



### 3.1 Overview

The wireless keyboard consists of a wired 102-key ps/2 keyboard, two Atmega32 microcontrollers, one 433.92MHz transmitter and one 433.92MHz receiver from Radiotoxic.

Our design can be divided into two parts: the keyboard side and the computer side.

On the keyboard side, we have one of the microcontrollers connected to the keyboard and to the transmitter. The keyboard communicates with the microcontroller and the microcontroller sends data out to the transmitter by using it's built in UART.

On the computer side, we have the other microcontroller connected to the computer and to the receiver. The computer communicates with the microcontroller and the microcontroller gets data from the receiver through its build in UART.

Note that we use the UART because they allow us to easily set the transmit and receive rate and they do all the transmitting and receiving in the background.



### 3.2 System Objectives

This project supports many ideas and objectives that can be summarized as follows:

- 1- The user can use the keyboard without cable constrain or distance, he can work with computer and set in the way he want .
- 2- To make wireless technology available in computer field so any one can use and run this technology with out difficulties and provide software required to run wireless circuit under computer area .

### 3.3 System Block Diagram

figure (3.1) shows block diagram for the system. It consists of several units to be accomplished and integrated to each other to form the final wireless keyboard system ,These units are:

Keyboard : When the user presses any key the signal Data, Clock, VCC, GND is generated from the microcontroller of the key board, this signal sent to the microcontroller of receiver circuit serially, where only one signal at a time .



Microcontrollers : The MC responsible to reads data from keyboard and it will be transmitted to transmitter circuit .

the ps2 protocol also allows the computer to talk to the keyboard . To do this, MC must do this work by telling keyboard to send data or not .

RT4-433: Is the chip that send the signal to the RR3-433 Receiver . The transmitter operates from a 4.5-5.5V supply, making it ideal for battery-powered applications.

RR3-433: It receives the signal from RT4-433 transmitter and sends it to MC then to computer to perform the tasks.

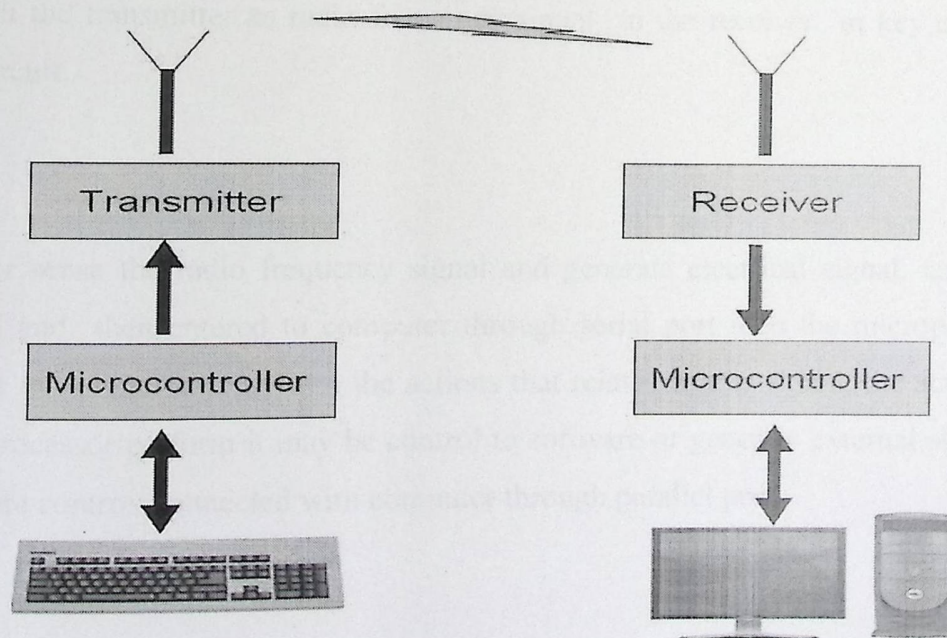


Figure ( 3.1 ) general block diagram of system



### 3.4 How Does The System Work?

When a key is pressed is not sufficient reason to generate a key code. A user may hold a key down for many tens of milliseconds before releasing it.

The keyboard controller must not generate a new key sequence every time it scans the keyboard and finds a key held down. Instead, it should generate a single key code value when the key goes from an up position to the down position (a *down key* operation).

Upon detecting a down key stroke, the microcontroller sends a keyboard *scan code* through the transmitter as radio frequency signal to the receiver in key computer side circuit.

receiver sense the radio frequency signal and generate electrical signal, amplified, filtered and then entered to computer through serial port then the microprocessor process this signal and perform the actions that related to this signal, the action that microprocessor perform it may be control to software or generate external signal for hardware control connected with computer through parallel port.

When the scan code arrives at the PC, a second microcontroller chip receives the scan code, does a conversion on the scan code, makes the scan code available at I/O



port 60h, and then interrupts the processor and leaves it up to the keyboard ISR to fetch the scan code from the I/O port.

