

Palestine Polytechnic University



**College of Engineering & Technology
Electrical and Computer Department**

Graduation Project

**Vehicle Ad-Hoc Network
(VANET)**

Project Team

Afnan Al-Adam
Reham Tel

Aysha Zarour
Fidaa Abu-Shunnar

Project Supervisor
Dr. Murad Abu-Subaih

Hebron – Palestine
June , 2011

Palestine Polytechnic University



**College of Engineering & Technology
Electrical and Computer Department**

Graduation Project

**Vehicle Ad-Hoc Network
(VANET)**

Project Team

Afnan Al-Adam
Reham Tel

Aysha Zarour
Fidaa Abu-Shunnar

Project Supervisor
Dr. Murad Abu-Subaih

Hebron – Palestine
June , 2011

**Vehicle Ad-Hoc Network
(VANET)**

Project Team

Afnan Al-Adam
Reham Tel

Aysha Zarour
Fidaa Abu-Shunnar

Project Supervisor

Dr. Murad Abu-Subaih

**This Graduation Project submitted to Telecommunication and
Electronic Engineering “Department in College of Engineering and
Technology”**

Palestine Polytechnic University

**For accomplishment the requirements of the bachelor degree in
Engineering field at Telecommunication and Electronic Engineering**

**Palestine Polytechnic University
Hebron – Palestine
June, 2011**

جامعة بوليتكنك فلسطين
الخليل – فلسطين
كلية الهندسة و التكنولوجيا
دائرة الهندسة الكهربائية والحاسوب

اسم المشروع
**Vehicle Ad-Hoc Network
(VANET)**

أسماء الطلبة

رهام إبراهيم ظل

أفنان نائل العدم

فداء يوسف أبو شنار

عائشة هاشم زعرور

بناء على نظام كلية الهندسة والتكنولوجيا و إشراف المشرف المباشر على المشروع وموافقة أعضاء اللجنة الممتحنة تم تقديم هذا المشروع إلى دائرة الهندسة الكهربائية والحاسوب و ذلك للوفاء بمتطلبات درجة البكالوريوس في الهندسة تخصص هندسة اتصالات و الكترونيات .

توقيع المشرف

.....

توقيع اللجنة الممتحنة

.....

توقيع رئيس الدائرة

.....

الإهداء

بسم الله الرحمن الرحيم
(قل اعملوا فسيرى الله عملكم ورسوله والمؤمنون)
صدق الله العظيم

إلهي لا يطيب الليل إلا بشكرك ولا يطيب النهار إلا بطاعتك .. ولا تطيب اللحظات إلا بذكرك ..
ولا تطيب الآخرة إلا بعفوك .. ولا تطيب الجنة إلا برويتك الله جل جلاله
إلى من بلغ الرسالة وأدى الأمانة .. ونصح الأمة .. إلى نبي الرحمة ونور العالمين ..
سيدنا محمد صلى الله عليه وسلم
إلى من كلله الله بالهبة والوقار .. إلى من علمني العطاء بدون انتظار .. إلى من أحمل اسمه
بكل افتخار ..
والذي العزيز
إلى ملاكي في الحياة .. إلى معنى الحب وإلى معنى الحنان والتفاني .. إلى بسملة الحياة وسر
الوجود إلى من كان دعائها سر نجاحي وحنانها بلسم جراحي إلى أغلى الحبايب ..
أمي الحبيبة

إلى جميع زملائنا وأصدقائنا أينما كانوا ..

إليكم جميعاً نهدي هذا العمل المتواضع

فريق العمل

Acknowledgment

We acknowledge Palestine Polytechnic University for giving us the possibility to show some of what we have learned from it.

And we acknowledge all the instructors in the electrical and computer department for their great impact in our education, especially the supervisor of this project Dr. Murad Abu-Subaih , and everybody who effected our education.

Great thanks for **scientific monograph society (Alresalla society)** which supported us financially, and removed the problem of our finance thinking.

We want to thank Eng.Sami Salamin , Eng.Waddah Sultan , Eng.Ra'fat aljunidi,and Eng Rasmi Said Ahmad for their help and support.

Finally we can't forget to acknowledge our great parents who scarified themselves for educating us and facilitate our life, and for all their clemency and tolerance.

Team work

Abstract:

VANET (Vehicular Ad-Hoc Network) is a kind of ad-hoc mobile network where is emerging standard for data communication among nearby vehicles, and between moving vehicles with fixed equipment, usually described as roadside equipment.

Congestion is one of the most pressing problems for transportation research.

This project tries to find a solution to improve safety situations and save time for passengers and drivers. In this project, we count the number of vehicles on roads and their speed. According to this number and speed the type of congestion is determined. The type is then transmitted using wifi technology, the near vehicles receives the signal and know the status of the road, then retransmit it to the other vehicle using ad-hoc communication .

List of Abbreviations:-

VANET	Vehicles Ad hock Network
LAN	Local Area Network
MANET	Mobile Ad hock Network
SANET	Sensor Ad hock Network
DSRC	Dedicated Short Range Communications
ITS	Intelligent Transportation System
IVC	Inter_Vehicle Communication
RVC	Road Side_to_Vehicle Communication
PWM	Pulse Width Modulation
MCU	Microcontroller Unit
Wi_Max	Worldwide Interoperability for Microwave Access
CPU	Central Processing Unit
EEPROM	Erased Electrical Erasable Programmable Read Only Memory
RAM	Random Access Memory
PIC	Peripheral Interface Controller
IDEs	Integrated Development Environment
ADC	Analog to Digital Convertor
Wi-Fi	Wireless fidelity

WLAN	Wireless Local Area Network
G	Gain
D	Directivity
E	efficiency
dB	decibel
HPBW	Half- Power Beam Width
dBi	decibel isotropic
TCP/IP	Transport Control Protocol over Internet Protocol
LCD	Liquid Crystal Display
CRT	Cathode Ray Tube
LSB	Least Significant Bit
mV	millivolt
cm	centimeter

List of Figures:

Figures	Page
Figure 2.1 Ad hoc network	10
Figure 2.2 Example of MANET	11
Figure 2.3The type of VANET network .	12
Figure 2.4 various intelligent transportation systems component	14
Figure 2.5 LV –MaxSonar –EZ0 Range Finder	16
Figure 2.6 intel 8 bit Microcontroller	17
Figure 2.7 PIC Microcontroller	18
Figure 2.8 PICDEM™ FS-USB Evaluation Kit for PIC18F4550	19
Figure 2.9 WiFly GSX RN-131G	21
Figure 2.10 Basic 20x2 Character LCD FSTN - Black on Green	23
Figure 3.1 System General Block Diagram.	25
Figure 3.2: Mounting of Ultrasonic Range-Measuring Sensors	27
Figure 3.3 PIC 18F4550 pin diagram	28
Figure 3.4 PIC18F4550 I/O Organization	29
Figure 3.5 PIC18F4550 ADC Inputs	30
Figure 3.6 Antenna block diagram	30
Figure 3.7 Wireless transceiver diagram	31
Figure 3.8 WiFly GSX RN-131G	31
Figure 3.9 WiFly GSX with host microcontroller	32
Figure 3.10 pins of the wifly GSX	33
Figure 3.11interfacing between WiFly GSX and LCD screen	34
Figure 3.12 system Design Block Diagram.	36
Figure 3.13 the system flow control	37
Figure 3.14 system flow control.	38
Figure 4.1 General System Interfacing Block Diagram.	40
Figure 4.2: PICA (connect sensor with theWiFly GSX).	41
Figure 4.3: PIC B (connect the Ad hocWiFly GSX with LCD).	41
Figure 4.4: Interfacing between ultrasonic sensor and PIC microcontroller.	42

Figure 4.5: Interfacing First WiFly GSX Wireless Transceiver with PIC.	43
Figure 4.6 Interfacing two WiFly GSX Wireless Transceiver with PIC.	44
Figure 4.7: Interfacing between LCD and PIC.	45
Figure 4.8: The interfacing between PIC, Wifly-GSX and LCD on car.	45
Figure 4.9 Battery Interfacing.	46
Figure 4.10: Interfacing Electrical Component.	47
Figure 5.1: MPLAB IDE Editor.	50
Figure 5.2: ICSP Programmer and PICkit software.	52
Figure 5.3 : Step A/D Conversions.	53
Figure 5.4 USART Library.	55
Figure 5.5 Flow chart of the first transceiver .	57
Figure 5.6 Flow chart of the second transceiver .	58
Figure 5.7 Starting a Telnet session using TeraTerm in adhoc.	59
Figure 5.8A Telnet session using TeraTerm.	59
Figure 5.9 WiFly in Command Mode.	60
Figure 5.10 Programming Auto-association Commands.	61
Figure 5.11 Programming Auto-association Commands	62
Figure 5.12 Disabling Auto sleep Mode.	62
Figure 5.13 Programming Transferring of Data commands.	63
Figure 5.14 Programming UART Settings.	63
Figure 5.15 Errors in Commands	64
Figure 5.16 A GET Command Example.	65
Figure 6.1 Sensing Unit	71
Figure 6.2 Beam shape of the sensor	72
Figure 6.3 Minimum Acquisition Time in the PIC18F4550 ADC.	72

Figure 6.4: Sensing unit with Access point	73
Figure 6.5 Components connection on the vehicle	73
Figure 6.6 Message shown on the LCD when congestion is high	74
Figure 6.7 : Message shown on the LCD when congestion is medium	74

List of Tables :

Tables	page
Table 1.1 hardware costs	4
Table 1.2 Project Timing Plan Table.	5
Table 1.3 Schedule Table	6
Table 2.1: Overview of the IEEE 802.11 standards.	20
Table 3.1 Interface pins of LCD	35
Table 4.1: Figures key	48
Table 6.1 Testing Schedule	69

List of contents:-

Subject	page
Dedication	
Acknowledgment	
Abstract	
List of abbreviations	
List of figures	
List of tables	
Chapter one :Introduction	
1.1 Overview	2
1.2 Problem Statement	2
1.3 Project Objective	2
1.4 Project Benefits	3
1.5 Literature review	3
1.6 Estimated Cost	4
1.7 Time planning	4
1.7.1 Time Schedule	5
1.7.2 Schedule Table	6
1.8 Report Content	7
Chapter tow: Theoretical background	
2.1 Introduction	9
2.2 Ad-Hock	9
2.3 MANET	10
2.4 VANET	11
2.4.1 Characteristic of VANETs	13

2.5 ITS	14
2.6 Traffic Flow Sensors	15
2.6.1 Ultrasonic sensor	15
2.7 Microprocessor	16
2.8 Wireless	19
2.8.1 Wi-Fi	20
2.9 Wireless transceiver	21
2.10 LCD Screen	23
Chapter three: System Design	
3.1 Project Objectives	25
3.2 General Block Diagram	25
3.3 System Operation	26
3.3.1 Software components:	26
3.3.2 Hardware components:	26
3.3.3 Ultrasonic sensor:	27
3.3.4 Sensor to Microcontroller	28
3.3.5 Access point	30
3.3.6 WLAN Transceiver:	31
3.3.7 Receiver to microcontroller	34
3.3.8 LCD screen	34
3.4 Detailed System Design and Flow Control	36
3.4.1 Detailed System Design	36
3.4.2 System Flow Control	37
Chapter four: Hardware System Design	
4.1 Introduction.	40
4.2 Interfacing and Connections between the components.	41

4.2.1 PIC Requirements.	41
4.2.2 Ultrasonic Sensor Interfacing.	42
4.2.3 Interfacing WiFly GSX Wireless Transceiver.	42
4.2.4 LCD Interfacing.	44
4.2.5 Battery.	46
Chapter five: Software Design Implementation	
5.1 Software System Design.	50
5.2 Detailed Design Description.	50
5.2.1 Sensor and PIC programming	50
5.2.1.1 MPLAB IDE	51
5.2.1.2 Programmer.	52
5.2.1.3 Sensor parameters.	53
5.2.1.3.1 Software description of sensor interfaces.	53
5.2.2 WiFly GSX transceiver programming.	56
5.2.3 LCD with PIC programming.	65
5.2.3.1 List of Component Modules.	65
5.2.3.2 How Using the Library Module in a Project.	65
5.2.3.4 . The Shared Parameters of LCD.	66
5.3 System Configuration.	66
5.3.4 LCD library configuration.	66
Chapter Six: System Testing	
6.1. Introduction.	69
6.2 Testing Scheduling.	69
6.3 Testing Procedures.	69
6.3.1 Testing Strategies.	71

Chapter seven: Conclusion and Recommendation	
7.1 Introduction .	77
7.2. System Achievements	77
7.3. Real Learning Outcomes.	
7.4. Recommendation.	78
Appendix	
Appendix A	80
Appendix B	101
Appendix C	114

1

Chapter One

Introduction

- 1.1 Overview**
- 1.2 Problem Statement**
- 1.3 Project Objective**
- 1.4 Project Benefits**
- 1.5 Literature Review**
- 1.6 Estimated Cost**
- 1.7 Time Planning**
- 1.8 Report Content**

Chapter One

Introduction

1.1 Overview

The time is an important factor in every aspect of our life. The most annoying thing that wastes time is road traffic. This leads to many difficulties. In large countries, this problem affects the economics of them. So, we need to solve this problem and reduce its effects as possible as we can. We need a mechanism that at least inform the drivers about the state of roads and give them a chance to find out other alternatives and save their time.

VANET networks allow users to communicate with each other while they are in their moving vehicles. This enables drivers to exchange information about the road state.

1.2 Problem Statement

Any country in the world has its importance about traffic and roads when they are clean, tidy, with no noise, no pollution, and not annoying for the drivers and the passengers. The most annoying thing in the roads is the congestion.

Congestion prevents traffic motion, wastes time, increases collision rates and creates a confluence of the annoyances of air pollution and noise, briefly congestion is The Real Road Problem.

This problem needs to be solved by good techniques; a simple one is implemented in our project.

1.3 Project Objective

The main purpose of this project is to give the drivers information about the state of roads, give them a chance to find out other alternatives, save their time, and to keep the roads out of danger.

1.4 Project Benefits

1. Reduce congestion control (monitor road traffic in order to eliminate congestion).
2. An attempt to solve the problem of wasting time due to road traffic.
3. Keep our city clean of noise especially at inter-section roads.
4. Investigate the possibility of using Wi-Fi communication between vehicles.

1.5 Literature review

Many projects have been developed in VANET, focusing on simulation, increasing reliability of broadcasting messages, and increase the security of the VANET.

For example: In [1] the authors studied the performance and usability of wireless Vehicular Ad hoc network, and they improved its performance, especially for broadcasting.

In [2] the author developed broadcast protocol to improve the reception rate of broadcast messages and determine the congestion of the node by simply analyzing the sequence numbers of packets that have recently received. Based on the percentage of packets that are successfully received in the last few seconds, a node dynamically adjusts the size of the contention window, and then the performance of the broadcast protocol improved.

In [3] the authors studied the performance of a forwarding scheme and a channel access mechanism for improving the efficiency of message broadcasting in VANETs.

In [4] the authors proposed a set of security primitives that can be used as the building blocks of secure applications.

All previous projects describe the way to connect between vehicles, and how to make security, so all previous projects are simulation. This project is applied project using broadcasting message to communicate between vehicles.

1.6 Estimated Cost

The following table shows the estimated hardware costs.

Table 1.1 Hardware cost.

1	Ultrasonic sensor	138
2	PIC Board	214
3	Wifly –GSX transceiver	496X3
4	LCD	26X2
5	Microcontroller(18f4550)	62X2
6	Regulator	35x3
7	PIC "bases"	10X3
8	PINs	20
9	Welding Board	70
10	Models+ Cars	400
11	Travelling	70
12	Documentation print	180
Total		2891 SH

1.7 Time planning:

The project plan follows the following time schedule, which includes the relation between the tasks of the study and the analysis system.

1.7.1 Time Schedule

The following table explains the expected timing plan.

Table 1.2 Project Timing Plan Table.

System Definition	6 Weeks
Requirement Analysis	2 Weeks
System Design and Documentation	6 Weeks
Documentation	8 Weeks
Software implementation	14 Weeks
Hardware implementation	11 Weeks
System testing	2 Weeks
Documentation	6 Weeks
Total	32 Weeks

1.7.2 Schedule Table

Table 1.3 Schedule Table.

Weeks Tasks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
System Definition																
Requirement Analysis																
System Design																
Documentation																
Weeks Tasks	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Software implementation																
Hardware implementation																
System Testing																
Documentation																

1.8 Report Content

This project is mainly divided into seven chapters:

Chapter one includes the introduction, which describes the general review of the system, problem statement, project benefits, estimated cost, and time planning, are exist to give readers a general view about the project.

Chapter two discusses the theoretical background. It starts with general information about the project, MANETs, Ad hoc, VANETs, wireless, wifi, and then discusses the important aspects of the system including PIC microcontroller, wireless transceivers, video camera and sensors.

Chapter three presents the general system design concepts. It includes system objectives, general system block diagram, and system flow control.

Chapter four presents the hardware design implementation; it includes a general description for the system interfacing, detailed interfacing techniques and requirements to connect the system components together.

Chapter five discussed the software design implementation. It contains a detailed block diagrams to show how the software requirements in this system were designed and implemented to perform the required function of the robot.

Chapter six presented the testing operation of the system. Testing time schedule, testing procedure and testing strategies presented through black box testing and white box testing.

Chapter seven suggests future work that could be added to this system.

2

Chapter Two

Theoretical Background

2.1 Introduction

2.2 Ad-Hock

2.3 MANET

2.4 VANET

2.5 Intelligent Transportation System (ITS)

2.6 Traffic Flow Sensors

2.7 Microprocessor

2.8 Wireless

2.9 Wireless transceiver

2.10 LCD Screen

Chapter Two

Theoretical Background

2.1 Introduction

This chapter defines the basic concepts implemented in this project.

In addition, it discusses each component used in this system; explain its characteristics, uses and types.

2.2 Ad-hoc network

An ad-hoc network is a local area network (LAN) that is built spontaneously as devices connect. Instead of relying on a base station to coordinate the flow of messages to each node in the network, the individual network nodes forward packets to and from each other. In Latin, ad hoc literally means "for this," meaning "for this special purpose" and also, by extension, improvised or impromptu. In the Windows operating system, ad-hoc is a communication mode (setting) that allows computers to directly communicate with each other without a router. [5]

In other definition an ad hoc network is one where there are no access points passing information between participants. Infrastructure networks pass information through a central information hub which can be a hardware device or software on a computer. Office networks, for example, generally use a server to which company workstations connect to receive their information. Ad hoc networks, on the other hand, do not go through a central information hub.

Ad hoc networks are generally closed in that they do not connect to the Internet and are typically created between participants. But, if one of the participants has a connection to a public or private network, this connection can be shared among other members of the ad hoc network. This will allow other users on the spontaneous ad hoc network to connect to the Internet as well.

An ad hoc network can be thought of as a peer-to-peer network for the wireless age. Peer-to-peer or workgroup style networks were used to create a network environment for early Windows computers. This allowed these early computers to connect to each

other to exchange information, usually in a smaller office environment without the need for domains and the additional management and overhead that comes with them. [6]

Ad-hoc network can be classified depending on the applications that use it to:

- 1-Vehicle ad hoc network (VANET).
- 2-Mobile ad hoc network (MANET).
- 3-Wireless mesh networks.
- 4-Wireless sensor networks (sensor or static (SANET)).

Characteristic of the ad-hoc networks are rapidly deployable, reconfigurable, high node mobility, low band width lack of centralized entity.



Figure2.1 Ad hoc network

2.3 MANET

A mobile ad hoc network (MANET), sometimes called a mobile mesh network, is a self-configuring network of mobile devices connected by wireless links. Each device in a MANET is free to move independently in any direction, and will therefore change its links to other devices frequently. Each must forward traffic unrelated to its own use, and therefore be a router. The primary challenge in building a MANET is equipping each device to continuously maintain the information required to properly route traffic. Such networks may operate by themselves or may be connected to the larger Internet.

MANETs are a kind of wireless ad hoc networks that usually has a routable networking environment on top of a Link Layer ad hoc network. They are also a type of mesh network, but many mesh networks are not mobile or not wireless.

The growth of laptops and 802.11/Wi-Fi wireless network have made MANETs a popular research topic since the mid- to late 1990s. Many academic papers evaluate protocols and abilities assuming varying degrees of mobility within a bounded space, usually with all nodes within a few hops of each other and usually with nodes sending data at a constant rate. Different protocols are then evaluated based on the packet drop rate, the overhead introduced by the routing protocol, and other measures. [7]

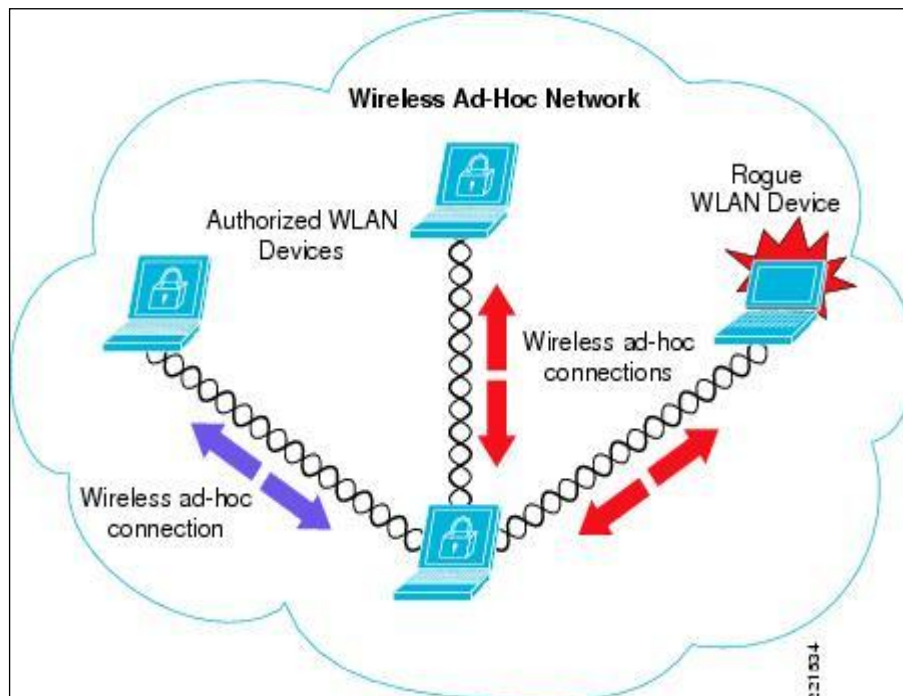


Figure2.2: Example of a MANET

2.4 VANET

A Vehicular Ad-Hoc Network, or VANET, is a subset of mobile ad-hoc network, which supports data communications among nearby vehicles and between vehicles and nearby fixed infrastructure, and generally represented as roadside entities. Depending on the range of data communications, nodes in VANET communicate

among themselves in type of short-range (vehicle-to-vehicle) or medium-range (vehicle-to-roadside) communications.