### 3.5 System Modeling

#### 3.5.1 Data Flow Model

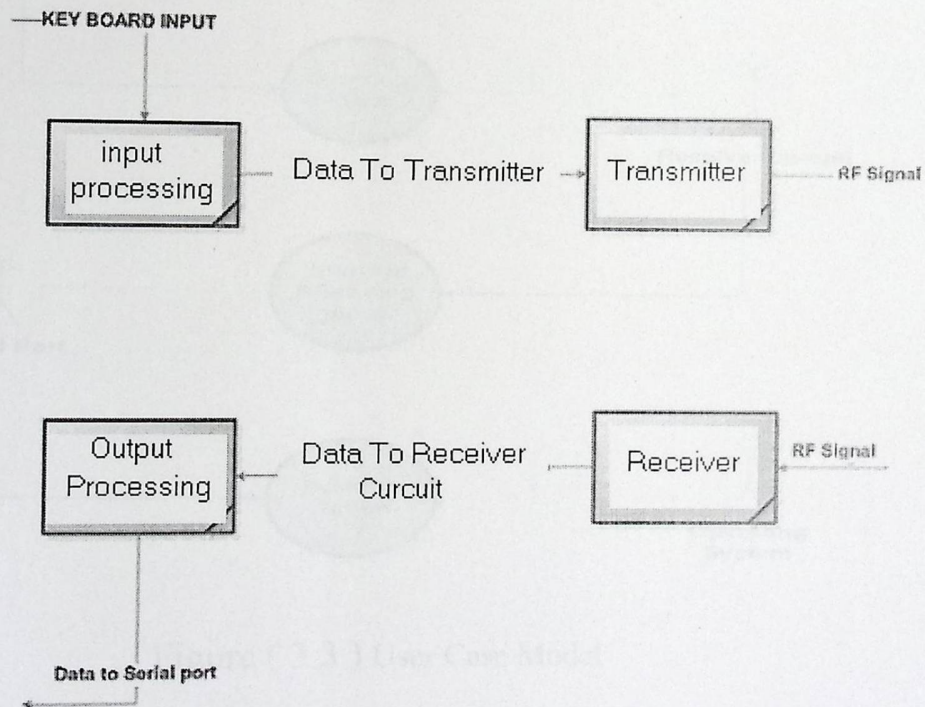


Figure ( 3.2 ) data flow model



### 3.5.2 User Case Model

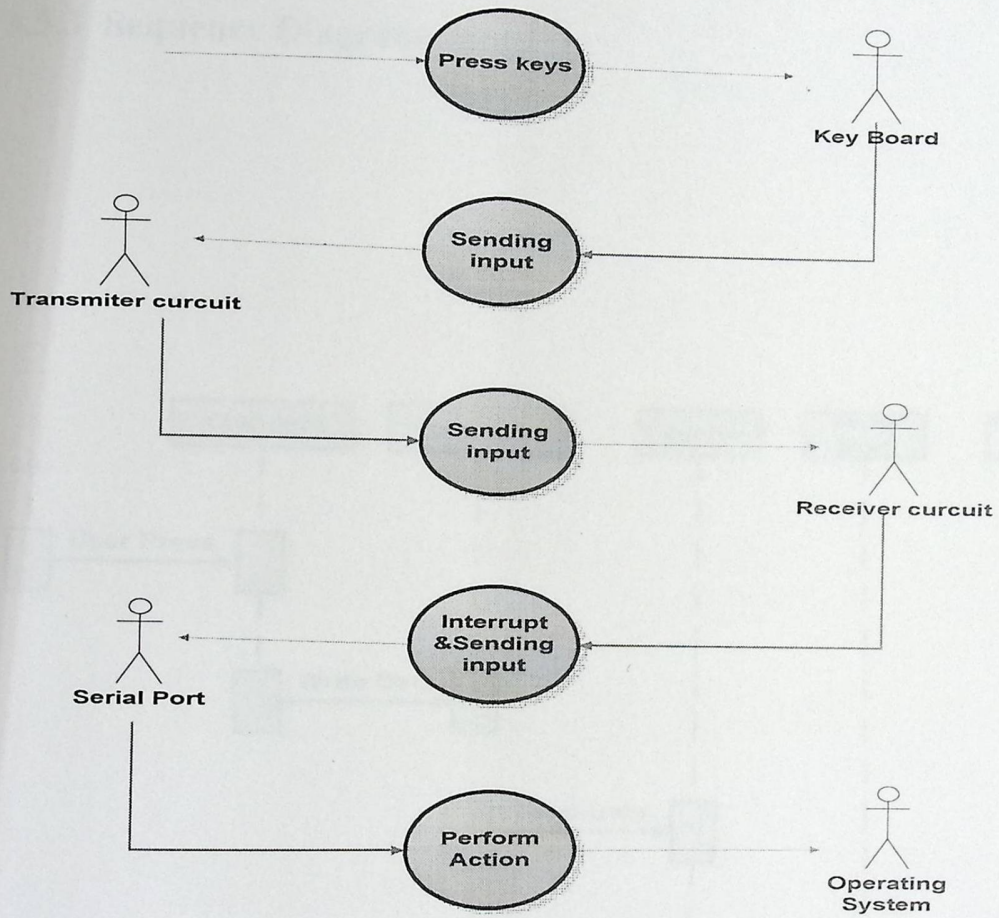


Figure ( 3.3 ) User Case Model



### 3.5.3 Sequence Diagram

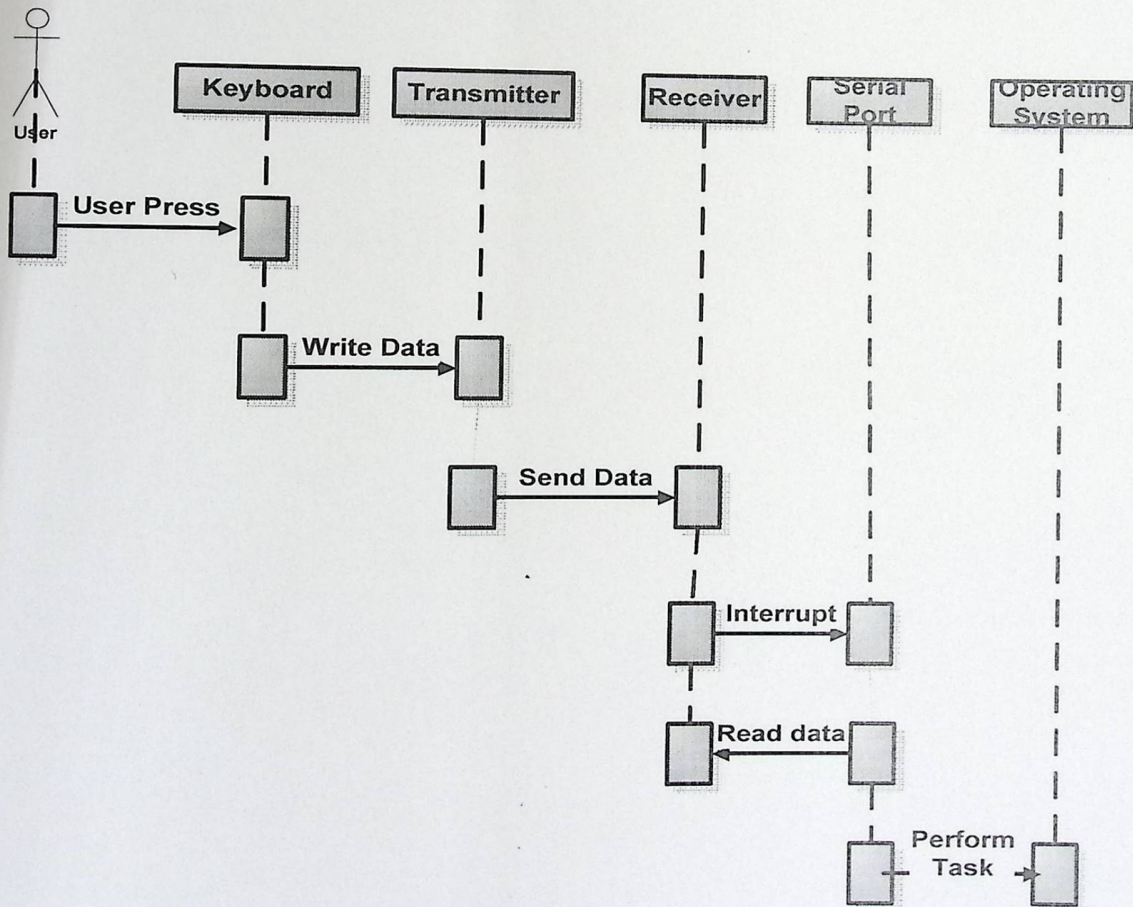
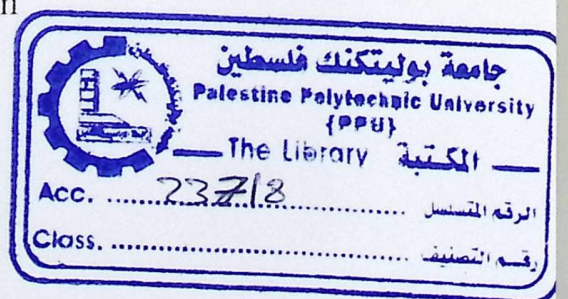


Figure ( 3.3) Sequence Diagram





## Summary

In this chapter we talk about the objective of our project , and the idea that make the wireless keyboard is better for user, then we explain the major component and subsystem of our project , and how this component work to gather to handle any keys was pressed .

## Detailed Technical Design

4.1. Detailed description of the project phases

4.2. Subsystem detailed design



# 4

---

## *Detailed Technical Design*

- 4.1. Detailed description of the project phases
- 4.2. Subsystem detailed design



#### 4.1. Detailed description of the project phases

this section describe the design details of the system as a whole and the details for each component in the system . the following figure generally view the main component in our system and how they are connect to each other .

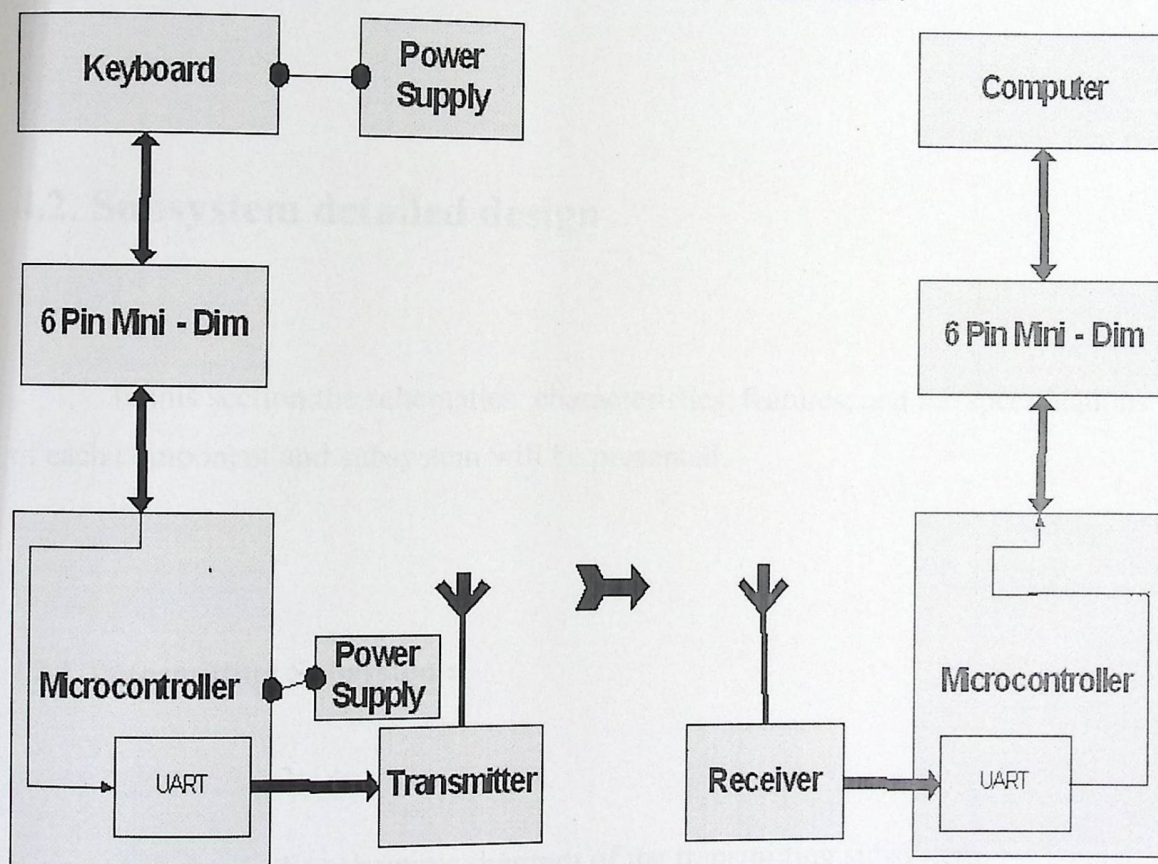


Figure 4.1: Overall System Block Diagram

Figure 4.1 when the user presses any keys on the keyboard, signal is generated to the MC via port D then via the same port will send the code to transmitter RT4-433.92 which will send the code to receiver.



In other side The computer connects to the microcontroller through a 6-pin min-DIN plug and provides power to run both the microcontroller and the receiver, thus there is no need for any batteries.

The microcontroller receives data from the wireless keyboard using the receiver (RR3-433.92). The data output of the receiver is passed to the RXD pin of the microcontroller.

## **4.2. Subsystem detailed design**

In this section the schematics, characteristics, features, and the specifications of each component and subsystem will be presented.

### **4.2.1 Transmitting Subsystem:**

Figure (4.1) shows the schematic diagram of the transmitting subsystem, which is constructed of the MC and the RF transmitter.



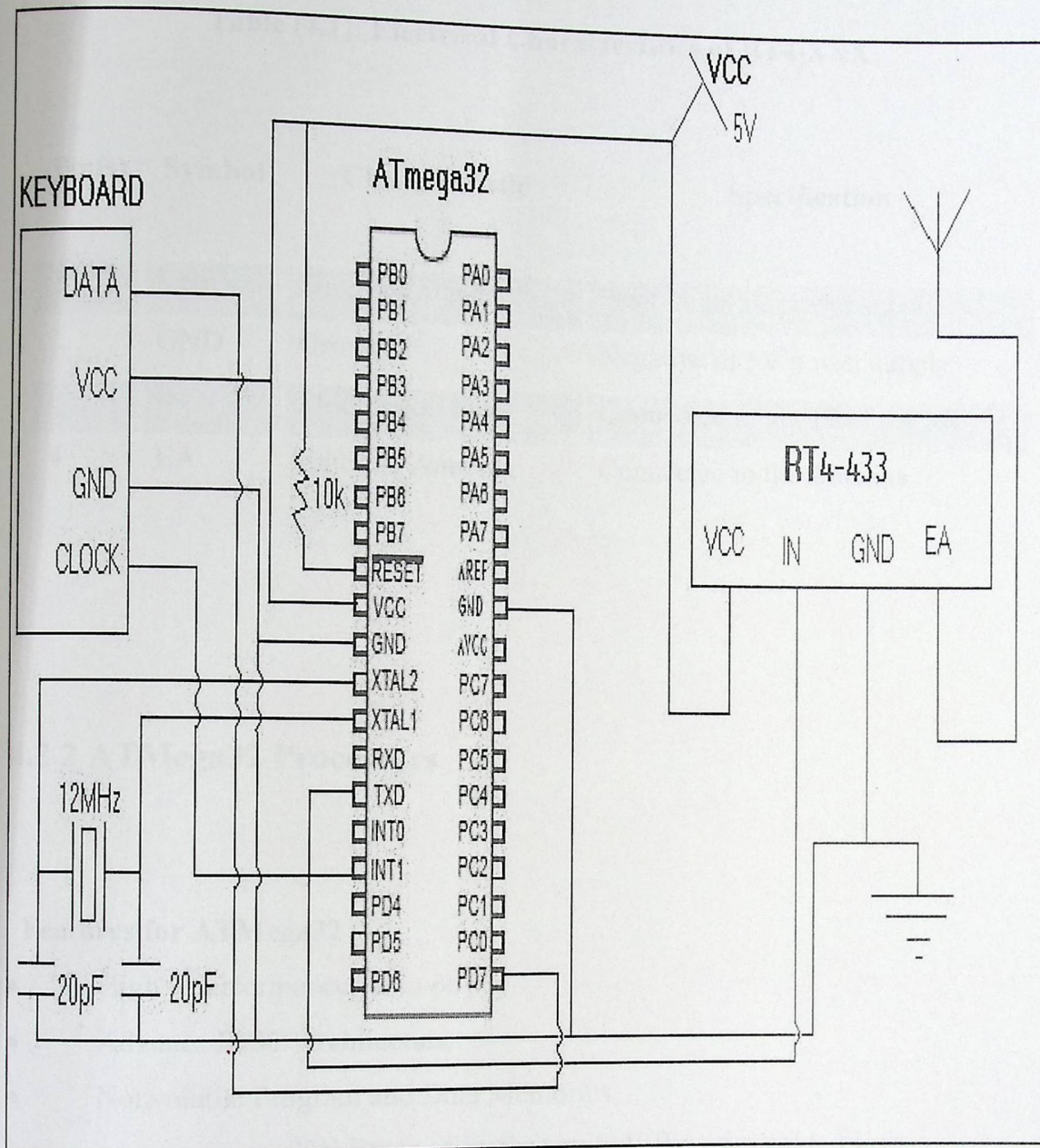


Figure (4.1): Schematic Diagram of Transmitting Unit

The pin connections of the transmitter according to our project requirements are listed in the Table (4.1).



**Table (4.1): Electrical Characteristics of RT4-XXX**

Pin(s)	Symbol	Characteristic	Specification
1	Vcc	Supply Voltage	Positive of 5V power supply
2	GND	Ground	Negative of 5V power supply
3	IN	Modulation Input	Connected to pin 17 of the MC
4	EA	External Antenna	Connected to the Antenna

## 4.2.2 ATmega32 Processors

### Features for ATmega32

- High – performance , low-power
- Advance RISC Architecture
- Nonvolatile Program and Data Memories
  - 1 - 32K Bytes of In-System Self-Programmable Flash
  - 2 - 1024 Bytes EEPROM Endurance: 100,000 Write/Erase Cycles
  - 3 - 2K Byte Internal SRAM
  - 4 - Programming Lock for Software Security



- Special Microcontroller Features

1 - External and Internal Interrupt Sources

2 - Internal Calibrated RC Oscillator

**Pin Descriptions:**

**VCC** : Digital supply voltage.

**GND** : Ground

**Port A (PA7..PA0):** Port A serves as the analog inputs to the A/D Converter. Port A also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used. Port pins can provide internal pull-up resistors (selected for each bit).

The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. When pins PA0 to PA7 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.



**Port B (PB7..PB0) :** Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability.

As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

**Port C (PC7..PC0):** Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability.

As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.

If the JTAG interface is enabled, the pull-up resistors on pins PC5(TDI), PC3(TMS) and PC2(TCK) will be activated even if a reset occurs.

**Port D (PD7..PD0):** Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability.

As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.



### 4.2.3 Receiving Subsystem:

Figure (4.2) presents the schematic diagram of the receiving subsystem, which is constructed of the decoder and the RF receiver.

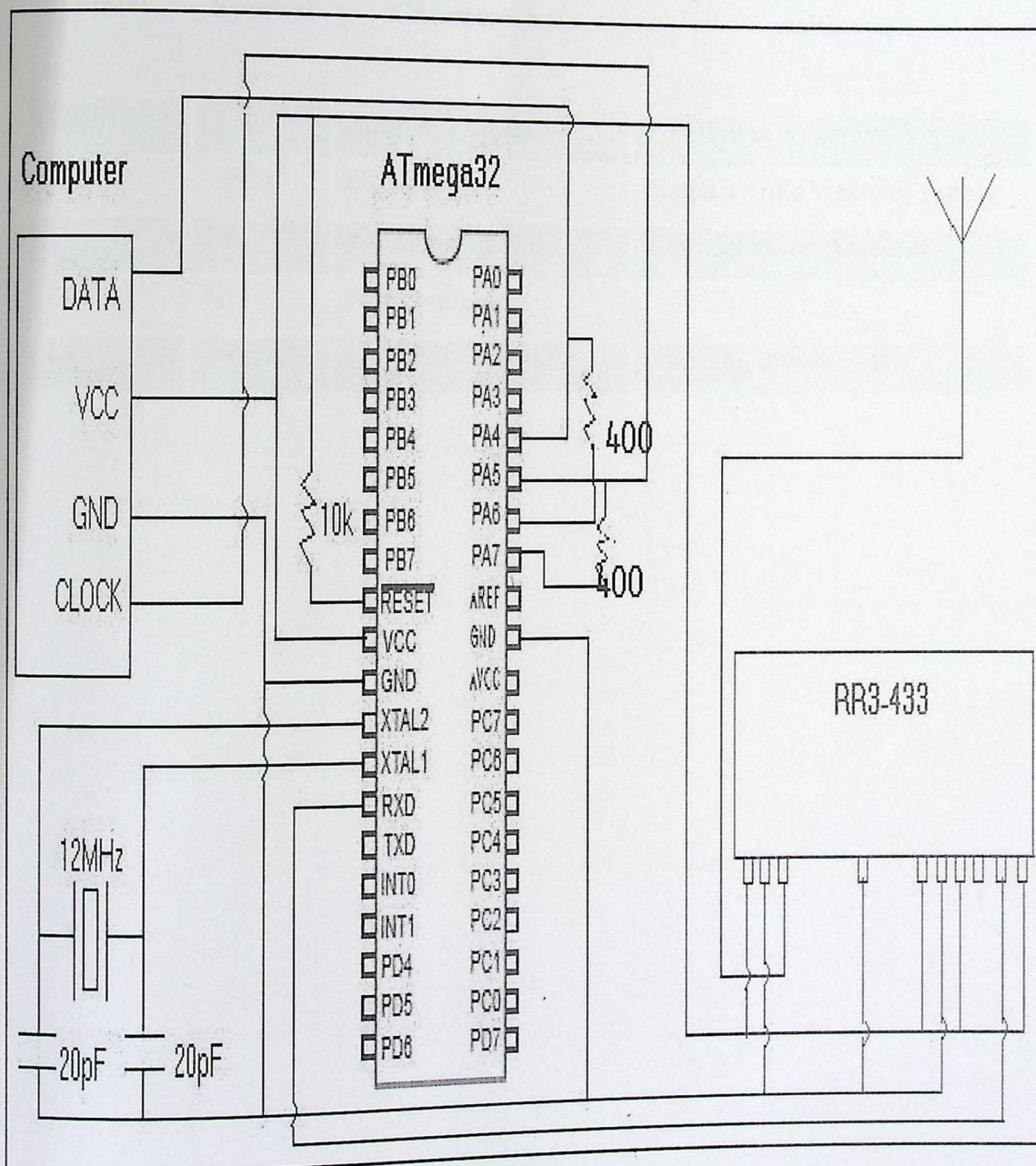


Figure (4.2): Schematic Diagram of Receiving Unit



The pin connections of the receiver according to our project requirements are shown in the Table (4.3).

**Table (4.3): Electrical Characteristics of RR3-XXX**

Pin(s)	Symbol	Characteristic	Specification
1,10,12,15	Vcc	Supply Voltage	Positive of 5V power supply
2,7,11	GND	Ground	Negative of 5V power supply
3	IN	Antenna Input	Connected the Antenna
13	NC	Not connected	.....
14	OUT	Output	Connected to the MC



## Summary

In this section we described the design details of our project as a whole the details for each component in the system and the two side of our project , computer side and keyboard side , the detail description of component and how it will work .

# Software System Design

- 1 Introduction
- 2 Software project
- 3 How Software System Work?
  - 3.1 Keyboard Side Software
  - 3.2 Computer Side Software
- 5.4 Code Listing



# 5

---

## *Software System Design*

### **5.1 Introduction**

### **5.2 Software project**

### **5.3 How Software System Work?**

#### **5.3.1 Keyboard Side Software**

#### **5.3.2 Computer Side Software**

### **5.4 Code Listing**



## 5.1 Introduction:

In this chapter we are going to describe the software system design and to describe some of the using methods and algorithm design as followed.

And it contains the flowchart for all processes in this project from first step in the project (initialize the ports & transmitting signal & receiving it ) to the final step (send signal to computer ).

## 5.2 Software Project

programmer was used to program the ATmega32 microcontroller , the program written in C language must be translated into hex value, where the microcontroller can understand it. CodeVisionAVR software is the software used to convert C code into hex code to be load on microcontroller. physically, program represent a file o the computer disk ( or in memory if it read in a microcontroller ), and is written according to the rule of the C language for microcontroller programming.

Translator interprets each instruction written in C language as a series of zero and ones which have meaning for the internal logic of the microcontroller. Figure (5.1) represent the basic phase of programming the ATmega32.



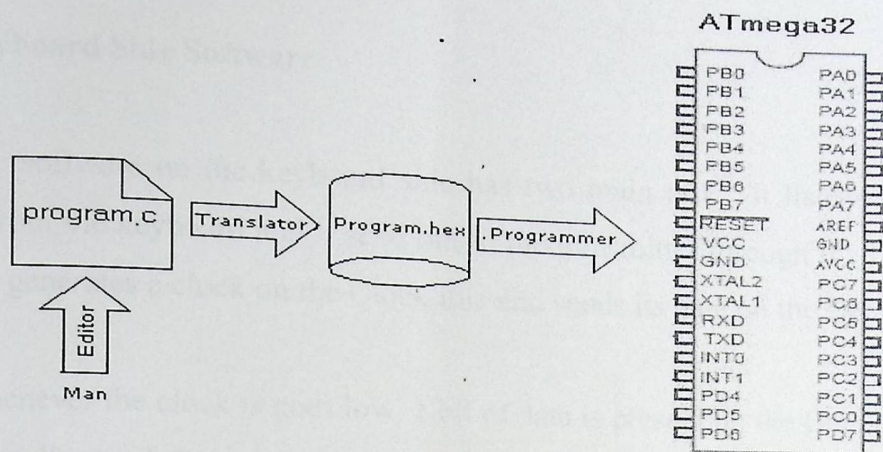


Figure (5.1): ATmega32 Programming Phases

Different programming languages access the port, and specify the operation microcontroller must perform. C language is a programming language with built in function , which is used for reading ,and writing to the port .

With C language, it is easy to work with microcontroller, we can read and write from port easily and we can use library of C language to perform the different logical and arithmetic operation. using C language is much better than assembly language since the size of code required to build the system is much less than the size of code in assembly language .

### 5.3 How System Software Works?

In this section, flow chart will be introduced to describe the program of both the computer side circuit, and the keyboard side circuit. flowcharting is a tool for analyzing processes .



### 5.3.1 keyboard Side Software

The software on the keyboard side has two main task : it listens to the data coming from the keyboard and it send out to the transmitter through the UART .The keyboard generates a clock on the Clock line and sends its data on the Data line.

Whenever the clock is goes low, 1 bit of data is present on the Data line and the microcontroller reads it.

Since the keyboard protocol sends a start bit, followed by 8 data bits, then a parity and a stop bit, we only shift in the data bits into a register. Once we have the 8 data bits, we place the byte into a queue.

To make sure that the system works correctly means it can reads the data bytes correctly, we will send tow byte that are 0xAA and then the byte 0xFF into the queue before each data byte.

The 0xFF byte indicates that the next byte is a data byte and should not be ignored. Furthermore, we maintain timer2 to keep track of how long it has been since we transmitted some data. If more than approximately 25mSec has elapsed, we transmit 0xAA after 0xFF.

Although the interrupt code appears quite long, there is only a piece of it that is executed every time the interrupt is taken. rest of the time, the program is inside the main while loop where we are constantly checking if the queue is empty and if the UART is free to send data.



The following flow chart , figure (5.3) , describe how the program should work to meet the hardware specification .