It is a technology which is used to move cars as joint in network to make a transportable network. Participating cars become a wireless connection or router through VANET and it allow the cars almost to connect 100 to 300 meters to each other and in order to create a wide range network, other vehicles and cars are connected to each other so the mobile internet is made. It is supposed that the first networks that will incorporate this technology are fire and police mobiles to interact with one another for security reasons. [8]

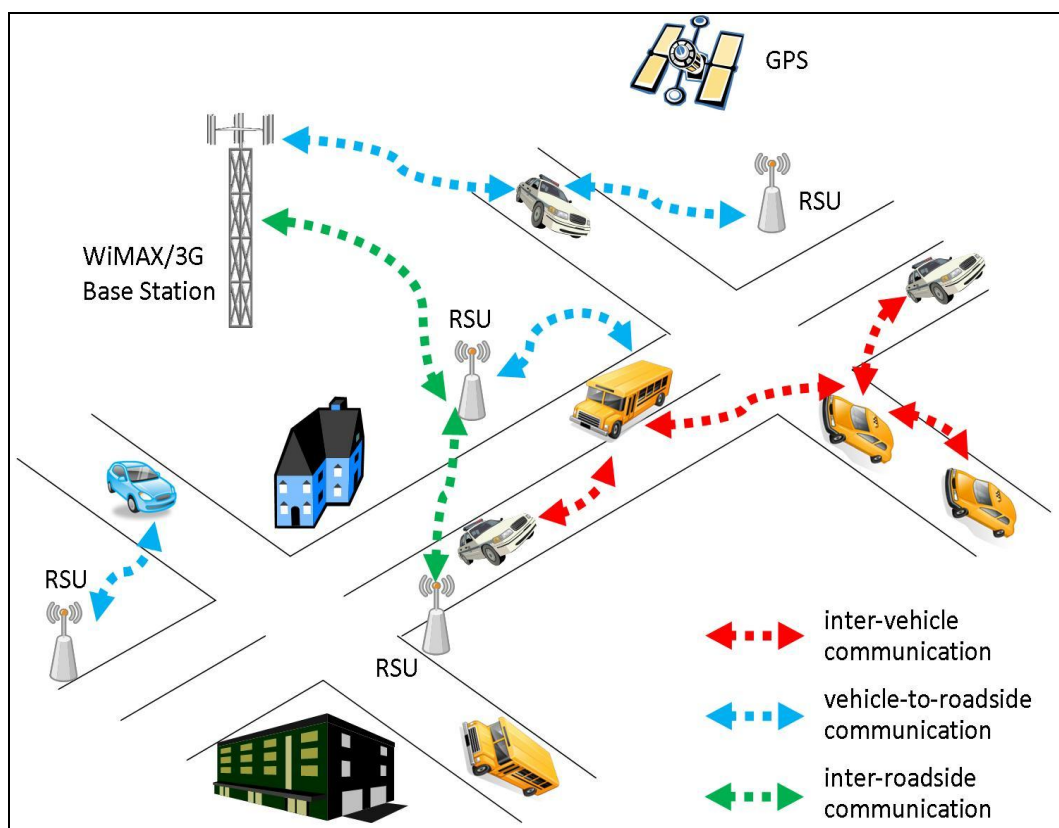


Figure 2.3: The types of the VANET network

Vehicular ad hoc network (VANET), a subclass of mobile ad hoc networks (MANETs), is a promising approach for future intelligent transportation system (ITS). These networks have no fixed infrastructure and instead rely on the vehicles themselves to provide network functionality. However, due to mobility constraints, driver behavior, and high mobility, VANETs exhibit characteristics that are dramatically different from many generic MANETs. [9]

2.4.1 Characteristics of VANETs:

It is similar to MANETs in some characteristics such as short radio transmission range, self-organization and self-management, and low bandwidth [10]

Some characteristics of VANET are as follows:

1-High mobility with the constraint of road topology: The nodes in VANETs usually are moving at high speed. The node motion is constrained by the road topology and layout. This leads to predictability of node positions, making protection of node privacy harder in VANETs. [11]

2-Frequently disconnected network: The connectivity of the MANETs could also be changed frequently. Especially when the vehicle density is low, it is highly probable that the network is disconnected. [12]

3-Mobility modeling and predication: Due to high mobility and dynamic topology, mobility model and predication play an important role in network protocol design for VANETs. Furthermore, Factors such as road configuration, traffic laws, safety limits, and physical limits affect the mobility of vehicles mobility pattern.[13]

4- Potentially unbounded network size: VANETs could involve the vehicles in one city, several cities, or even a country. Thus, it is necessary to make any protocols for VANETs scalable in order to be practical.

5- Better physical protection: The VANET nodes are better protected than those nodes in other MANETs. Thus, VANET nodes are more difficult to compromise, which is also good news for security provisioning in VANETs.

Vehicular Ad-hoc Networks are expected to implement variety of wireless technologies such as Dedicated Short Range Communications (DSRC) which is a type of Wi-Fi. Other candidate wireless technologies are Cellular, Satellite, and WiMAX. Vehicular Ad-hoc Networks can be viewed as component of the Intelligent Transportation Systems (ITS).

Vehicular Networks are an envision of the Intelligent Transportation Systems (ITS). Vehicles communicate with each other via Inter-Vehicle Communication (IVC) as well as with roadside base stations via Roadside-to-Vehicle Communication (RVC). The optimal goal is that vehicular networks will contribute to safer and more efficient

roads in the future by providing timely information to drivers and concerned authorities.

2.5 ITS

The term intelligent transportation system (ITS) refers to efforts to add information and communications technology to transport infrastructure and vehicles in an effort to manage factors that typically are at odds with each other, such as vehicles, loads, and routes to improve safety and reduce vehicle wear, transportation times, and fuel consumption.

Interest in ITS comes from the problems caused by traffic congestion and a synergy of new information technology for simulation, real-time control, and communications networks. Traffic congestion has been increasing worldwide as a result of increased motorization, urbanization, population growth, and changes in population density. Congestion reduces efficiency of transportation infrastructure and increases travel time, air pollution, and fuel consumption. [14]



Figure 2.4: Various intelligent transportation systems components.

2.6 Traffic Flow Sensors

A traffic flow sensor is a device that indicates the presence or passage of vehicles and provides data or information that supports traffic management applications such as freeway, mainline, incident detection, and gathering of vehicle volume and classification data to meet state and federal reporting requirements.

Sensor types are divided into in-roadway and over-roadway classes, defined as follows:

*An **in-roadway sensor** is one that is embedded in the pavement of the roadway, embedded in the subgrade of the roadway, or taped or otherwise attached to the surface of the roadway. Examples of in-roadway sensors include inductive loop detectors, which are sawcut into the pavement; and magnetometers, which may be placed underneath a paved roadway or bridge structure.

*By contrast, an **over-roadway sensor**, or non-intrusive sensor, is one that is mounted above the surface of the roadway either above the roadway itself or alongside the roadway, offset from the nearest traffic lane by some distance. Examples of over-roadway sensors are video image processors that utilize cameras mounted on tall poles adjacent to the roadway or traffic signal mast arms over the roadway; microwave radar, ultrasonic, and passive infrared sensors mounted in a similar manner; and laser radar sensors mounted on structures that span the lanes to be monitored.[15]

In our project we decided to use an over-roadway sensor: ultrasonic sensor.

2.6.1 Ultrasonic sensor

Ultrasonic sensors can be used to detect the presence of something in the range of the sensor and the distance between the sensor and that body, sometimes called range finders. There are many types of these sensors; the type implemented in this system is the LV-MaxSonar-EZ0 (see figure 2.5).



Figure 2.5 LV-MaxSonar-EZ0 Range Finder.

The LV-MaxSonar-EZ0 sensor operates at range of voltage between 2.5V to 5.5V provide ranging up to 6.45 meter (255 inch) with a controlled beam width. This sensor output ranges in three manners, as a Pulse Width Modulation (PWM) representing the range with scaling factor of 147uS per inch. Another output as a serial data represented in ASCII code. The third output as an analog voltage with sensitivity depending on the voltage source, for example with a supply of 5V yields ~9.8mV/in. and 3.3V yields ~6.4mV/in.[12] The analog output is the type used in this system, due to the simplicity of reading the range in analog signal using the PIC microcontroller that provides up to 13 analog inputs.

2.7 Microcontroller

A microcontroller (also microcontroller unit, MCU or μC) is “a small computer on a single integrated circuit consisting of a relatively simple CPU combined with support functions such as a crystal oscillator, timers, watchdog timer, serial and analog I/O etc. Program memory in the form of EEPROM (Electrically Erasable Programmable Read Only Memory) or ROM (Read Only Memory) is also often included on chip, as well as a typically small amount of RAM (Random Access Memory). [16]

Microcontrollers have long been a convenient interface for embedded systems; they represent the core of the control system for electronic devices in dedicated applications. Thus, in contrast the microprocessors that are used in general purpose applications like personal computers that need high-performance, and multitasking. Microcontrollers contain data and program memory, serial and parallel I/O, timers, external and internal interrupts, and peripherals. These make them a strong choice when implementing control systems.

Microcontrollers are used in automatically controlled products and devices, such as automobile engine control systems, remote controls, office machines, appliances, power tools, and toys. By reducing the size and cost compared to a design that uses a separate microprocessor, memory, and input/output devices, microcontrollers make it economical to digitally control even more devices and processes. Mixed signal microcontrollers are common, integrating analog components needed to control non-digital electronic systems. [17]

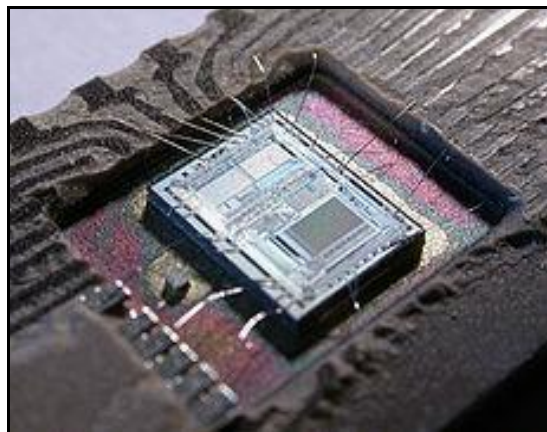


Figure 2.6 Intel 8-bit Microcontroller

Microcontrollers exist in many types and forms, for different applications and costs. And here is a list of some microcontrollers' types that are known in the market:

1. IMPS (32-bit PIC32)
2. ARM processors
3. 8051
4. PIC (8-bit PIC16, PIC18, 16-bit dsPIC33 / PIC24)

In this system the PIC microcontrollers are used.

The PIC microcontrollers are a family from Harvard architecture microcontrollers that is manufactured by Microchip Technology. The name PIC at the beginnings stood for "Programmable Interface Controller" and shortly after was replaced with the name "Programmable Intelligent Computers".

PICs are popular with both industrial developers and hobbyists alike due to their low cost, wide availability, large user base, extensive collection of application notes, availability of low cost or free development tools, and serial programming (and re-programming with flash memory) capability.[18]

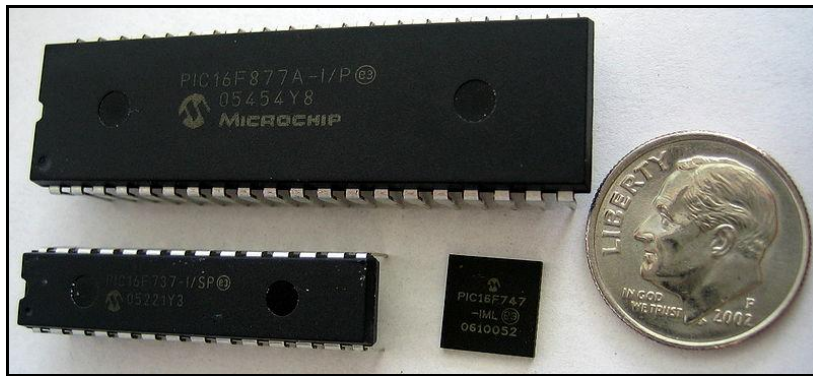


Figure 2.7 PIC Microcontrollers

The PIC chips (or PICmicro chips) are as stated previously, programmable chips, programmed to perform a special operations for dedicated applications in embedded systems, they provide strong interfacing abilities with many peripherals and other microcontrollers in other embedded systems.

PIC microcontrollers are a simple three steps process, write the code, compile the code, and upload the code into a microcontroller. Writing the code can be developed in many Integrated Development Environments (IDE's) for example, MPLAB IDE, which is software, developed for the Microchip appliances like the PIC microcontrollers. Compiling the code can be done by the compiler of the MPLAB IDE.

There are different compilers associated to work with PIC chips, C compiler, or assembler for assembly language codes, and many more. The decision of which compiler to use, is a developer choice, depending on the application which the PIC is a part of. The final step of programming the PIC chip is uploading the code into the microcontroller. This can be done also in MPLAB IDE or in different programs that are connected to the PIC kit (figure 2.8). The uploading process can be done through a USB cable or other connecting technique depending on the kit that contains the PIC chip.



Figure 2.8 PICDEM™ FS-USB Evaluation Kit for PIC18F4550

The PIC microcontroller architecture makes interfacing most peripherals with the PIC a far from hard task, the I/O's are organized as ports, PORTA, PORTB, etc. each port can be treated as a unit or as single I/O pins, 8-bit, 5-bit or other organization. Each port was initially configured to do a specific operation, serial data operations, Analog-to-Digital conversion, and many more, but it's not always necessary to stick with the initial configuration, each port or single I/O pin can be configured to do different operation from what initially configured.

In this project the PIC18F4550 will be used. This is due to its availability, cheap cost, easy programming, it has all what is needed for the implementation of this project, enough I/O Ports, ADC, timers and counters, and variable operating frequency up to 48 MHZ.

2.8 Wireless

In telecommunications, wireless communication is the transfer of information without the use of wires. It use electromagnetic waves to carry the signal over part or all of the communication path. The distances involved may be short (a few meters as in television remote control) or long (thousands or millions of kilometers for radio communications).

The term "wireless" can be rather ambiguous, since it may refer to several different wireless technologies. The two most common types of wireless capabilities mobiles are Wi-Fi and Bluetooth.

2.8.1 Wi-Fi

Wi-Fi (Wireless Fidelity) is a generic term that refers to the IEEE 802.11 communications standard for Wireless Local Area Networks (WLANs). It uses Radio Technologies to transmit & receive data at high speed. It operates at 2.4 GHz radio spectrum or operates at 5 GHz (less popular), which the usage of the frequency depend on the technology of Wi-Fi is used Wi-Fi refers to any system that uses the 802.11 standard , the term Wi-Fi is synonymous with 802.11b, as 802.11b was the first standard in that family to enjoy widespread popularity. Today, however, Wi-Fi can refer to any of the established standards: 802.11a, 802.11b, 802.11g and 802.11n. [19]

The IEEE 802.11 standards and task groups:

All of the 802.11 extensions can be found in the table 2.1.

Standard	Description
802.11a	5GHz OFDM PHY – 54 Mbps
802.11b	2.4GHz CCK PHY – 11 Mbps
802.11c	802.11 bridging
802.11d	International roaming
802.11e	QoS/efficiency enhancements
802.11f	Inter AP protocol
802.11g	2.4GHz OFDM PHY – 54 Mbps
802.11h	5GHz regulatory extensions
802.11i	Security enhancements
802.11j	Japan 5GHz band extensions
802.11k	Radio resource measurement
802.11m	Maintenance
802.11n	High throughput PHY

Table 2.1: Overview of the IEEE 802.11 standards

Which to take, 802.11a or 802.11g?

IEEE 802.11b enjoys international acceptance, as the 2.4-GHz radio frequency band is almost universally available and no license is available. 802.11b hardware can transmit data at speeds of up to 11 megabits per second.

IEEE 802.11g operates in the same frequency band as 802.11b, and is therefore backwards compatible with most of the older WLAN hardware. 802.11g+ hardware can transfer data at up to 108 Mbps, or at 11Mbps if operating with 802.11b devices.

IEEE 802.11a, which operates around the 5 GHz band 802.11a also provides for up to 54 Mbps throughput, but is not interoperable with 802.11b.

In this project the 802.11g is used, because it has some advantages rather than different types such as mobility, ease of Installation, flexibility, cost, reliability, use unlicensed part of the radio spectrum, roaming and speed references.

2.9 Wireless Transceivers

The wireless transceiver is a device that contains both transmitter and receiver which are combined in a common circuitry. It transmits and receives data wirelessly. In this project, the transceiver which is in the vehicle accepts signal from another transceiver which used as access point, then transmit signal to other vehicles.

After researching, it was found that the **WiFly GSX** transceiver which shown in figure 2.9, is the most suitable wireless transceiver available in the markets that could be used in this project.



Figure 2.9 WiFly GSX RN-131G

The WiFly GSX module is a stand alone, embedded wireless 802.11 networking module. Because of its small form factor and extremely low power consumption, the RN-131G is perfect for mobile wireless applications such as asset monitoring, GPS tracking and battery sensors. The WiFly GSX module incorporates a 2.4GHz radio, processor, TCP/IP stack, real-time clock, crypto accelerator, power management and analog sensor interfaces. This complete solution is preloaded with software to simplify integration and minimizes development of application. In the simplest configuration the hardware only requires four connections (PWR, TX, RX, GND) to create a wireless data connection.

Additionally, the sensor interface provides temperature, audio, motion, acceleration and other analog data without requiring additional hardware. The WiFly GSX module is programmed and controlled with a simple ASCII command language. Once the WiFly GSX is setup it can scan to find an access point, associate, authenticate and connect over Wi-Fi network [21].

This type of transceiver that used in our project due to a number of factors. First, it has an extremely small size and light weight, also its low cost and availability in the market.

One of the important factors of choosing the wireless transceiver is that it can be used as access point. This is because the system depends on the access point to receive the commands, and to transmit signals to surrounding cars. It can be also connected to PIC microcontroller, and support serial communication between them. Further interfacing details will be discussed in chapter 3.

It is very important to think about the power consumption for the system. As described earlier, this system is a mobile system, there are no wires attached to it, so the power consumption has to be limited as much as possible. The **WiFly GSX** transceiver was considered as a breakthrough in the wireless communication, because it has the least power consumption between the wireless transceivers, it has three modes, sleep mode of 4 μ A, standby of 15mA and active mode 40-212 mA. With this little power consumption the **WiFly GSX** can live on the regular AA batteries for many years.

2.10 LCD screen

A liquid crystal display (LCD) is a thin, flat electronic visual display that uses the light modulating properties of liquid crystals (LCs). LCs do not emit light directly.

They are used in a wide range of applications including computer monitors, television, instrument panels, aircraft cockpit displays, signage, etc. They are usually more compact, lightweight, portable, less expensive, more reliable, and easier on the eyes. They are available in a wider range of screen sizes than CRT and plasma displays, and since they do not use phosphors, they cannot suffer image burn-in.

LCDs are more energy efficient and offer safer disposal than CRTs. Its low electrical power consumption enables it to be used in battery-powered electronic equipment. It is an electronically-modulated optical device made up of any number of pixels filled with liquid crystals and arrayed in front of a light source (backlight) or reflector to produce images in colors or monochrome.[22]

In this project the Basic 20×2 character LCD FSTN –Black on Green (see figure2.10) is used. This is a basic 20 character by 2 line display. Utilizes the extremely common HD44780 parallel interface chipset. Interface code is freely available. It needs ~11 general I/O pins to interface to this LCD screen. No backlight.



Figure2.10 Basic 20x2 Character LCD FSTN - Black on Green

3

Chapter Three

System Design

3.1 Project Objectives.

3.2 General Block Diagram.

3.3 System operation.

3.4 System Flow Control.

Chapter Three

System Design

This chapter discusses the design concept of the system. It describes the project objectives, the system block diagram as well as presenting system entities.

3.1 Project Objectives

Project objectives can be summarized in these points:

- 1-To give continuous information about the congestion in the road.
- 2-To connect vehicles in road by an ad-hoc network.
- 3- To inform drivers about roads conditions.
- 4-To keep roads safety.

3.2 General Block Diagram

In this section, a general system block diagram is presented, and it is shown in figure 3.1

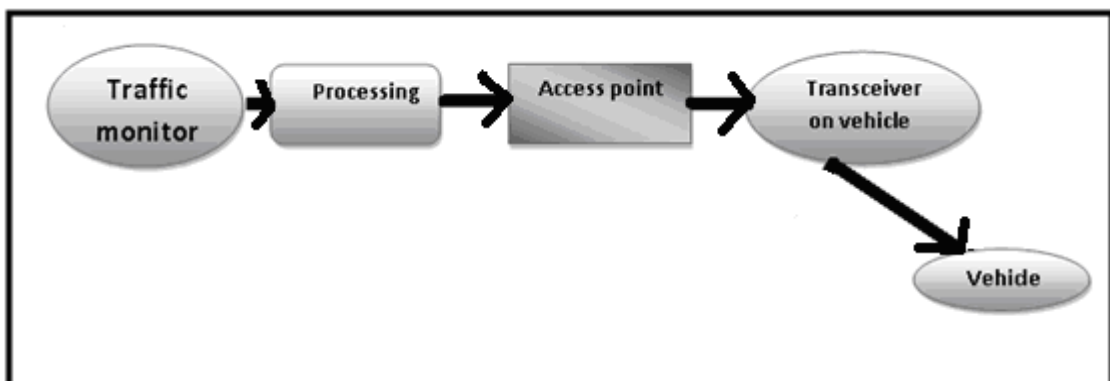


Figure 3.1 System General Block Diagram.

3.3 System Operation

An ultrasonic sensor mounted on the roadside detected the number of vehicles and processing it using PIC microcontroller, the decision taken depending on the results.

When the congestion type is determined by the processing unit, the results transmitted by an antenna from processing unit to near vehicles, this result shown on LCD in each vehicle that receives signals and retransmit it to others by ad-hoc network.

In order to understand each component, they are specified into two types, software, and hardware components.

3.3.1 Software components:

Software is needed to implement the hardware component connections in the system.

3.3.2 Hardware components:

The physical components of the project are:

1. Ultrasonic sensor.
2. PIC microcontroller.
3. Transmitter at processing unit.
4. Transceiver at vehicle.
5. LCD screen.