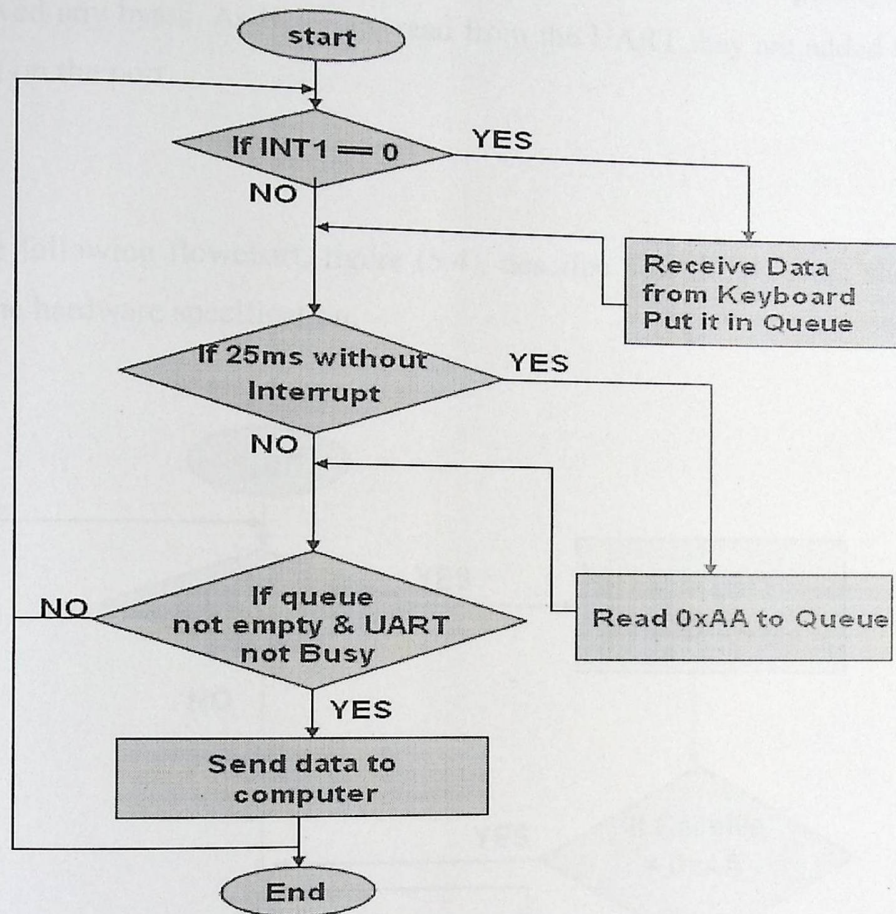


Figure (5.3) : flow chart Keyboard side

### 5.3.2 Computer Side Software

The software on the computer side has two main tasks. First, it listens for incoming data on the UART, determining which bytes correspond to keyboard code. Second, it echoes these codes to the computer.



There is an initial 0xAA. Then 0xFF is sent. This indicates that the next byte will be a valid data byte. Finally, a data byte is sent. Since the microcontroller uses the UART to receive data from the receiver, it polls the UART regularly to see if it has received any bytes. As bytes are read from the UART they are added to a queue to be sent on the port.

The following flowchart, figure (5.4), describe how the program should work to meet the hardware specification.

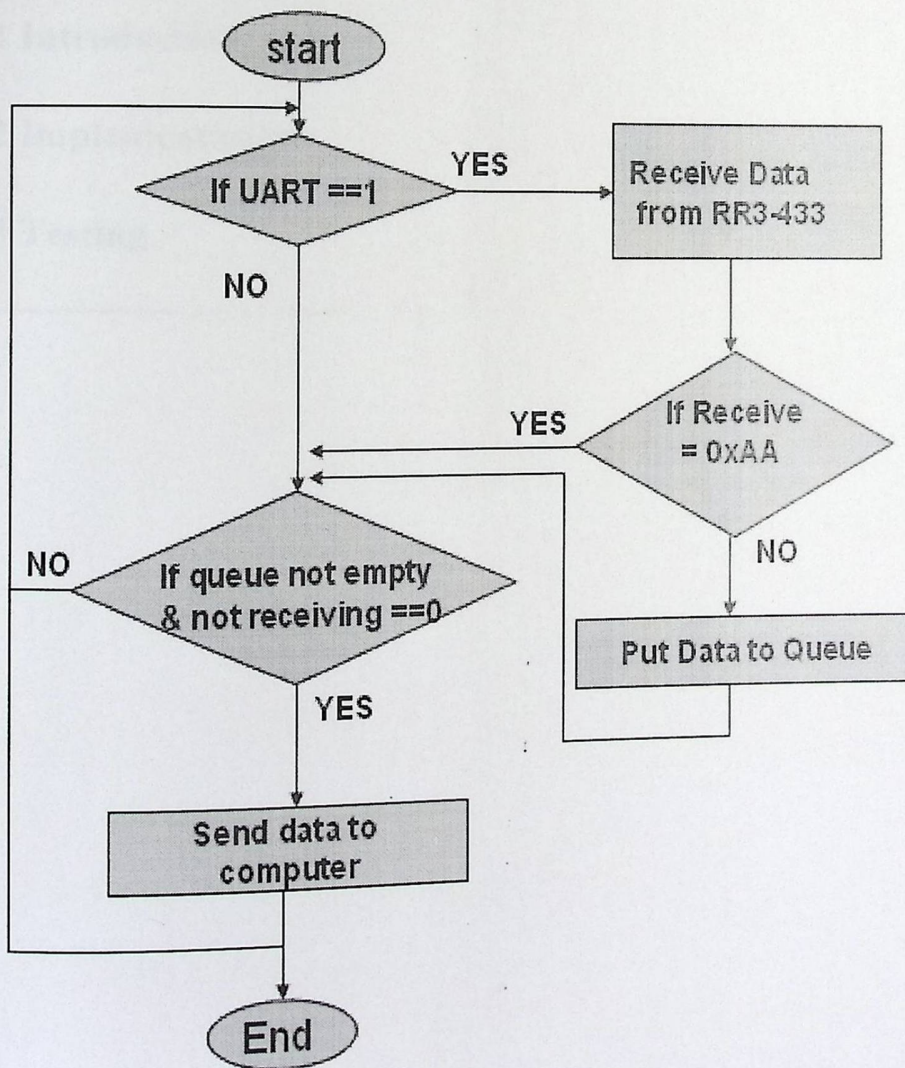


Figure (5.4): flowchart Computer side



# 6

---

## *Implementation And Testing*

**6.1 Introduction**

**6.2 Implementation**

**6.3 Testing**



## 6.1 Introduction

In this chapter we are going to show the implementation and testing processes for the system. The implementation and testing was done by using the following tools and components:

- IC stands.
- 10\*20 cm broad board.
- All the ICs that are depicted in the design chapter (see chapter 4).
- Wrapper tool for wrapping the connectors on the ICs stands.
- wire cutter.
- A digital Millimeter for testing.
- Oscilloscope .

This has more than one issue to be tested. some testing part reflect a software or hardware.

In addition, testing procedure concentrate on a single device independent from overall system.

## 6.2 Implementation

The implementation process is synchronized with the testing operation, since each implementation phase will take many testing steps to ensure that are no errors. The actual project implementation was a hardware prototype.



The first step for implementing the system is done through the simulator, where the simulator give us an accurate implementation since there is no possibility of hardware risk. after writing any program to be implemented, first, we compile it and then simulate them on the simulator, the simulator where the program written correctly give us an accurate result with no error.

The second step is implement the system on the bred board, implementation on the bred board is difficult some thing and not always give us the same result as on the simulator.

To obtain the same result there many thing to do, since microcontroller need some tutorial must be done, the first time we implement the program on bred board we go nervous and we don't know what happen, where is it work good on simulator, but after we made the tutorial it is become better.

### **6.3 Testing**

Here are the testing issue for the system. They are not ordered in any manner, rather they represent a way of system integrity and operation .there is many type of testing have to be explained for the system :



## 6.3.1 Hardware Testing

### 6.3.1.1 Block Testing

#### 6.3.1.1.1 Microcontroller testing

Microcontroller is one of the major component used in the system, there is many configuration must be established for the microcontroller to work successfully.

For ATmega32 microcontroller in our system, we test the ATmega32 by configuring the port C of ATmega32 as output, and assume that there is digital data over it. we specify any known value on the port C and connect parallel led's to the port, to ensure that the ATmega32 work successfully the value specified in the program must be appear on the led's.

Here is the program used to test the microcontroller, written by C language.

```
#include <Mega32.h>
void main(void)
{
    DDRC = 0xFF;           //set LEDs to all output
    PORTC=0x00;           //set all LED's to off
}
```



### 6.3.1.1.2 Transmitter and Receiver Testing

Here we test the transmitter and receiver which is one of major component of the system, transmitter send data from keyboard\_side circuit to the computer\_side circuit, while the receiver receive data from transmitter in the computer side circuit. system to be work successfully, transmitter must send data correctly and the receiver receive it in the same rate. so, before any step of building the system we must sure that the data transfer accurately and without noise.

The best way to test the transmitter and receiver is through the oscilloscope, we assume that serial data input to the transmitter to be transferred as analog signal, the receiver in the other side must accept the analog signal and convert it to digital data, the system will transmit 0x55 and receives 0x55 as shown in figure(6.1) the data input to the transmitter.

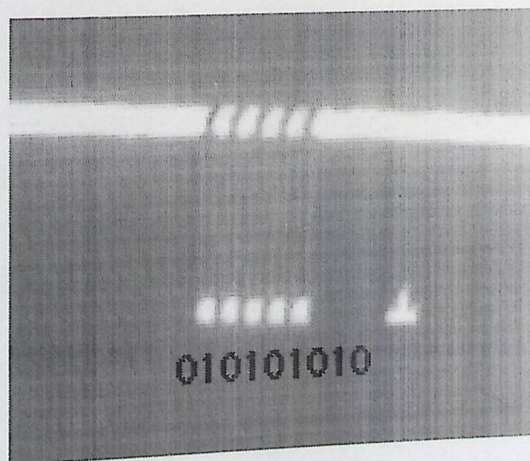


Figure (6.1) data input to the transmitter



figure (6.2) show the output data from receiver

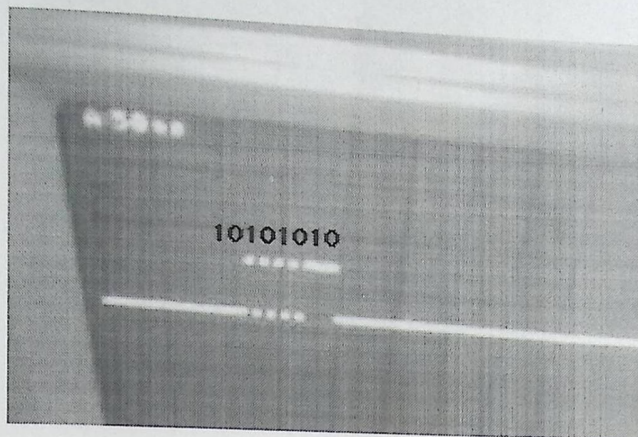


Figure (6.2) output data from receiver

we note that the data input and data output is the same , so the transmitter and receiver test is passes .

### 6.3.1.1.3 keyboard

keyboard that provide the system for digital data must tested here by oscilloscope, we want to ensure that the keyboard interrupts the microcontroller eleventh time when any key pressed, and eleventh bit transmitted serially from keyboard .

The figure (6.3) view the eleventh interrupt when any key pressed, the clock from keyboard is represent the interrupts.