3.3.3 Ultrasonic sensor:

Ultrasonic sensors transmit pressure waves of sound energy at a frequency between 25 and 50 kHz, which are above the human audible range. Most ultrasonic sensors operate with pulse waveforms and provide vehicle count, presence, and occupancy information. Pulse-shape waveforms measure distances to the road surface and vehicle surface by detecting the portion of the transmitted energy that is reflected towards the sensor from an area defined by the transmitter's beam width. When a distance other than that to the background road surface is measured, the sensor interprets that measurement as the presence of a vehicle.

Pulsed energy transmitted at two known and closely spaced incident angles allows vehicular speed to be calculated by recording the time at which the vehicle crosses each beam and how much the car will spend under the sensor. Since the beams are a known distance apart, the speed can be calculated as beam separation distance divided by the time to traverse the beams. The preferred mounting configurations for range-measuring, pulsed ultrasonic sensors are looking from an overhead position and side viewing as shown in Figure 3.2.



Figure 3.2: Mounting of Ultrasonic Range-Measuring Sensors

3.3.4 Sensor to Microcontroller:

Sensor plays an important role since the whole work begins with its operation, however, use of such a sensor is not enough, it must be interfaced to a microcontroller to form a complete system.

The ultrasonic sensor sends micro waves continuously, these waves are cut by the vehicle, each cut of the wave manipulated using microcontroller to determine the road state (congestion is high, medium, or low).

Features of PIC18F4550 that provide such facilities are described in the following figure.

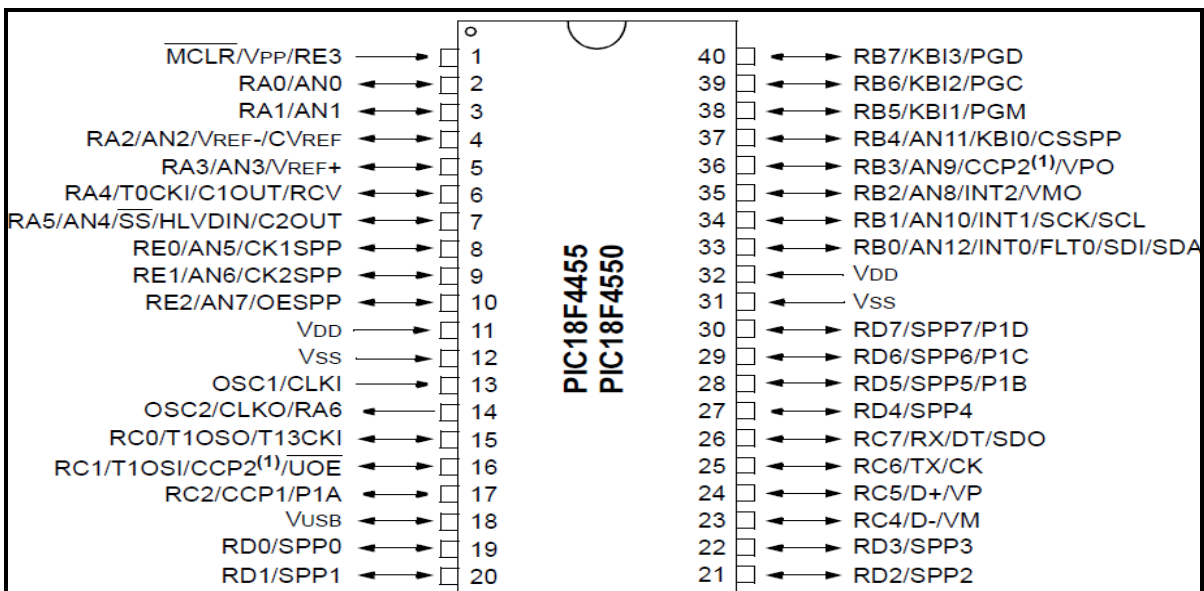


Figure 3.3 PIC18F4550 Pin Diagram

As shown in figure 3.3, the PIC18F4550 provides 33 single I/O pins, divided into 5 ports,

PORTA 6 pins, PORTB, PORTC, and PORTD 8 pins, and finally PORTE 3 pins.

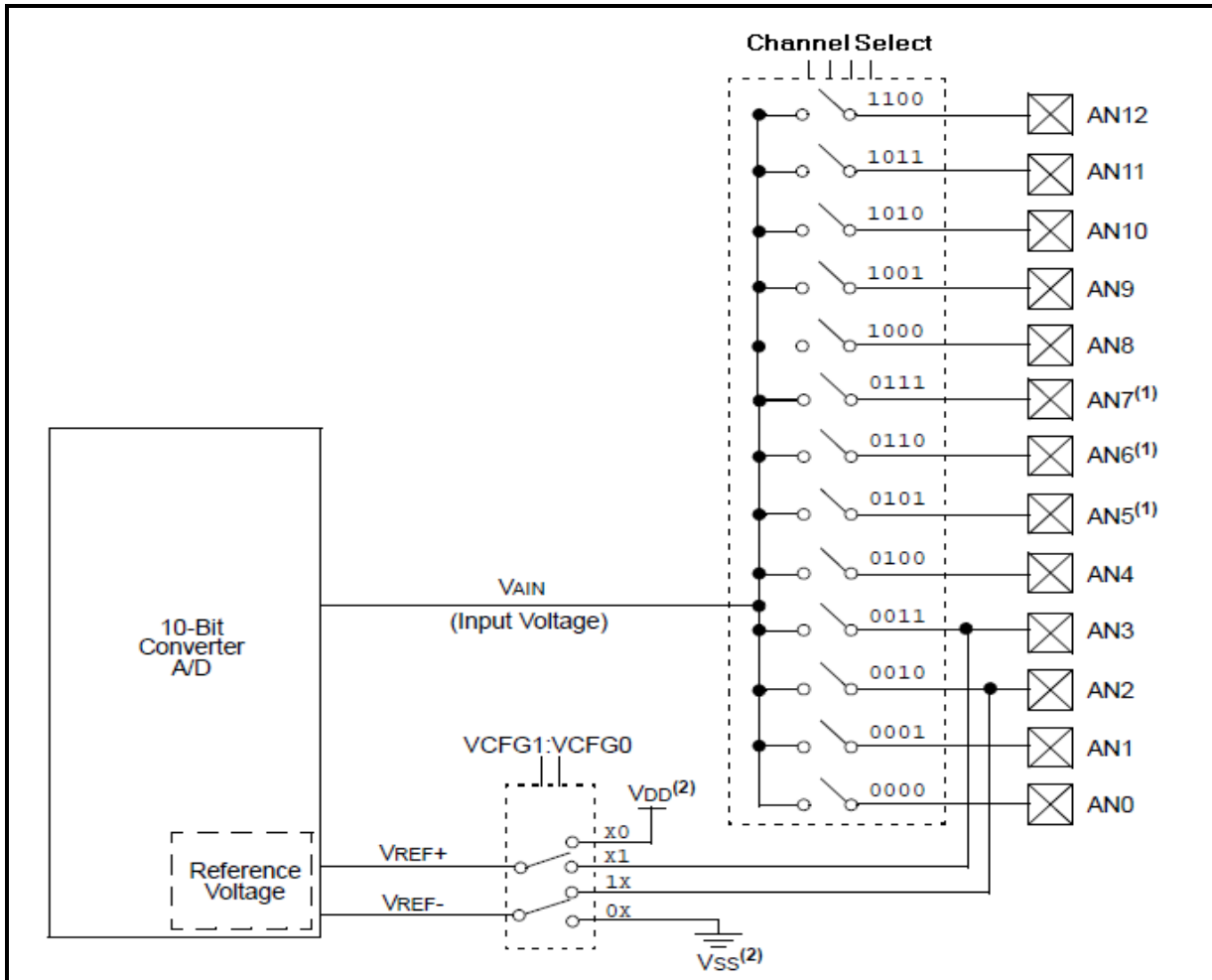


Figure 3.4 PIC18F4550 I/O Organizations

The PIC18F4550 as shown in figure 3.4, contains a built on Analog-to-Digital converter (ADC) with 10-bit resolution, up to 13 channels, and three timers. Figure 3.5 shows the interfacing of the ADC with the 13 input analog channels. The ADC is used to connect the ultrasonic sensor which gives analog readings for the distance of the vehicles in the range of the sensor.

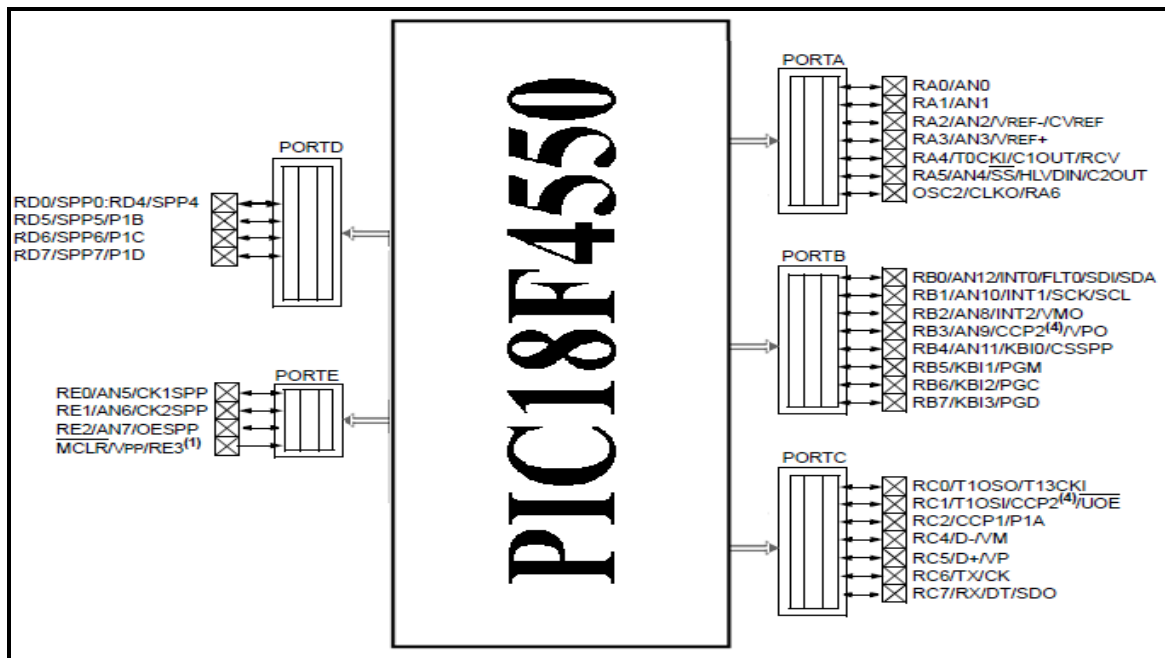


Figure 3.5 PIC18F4550 ADC Inputs

3.3.5 Access point

The WiFly GSX which acts as access point receive the data from PIC microcontroller, then transmits signal in all directions, the nearest vehicles receive this signal, as shown in figure 3.6.



Figure 3.6 Access point block diagram

3.3.6 WLAN Transceiver:

The vehicle is a moving object, for that we need to connect it with wireless transceiver (WLAN). Two scenarios for communication in this project are proposed which shown in figure 3.7.

First one is between access point and the transceiver in vehicle:

The access point transmit signal and the transceiver in vehicle receives this signal, which includes information about the traffic congestion. **Second** scenario is implemented between vehicles.

The second scenario includes receiving signal and retransmitted it to other vehicle near it by using ad-hoc principle.

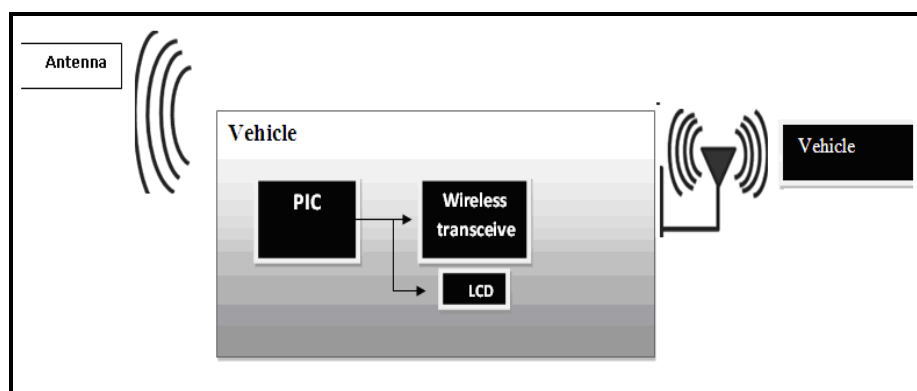


Figure 3.7 Wireless transceiver Diagram.

The WLAN transceiver in this system is Wifly-GSX, which shown in the figure 3.8. It operates in two types of communication, receives signal from access point using Wi-Fi technology, the other type is transmit and receive ad hoc between vehicles.

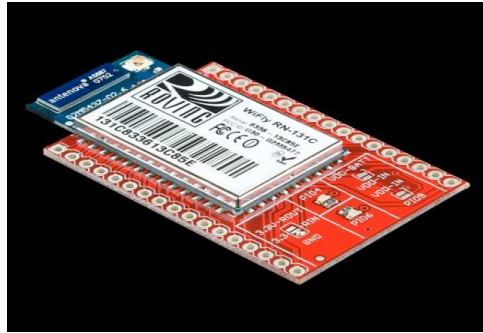


Figure3.8 : WiFly GSX RN-131C

The **WiFly GSX** wireless transceiver is a programmed module; it has a specific command which it understands if it received them. The system designer can communicate with the transceiver through the CMD on any computer with windows on it, or software can be downloaded to the computer, such as the **TeraTerm** software. This software creates a connection between the computer and the transceiver wirelessly, and after the connection is completed, the transceiver is able to receive from the developer. Or it can be connected to a PIC microcontroller and receives ASCII code form it as shown in figure 3.9.

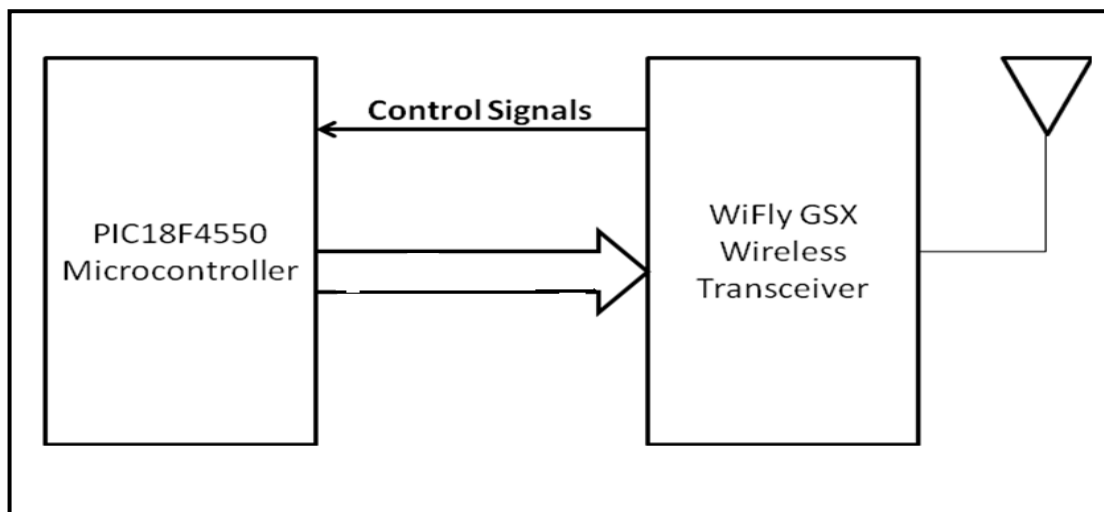


Figure 3.9 WiFly GSX with host microcontroller

As shown on figure 3.10, the WIFLY transceiver has 44 pins. One of these pins is left unconnected (PIN 35), some of them are connected to the ground (PIN 19) and (PIN

36-44) these pins must be connected for proper antenna performance. Some pins are used for a sensor interface, with analog input.

There are two sources for the input power to the transceiver, it could be from a battery (2-3.3 V) which could be connected into (VDD-BATT PIN20), or a (3.3–3.7 V) voltage source which could be connected into (VDD-IN PIN 21).

To achieve the lowest power consumption (4uA) in sleep mode connect a weak pull down (100K resistor to GND) on the pin 22 - DMA-TX.

There is a dedicated pin for forcing the transceiver to wake up from the sleep mode which is (FORCE_AWAKE PIN 9).

The pins that are used to connect the transceiver with the PIC microcontroller are the UART-RX and the UART-TX (PINs 12 & 13). The UART-TX sends data from the module to the PIC, and vice versa for UART-RX.

This transceiver has an on-board ceramic antenna that operates at the 2.4GHz for transmission and reception, and other U.FL connector for optional external antenna.

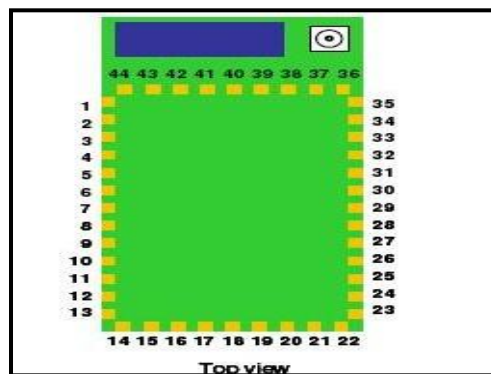


Figure 3.10 pins of the wifily GSX

3.3.7 Receiver to microcontroller:

After the vehicles receive the data from others, this data must be known to the driver. So the microcontroller is applied to transfer the surrounding state to an LCD in each vehicle, then the alert will be readable.

PIC18F4550 is implemented to do the task.

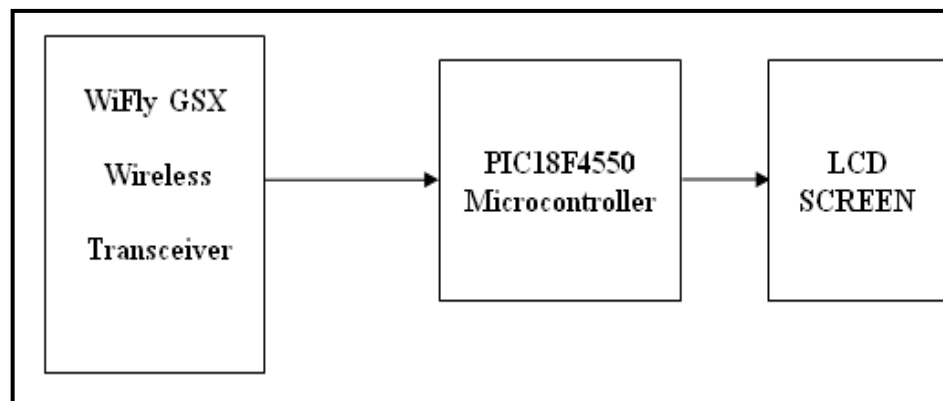


Figure 3.11 Interfacing between WiFly GSX and LCD screen

3.3.8 LCD screen

LCD screen is connected easily to the microcontroller via connecting the related pins to the microcontroller as describes below, then programming the LCD using MPLAB program.

Table 3.1 Interface pins of LCD

Pin NO.	Symbol	Level	Description
1	VSS	0V	Ground
2	VDD	5.0V	Supply voltage for logic
3	VO	---	Input voltage for LCD
4	RS	H/L	H : Data signal, L : Instruction signal
5	R/W	H/L	H : Read mode, L : Write mode
6	E	H, H → L	Enable signal for KS0076
7	DB0	H/L	Data bit 0
8	DB1	H/L	Data bit 1
9	DB2	H/L	Data bit 2
10	DB3	H/L	Data bit 3
11	DB4	H/L	Data bit 4
12	DB5	H/L	Data bit 5
13	DB6	H/L	Data bit 6
14	DB7	H/L	Data bit 7
15	NC	---	No connection
16	NC	---	No connection

3.4 Detailed System Design and Flow Control:

3.4.1 Detailed System Design:

The design that implemented in this project can be shown in figure 3.12

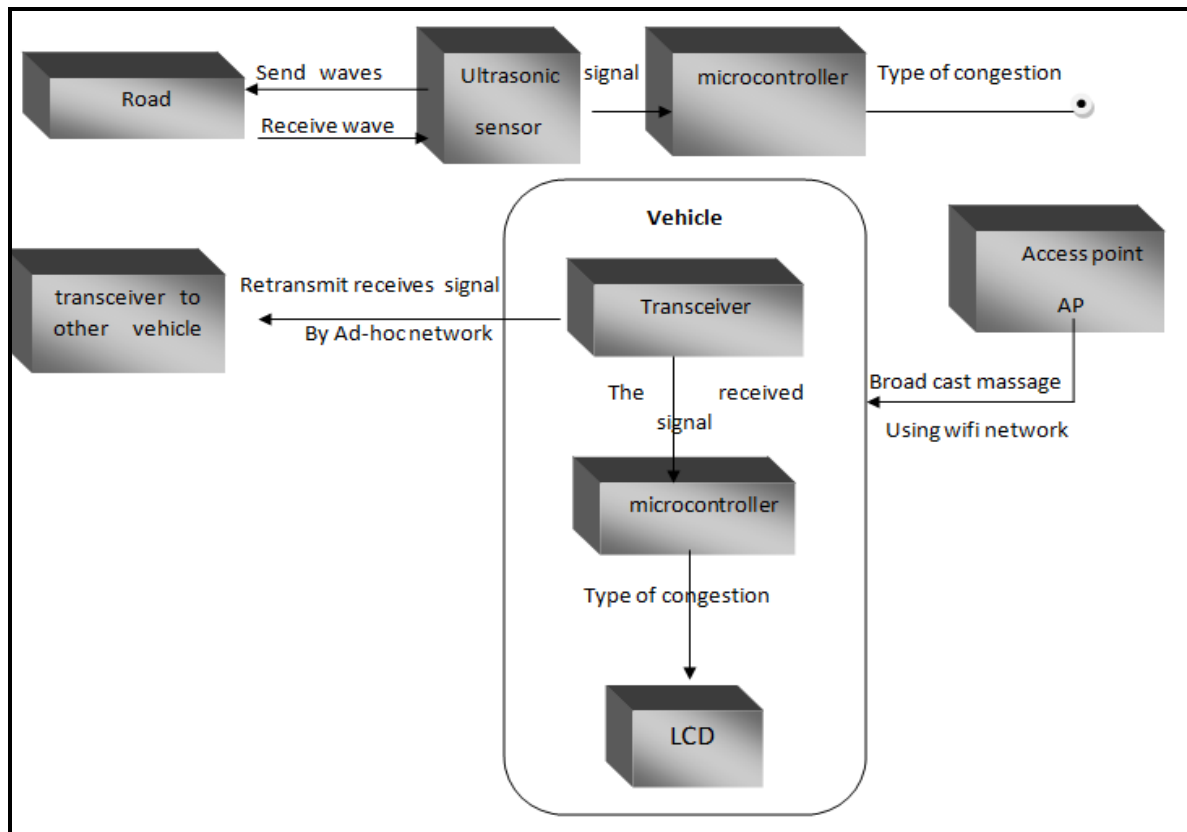


Figure3.12 System Design Block Diagram

Ultrasonic sensor which mounted on a roadside sends a wave to the road, and then receives it with changes if there is a vehicle on road, this wave treated as signal consists of sequence pulses, passes to the PIC microcontroller.

The PIC depending on feedback from sensor, counts the number of vehicles, and then determines the type of congestion which is high, medium or low; also it determines which road has congestion.

According to the type of congestion if it is high or medium the access point transmits a broadcast message using Wi-Fi network to nearest vehicles on road.

The near vehicles receive the signal from access point on transceiver, and retransmit the results to other vehicles near it, using Ad-hoc network.

Each transceiver on vehicle connects to other PIC microcontroller to appear the results on a LCD for driver.

The details shown in figure 3.13.

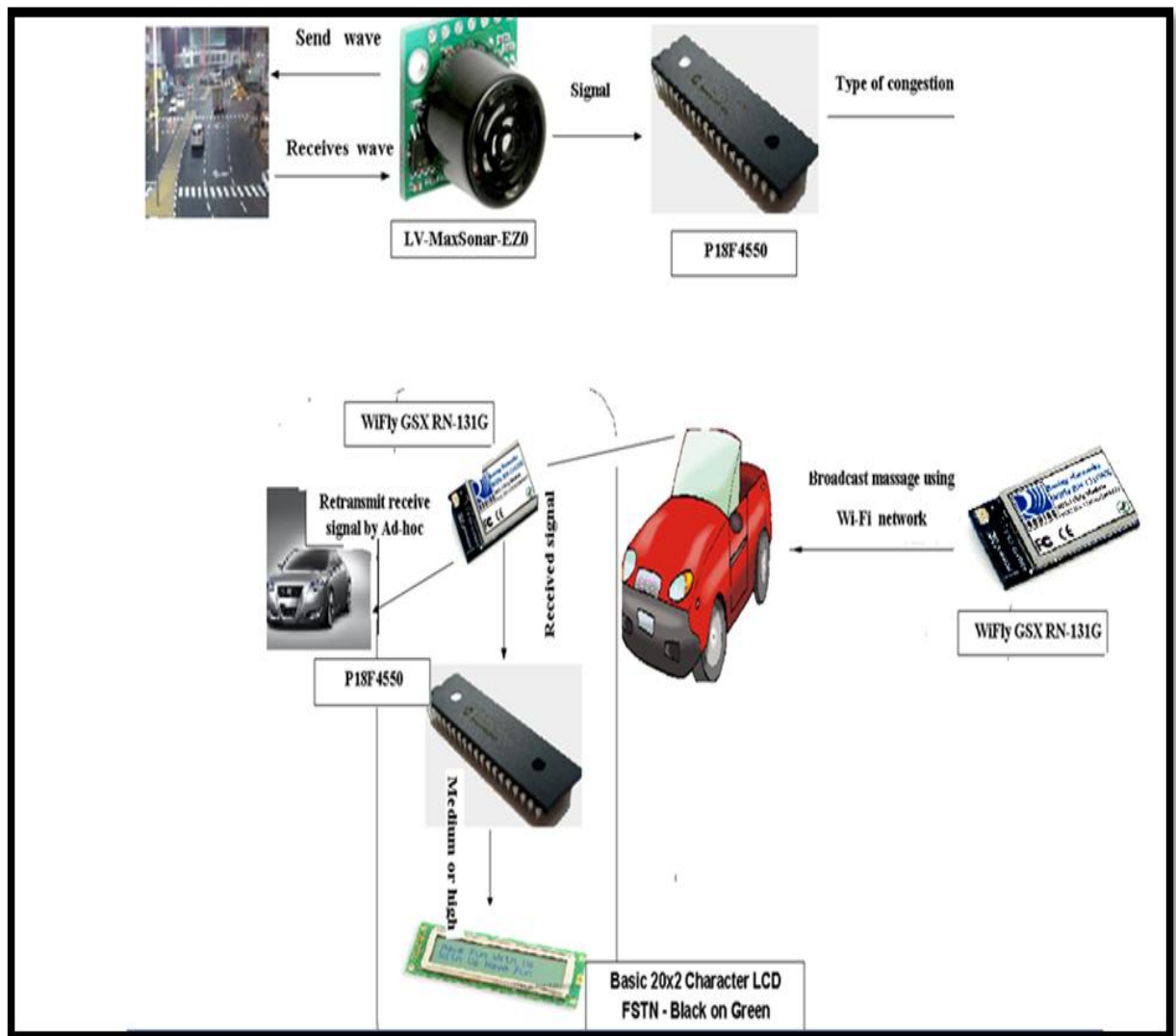


Figure 3.13 Detailed System Design

3.4.2 System Flow Control

Figure 3.14 shows the scenario of the system works.

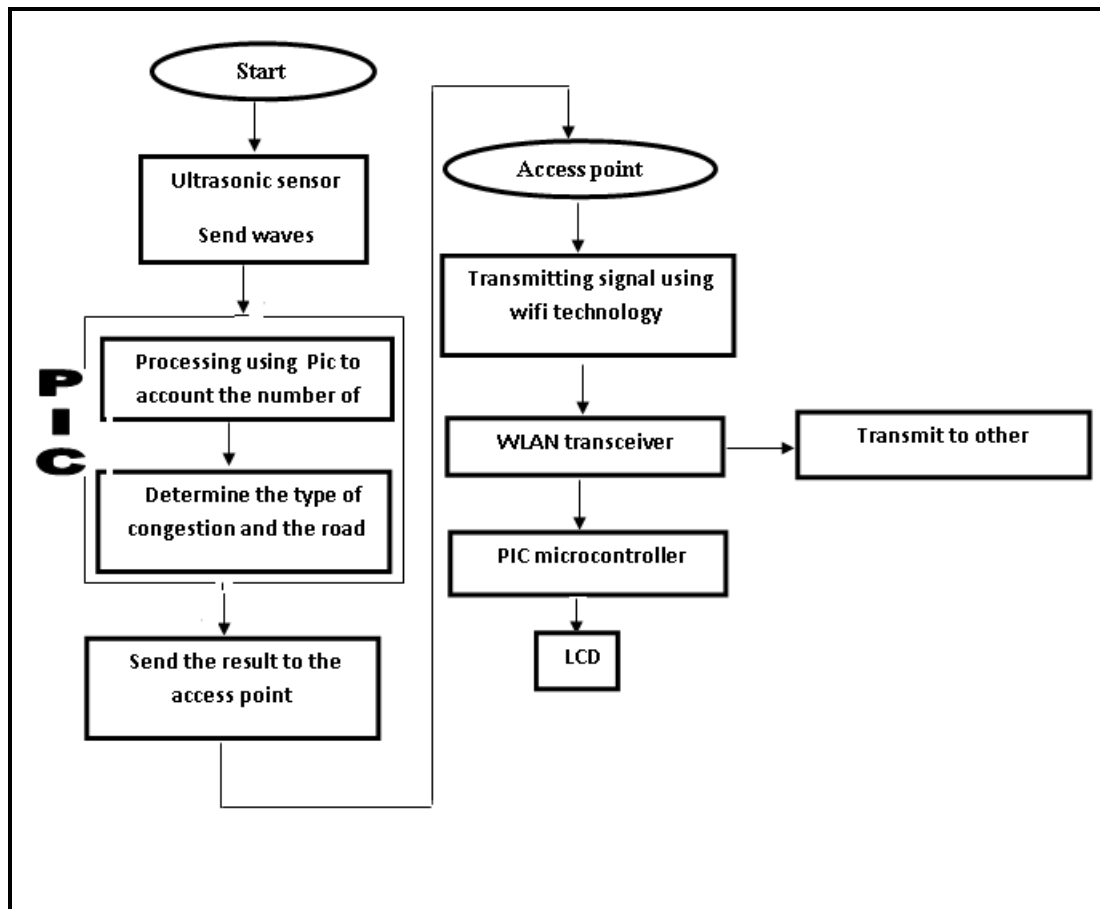


Figure 3.14 Scenario of the System work

4

Chapter Four

Hardware System Design

4.1 Introduction

4.2 Interfacing and Connections between the Components

Chapter Four

Hardware System Design

4.1 Introduction

In this project, the hardware design will be divided into two parts, the road part and the vehicle part. The road part includes the components that senses the road state and broadcasts it, which are: The ultrasonic sensor, the PIC microcontroller, and wireless transceiver. The vehicle part includes the wireless transceiver, the PIC microcontrollers (differ from that of the road part), and the LCD. A division must be made between these parts in order to build the system.

In figure 4.1, a general system interfacing block diagram is shown, it represents the interfacing of the whole system. In the coming sections, detailed interfacing techniques and requirements will be discussed.

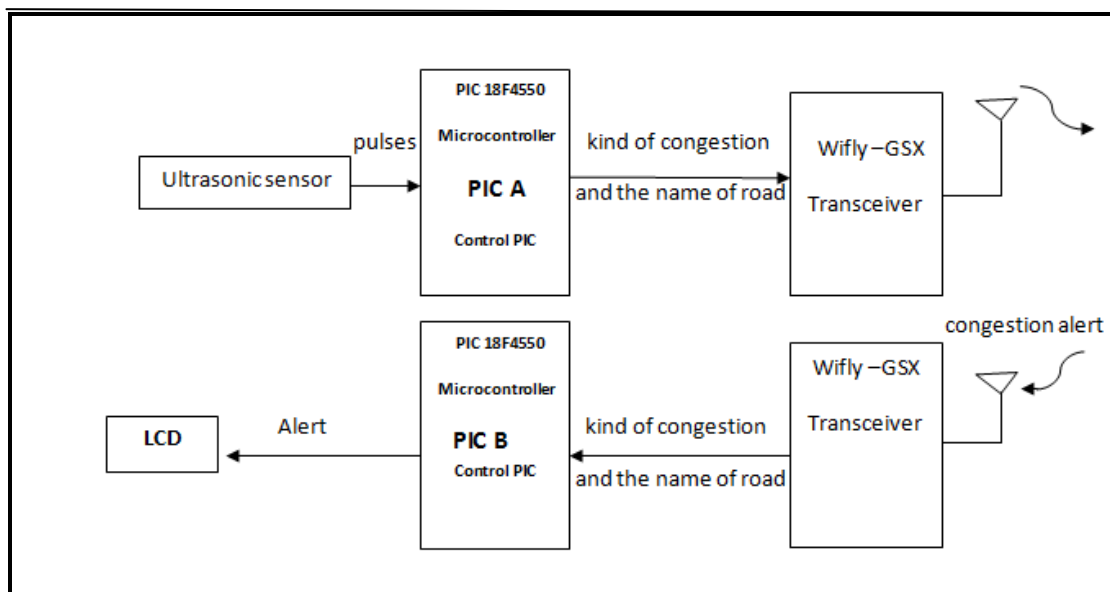


Figure 4.1 General System Interfacing Block Diagram.

4.2 Interfacing and Connections between the components

In this section, the interfacing techniques, requirements, and ideas to connect the electrical parts together, the wireless transceiver, the ultrasonic sensor, LCD screen, and the battery all with the PIC microcontroller, will be discussed.

4.2.1 PIC Requirements

This system requires two WiFly GSX and two PICs, the first PIC is connected to the ultrasonic sensor and is responsible for sending the results to the broadcasting WiFly GSX. This PIC is called the **PIC A** figure (4.2). The other PIC is connected to the Ad hoc WiFly GSX and is responsible for sending the result to the LCD. This PIC is called the **PIC B** figure (4.3).

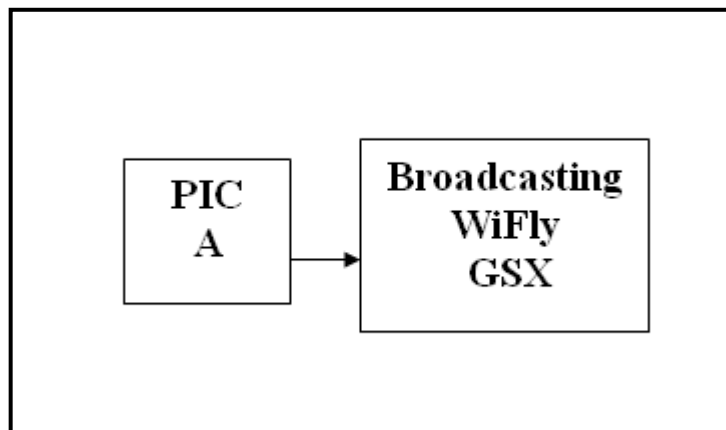


Figure 4.2: PICA (connect sensor with the WiFly GSX).

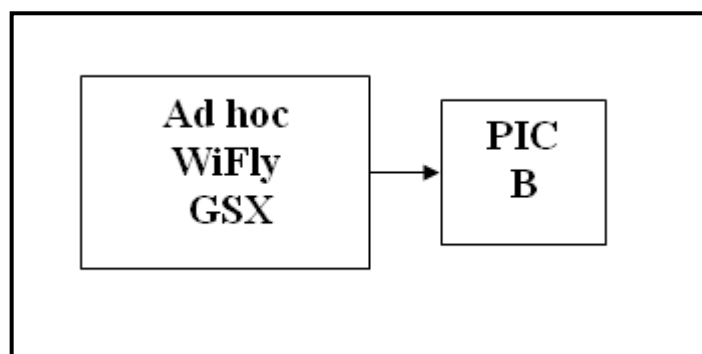


Figure 4.3: PIC B (connect the Ad hoc WiFly GSX with LCD).

4.2.2 Ultrasonic Sensor Interfacing

Ultrasonic sensor is very important component in this project, since it used to determine the kind of congestion on the road. Use LV-MaxSonar –EZ0, the suitable beam width angle; for a good result, using 36 degree which give 3.5 inch diameter of target.

Serial data is sent at the end of the measurement cycle, so the analog voltage is set right after the target is detected.

The common pins we use in this project, pin GND is used to connect circuit to ground, +5V to connect to +5V supply ,AN output used to connect the ultrasonic sensor to analog to digital pins on our microcontroller which is pin (RA0/AN0) . The pin TX use to connect with PC , PW pin since it not easy to connect like TX or AN pins so connect it with oscilloscope or can be timed with microcontroller .

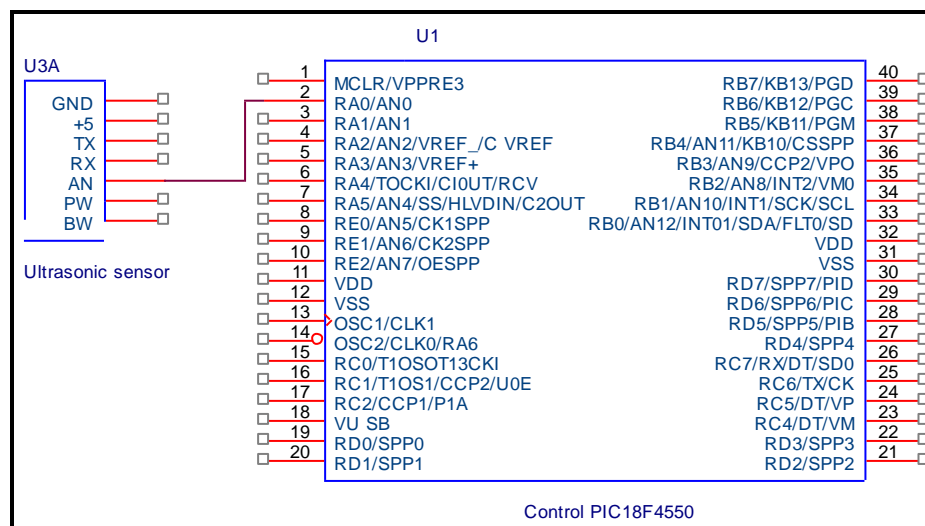


Figure 4.4: Interfacing between Ultrasonic sensor and PIC microcontroller.

4.2.3 Interfacing WiFly GSX Wireless Transceiver

In order to connect the wireless transceiver to the PIC microcontroller, a brief study was made about the pins for both the PIC and the WiFly GSX wireless transceiver, and the idea was that serial communication can be applied between them. The PIC18F4550 supports serial communication through the built in USART, two pins are available, one serial data input (RC7/RX), and the other is the serial data

output (RC6/TX). The WiFly GSX also supports serial communication through the UART_TX (output) and UART_RX (input) pins.

The connection to the sensor PIC is done through the UART_RX pin connected with the RC7/RX pin, because the sensor result is an output from the PIC to the WiFly GSX, and connection to the LCD PIC is done through the UART_TX pin connected with the RC6/TX pin, in order to send information to the PIC.

Due the difference in the voltage levels between the PIC Microcontroller (5V) and the WiFly GSX transceiver (3.3V), connecting the USART_TX of the PIC with the UART_RX of the transceiver is done through three 10KΩ resistors connected serially, and the wire of the UART_RX is taken from a 20KΩ. The values of these resistors were found using the voltage divider rule.

The WiFly GSX baud rate can be configured to be from 2400 bps, to 1Mbps which is suitable to the project requirements.

Three Wifly GSX transceiver in this project need to connect with pic18f-4550, one of them send broadcast message while other sent ad-hoc message between vehicles, so their connection differ with PIC depend on the action of the Wifly GSX.

First Wifly GSX receives from RC6/TX pin on PIC to its USART_RX pin, figure (4.5). The other receives from the Wifly GSX at the USART-RX pin, then the transceiver send from USART_TX pin to the RC7/RX on the PIC, figure (4.6).

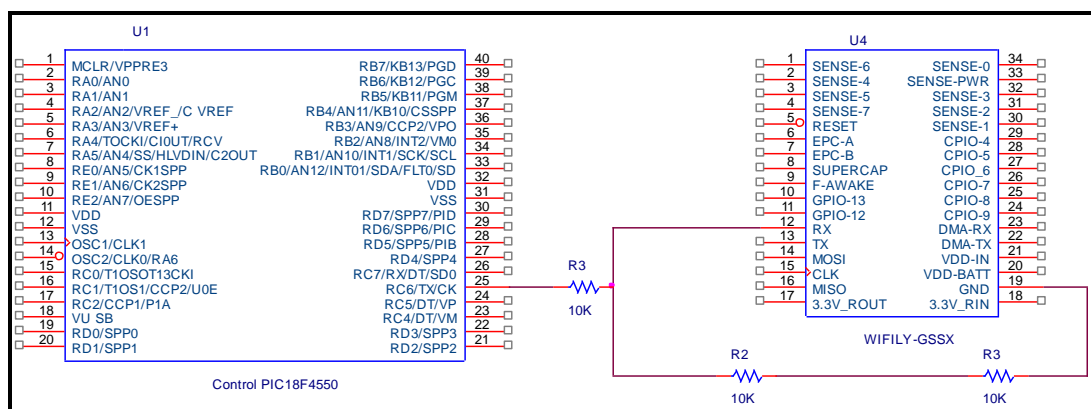


Figure 4.5: Interfacing First WiFly GSX Wireless Transceiver with PIC

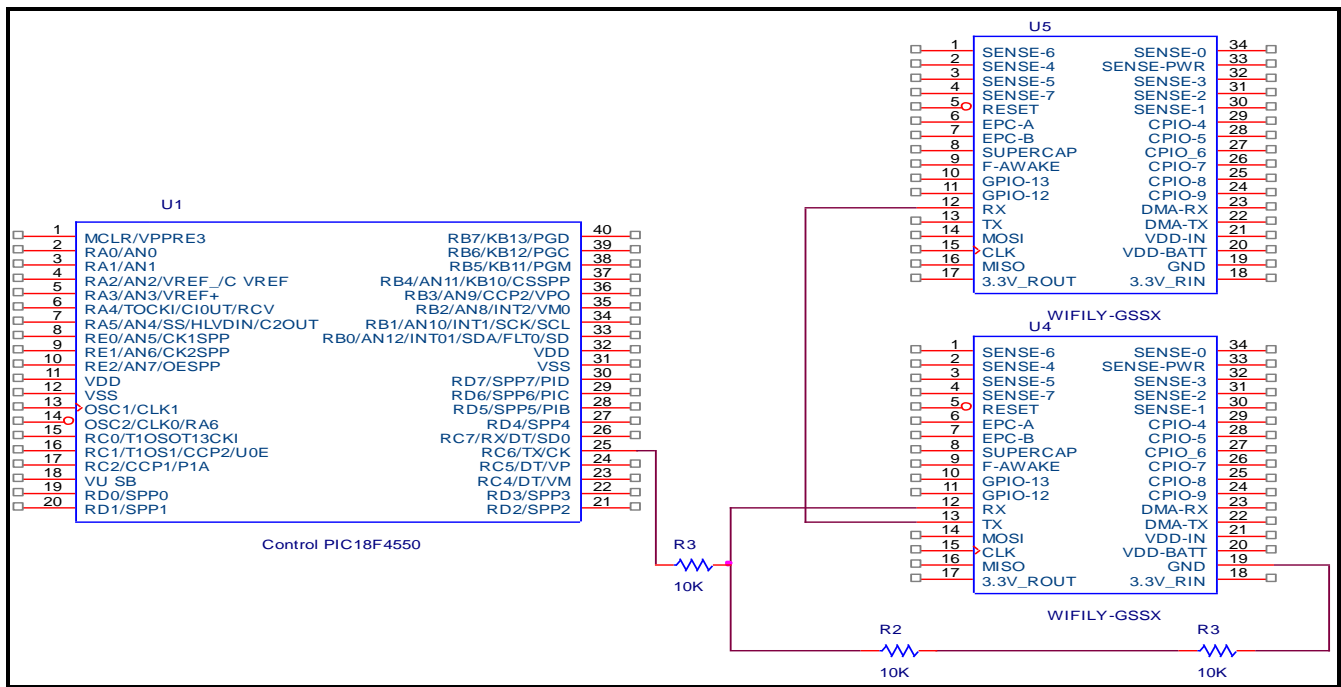


Figure 4:6 Interfacing two WiFly GSX Wireless Transceiver with PIC

4.2.4 LCD Interfacing

The LCD screen is a very important component in this system. It is responsible for making the results of the project readable to the driver.