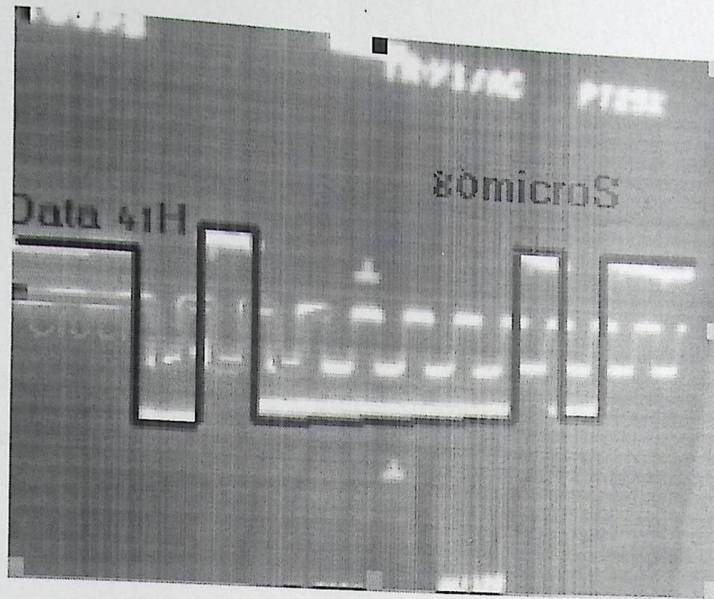


Figure (6.3) keyboard clock and data

### 6.3.1.2 Integration Testing

#### 6.3.1.2.1 Input Circuit Testing

The input circuit is the keyboard\_side circuit, where the data from keyboard processed and send out to the receiver, for the input circuit we need to ensure that the data is stored in the microcontroller and send out through the UART to the transmitter, the way used here to test the input circuit is, when a key (A) on the keyboard pressed the value (C1H) must stored in the query.

We build the program so that when a key (A) pressed, the serial data from keyboard to the port stored in a query, so we compare the value on the query with (C1H), if it's equal ,the led attached to the port must go on, other wise, is go off .



### 6.3.1.2.2 Output Circuit Testing

The output circuit is the computer\_side circuit, in this circuit the ATmega32 microcontroller must accept the data and process it to be transmitted to the computer, we build a program assuming that an value of character like (A) stored and must out through the port serially.

### 6.3.1.2.3 subsystem Testing

After testing the component of the system keyboard, microcontroller, transmitter and receiver . First , we connect the microcontroller with transmitter in the keyboard side circuit , and assume that there is a character stored in queue have to be transmitted to the transmitter through the UART.

Here we make sure that the data reach the transmitter through the UART , by connect the output pin from UART with led , we note the series of bits reach the transmitter when sequence light appear on the led . see figure (6.4) .





Figure (6.4) ATmega32 and transmitter testing

Second , we connect the microcontroller and the receiver in the computer circuit side , Here we want to ensure that the data come from transmitter is accepted by the receiver and the receiver convert it to sequence of digital data . output data from receiver connected to the led to ensure that receiver accept analog signal and convert it to digital data ,sequence of light also appear here .see figure (6.5) .

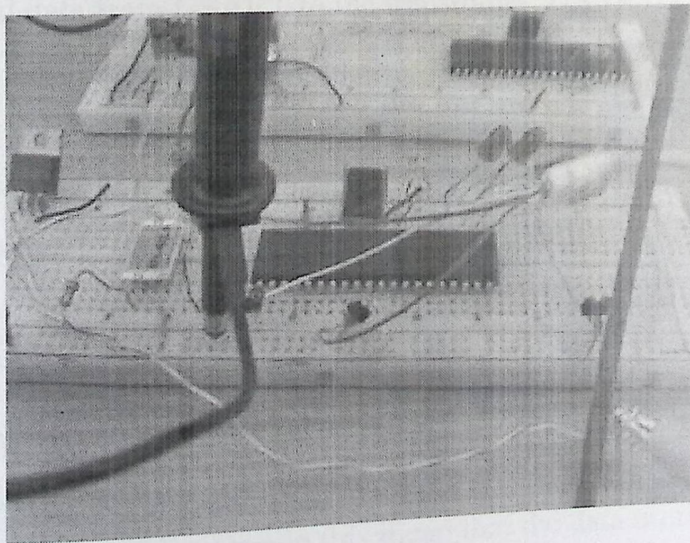


Figure (6.5) ATmega32 ,transmitter and receiver testing



### 6.3.2 Software Testing

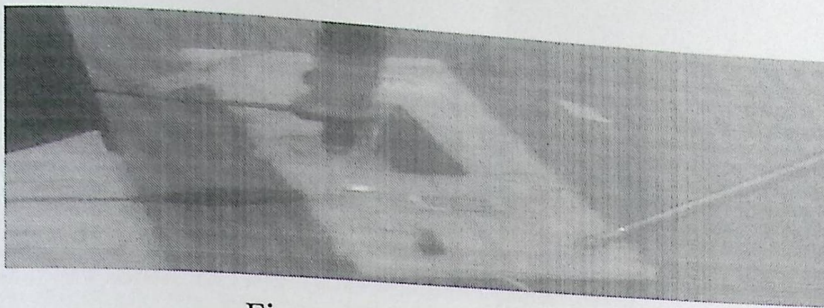
The goal of testing software is to establish confident that the software system is fit for purpose of the system, this mean that the system must be good enough for its intended use . The level of required confidence depend on the system purpose ,the expectation of the system user and the current marketing environment for the system .

Determining is the most problematic task , it took us long time and several trials working on it . Finally we managed to determine the time for delay , delay is must be very accurate since we read the serial data depending on the delay between each bit and the increasing or decreasing of the delay cause problem .

Next, it significantly important to test the USART . Therefore, a simple program was designed to read data from USART to the transmitter .USART and transmitter or receiver operate with different baud rate ,in communication with each other must operate with the same baud rate , but the baud rate of transmitter and receiver is fixed while the baud rate of the USART is dynamic and must be change by programmer to be the same with the transmitter and receiver baud rate.

Its simply to make it same but this not mean that the chip must be always work successfully , as shown in the figure (6.7) we assume that specific value transmitted by USART , when the led become on this mean that the data transmitted successfully and baud rate is the same in both side . because if baud rate is wrong no data will be transmitted .



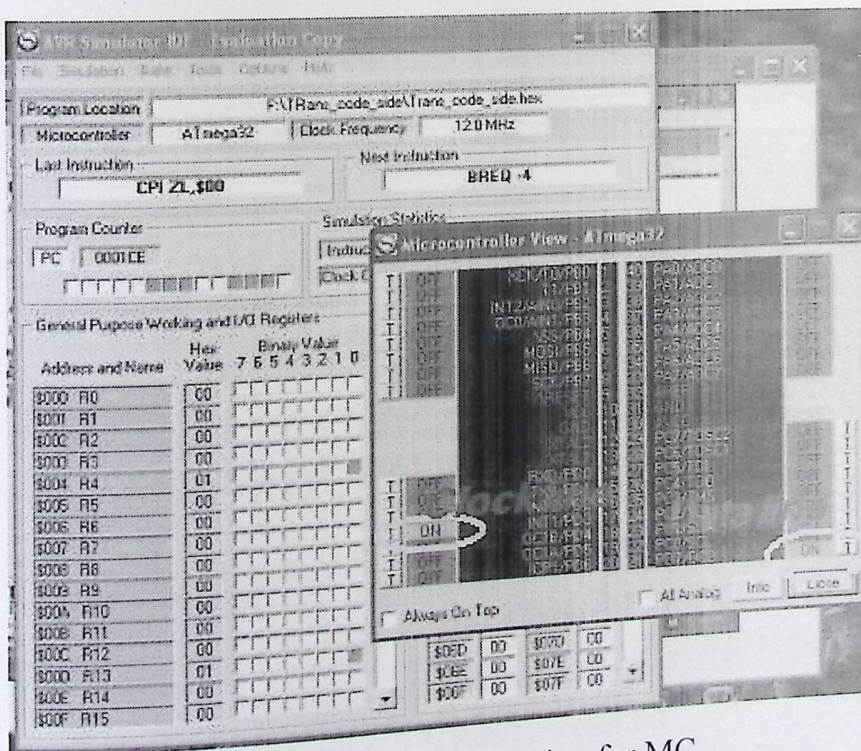


Figure( 6.7 ) baud rate test

in other hand the other software test of the part of the system like ports and rest circuit can simply made.

we tested all programs via AVRsimulatorIDE to determine all programs work correctly.

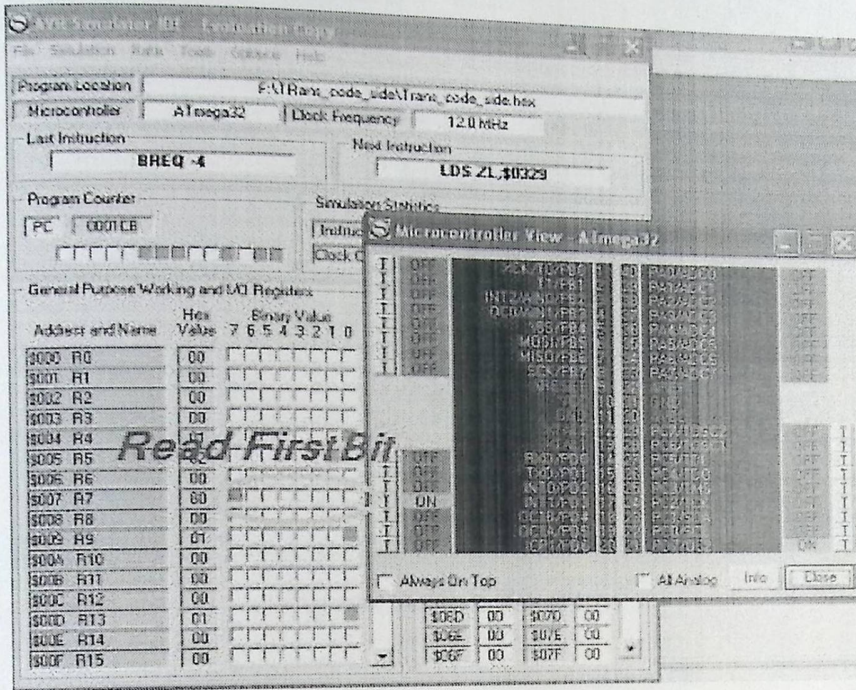
The following figure show how the MC receives data from the keyboard when start program the clock line and data line will be high as figure(6.7)



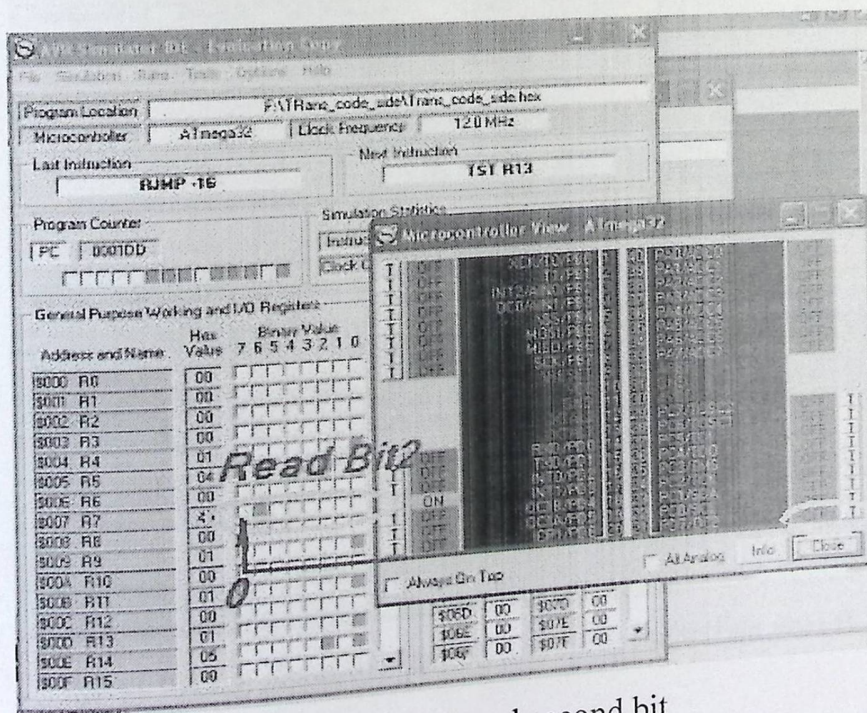
Figure( 6.7 ): first situation for MC



The following figure show what happen when clock line goes low the MC will read the data on data line (0 or 1) after that goes to high like the clock generate by keyboard.