The Basic 20×2 character LCD FSTN –Black on Green which is the one used in this system has 16 pins.

Connecting the PIC to the LCD screen is done through the serial data line from the LCD (D11-D14) to the LCD PIC through the PORTB lower input pins, figure (4.7)

The register selection pin (RS) is connected to the LCD PIC through out the pin RB4, the control enable pin (E) from the LCD is connected to pin RB5 of the PIC, Vo pin is connected to ground like R/W pin, and the remaining pins are not connected.

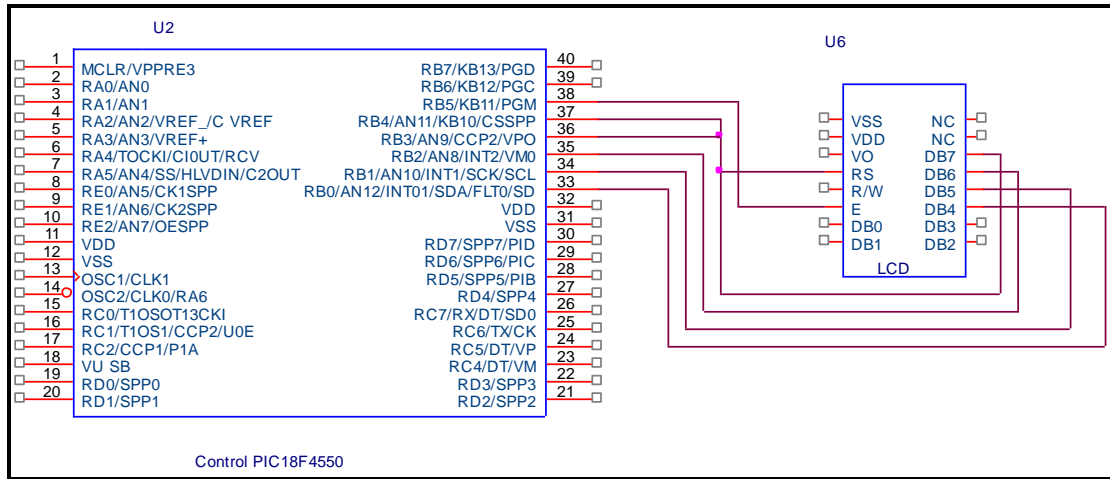


Figure 4.7: Interfacing between LCD and PIC

On each vehicle we have the following component interfacing: LCD, PIC and Wifly GSX. Figure (4.8)

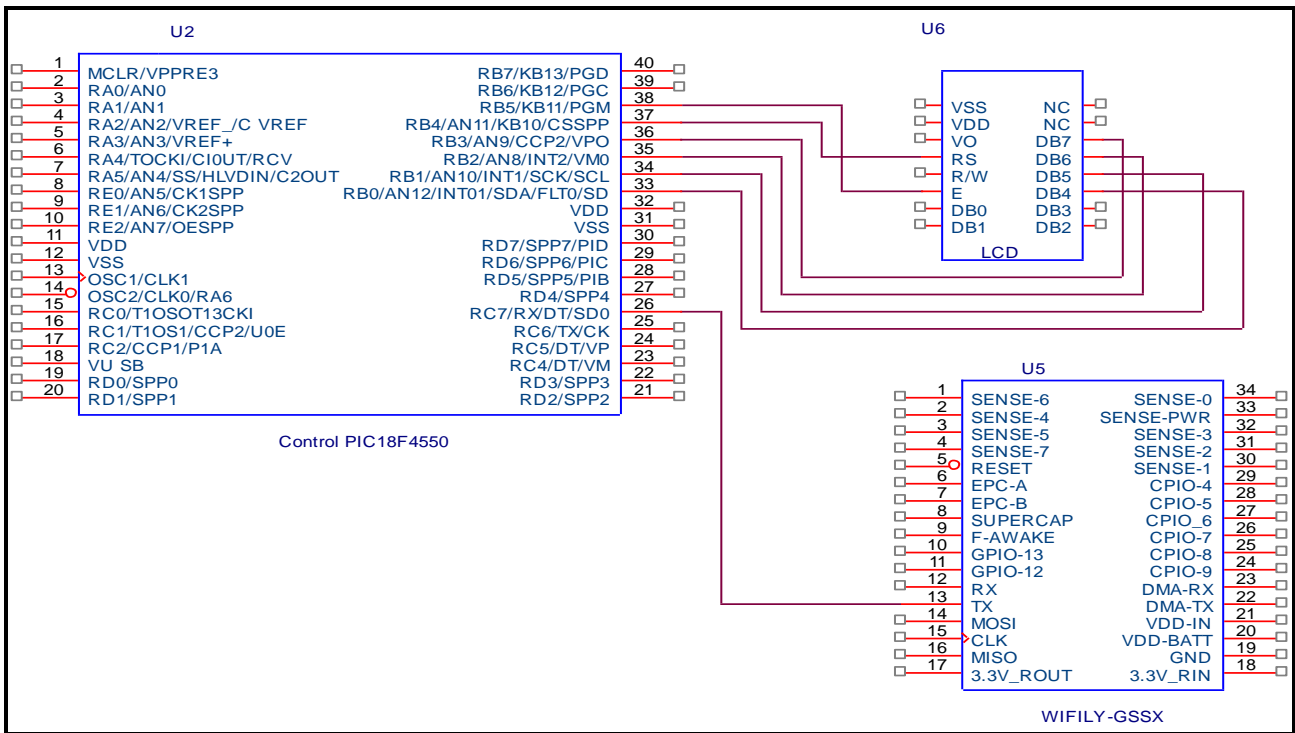


Figure 4.8: The Interfacing between PIC, Wifly-GSX and LCD on Vehicle

4.2.5 Battery

The battery is the portable power source for the system; it feeds all the system components with power.

In order to connect some components with the battery, regulators are needed to control the power entering the components, to avoid any problems of overloading the components with power, and burning them.

Figure 4.9 presents the connection of the electrical components with the battery.

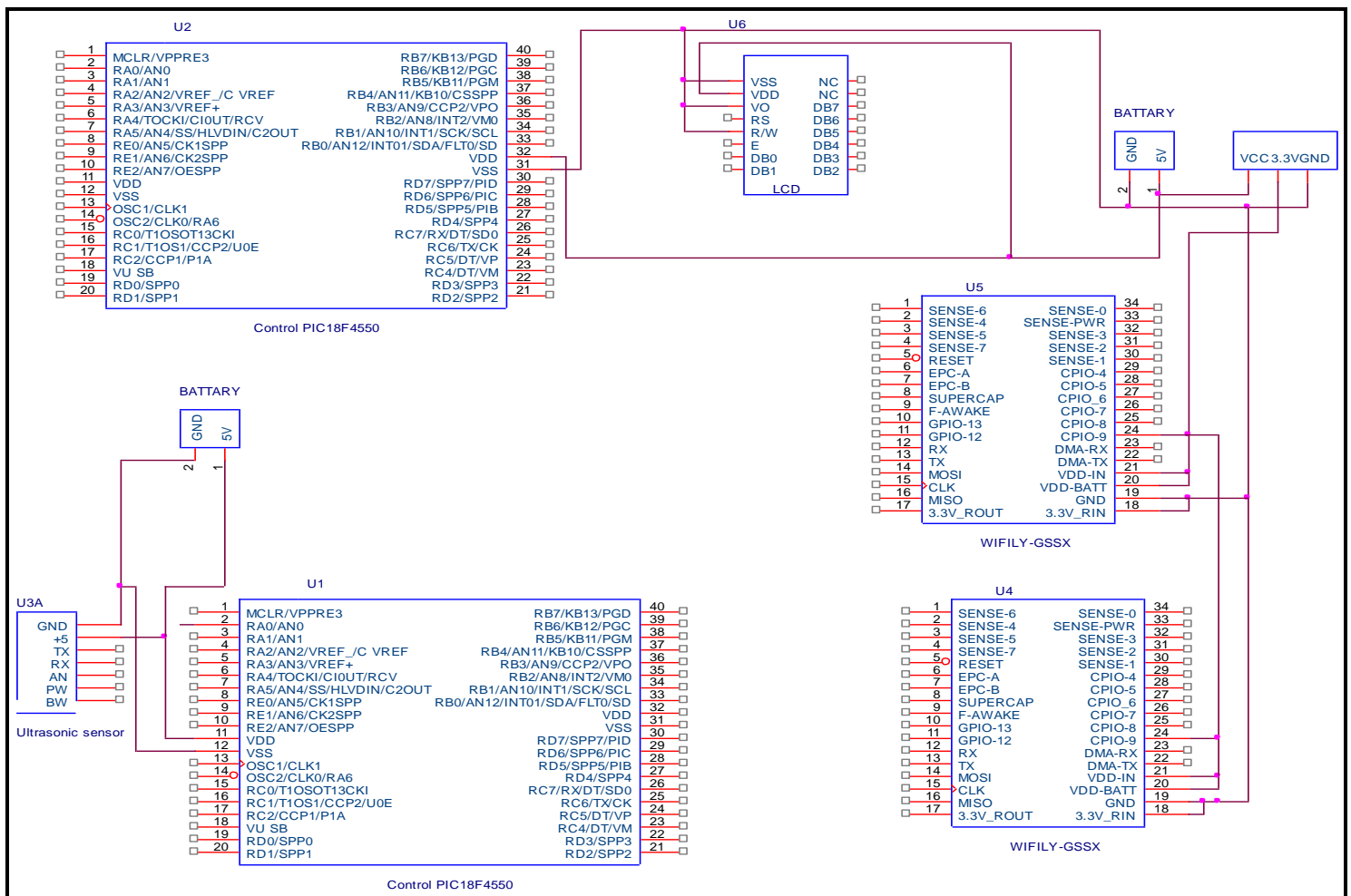


Figure 4.9 Battery Interfacing.

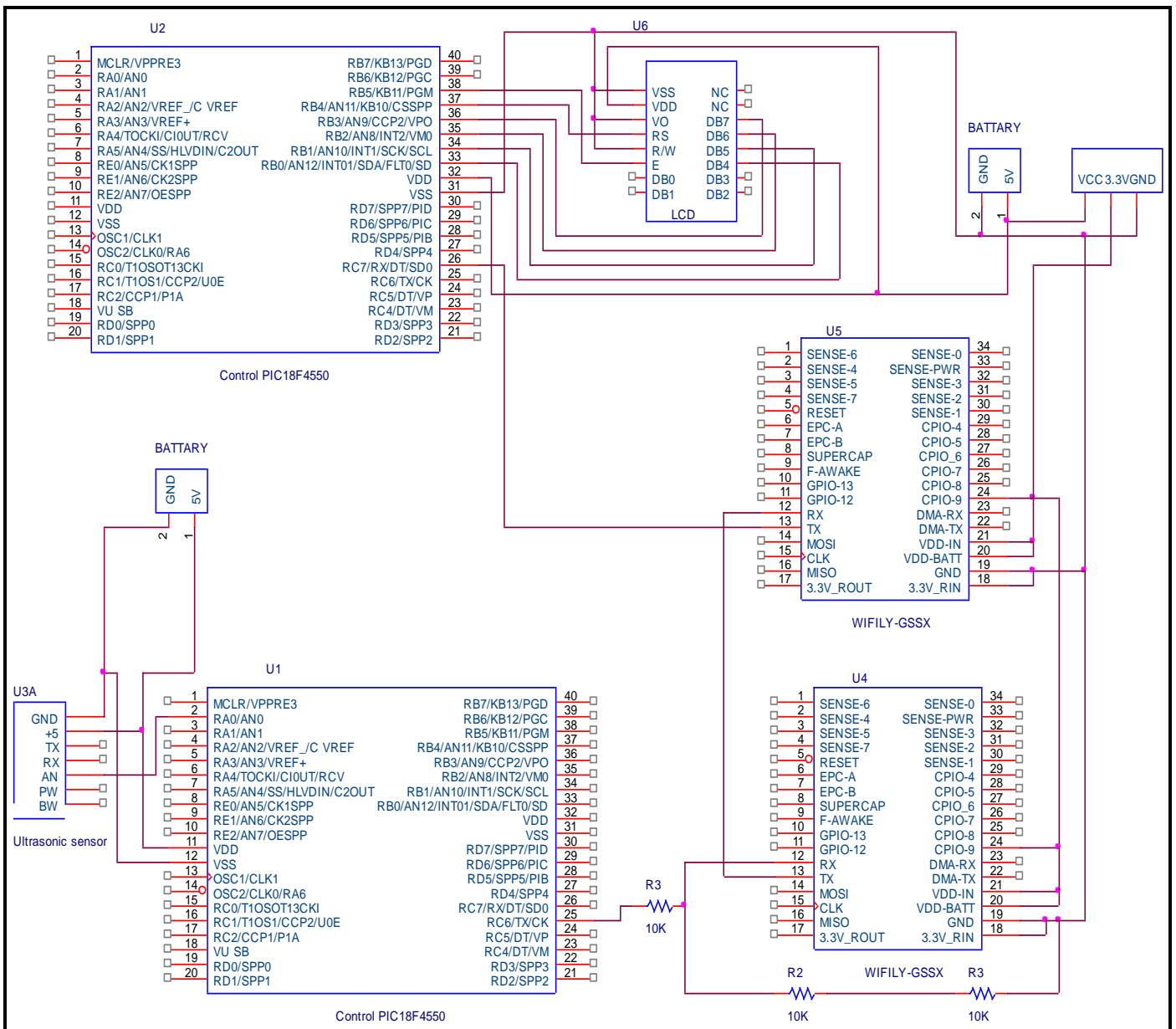


Figure 4.10: Interfacing Electrical Component

Table 4.1: Figures key

U1	PIC A
U2	PIC B
U3A	Ultrasonic Sensor
U4	Wifly GSX 1
U5	Wifly GSX 2
U6	LCD

5

Chapter Five **Software Design Implementation**

5.1 Software System Design

5.2 Detailed Design Description

5.3 System Configuration

Chapter Five

Software Design Implementation

5.1 Software System Design

In order to complete this project, software implementation needs to be done. All the important task such as programming PIC and designing circuitry required several software installations. For software implementation, Micro C and MPLAB IDE are used to program microcontroller in C language. Besides, Tera term is used to program the Wifly transceiver.

5.2 Detailed Design Description

In this section a detailed description about each software component is presented. It discusses the importance of each component and its rule.

5.2.1 Sensor and PIC programming

Microcontroller acts as brain of the whole system. It receives the desired data from the sensor that interface with it .An algorithm is developed to make the microcontroller able to read the input and respond accordingly. Therefore, the algorithm is established and represented by a flowchart in chapter three. These flowcharts are then translated into C language and compiled using Micro C, the PIC18F4550 software development tool. The flow chart is shown in figure 5.1.

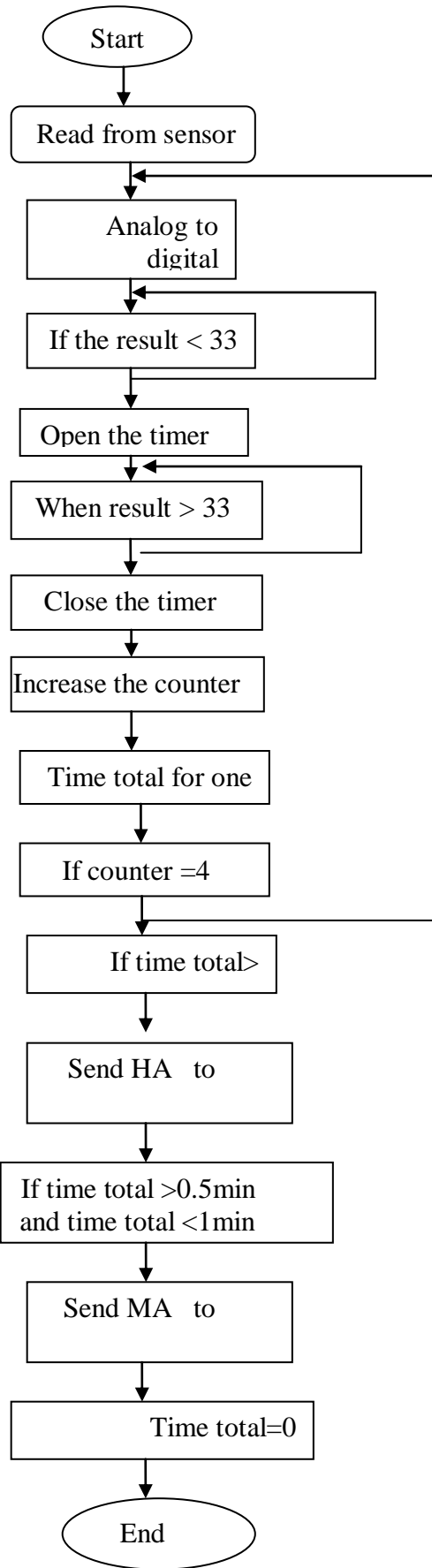


Figure 5.1: Flow chart of sensor and PIC

5.2.1.1 MPLAB IDE

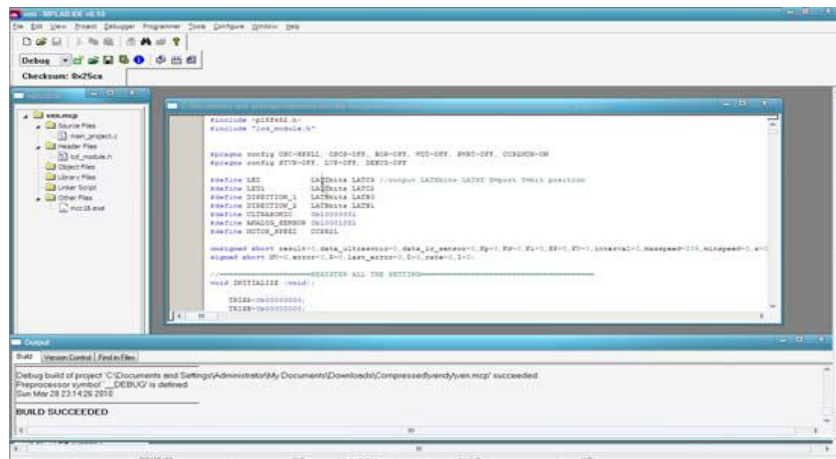


Figure 5.2: MPLAB IDE Editor

Figures 5.2 show the MPLAB IDE editor .MPLAB IDE is a Windows based Integrated Development Environment for microchip Technology Incorporated PIC microcontroller (MCU) and dsPIC digital signal controller (DSC) families.

In the MPLAB IDE, we can:

1. Create source code using the built-in editor.
2. Assemble, compile and link source code using various language tools.

An assembler, linker and librarian come with MPLAB IDE. C compilers are available from Microchip and other third party vendors.

3. Debug the executable logic by watching program flow with a simulator, such as MPLAB SIM, or in real time with an emulator, such as MPLAB IDE. Third party emulators that work with MPLAB IDE are also available.
4. Make timing measurements.
5. View variables in Watch windows.
6. Program firmware into devices with programmers such as PICKit.

5.2.1.2 Programmer

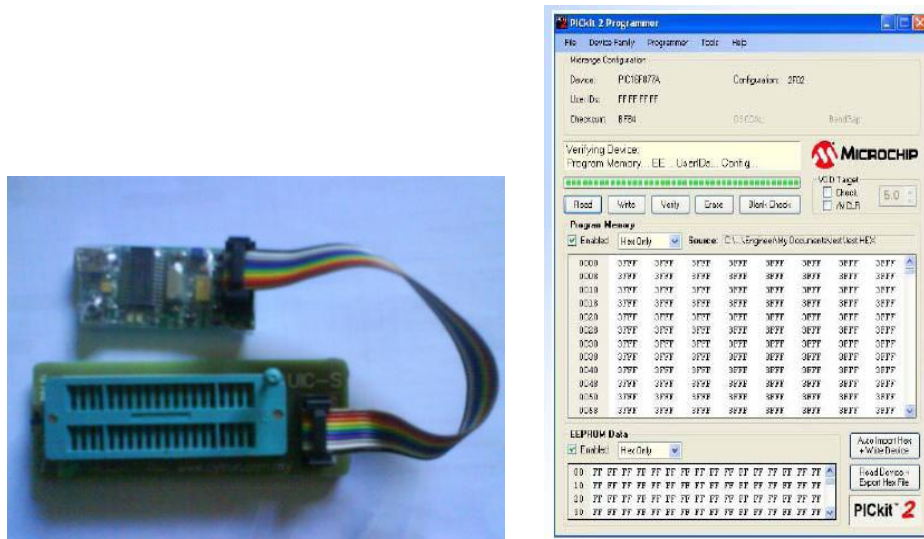


Figure 5.3: ICSP Programmer and PICKit software

PICKit is software used to load the program to the microcontroller. The circuit is interface with ICSP programmer first before loading the hex file into the PIC18F452. Figure 5.3 shows PICKit editor and the ICSP programmer. The important icon such as open, write and verified has been explained in detail as below:

1. After connecting the circuit with the ICSP programmer, the icon ‘Verified’ was pressed to detect the PIC.
2. Then the hex file was attached from icon ‘open’
3. To load the file into the microcontroller, button ‘Write’ was selected. Successful message will appear if the loading was finish and success.

5.2.1.3 Sensor parameters

5.2.1.3.1 Software description of sensor interfaces:

Ultrasonic sensor which describes before, send analog signal to PIC then by using code in C language convert the analog to digital and obtain the result which determine the distance below the sensor, if it lower than determine distance ,which involve to distance from ground to sensor, the code know that there is a vehicle under the sensor.

While the vehicle under the sensor the timer in code increases. If the vehicle moved the sensor returns to determined distance. So the counter increases.

Depending on the counter and the timer the code determine the congestion, to be high or low or medium.

Since the congestion is high the code sent L to usart pin, if it is medium M sent to usart pin. Also the name of road is sent to usart pin. Where usart pin is connected to transceiver.

The library needed in MPLAB for the above description code:

1- Analog to digital converter library can be found on ADC.h

Following figure show the main function used in ADC

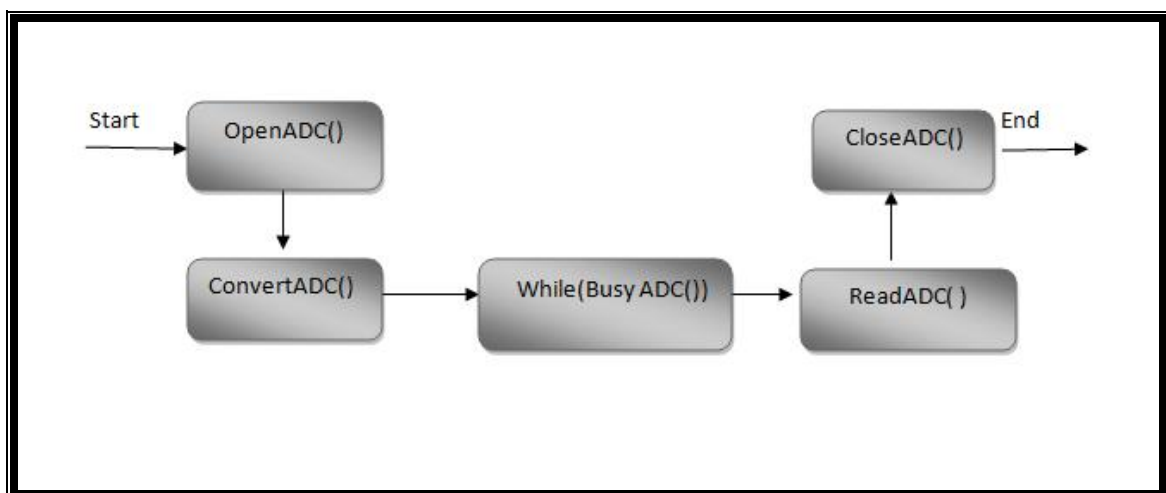


Figure 5.4: Step A/D Conversions

OpenADC

ADC_RIGHT_JUST → we have two register store value of conversion so we can determine if read left justified or else. (Result in Least Significant bits)

ADC_2_TAD → 2 Tad

ADC_CH0 → Channel 0 (AN0)

ADC_INT_OFF → Interrupts disabled

To program the ADC to read the range, it is important to determine the Least Significant value of the ADC, which is the value of voltage that is detected by the module.

$$\text{LSB} = \text{Input Voltage} / 2^{\text{ADC-resolution}}. \text{ADC-resolution} = 10 \text{ bit}$$

$$\text{LSB} = 5\text{V} / 1024 = 4.8828125\text{mV}$$

Then to calculate the value of the ADC that represents the range in inches the sensors sensitivity is important. So, the distance of the obstacle formula is:

$$\text{Value of ADC (integer)} = \text{Distance (inches)} \times \text{Sensitivity} / \text{LSB};$$

Sensors sensitivity: A supply of 5V yields ~9.8mV/in.

For example if there is no vehicle under the sensor then the distance between the sensor and the ground it is 30cm. the value of the ADC will be:

$$\text{ADC} = ((42\text{cm} / 2.5)(\text{inches}) \times 9.8) / 4.8828125 = 33.7.$$

2. USART library:

Main step for USART using C18 library as shown in figure 5.5

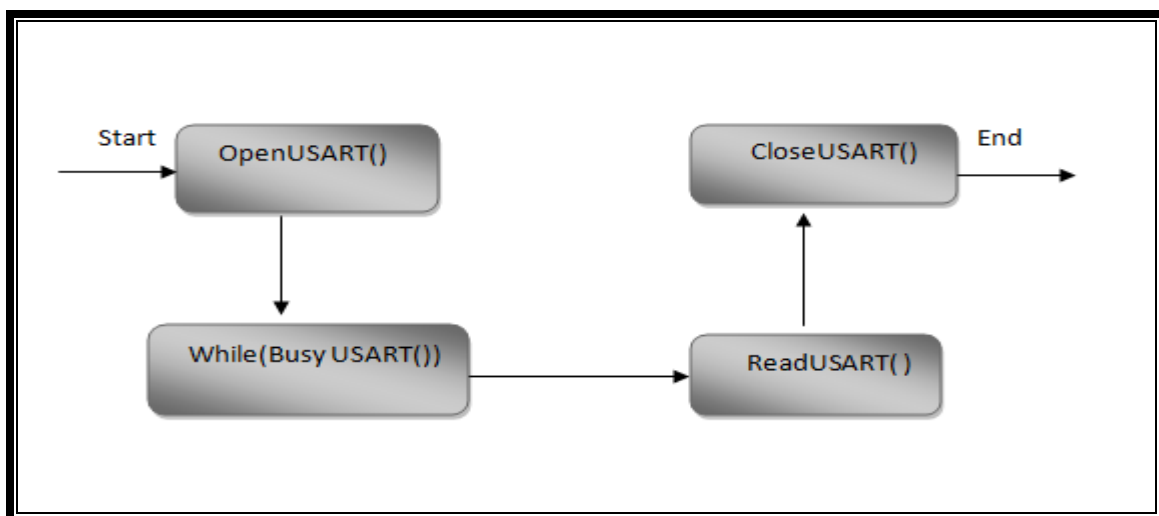


Figure 5.5 USART Library

Open USART include:

USART_TX_INT_OFF: Transmit interrupt OFF.

USART_RX_INT_OFF: Receive interrupt OFF.

USART_ASYNC_MODE: Asynchronous Mode.

USART_EIGHT_BIT: 8-bit transmit/receive.

USART_BRGH_HIGH: High baud rate.

12: value calculated for register spbrg.

3. Timer library:

OpenTimer1 include:

TIMER_INT_ON: Interrupt enable.

T1_16BIT_RW: Set timer1 as two 16-bit registers.

T1_SOURCE_INT: Choose internal clock source (TOSC).

T1_PS_1_8: Prescale Value: 1:8.

Also we use the interrupt in MPLAB:

The PIC18F2455/2550/4455/4550 devices have multiple interrupt sources and an interrupt priority

Feature that allows each interrupt source to be assigned a high priority level or a low priority level. The high priority interrupt vector is at 000008h and the low priority interrupt vector is at 000018h. High priority interrupt events will interrupt any low priority interrupts that may be in progress.

There are ten registers which are used to control interrupt operation. These registers are:

- RCON
- INTCON
- INTCON2
- INTCON3
- PIR1, PIR2
- PIE1, PIE2
- IPR1, IPR2

It is recommended that the Microchip header files supplied with MPLAB® IDE be used for the symbolic bit names in these registers. This allows the assembler/compiler to automatically take vehicle of the placement of these bits within the specified register. Each interrupt source has three bits to control its operation. The functions of these bits are:

- Flag bit to indicate that an interrupt event occurred.
- Enable bit that allows program execution to branch to the interrupt vector address when the flag bit is set.
- Priority bit to select high priority or low priority.

Simple calculations calculate to get one second for timer1 by using interrupt:

The timer increment frequency = $(F_{osc}/4) * (1/8) = (48\text{MHz}/4) * (1/8) = 1.5\text{MHz}$.

This time, we want to generate one interrupt every 20ms. Then, the amount of the timer register value for 20 ms = $(20 * e^{-3}) * (1.5 * e^6) = 30000$.

The timer0 starting value = $0\text{xFFFF} - 30000 = 65535 - 30000 = 35535 = 0\text{x8ACF}$.

Now we know every 20ms, one interrupt can be generated.

Then we can count the interrupts, if there have 50 interrupts been generated, we can get 1 second.

5.2.2 WiFly GSX Transceiver Programming

Upon power up, the transceiver's behavior can be described in two scenarios. In the first scenario which shown in figure 5.6, when the transceiver turned on it will scan for a wireless networks, the transceiver has 13 channels at which scan for WLAN's occurs. After the scan is completed, the transceiver will try to associate with the specified network. If a connection is established, the transceiver which connected with **PIC A** will send the alert congestion to the stored IP address of the transceiver, after this the connection will close.

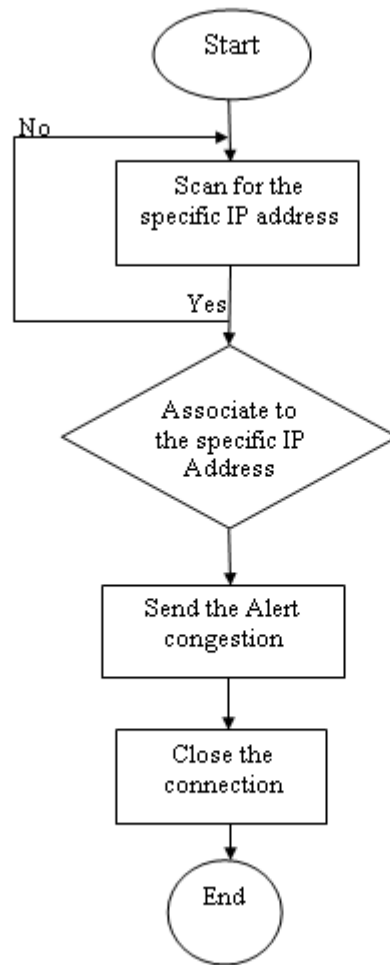


Figure 5.6: Flowshart of the first transceiver

In the second scenario the transceiver which connected to **PIC B** will receive the alert congestion and send it serially to **PIC B**, which then programmed it to view the alert congestion on the LCD screen this scenario is shown in figure 5.7.

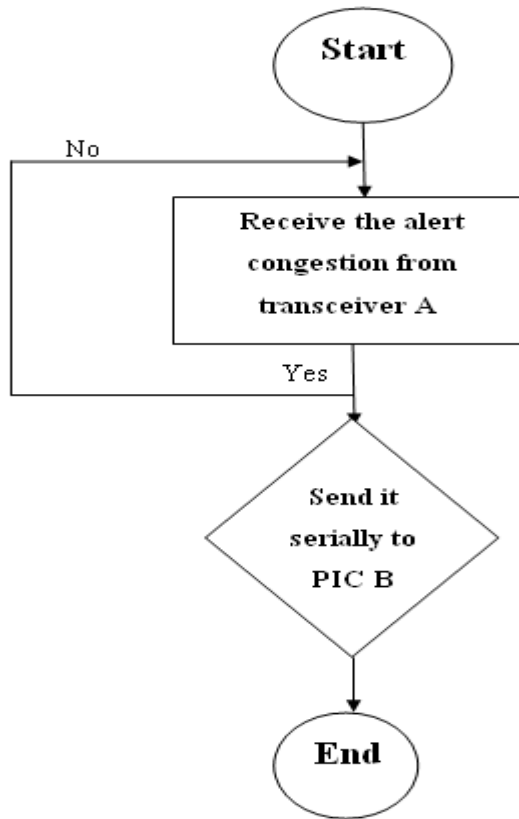


Figure 5.7: Flowshart of the second transceiver

At LCD when MA is receives on **PIC B** , then medium congestion and road A is appears on the LCD, also when **PIC B** receives data which HA the high congestion and road A appears on LCD as shown in table 5.1.

Table 5.1: Massage on LCD

Message receives on PICB	On LCD
MA	Medium congestion /Road A
HA	High congestion/ Road B

For the transceiver to function as required, it must be programmed by typing a number of commands in a terminal program using Telnet service, for example

TeraTerm software. When the transceiver is programmed and the configurations are set, it should be able to access Wi-Fi networks and transfer or receive data.

In this project connecting with the WiFly transceiver at the first time was done wirelessly by connecting with the transceiver in the ad-hoc mode. Enabling ad-hoc mode via hardware was done by connecting PIO9 pin to the 3.3V line at power up.

When the module powers up in ad-hoc mode the WiFly module creates an ad-hoc network with the following information:

SSID: WiFly-GSX-XX where XX are the final two bytes of the transceiver's MAC address.

IP address: 169.254.1.1

Starting a Telnet session using TeraTerm to set up a connection with the module was done as follows:

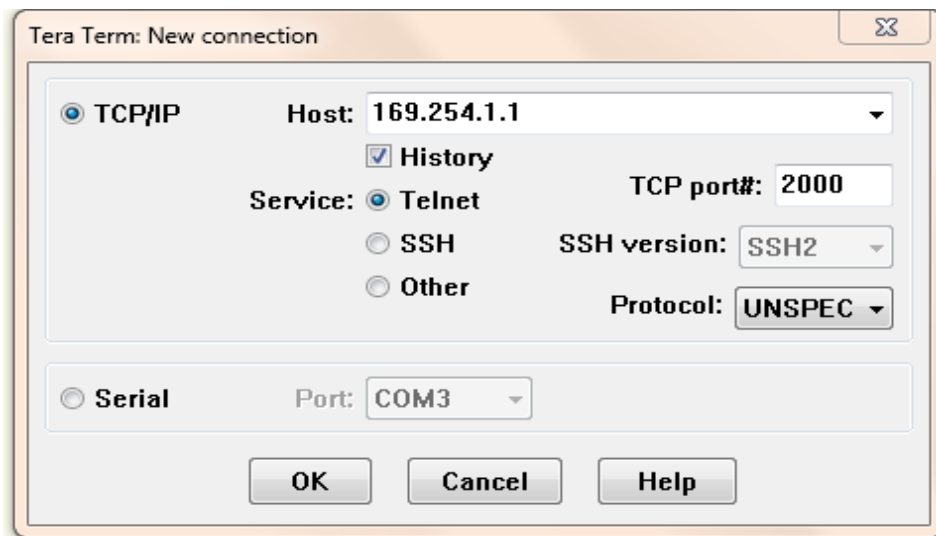


Figure 5.8 Starting a Telnet session using TeraTerm in adhoc

As shown in figure 5.9 the module replied with the string *HELLO* indicating that the connection has been established.

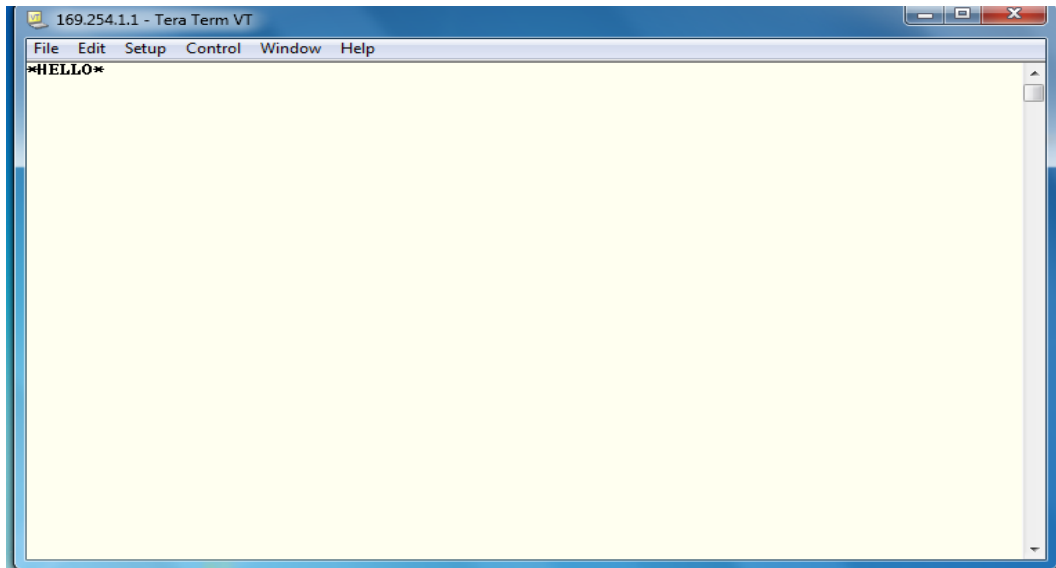


Figure 5.9 A Telnet session using TeraTerm

In terminal window, the WiFly GSX module was entered into command mode by typing \$\$\$. A CMD reply confirmed that the transceiver was running in the command mode.

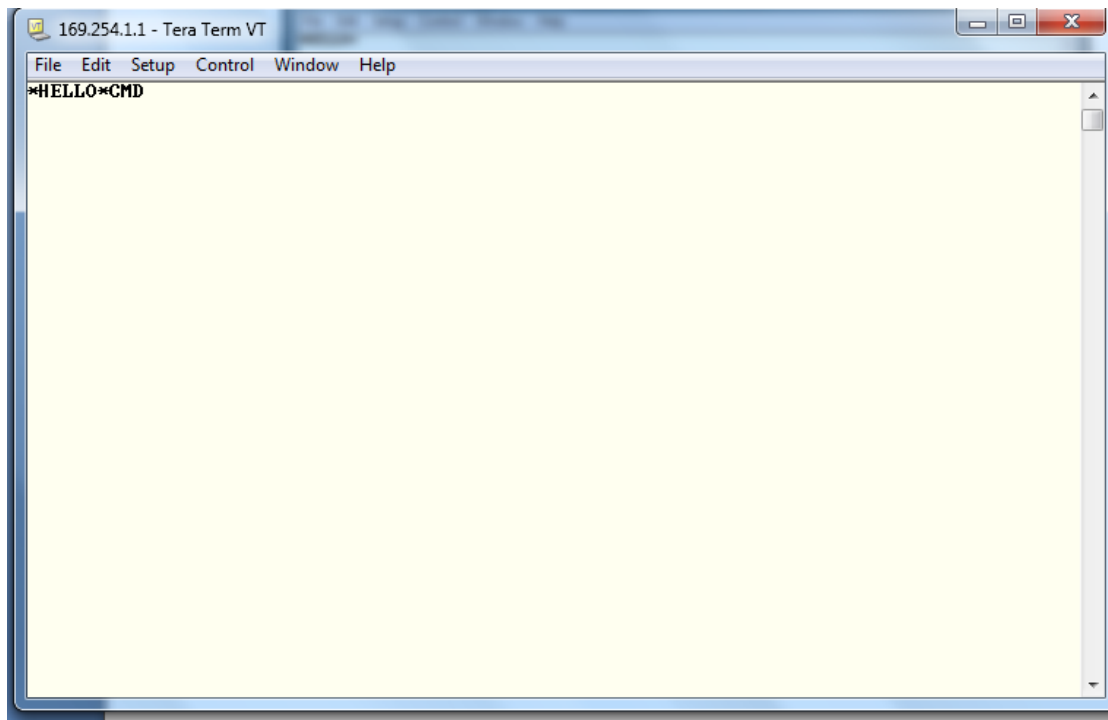


Figure 5.10 WiFly in Command Mode

From command mode we configure the module for adhoc mode using the join command. We also set the name and channel of the adhoc network:

set wlan join 0	<Puts the module in adhoc Mode>
set wlan ssid vanet1	<Sets the name of the network>
set wlan chan 0	<Sets the channel of the network>

The (**set wlan join**) command is used to determine the policy for association with network access point. If the user wants to join with a network manually, the command must be followed by "0", this prevents the transceiver to join a network automatically. If the user wants to join a network with the stored SSID, then the command must be followed by "1". To associate with Adhoc network (**set wlan join 0**) is used, in order to connect to the required network.

The (**Set wlan ssid**) command is used to set the name of the network, in this project the (**Set wlan ssid vanet1**) is used.

The (**set wlan chan**) command is used to determine in how many channels between the 13 channels of the transceiver the scan can be performed. If this command was followed by a "0", this means that the scan will be performed across all the 13 channels returning the SSID for the WLAN's in range. In this project the (**set wlan chan 0**) command is used to perform the scan at all the 13 channels.

```

169.254.1.1 - Tera Term VT
File Edit Setup Control Window Help
*HELLO*CMD
set wlan join 0
AOK
<2.21> set wlan ssid vanet1
AOK
<2.21> set wlan chan 0
AOK
<2.21> save
Storing in config
<2.21>

```

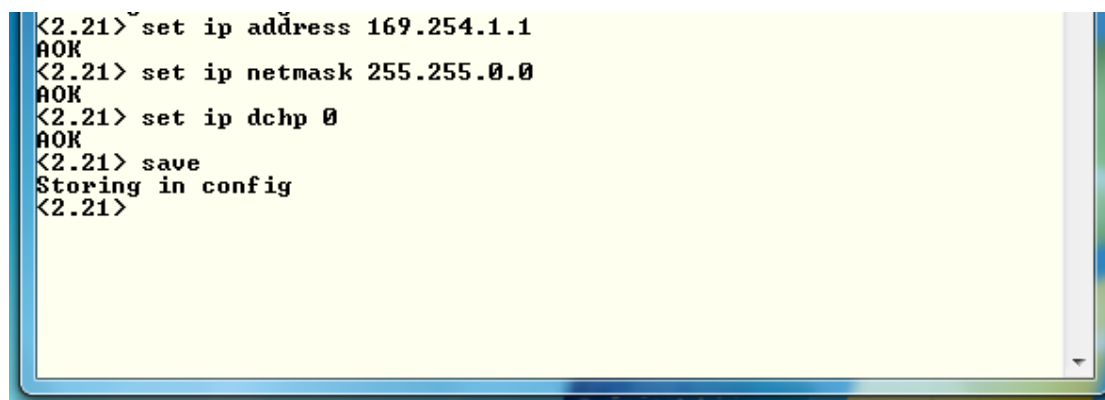
Figure 5.11 Programming Auto-association Commands

Next we turn off DHCP and set the IP address and netmask so other devices know where to connect to the WiFly GSX. Since auto IP fixes the first two bytes of the IP address you want to use the netmask of 255.255.0.0 so that other device connecting to the module can be reached.

```
set ip address 169.254.1.1          <Set the module's IP address>
set ip netmask 255.255.0.0        <Set the module's IP netmask>
set ip dhcp 0                      <Turn off DHCP>
```

The transceiver must have an IP address to be a part of any network, this IP could be assigned manually using the (**set ip address**) command followed by the wanted IP. In this project the ip address of the wifly is assigned manually and given the following ip address: 169.254.1.1

The DHCP client can be configured using the (**set ip dhcp**) command. If this command was followed by “0”, then the DHCP client is off, and the transceiver will use the stored static IP. If the command was followed by “1”, this means that the DHCP client is on and the transceiver will be given an IP address and use the gateway address of the access point. In this project, the (**set ip dhcp 0**) is used, DHCP OFF, use stored static IP address.



```
<2.21> set ip address 169.254.1.1
AOK
<2.21> set ip netmask 255.255.0.0
AOK
<2.21> set ip dhcp 0
AOK
<2.21> save
Storing in config
<2.21>
```

Figure 5.12 Programming Auto-association Commands.

Since in this project the sleep mode will not be implemented, the auto sleep configuration must be disabled, this is done by using the (**set sys autosleep 0**) and the sleep timer must be zero, this is done using the command (**set sys sleep 0**). In this case the transceiver will always be in the active mode.

```
<2.21> set sys autosleep 0
AOK
<2.21> set sys sleep 0
AOK
<2.21> save
Storing in config
<2.21>
```

Figure 5.13 Disabling Autosleep Mode.

The WiFly GSX has an option that if the transceiver has been idle for a specific time, the connection will be disconnected. The command (**set comm idle**) sets the idle disconnect timer. If the command was followed by “0”, the idle disconnect will be disabled, as the case in this system.