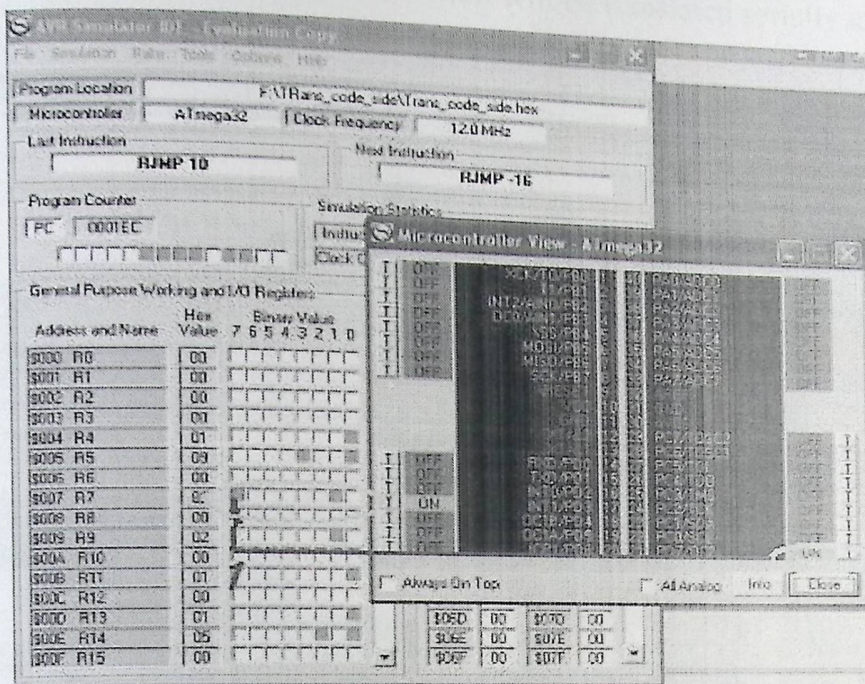
Figure( 6.8 ): Read first bit



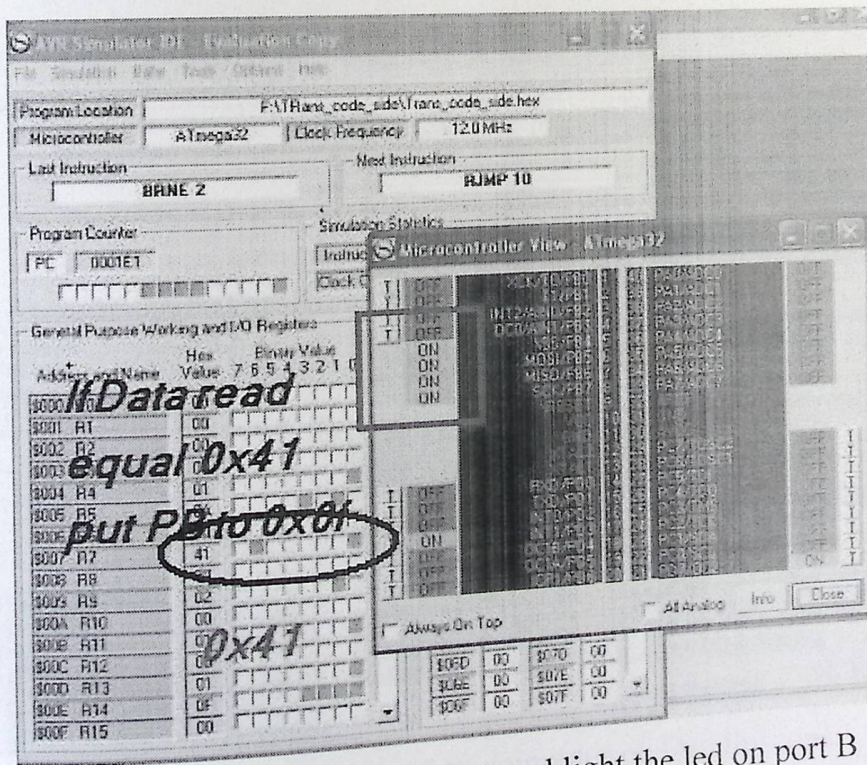
Figure( 6.9 ): Read second bit



The following show when the MC read the bit 7 .



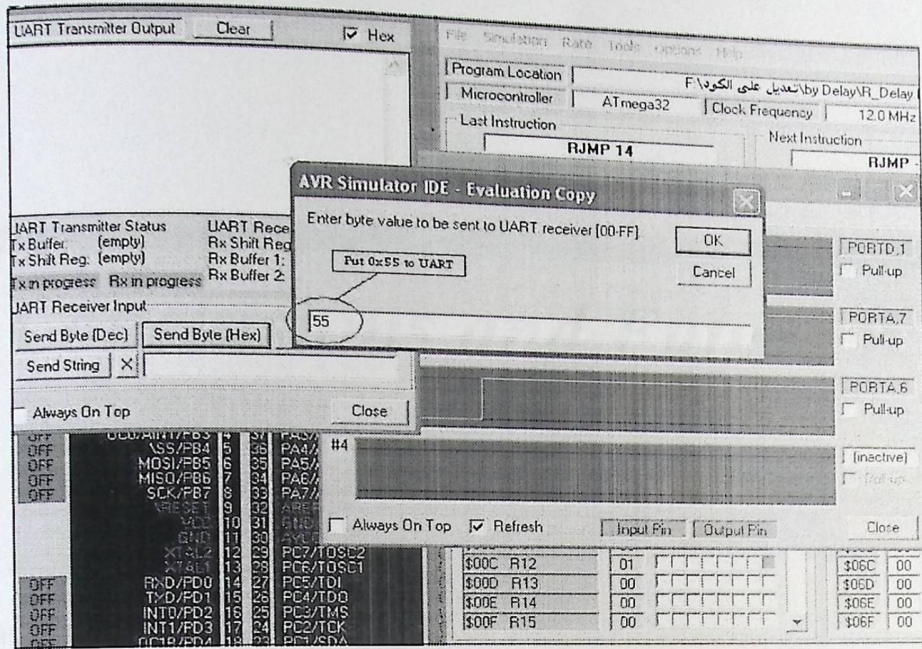
Figure( 6.10 ): Read bit7



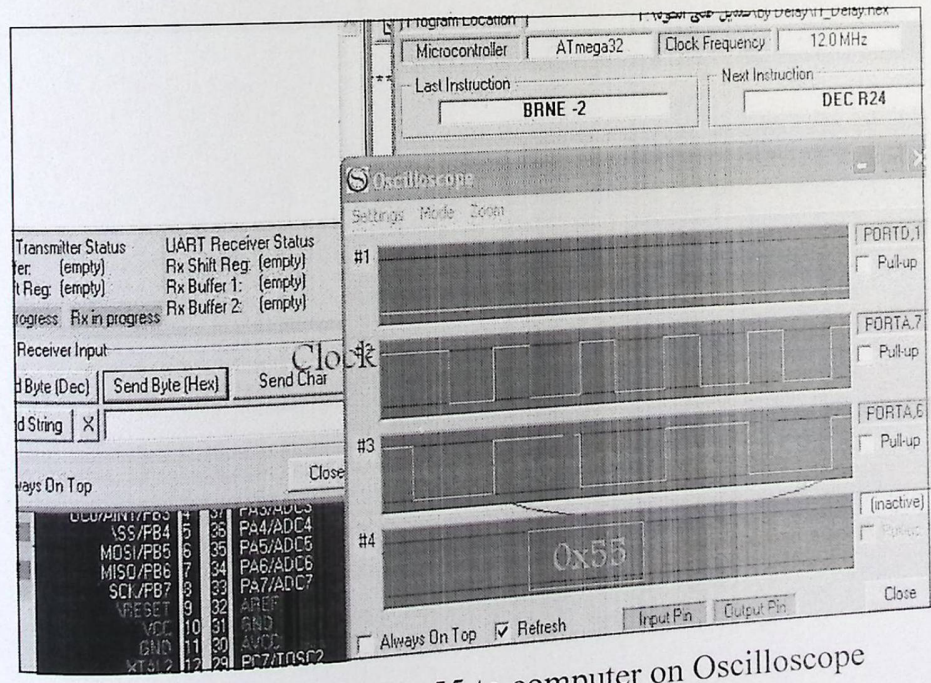
Figure( 6.11 ): Read the last bit and light the led on port B



The figure( 6.12 ) assume that the computer side receive 0x55 hex by the UART and figure ( 6.13 ) show how the 0x55 will be translated serially to computer.



Figure( 6.12 ):Receive 0x55 by the UART



Figure( 6.13 ): Send 0x55 to computer on Oscilloscope



# 7

---

## *Conclusions and Future Works*

**7.1 Conclusions**

**7.2 Problems**

**7.3 Future Work:**



## 7.1 Conclusions

Many experiences were added to the team cognitive knowledge through this project. Many conclusions can be stated here, but only significant and important ones are described:

- Each subsystem was implemented in its own circuit and tested by means of Hardware and Software. At the next testing stage two or more subsystems were combined together to check the influence of their outputs on each other. to check its work and test the complete system program on it.
- Building Wireless Keyboard was a great way of combining all we have learned over the five years at PPU. We were able to design, build and implement the project. We learned how to deal with serial communication.
- Starting the work in any project is hard ,so, the beginning must be in more accurately and carefully .
- There is a big difference between the theories and the real world implementation.
- This project proved that our ideas were viable for implementation in a project that resembles a real-life problem.



- Each device was tested individually in small circuit( with certain purpose) to study its behavior and make sure it works properly and can do its expected job.

## 7.2 Problems

System completion in regard to its objectives is an implementation dependent issue. Problems are natural things. Encountering problems and limitations is very common in such a huge project specially when dealing with electronic devices.

Here are problems faced the project team during the system implementation:

- Choosing the appropriate chip requires long time , while we study and plan the project many thing may change , some feature of programming may found in some kind of chips .
- When we started our work on the project, we had built prototypes for all circuits using old breadboards from the university lab, which led into damage for some integrated chips.
- Internal damage in some devices due to the wrong connections, or high voltages, or currents supplied to the devices during the implementation.



- Time and effort were wasted because of the bad choices we have made, such as:
  - ✓ In the first semester we decided to use ATmega32, but its unavailable in Market. Then we decided to use PIC18F452 .
  - ✓ During the test this chip was damaged, and then the university brings ATmega32 then we decided to return to it.

### 7.3 Future Work:

We tried our best to choose the rational design to achieve the objectives of our project. We also believe that any work can not reach the perfection. Still a lot of thought and ideas can be utilized to enhance the current work achieved. Here are some of them:

- In this system we have used keyboard from the market in the future we can build a special design for a keyboard.
- Providing a Mouse beside the keyboard so the controlling on the computer will be completed by using the same chip receiver and transmitter.



## References

### Books:

Author Name	Book Name	Publisher	Year
Handbook of RF and wireless technology	dowla_farid	Lawrence Livermore National Laboratory	2004
Modern wireless technology	hagkin_sireon	Prentice Hall	2004
Wireless communication system	wany_xiaodony	Prentice Hall	2001
Wireless communication principle and practice	rappaport_theodare	Prentice Hall	2003
C باستخدام لغه atmel avr برمجہ	شد -محمد عبد المعطي	شعاع للنشر والعلوم	2005
avr متحكمات	عدنان_عمار	شعاع للنشر والعلوم	2005



**Web Site:**

- 1- [http://en.wikipedia.org/wiki/Computer\\_keyboard](http://en.wikipedia.org/wiki/Computer_keyboard)
- 2- [http://www.osdev.org/wiki/Keyboard\\_Input](http://www.osdev.org/wiki/Keyboard_Input)
- 3- <http://computer.howstuffworks.com/keyboard.htm>
- 4- [http://www.atmel.com/dyn/resources/prod\\_documents/doc2503.pdf](http://www.atmel.com/dyn/resources/prod_documents/doc2503.pdf)
- 5- <http://www.telecontrolli.com/pdf/receiver/rr3.pdf>
- 6- <http://www.telecontrolli.com/pdf/transmitter/rt4.pdf>
- 7- <http://www.astro.virginia.edu/class/skrutskie/astr174/a174intro.s07.pdf>
- 8- [http://en.wikipedia.org/wiki/Radio\\_frequency](http://en.wikipedia.org/wiki/Radio_frequency)
- 9- <http://electronics.howstuffworks.com/radio>
- 10- [http://en.wikipedia.org/w/index.php?title=Radio\\_frequency&action=history](http://en.wikipedia.org/w/index.php?title=Radio_frequency&action=history)



# Appendices

**Appendix A: Datasheets of Project Components**

**Appendix B: Source Code**



# Appendix A

## Datasheets of Project Components



# Appendix B

## Source Code

Computer\_side\_code

Keyboard\_side\_code



## Computer\_side\_code

```
#include <Mega32.h>

#include <delay.h>

#define CLK_TIME 180

#define RESPOND_WAIT 528

//define ps/2 data line states

#define IDLE 0

#define INHIBIT 1

#define BUSY 2

#define REQUEST 3

//define some useful maps to bits

#define UART_IN UCSRA.7

#define CLK_OUT PORTA.7

#define DATA_OUT PORTA.6

#define CLK_IN PINA.5

#define DATA_IN PINA.4

//state of the data line

char transmitting, receiving, waiting;

//variables for writing to PS/2 port

char position, PS2byte, PS2parity;

char clockState;
```



```
//define clock states

#define HIGH 0

#define FALLING 1

#define LOW 2

#define RISING 3

//define queue values

#define QUEUELEN 100

#define TRUE 1

#define FALSE 0

char queue[QUEUELEN]; //queue of data sent by keyboard. (first in, first out)

char queueFull; //indicates if queue is full

char queueEmpty; //indicates if queue is empty

char queueIn; //indicates where to put data into queue

char queueOut; //indicates where to take data out of queue

char queuePut(char d); //put data into queue

char queueGet(void); //get data from queue

void send(void); /// send data to computer

//0 = waiting

//1 = received AA last

//2 = received FF last next in is good

// char preamble_state;

//break code was received
```



```

char break_code;

char parity(char x)//calculate the parity of a character
{ char temp, i;

temp=1;

for(i=0;i<8;i++)
{
temp=temp^(x&1);

x>>=1;

}

return temp;

}

//insert data into queue. Return 1 if queue full, or 0 if inserted data sucessfully
char queuePut(char d)
{ if (queueFull==TRUE)//check if queue is full

return(TRUE);

queue[queueIn]=d;//insert d into queue

queueIn++;//increment where to stick in the next d value

queueEmpty=FALSE;//indicate queue isnt' empty anymore

if (queueIn==QUEUELEN) //if reached the end of the queue

queueIn=0; //wrap around to the beginning

if (queueIn==queueOut) //if queueIn caught up to queueOut

queueFull=TRUE; //indicate queue is full

```



```

        return(0);
    }

//get data out of queue. Return 0 if queue empty or the actual data if not empty
char queueGet(void)
{ char d;

    if (queueEmpty==TRUE) //check if queue is empty
        return(0);

    d=queue[queueOut]; //get data out of queue
    queueOut++; //increment location where to get next d value
    queueFull=FALSE; //indicate queue isn't full anymore
    if (queueOut==QUEUELEN) //if reached the end of the queue
        queueOut=0; //wrap around to the beginning
    if (queueOut==queueIn) //if queueOut caught up to queueIn
        queueEmpty=TRUE; //indicate queue is empty

    return(d); //return the data from queue
}

//determine the state of the data line
char clockStateNow(void)
{
    /*
    if(transmitting || receiving || waiting) //I'm doing something

    return(BUSY);

```



```

if(CLK_IN==0)//computer is inhibiting communication
return(INHIBIT);

if(DATA_IN==0)//computer wants to send something
return(REQUEST);

    */

return(IDLE);//nothing happening
}

void initChip(void)
{
OCR1A = CLK_TIME; //compare match to drive the clock to PC
OCR1B = RESPOND_WAIT;

TCCR1B = 0x01; //Run counter at full speed
TCCR1A = 0x00; //nothing needed here, no output

TIMSK = 0x18; //Set the compare A interrupt and compare B interrupt

queueFull=FALSE;//queue is not full

queueEmpty=TRUE;//queue is empty

queueIn=0; //where to insert into queue

queueOut=0; //where to take out of queue

PORTA = 0xC0; //initialize outputs to high

DDRD = 0x00;

DDRB = 0x00;

DDRA = 0xC0; //set data direction (see initial wiring comments)

DDRC = 0xFF; //set LEDs to all output

```



```

PORTC=0x00;    //siet all LED's to off
UCSRB = 0x18 ; //enable only the receiver, no interrupts
UCSRC=0xB6; //odd parity, 8-bit data segments
UBRRH = 0x01;
UBRRL = 0x76; //need to BR 2000

#asm
    sei
#endasm
}

void send(void)
{
    if(transmitting)
    {
        int i =1;
        if(clockState==HIGH) //write data out here
        {
            // start bit
            DATA_OUT=0;
            delay_us(3); // wait 40 mics
            CLK_OUT=0;
            for(;i<9;i++)//data bits
            { delay_us(3); // wait 40 mics
              DATA_OUT=PS2byte&0x01;
            }
        }
    }
}

```



```

PS2byte = PS2byte>>1;

CLK_OUT=1;

delay_us(3);

CLK_OUT=0;

DDRB = 0xFF;

PORTB = 0xAA;

}

//***** Parity Bit *****

delay_us(3); // wait 40 mics

DATA_OUT=PS2parity;

CLK_OUT=1;

delay_us(3); // wait 40 mics

CLK_OUT=0;

delay_us(3); // wait 40 mics

//***** Stop Bit *****

DATA_OUT=1;

CLK_OUT=1;

delay_us(3); // wait 40 mics

CLK_OUT=0;

delay_us(3); // wait 40 mics

CLK_OUT=1;

DDRC = 0xFF;

PORTC=0x7f;

```



```

// transmitting=0;
}
else
{
// CLK_OUT=1;
// DATA_OUT=1;
// clockState=HIGH;
}
}
}

void main(void)
{
char inFromRF;
char myState;
initChip();//initialize chip
while(1)
{
if(UART_IN)
{ //f1
//handle uart, make sure you receive the preamble before registering a symbol
inFromRF=UDR;
if(!UCSRA.2)
{ //f2

```



```

    if(inFromRF==0xAA);
    else
    {
        queuePut(inFromRF);
    }
    } //ef2
} //ef1

myState=clockStateNow();//check if the line is idle
/* if(myState==REQUEST)
{
    DDRC = 0xFF;
    PORTC=0x00;
    receiving=1;
    position=0;
}*/

if( (myState==IDLE) && !queueEmpty )//is it idle and is there something to
// send

{ //1
    PS2byte = queueGet();
    if(PS2byte!=0)
    { //2
        transmitting=1;
        position=0;
    }
}

```



```
clockState=HIGH;
PS2parity = parity(PS2byte);
send();
} //e2
} //e1
} // end_while
} //e_main
```



## Keyboard\_side\_code:

```
#include <Mega32.h>

#define rDAT PIND.7

#define rCLK PIND.3

#define wDAT PORTD.7

#define wCLK PORTD.3

#define TRUE 1

#define FALSE 0

define QUEUELEN 100

char receive; //indicate that int2 interrupt should work as reciever (if 1) or
              //tranmsitter (if 0)

char countIn; //count how many Low CLK pulses we have seen as we receive
              //data from keyboard

char countOut; //count how many Low CLK pulses we have seen as we transmit
              //data to keyboard

unsigned char dataIn; //will store the data bits coming in from the keyboard

unsigned char dataOut; //will store the data bits we are sending out to the keyboard

char parityIn; //will store the calculated parity of data coming in from the keyboard

char parityOut; //will store the calculated parity of data we are sending out to the
keyboard

char pError; //will store if there was a parity error in data sent by keyboard

char queue[QUEUELEN]; //queue of data sent by keyboard. (first in, first out)
```



```

char queueFull; //indicates if queue is full
char queueEmpty;//indicates if queue is empty
char queueIn; //indicates where to put data into queue
char queueOut; //indicates where to take data out of queue
char rxDone; //indicate done receiving frame
char txDone; //indicate done transmitting frame
char t2visits; //times gone into timer2 interrupt without any resets of the counter

void initialize(void); //initialize mcu

void initializeKyBd(void);//initialize keyboard

char queuePut(char d); //put data into queue

char queueGet(void); //get data from queue

void TXtoKyBd(char d); //send command to keyboard

void RXfromKyBd(void); //receive data from keyboard

//insert data into queue. Return 1 if queue full, or 0 if inserted data sucessfully

char queuePut(char d)
{ if (queueFull==TRUE)//check if queue is full

    return(TRUE);

    queue[queueIn]=d; //insert d into queue

```



```

queueIn++; //increment where to stick in the next d value
queueEmpty=FALSE; //indicate queue isnt' empty anymore
if (queueIn==QUEUELEN) //if reached the end of the queue
    queueIn=0; //wrap around to the beginning
if (queueIn==queueOut) //if queueIn caught up to queueOut
    queueFull=TRUE; //indicate queue is full
return(0);
}