The default case for the transceiver is to receive a flush size value before forwarding, this flush size value is the number of bytes that should be received on the UART of the transceiver before starting to forward to the TCP connection, this value is determined by the command (**set comm size**), the default value for this command is 64 bytes, and the maximum is 1420 bytes. If this command can be followed by “0” or “1” as the case in this system, this case makes the transceiver start forwarding immediately without the need of the flush size.

```
<2.21> set comm idle 0
AOK
<2.21> set comm size 0
AOK
<2.21> save
Storing in config
<2.21> █
```

Figure 5.14 Programming Transferring of Data commands.

For a successful serial communication between the transceiver's UART and the PIC microcontroller USART, they must have the same baud rate. The command that sets the transceiver's baud rate is (**set uart baud**) followed by the value of the baud rate that is the same in both the transceiver and the PIC microcontroller which in this system is (**9600**) bytes per second (Bps).

```
<2.21> set uart baud 9600
AOK
<2.21> save
Storing in config
<2.21> █
```

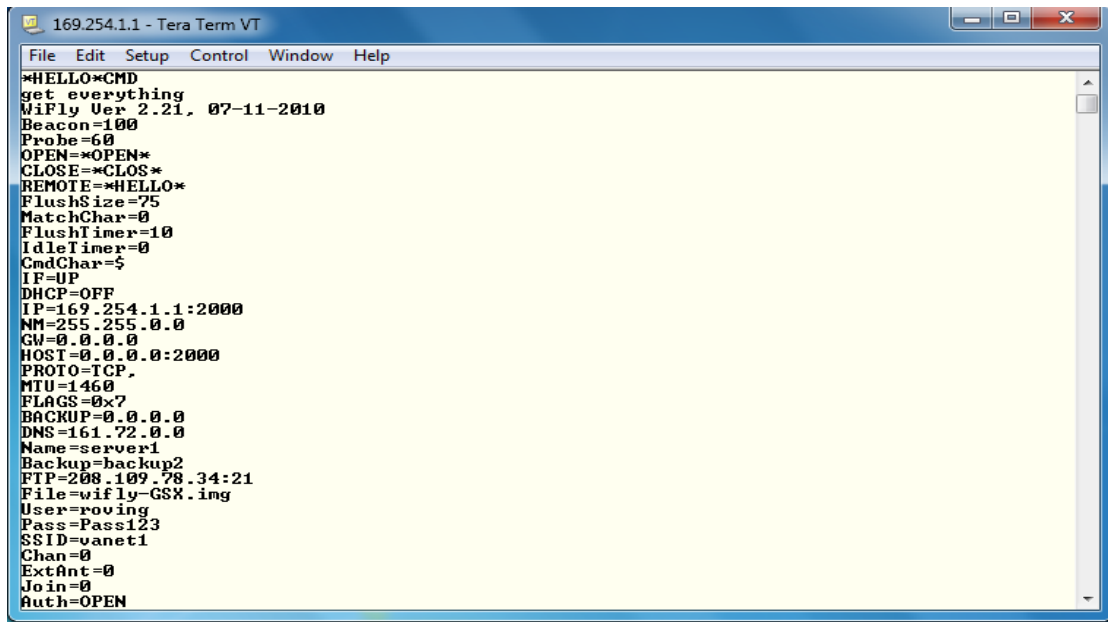
Figure 5.15 Programming UART Settings.

For each command to be successfully accepted by the transceiver, a (**save**) command must be used after any new commands. If the commands were successfully handled by the transceiver, it will reply with (**AOK**), if any error in the syntax of the command, the transceiver will reply with (**ERR: ?- Cmd**) , and (**ERR: Bad Args**) in case of error in the parameters.

<pre>169.254.1.1 - Tera Term VT File Edit Setup Control Window Help *HELLO*CMD set wlan join 0 ERR: ?-Cmd <2.21></pre>	<pre><2.21> set wlan ssidvanet ERR: 2few Args <2.21></pre>
--	--

Figure 5.16 Errors in Commands.

All the commands that were mentioned above are from the **set** category of the commands. There are another four categories, **get**, **status**, **action**, and **IO** commands.



```
169.254.1.1 - Tera Term VT
File Edit Setup Control Window Help
*HELLO*CMD
get everything
WiFly Ver 2.21, 07-11-2010
Beacon=100
Probe=60
OPEN=*OPEN*
CLOSE=*CLOS*
REMOTE=*HELLO*
FlushSize=75
MatchChar=0
FlushTimer=10
IdleTimer=0
CmdChar=$
IP=UP
DHCP=OFF
IP=169.254.1.1:2000
NM=255.255.0.0
GW=0.0.0.0
HOST=0.0.0.0:2000
PROTO=TCP
MTU=1460
FLAGS=0x7
BACKUP=0.0.0.0
DNS=161.72.0.0
Name=server1
Backup=backup2
FTP=200.109.78.34:21
File=wifly-GSX.img
User=roving
Pass=Pass123
SSID=vanet1
Chan=0
ExtAnt=0
Join=0
Auth=OPEN
```

Figure 5.17 A GET Command Example.

5.2.3 LCD with PIC programming.

5.2.3.1 List of Component Modules

XLCD.P18.ex.txt this is main the test file developed to demonstrate the use of the library functions for the PIC18 family

XLCD.c this is the LCD code implementation file. One needs to include this into their project.

XLCD.h This is the header file, all functions and control ports are defined here

One needs to include this into their project.

5.2.3.2 How Using the Library Module in a Project

We follow the steps below to use the library module in our project:

1. Use the Application Maestro™ software to configure the code as required.
2. At the Generate Files step, save the output to the directory where the code of the project resides.
3. Launch the MPLAB® IDE, and open the project's workspace.

4. Verify that the Microchip language tool suite is selected (*Project>Select Language Toolsuite>Microchip C18Toolsuite*).
5. Got to (*Project>build options>project*) and select the path for linker, library etc.
6. In the Workspace view, right-click on the “Source Files” node. Select the “Add Files” option. Select XLCD.C and click OK.
7. Now right-click on the “Linker Scripts” node and select “Add Files”. Add the appropriate linker file (.lkr) for the project’s target microcontroller.
8. Add any other files that the project may require. Save and close the project.
9. To use the module in your application, invoke the functions as needed.

5.2.3.4. The Shared Parameters of LCD

To initialize the LCD module according to the Application Maestro options XLCDInit() is used , XLCDPutRomString(addr) is used to displays String in Program memory, to send clocking signal and instructions to the LCD XLCDCommand(Command) is used.

The command XLCDClear () is added in the code to Clear the DDRAM content of the LCD and points to the 00 address location.

5.3 System Configuration

5.3.4 LCD library configuration

Before writing the program and test it, the LCD library is configured first by following these steps:-

- Selecting the interface between the LCD module and the microcontroller, i.e. whether to
Select 8 or 4 bit interface.
- Selection for upper or lower nibble in case of 4-bit interface.
- Port selection for data transfer.
- Port and pin selection for the control signals.
- Facility to ground the R/W pin of the LCD (if read not required), which can help in saving a port pin for the microcontroller.
- Selection of the mode, whether the user wants delay or read busy flag in between commands.

- Selection of Blocking or non-Blocking functions.
- Configuring the parameters like single-line or two-lines, font selection, cursor on, blink on, etc.

6

Chapter Six

System Testing

6.1 Introduction

6.2 Testing Scheduling

6.3 Testing Procedure

Chapter Six

System Testing

6.1. Introduction

The whole testing stage will be described in this chapter, that's mean how to operate each part independently, in addition how to interface and test these parts together in order to achieve our goals.

6.2 Testing Scheduling

The table below shows testing time schedule.

Table 6.1 Testing Schedule

Days Testing Process	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Unit Testing														
Sub-System Testing														
System Testing														
Acceptance Testing														

6.3 Testing Procedures

The system testing procedures are described below.

Unit testing:

To implement such testing, each unit must be tested, so each task of the project would be tested individually. The Ultrasonic was tested alone, Wifly GSX transceiver, the PIC microcontroller, and LCD each was tested separately. Additionally, the connecting cables and all the pins were tested.

The software components also were tested.

Sub-System Testing:

The main aim of this testing part is to test the main operations in the system. There are fifth main operations in the system. The first one is the sensors readings transfer to the PIC. The second operation is exchanging of data between the WiFly transceiver and the PIC microcontroller. The third is sending result from Wifly transceiver which acts as access point to another WiFly GSX mounted on vehicle. The fourth operation is exchanging of data between the WiFly transceiver and the PIC microcontroller on received vehicle. The last one is sending the result from PIC to LCD. All the operations mentioned before fulfilled the expectations.

The major software components were tested also.

System Testing:

This section is to test system work. It should be tested by using a complete process of connecting the Ultrasonic sensor, PIC microcontroller, Wifly GSX, and LCD. By implementing this process, the system works very well.

The software component in the system was tested. Start from the sensor reading software, connection between transceivers programming, and the software used to view the result on the LCD.

Acceptance testing:

In this section, the system will be tested to see if it meets the requirements the system was built for, and at this point the system did meet the needed task of the system.

6.3.1 Sub-System Testing:

We divide this project into parts:

The first part is determining the congestion type by using the ultrasonic and PIC as shown in the figure below. This part work well in this project with small percentage of error.

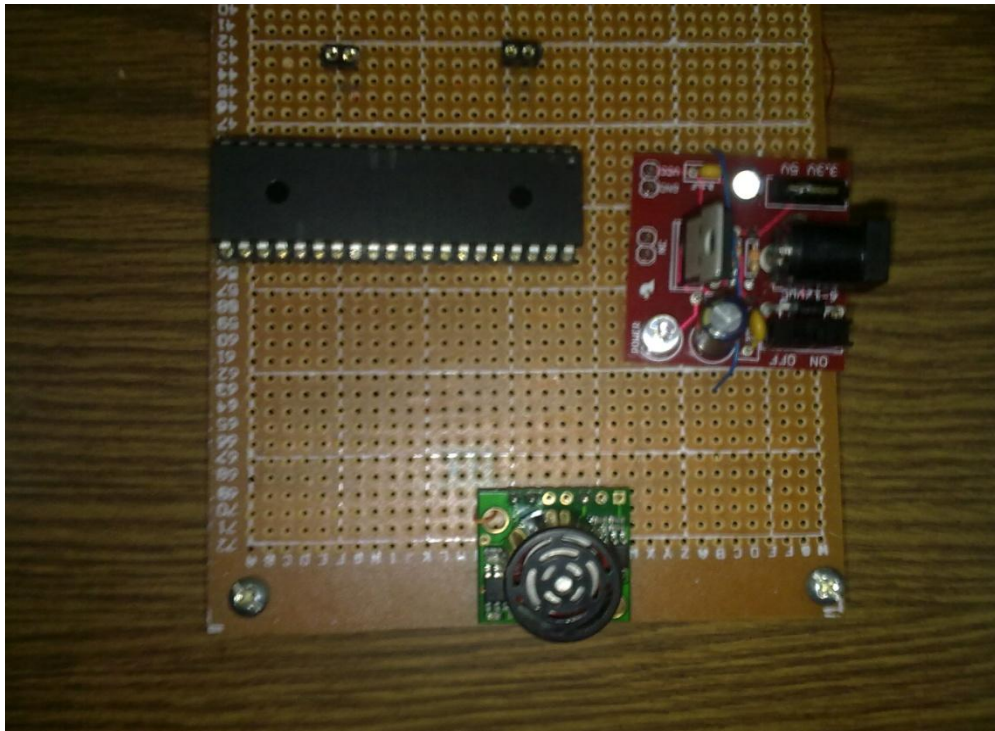


Figure 6.1 Sensing unit

Problems:

The sensitivity of the sensor from the surrounding area is the most problem we face it.

When power up the sensor, it will sends a beam which reach to around 6.45m the shape of this beam is shown in figure 6.2 ,because of this shape the sensor may detect another objects which is not the target object.

In order to avoid this problem, the sensor mounted on overhead position and isolated from other components in this project, so it sends beams towards the road which is one way road and detects only the vehicle on this road.

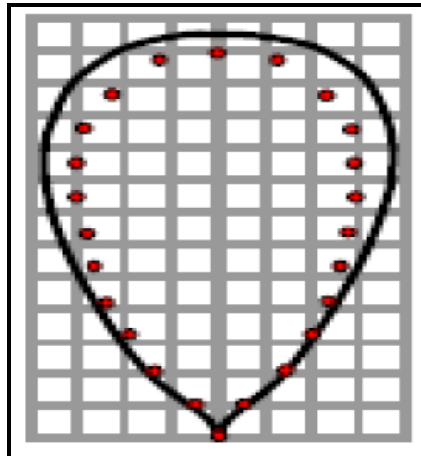


Figure 6.2 Beam shape of the sensor

The ultrasonic sensor is connected to PIC which operates at **8MHz** frequency or **0.125 μS** cycle duration, and as known that PIC's are from the RISC architecture which means, one instruction per cycle, except branching instructions.

The processing delay is not related only to the PICs, there is Analog-to-Digital converter (ADC) which consumes time when doing conversions.

There is a minimum acquisition time for the ADC that must be taken into consideration. Figure 6.3 shows a snipped image form the PIC18F4550 datasheet contains the calculations for the minimum acquisition for the ADC.

TACQ	=	Amplifier Settling Time + Hold Capacitor Charging Time + Temperature Coefficient
	=	TAMP + TC + TCOFF
	=	2 μs + TC + [(Temperature - 25°C)(0.05 μs/°C)]
TC	=	CHOLD (RIC + RSS + RS) ln(1/2047)
	=	- 120 pF (1 kΩ + 7 kΩ + 10 kΩ) ln(0.0004885)
	=	16.47 μs
TACQ	=	2 μs + 16.47 μs + [(50°C - 25°C)(0.05 μs/°C)]
	=	19.72 μs

Figure 6.3 Minimum Acquisition Time in the PIC18F4550 ADC.

The ADC acquisition time value is essential when writing the code, to avoid incorrect readings for the ADC.

The second part is the WiFly transceiver which acts as access point that received the message of the road state from the first part and then transmits this message to another transceiver as shown in the figure below.

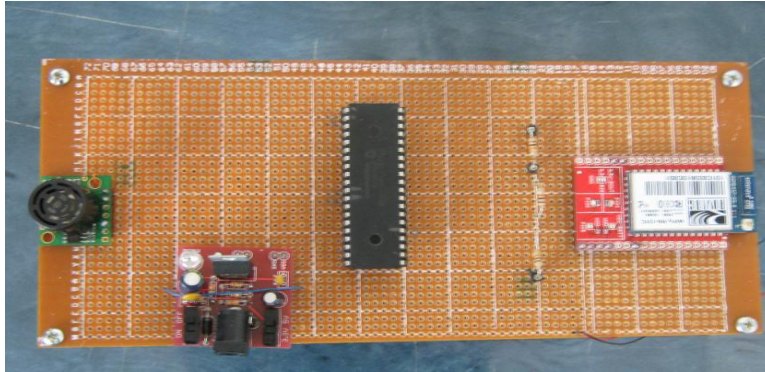


Figure 6.4: Sensing unit with Access point

The third part is mounted on the vehicle which consists of transceiver, PIC, LCD as shown in the figure 6.5 in this part the WiFly received the message from the access point and send it serially to the PIC which will show the result on LCD. This part work well without any problem.

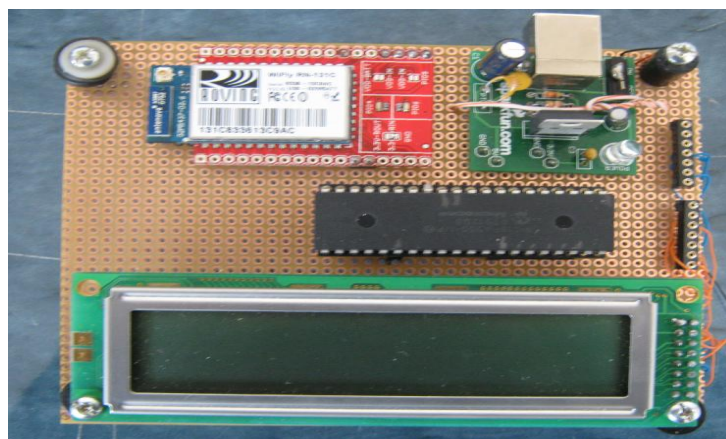


Figure 6.5 Components connection on the vehicle

The message which shown on the LCD contain the road name and type of congestion which may high or low as shown in the figures below .

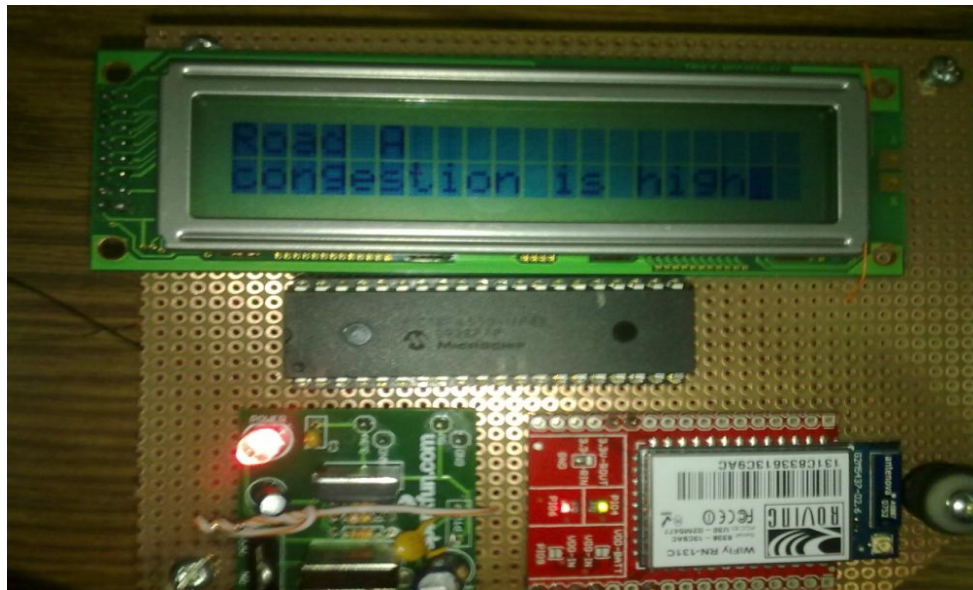


Figure 6.6: Message shown on the LCD when congestion is high



Figure 6.7: Message shown on the LCD when congestion is medium

Challenges:

Purchasing components and there latency arrival was the most confused challenge. The other challenge was the compatibility between the components, all parts of the project works well individually but it was difficult, need more accuracy in connection and programming when we connect all parts as one system.

7

Chapter Seven

Conclusion and Recommendation

7.1 Introduction

7.2. System Achievements

7.3. Real Learning Outcomes

7.4. Recommendation

Chapter Seven

Conclusion and Recommendation

7.1 Introduction

The project that has been done was a step for developing the idea of VANET network. Also the project was a good step in implementing a system for congestion limitation. Meanwhile we have some recommendations and suggestions for the future work. The following section will discuss them.

7.2. System Achievements

Almost all the goals of our system have been achieved. In this point the main achievements of the system are discussed and the ways of achieving it.

We build an ultrasonic sensor that read the congestion range by counting the number of vehicles and send them to the transceiver.

We can run our application continuously after a small waiting time needed for the Programming component to restart, in order to achieve project goals to get real time or near real time response as soon as the signal is arrived to LCD.

7.3. Real Learning Outcomes

After the implementation of the project we have an expert in the following points:

- Learn how to use and program 18F4550 microcontroller.
- How to control using devices.
- Learn how to program ultrasonic sensor.
- Transceiver programming.
- LCD programming.
- Faces many problems with communicating with the wireless transceiver and learn how to solve it.

7.4. Recommendation

In this project, there are some ideas that could be done or added to improve its performance, or add some capabilities, some techniques that are efficient and meet worldwide needs. Some of these ideas are mentioned below:

System could be improved to handle number of cross sections in the same country, controlled by one central station at the same time. This could be accomplished by updating the system software and hardware for that manner.

- The system could be improved to use the mobile phone instead of the LCD to show the road state to the driver.
- A new capability could be added to the system, to enable the driver to have a complete road map.
- An improvement to the system could be applied, by using a high resolution camera, or any other microcontroller or microprocessor with higher frequency to enable the driver to see the cross section road.
- The two transceivers could be replaced with server system that supports multitasking, and high performance.

References

- [1] Rainer Baumann, "Engineering and simulation of mobile ad hoc routing protocols for VANET on highways and in cities", ETH Zurich 2004.
- [2] Nathan Balon, Jinhua Guo, "Increasing Broadcast Reliability in Vehicular Ad Hoc Networks", University of Michigan, Dearborn, MI, 2006
- [3] Politecnico Torino Dipartimento Elettronica, "Smart Broadcast of Warning Messages in Vehicular Ad Hoc Networks", 10129 Torino, Italy.
- [4] Bryan Parno, Adrian Perrig, "Challenges in Securing Vehicular Networks", November 14-15, 2005.
- [5] <http://searchmobilecomputing.techtarget.com/definition/ad-hoc-network>.
- [6] <http://www.wisegeek.com/what-is-an-ad-hoc-network.htm>.
- [7] http://en.wikipedia.org/wiki/Mobile_ad_hoc_network.
- [8]:<http://www.wifinotes.com/mobile-communication-technologies/what-is-vanet.html>
- [9] Saleh Yousefi, Mahmoud Siadat Mousavi, Mahmood Fathy, "Vehicular Ad Hoc Networks: Challenges and Perspectives", Iran Univ. of Sci. & Technol., Tehran 2006.
- [10] Saleh Yousefi, Mahmoud Siadat Mousavi, Mahmood Fathy, "ITS Telecommunications Proceedings", International Conference on June Iran Univ. of Sci. & Technol., Tehran 2006.
- [11] Y. Wang, F. Li, "Guide to Wireless Ad Hoc Network: Vehicular Ad Hoc Networks", Springer London, pp.503-525 March, 2009.
- [12] Z. Li, Z. Wang, and C. Chigan, "Security of Vehicular Ad Hoc Networks in Intelligent Transportation Systems, in Wireless Technologies for Intelligent Transportation Systems", Nova Science Publishers, 2009.
- [13] http://en.wikipedia.org/wiki/Intelligent_transportation_system
- [14] http://www.ops.fhwa.dot.gov/.../chapter15_01.htm.
- [15] Dogan Ibrahim Advanced PIC Microcontroller Projects in C, Elsevier Publishing, March 2008.
- [16] <http://en.wikipedia.org>, Microcontroller.
- [17] <http://en.wikipedia.org>, PIC Microcontroller.