```

//get data out of queue. Return 0 if queue empty or the actual data if not empty

```
char queueGet(void)
```

```
{ char d;
```

```
if (queueEmpty==TRUE) //check if queue is empty
```

```
return(0);
```

```
d=queue[queueOut]; //get data out of queue
```

```
queueOut++; //increment location where to get next d value
```

```
queueFull=FALSE; //indicate queue isn't full anymore
```

```
if (queueOut==QUEUELEN) //if reached the end of the queue
```



```

queueOut=0; //wrap around to the beginning
if (queueOut==queueIn) //if queueOut caught up to queueIn
    queueEmpty=TRUE;//indicate queue is empty
return(d); //return the data from queue
} //on every falling clock edge generated by the keyboard, either receives or sends
//data

interrupt [EXT_INT1] void external_int1(void)
{ //when receiving data
    if(receive==TRUE)
    {
        countIn++; //count how many clock pulses we have seen
        //if countIn=1, then seeing the start bit
        //if countIn=2 to 9, then seeing data bits 0 to 7
        //if countIn=10, then seeing the parity bit
        //if countIn=11, then seeing the stop bit
        if (countIn==1)//seeing start bit
        {
            dataIn=0;
            rxDone=FALSE; //receiving not done (just starting)
            //set up timer0 to check for no clock transition in 200uSec
            TCNT0=0; //reset timer0

```



```

TCCR0=0x0B;//put timer0 into compare match mode, prescaler of 64;
TIMSK=(TIMSK&0xFD)|0x02; //make bit1=1 to enable compare match
//interrupt;
TIFR=TIFR&0xFD; //make bit1=0 to clear the timer0 comp match flag
}
else if ((countIn>1) && (countIn<10)) //data bits being sent
{ TCNT0=0; //reset timer0 since saw a clock signal

TIFR=TIFR&0xFD;//make bit1=0 to clear the timer0 comp match
dataIn=dataIn>>1;//shift data right by 1

if (rDAT==1) //if DAT line is 1

{dataIn=dataIn|0x80; /shift in a received bit of 1
parityIn++; /update the parity

}

}

else if (countIn==10) //parity bit being sent

{

TCNT0=0; //reset timer0 since saw a clock signal

TIFR=TIFR&0xFD; //set bit1=0 to clear the timer0 comp match

//partiy should be odd

if ((parityIn+rDAT)&0x01) //if (calculated parity+ rDAT) is odd

```



```

pError=FALSE; //then no parity error
else
    pError=TRUE; //otherwise there was a parity error
    }
else if (countIn==11) //stop bit being sent
    {
        if(dataIn == 0xff)
            {
                PORTC=0xff;
                DDRC=0x07;
            }
        TCCR0=0x00; //stop timer0 since seen last clock
        TCNT0=0; //reset timer0
        TIMSK=TIMSK&0xFD; //set bit1=0 to disable timer0 interrupt
        TIFR=TIFR&0xFD; //set bit1=0 to clear the timer0 comp match
        if(!pError && (dataIn!=0xAA) && (dataIn!=0xFA) && (dataIn!=0xFC))
            {
                queuePut(dataIn); //insert data into queue
            }
        rxDone=TRUE; //indicate that we have received a frame

```



```

countIn=0; //reset count since have seen the end of frame
parityIn=0; //clear parity for next time
}
}
else {
countOut++; //count how many clock pulses we have seen
//if countOut=1, then seeing ourselves pulling line low to inhibit communication
//if countOut=2 to 9 then we will be sending data bits 0 to 7
//if countOut=10, then we will be sending out the parity bit
//if countOut=11, then we will be sending out the stop bit
//if countOut=12, then seeing the "ack" bit
if (countOut==1) //seeing ourselves pulling the line low to inhibit communication
{
//set up timer so that we keep CLK low for at least 100uSec
TCCR1A=0x00; //normal waveform generation
TCCR1B=0x01; //normal waveform generation and full clock speed (12MHz)

TCNT1=0; //reset timer1 to 0

```



```

OCR1A=1680; //140uSec*12MHz=1680cycles
TIMSK=(TIMSK&0xEF)|0x10; //set bit4=1 to 1 to enable Output Compare A
//match interrupt
//set up timer0 to check for no clock transition in 200uSec
TCNT0=0; //reset timer0
TCCR0=0x0B; //put timer0 into compare match mode, prescaler of 64;
TIMSK=(TIMSK&0xFD)|0x02; //set bit1=1 to enable compare match interrupt;
TIFR=TIFR&0xFD; //set bit1=0 to clear the timer0 comp match flag
parityOut=0; //reset parity
txDone=FALSE; //transmitting not done(just starting)
}
else if (countOut>1 &&countOut<10) //data bits being sent
{
TCNT0=0; //reset timer0 since saw a clock signal
TIFR=TIFR&0xFD; //set bit1=0 to clear the timer0 comp match flag
if (dataOut&0x01)//if lowest bit is a 1
{ wDAT=1; //write out a 1 to the keyboard
parityOut++; //calculate the parity
}
}

```



else

```
wDAT=0; //write out a 0 to the keyboard
```

```
dataOut=dataOut>>1;//shift data right by 1 to get next bit next time
```

```
}
```

```
else if (countOut==10)//send parity bit
```

```
{
```

```
TCNT0=0; //reset timer0 since saw a clock signal
```

```
TIFR=TIFR&0xFD; //set bit1=0 to clear the timer0 comp match flag
```

```
if(parityOut&0x01) //if parity is odd
```

```
wDAT=0;//write out a 0 to the keyboard
```

else

```
wDAT=1; //write out a 1 to the keyboard to make it odd parity
```

```
}
```

```
else if (countOut==11)//sending out the stop bit
```

```
{
```

```
TCNT0=0; //reset timer0 since saw a clock signal
```



```

TIFR=TIFR&0xFD; //set bit1=0 to clear the timer0 comp match
wDAT=1;//write the stop bit (when change DDR, this will also activate the pullups)
DDRD=0x00; //make DAT line input, CLK input already. Do this so kybd can send
        // ack bit
    }

else if (countOut==12)//getting the ack bit of 0 on DAT
{
    TCCR0=0x00;//turn off timer0 since seen last clock

    TIMSK=TIMSK&0xFD; //set bit1=0 to turn of timer0 comp match interrupt
    TIFR=TIFR&0xFD;      //set bit1=0 to clear timer0 comp match flag

    txDone=TRUE;//transmitting done

    countOut=0; //reset countOut for next time

    parityOut=0;//reset parity for next time
}
}

GIFR=0;

}

//used to make CLK low for at least 100uSec so can send stuff to keyboard

interrupt [TIM1_COMPA] timer1_compA(void)

```



```

{
    TIMSK=TIMSK&0xEF; //set bit4=0 to turn off interrupt
    TCCR1B=0x00; //turn off timer (saves power)
    DDRD=0x80; //make CLK an input, keep DAT as output
    wCLK=1; //write 1 to B.2 so pullup turn on
    wDAT=0; //write 0 to DAT, this is the start bit...kybd will start generating a
            // clock any moment now
}

```

//if this executes, then it has been at least 200uSec between clk pulses while receiving or transmitting data to keyboard

```

interrupt [TIM0_COMP] timer0_comp(void)

```

```

{
    TIMSK=TIMSK&0xFD; //set bit1=0 to turn off timer0 comp match interrupt
    TCCR0=0x00; //turn off timer0
    TIFR=TIFR&0xFD; //set bit1=0 to clear timer0 comp match flag (just in
                    //case its set)
    countIn=0; //reset counters
    countOut=0;
    parityIn=0; //reset parity calculations
    parityOut=0;
    txDone=TRUE; //there was some error, but pretend you are done
}

```



```

rxDone=TRUE;

receive=TRUE; //put system into receive mode (just a default option)

wDAT=1; //turn on pull ups

wCLK=1;

DDRD=0x00; //set DAT and CLK to input
}

//used to make sure we send something approximately every 25msec
interrupt [TIM2_OVF] timer2_overflow(void)
{
    t2visits++;

    //after visit this interrupt without resetting 6 times, then about 25msec

    //elapsed since last time we sent something

    if (t2visits==15)
    {
        queuePut(0xAA);

        t2visits=0; //reset number of visits
    }
}

//initialize the mcu

```



```

void initialize(void){
    DDRD=0x00; //make PORTD pin D.3 and D.7 input for CLK and DAT
    PORTD=0x88; //write 1 to B.7 and B.3 so pullups on for wCLK and wDAT
    MCUCR=0x08; //int1 triggered on falling edge
    GICR=0x80; //int1 enable

    //Timer0 set up. The rest will be done inside the external interrupt
    OCR0=50; //Timer0 Outoput compare when
    //50 (decimal)....1/(16Mhz/64)*50= 200u Sec.

    //Timer2 set up.
    TCCR2=0x06; //prescal by 256
    TIMSK=(TIMSK&0xBF)|0x40; //set bit6=1 to enable overflow interrupt

    //UART
    UCSRB=0x08; //enable transmitter only
    UCSRC=0xB6; //Asynchronous mode, 8bit words, 1 stop bit ,odd parity
    UBRRH =0x01;
    UBRL =0x76 ; //need to change this 374 BR 2000

    DDRC=0xff; //for testing, set portc to output
    PORTC=0xff; //for testing,set LEDs off
    DDRA=0xff; //for testing set porta.0 to output

```



```
PORTA=0x00; //for testing,set to 0
receive=TRUE; //put int1 interupt into recieve mode
countIn=0; //no low clock pulses seen yet
countOut=0;

parityIn=0; //calculated parity at 0
parityOut=0;

//set up queue
queueFull=FALSE;//queue is not full
queueEmpty=TRUE;//queue is empty
queueIn=0; //where to insert into queue
queueOut=0; //where to take out of queue
rxDone=FALSE; //haven't finished receiving
txDone=FALSE; //haven't finished transmitting

    #asm
        sei

    #endasm
}

void initializeKyBd(void)
```



```

{
    dataIn=0;

    countIn=0;

    RXfromKyBd(); //put into receive mode

    while(!rxDone); //wait until get 0xAA

    TXtoKyBd(0xFF); //put into transmit mode, tell kybd to reset itself

    while(!txDone); //wait until transmitting done

    RXfromKyBd(); //put into receive mode

    while(!rxDone); //wait until get 0xFA (ack) from kybd that it got the command

    RXfromKyBd(); //put into receive mode

    while(!rxDone); //wait until get 0xAA (reset successful) from kybd

    TXtoKyBd(0xF4); //put into transmit mode, tell kybd to enable all keys

    while(!txDone); //wait until transmitting done

    RXfromKyBd(); //put into receive mode

    while(!rxDone); //wait until get 0xFA (ack) from kybd that it got the command

}

//set interrupt to be in receive mode

void RXfromKyBd(void)

{
    receive=TRUE; //set int2 to receive mode

```



```

    rxDone=FALSE; //reset

    wDAT=1; //pull ups on
    wCLK=1; //pull ups on
    DDRD=0x00; //set DAT and CLK to input
    countIn=0;
}

//sets interrupt to be in transmit mode
void TXtoKyBd(char d)
{
    receive=FALSE; //set int2 into transmit mode
    txDone=FALSE; //reset
    dataOut=d; //the data to transmit
    DDRD=0x88; //set DAT and CLK to output
    wDAT=1; //keep DAT high
    wCLK=0; //pull CLK low to signal that you want to communicate. Int2
            //will trigger right away
}

void main(void)
{
    initialize(); //initialize mcu
    initializeKyBd(); //initialize keyboard
}

```



```
while(1) {  
    if(!queueEmpty) && (UCSRA&0x20) //if queue not empty and UART is available  
    {  
        t2visits=0; //reset timer2 count value  
        TCNT2=0; //reset timer2  
        TIFR=TIFR&0xBF; //set bit6=0 to reset overflow flag  
        UDR=queueGet(); //send data out  
    }  
}  
}
```