[18] <http://en.wikipedia.org/wiki/Wireless>

[19] http://en.wikipedia.org/wiki/Omnidirectional_antenna

[20] <http://www.sparkfun.com/datasheets/Wireless/WiFi/rn-131-ds.pdf>

[21] http://en.wikipedia.org/wiki/Liquid_crystal_display

Appendix

Appendix A: Data Sheet

Appendix B: Codes

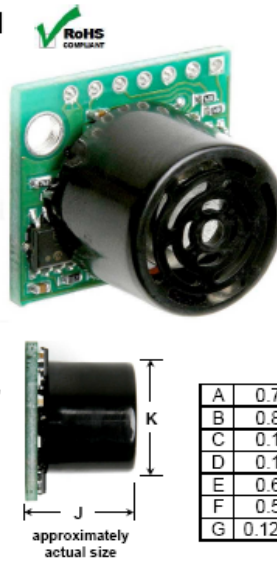
Appendix C: Flowchart

Appendix A
Data Sheet for ultrasonic sensor, PIC18f4550,
WiFly GSX, and LCD

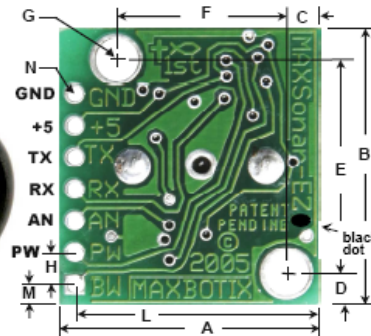
EZO-MaxSonar Ultrasonic Sensors

LV-MaxSonar®-EZ0™ High Performance Sonar Range Finder

With 2.5V - 5.5V power the LV-MaxSonar®-EZ0™ provides very short to long-range detection and ranging, in an incredibly small package. The LV-MaxSonar®-EZ0™ detects objects from 0-inches to 254-inches (6.45-meters) and provides sonar range information from 6-inches out to 254-inches with 1-inch resolution. Objects from 0-inches to 6-inches range as 6-inches. The interface output formats included are pulse width output, analog voltage output, and serial digital output.



LV-MaxSonar®-EZ0™ Data Sheet



A	0.785"	19.9 mm	H	0.100"	2.54 mm
B	0.870"	22.1 mm	J	0.645"	16.4 mm
C	0.100"	2.54 mm	K	0.610"	15.5 mm
D	0.100"	2.54 mm	L	0.735"	18.7 mm
E	0.670"	17.0 mm	M	0.065"	1.7 mm
F	0.510"	12.6 mm	N	0.038" dia.	1.0 mm dia.
G	0.124" dia.	3.1 mm dia.	weight, 4.3 grams		

values are nominal

Features

- Continuously variable gain for beam control and side lobe suppression
- Object detection includes zero range objects
- 2.5V to 5.5V supply with 2mA typical current draw
- Readings can occur up to every 50mS, (20-Hz rate)
- Free run operation can continually measure and output range information
- Triggered operation provides the range reading as desired
- All interfaces are active simultaneously
 - Serial, 0 to Vcc
 - 9600Baud, 81N
 - Analog, (Vcc/512) / inch
 - Pulse width, (147uS/inch)
- Learns ringdown pattern when commanded to start ranging
- Designed for protected indoor environments
- Sensor operates at 42KHz
- High output square wave sensor drive (double Vcc)

Benefits

- Very low cost sonar ranger
- Reliable and stable range data
- Sensor dead zone virtually gone
- Lowest power ranger
- Quality beam characteristics
- Mounting holes provided on the circuit board
- Very low power ranger, excellent for multiple sensor or battery based systems
- Can be triggered externally or internally
- Sensor reports the range reading directly, frees up user processor
- Fast measurement cycle
- User can choose any of the three sensor outputs

Beam Characteristics

The LV-MaxSonar®-EZ0™ has the most sensitivity of the MaxSonar product line, yielding a controlled wide beam with high sensitivity. Sample results for measured beam patterns are shown below on a 12-inch grid. The detection pattern is shown for:

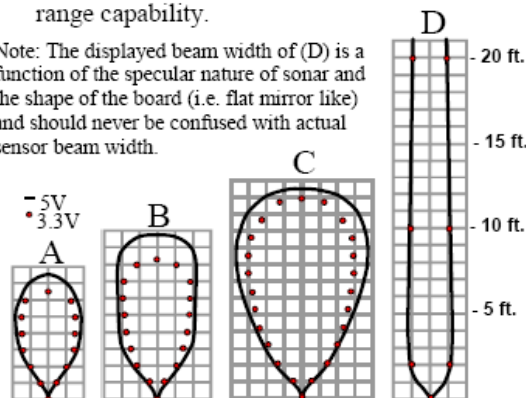
(A) 0.25-inch diameter dowel, note the narrow beam for close small objects,

(B) 1-inch diameter dowel, note the long narrow detection pattern,

(C) 3.25-inch diameter rod, note the long controlled detection pattern,

(D) 11-inch wide board moved left to right with the board parallel to the front sensor face and the sensor stationary. This shows the sensor's range capability.

Note: The displayed beam width of (D) is a function of the specular nature of sonar and the shape of the board (i.e. flat mirror like) and should never be confused with actual sensor beam width.



beam characteristics are approximate

MaxBotix® Inc.

MaxBotix, MaxSonar & EZ0 are trademarks of MaxBotix Inc.
LV-EZ0™ • v3.0c • 07/2007 • Copyright 2005 - 2007

Email: info@maxbotix.com

Web: www.maxbotix.com

LV-MaxSonar®-EZ0™ Pin Out

GND – Return for the DC power supply. GND (& Vcc) must be ripple and noise free for best operation.

+5V –Vcc – Operates on 2.5V - 5.5V. Recommended current capability of 3mA for 5V, and 2mA for 3V.

TX – When the *BW is open or held low, the TX output delivers asynchronous serial with an RS232 format, except voltages are 0-Vcc. The output is an ASCII capital “R”, followed by three ASCII character digits representing the range in inches up to a maximum of 255, followed by a carriage return (ASCII 13). The baud rate is 9600, 8 bits, no parity, with one stop bit. Although the voltage of 0-Vcc is outside the RS232 standard, most RS232 devices have sufficient margin to read 0-Vcc serial data. If standard voltage level RS232 is desired, invert, and connect an RS232 converter such as a MAX232. When BW pin is held high the TX output sends a single pulse, suitable for low noise chaining. (no serial data).

RX – This pin is internally pulled high. The EZ0™ will continually measure range and output if RX data is left unconnected or held high. If held low the EZ0™ will stop ranging. Bring high for 20uS or more to command a range reading.

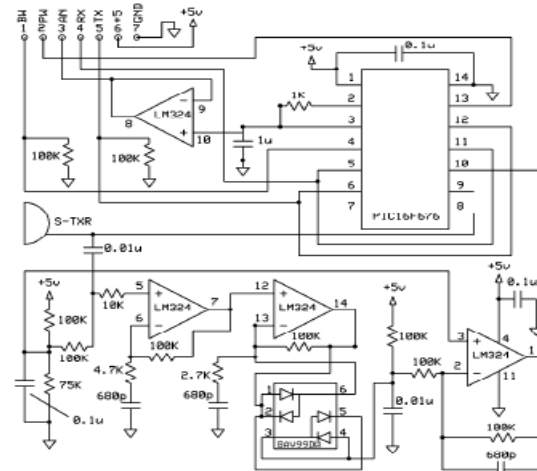
AN – Outputs analog voltage with a scaling factor of (Vcc/512) per inch. A supply of 5V yields ~9.8mV/in. and 3.3V yields ~6.4mV/in. The output is buffered and corresponds to the most recent range data.

PW – This pin outputs a pulse width representation of range. The distance can be calculated using the scale factor of 147uS per inch.

BW – *Leave open or hold low for serial output on the TX output. When BW pin is held high the TX output sends a pulse (instead of serial data), suitable for low noise chaining.

LV-MaxSonar®-EZ0™ Circuit

The LV-MaxSonar®-EZ0™ sensor functions using active components consisting of an LM324, a diode array, a PIC16F676, together with a variety of passive components.



LV-MaxSonar®-EZ0™ Timing Description

250mS after power-up, the LV-MaxSonar®-EZ0™ is ready to accept the RX command. If the RX pin is left open or held high, the sensor will first run a calibration cycle (49mS), and then it will take a range reading (49mS). Therefore, the first reading will take ~100mS. Subsequent readings will take 49mS. The LV-MaxSonar®-EZ0™ checks the RX pin at the end of every cycle. Range data can be acquired once every 49mS.

Each 49mS period starts by the RX being high or open, after which the LV-MaxSonar®-EZ0™ sends thirteen 42KHz waves, after which the pulse width pin (PW) is set high. When a target is detected the PW pin is pulled low. The PW pin is high for up to 37.5mS if no target is detected. The remainder of the 49mS time (less 4.7mS) is spent adjusting the analog voltage to the correct level. When a long distance is measured immediately after a short distance reading, the analog voltage may not reach the exact level within one read cycle. During the last 4.7mS, the serial data is sent. The LV-MaxSonar®-EZ0™ timing is factory calibrated to one percent at five volts, and in use is better than two percent. In addition, operation at 3.3V typically causes the objects range, to be reported, one to two percent further than actual.

LV-MaxSonar®-EZ0™ General Power-Up Instruction

Each time after the LV-MaxSonar®-EZ0™ is powered up, it will calibrate during its first read cycle. The sensor uses this stored information to range a close object. It is important that objects not be close to the sensor during this calibration cycle. The best sensitivity is obtained when it is clear for fourteen inches, but good results are common when clear for at least seven inches. If an object is too close during the calibration cycle, the sensor may then ignore objects at that distance.

The LV-MaxSonar®-EZ0™ does not use the calibration data to temperature compensate for range, but instead to compensate for the sensor ringdown pattern. If the temperature, humidity, or applied voltage changes during operation, the sensor may require recalibration to reacquire the ringdown pattern. Unless recalibrated, if the temperature increases, the sensor is more likely to have false close readings. If the temperature decreases, the sensor is more likely to have reduced up close sensitivity. To recalibrate the LV-MaxSonar®-EZ0™, cycle power, then command a read cycle.

Product / specifications subject to change without notice. For more info visit www.maxbotix.com/MaxSonar-EZ1_FAQ

MaxBotix® Inc.

MaxBotix, MaxSonar & EZ0 are trademarks of MaxBotix Inc.

PIC18F4550 Datasheet



MICROCHIP PIC18F2455/2550/4455/4550

28/40/44-Pin, High-Performance, Enhanced Flash, USB Microcontrollers with nanoWatt Technology

Universal Serial Bus Features:

- USB V2.0 Compliant
- Low Speed (1.5 Mb/s) and Full Speed (12 Mb/s)
- Supports Control, Interrupt, Isochronous and Bulk Transfers
- Supports up to 32 Endpoints (16 bidirectional)
- 1-Kbyte Dual Access RAM for USB
- On-Chip USB Transceiver with On-Chip Voltage Regulator
- Interface for Off-Chip USB Transceiver
- Streaming Parallel Port (SPP) for USB streaming transfers (40/44-pin devices only)

Power-Managed Modes:

- Run: CPU on, peripherals on
- Idle: CPU off, peripherals on
- Sleep: CPU off, peripherals off
- Idle mode currents down to 5.8 μ A typical
- Sleep mode currents down to 0.1 μ A typical
- Timer1 Oscillator: 1.1 μ A typical, 32 kHz, 2V
- Watchdog Timer: 2.1 μ A typical
- Two-Speed Oscillator Start-up

Flexible Oscillator Structure:

- Four Crystal modes, including High Precision PLL for USB
- Two External Clock modes, up to 48 MHz
- Internal Oscillator Block:
 - 8 user-selectable frequencies, from 31 kHz to 8 MHz
 - User-tunable to compensate for frequency drift
- Secondary Oscillator using Timer1 @ 32 kHz
- Dual Oscillator options allow microcontroller and USB module to run at different clock speeds
- Fail-Safe Clock Monitor:
 - Allows for safe shutdown if any clock stops

Peripheral Highlights:

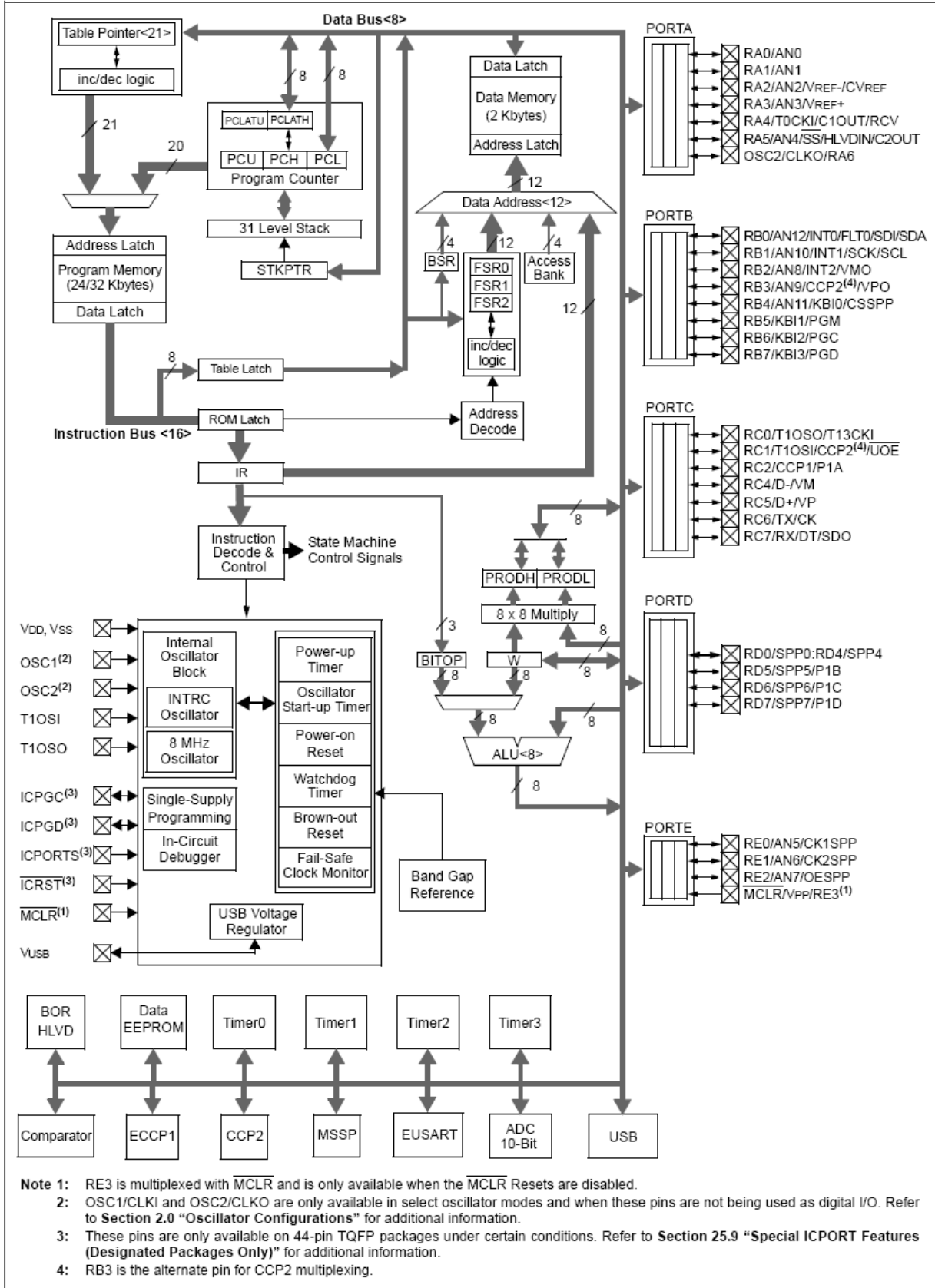
- High-Current Sink/Source: 25 mA/25 mA
- Three External Interrupts
- Four Timer modules (Timer0 to Timer3)
- Up to 2 Capture/Compare/PWM (CCP) modules:
 - Capture is 16-bit, max. resolution 5.2 ns ($T_{CY}/16$)
 - Compare is 16-bit, max. resolution 83.3 ns (T_{CY})
 - PWM output: PWM resolution is 1 to 10-bit
- Enhanced Capture/Compare/PWM (ECCP) module:
 - Multiple output modes
 - Selectable polarity
 - Programmable dead time
 - Auto-shutdown and auto-restart
- Enhanced USART module:
 - LIN bus support
- Master Synchronous Serial Port (MSSP) module supporting 3-wire SPI (all 4 modes) and I²C™ Master and Slave modes
- 10-bit, up to 13-channel Analog-to-Digital Converter module (A/D) with Programmable Acquisition Time
- Dual Analog Comparators with Input Multiplexing

Special Microcontroller Features:

- C Compiler Optimized Architecture with optional Extended Instruction Set
- 100,000 Erase/Write Cycle Enhanced Flash Program Memory typical
- 1,000,000 Erase/Write Cycle Data EEPROM Memory typical
- Flash/Data EEPROM Retention: > 40 years
- Self-Programmable under Software Control
- Priority Levels for Interrupts
- 8 x 8 Single-Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
 - Programmable period from 41 ms to 131s
- Programmable Code Protection
- Single-Supply 5V In-Circuit Serial Programming™ (ICSP™) via two pins
- In-Circuit Debug (ICD) via two pins
- Optional dedicated ICD/ICSP port (44-pin devices only)
- Wide Operating Voltage Range (2.0V to 5.5V)

Device	Program Memory		Data Memory		I/O	10-Bit A/D (ch)	CCP/ECCP (PWM)	SPP	MSSP		EAUSART	Comparators	Timers 8/16-Bit
	Flash (bytes)	# Single-Word Instructions	SRAM (bytes)	EEPROM (bytes)					SPI	Master I ² C™			
PIC18F2455	24K	12288	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F2550	32K	16384	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F4455	24K	12288	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3
PIC18F4550	32K	16384	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3

FIGURE 1-2: PIC18F4455/4550 (40/44-PIN) BLOCK DIAGRAM



28.0 ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings^(†)

Ambient temperature under bias	-40°C to +85°C
Storage temperature	-65°C to +150°C
Voltage on any pin with respect to V _{SS} (except V _{DD} , $\overline{\text{MCLR}}$ and RA4)	-0.3V to (V _{DD} + 0.3V)
Voltage on V _{DD} with respect to V _{SS}	-0.3V to +7.5V
Voltage on $\overline{\text{MCLR}}$ with respect to V _{SS} (Note 2)	0V to +13.25V
Total power dissipation (Note 1)	1.0W
Maximum current out of V _{SS} pin	300 mA
Maximum current into V _{DD} pin	250 mA
Input clamp current, I _{IK} (V _I < 0 or V _I > V _{DD})	±20 mA
Output clamp current, I _{OK} (V _O < 0 or V _O > V _{DD})	±20 mA
Maximum output current sunk by any I/O pin	25 mA
Maximum output current sourced by any I/O pin	25 mA
Maximum current sunk by all ports	200 mA
Maximum current sourced by all ports	200 mA

Note 1: Power dissipation is calculated as follows:

$$P_{dis} = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD} - V_{OH}) \times I_{OH}\} + \sum (V_{OL} \times I_{OL})$$

- 2:** Voltage spikes below V_{SS} at the $\overline{\text{MCLR}}$ /V_{PP}/RE3 pin, inducing currents greater than 80 mA, may cause latch-up. Thus, a series resistor of 50-100Ω should be used when applying a "low" level to the $\overline{\text{MCLR}}$ /V_{PP}/RE3 pin, rather than pulling this pin directly to V_{SS}.

† **NOTICE:** Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

FIGURE 28-1: PIC18F2455/2550/4455/4550 VOLTAGE-FREQUENCY GRAPH (INDUSTRIAL)

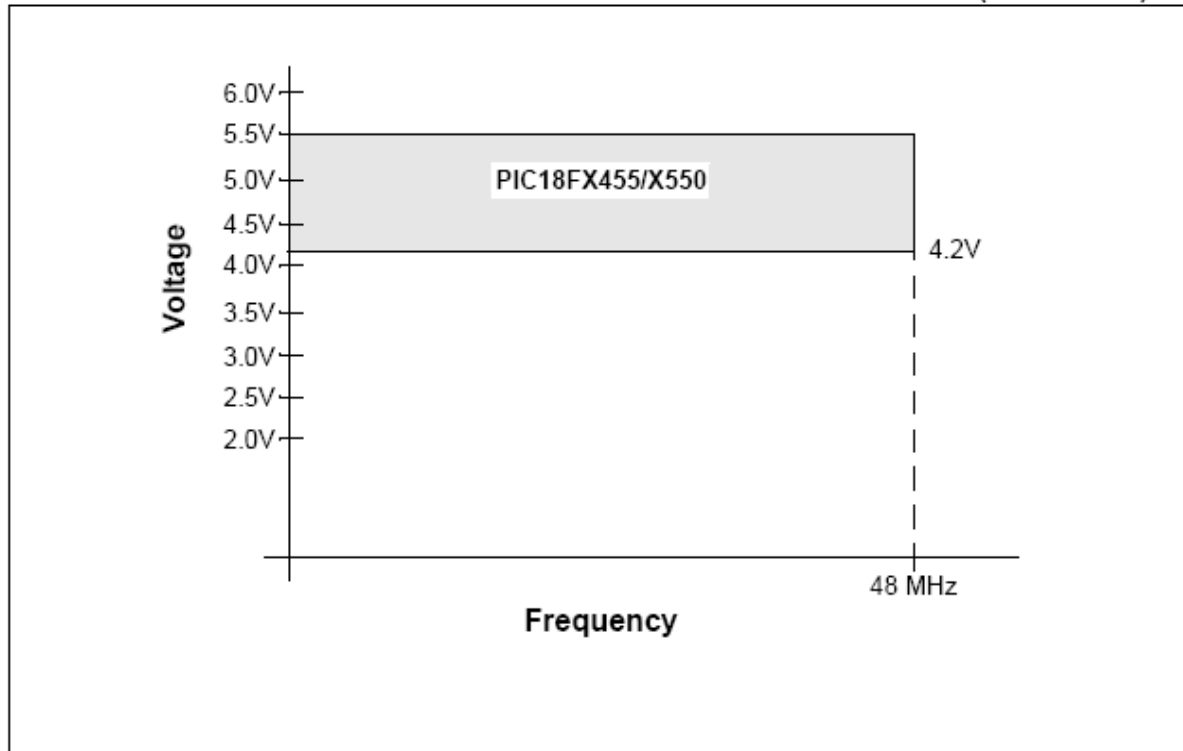
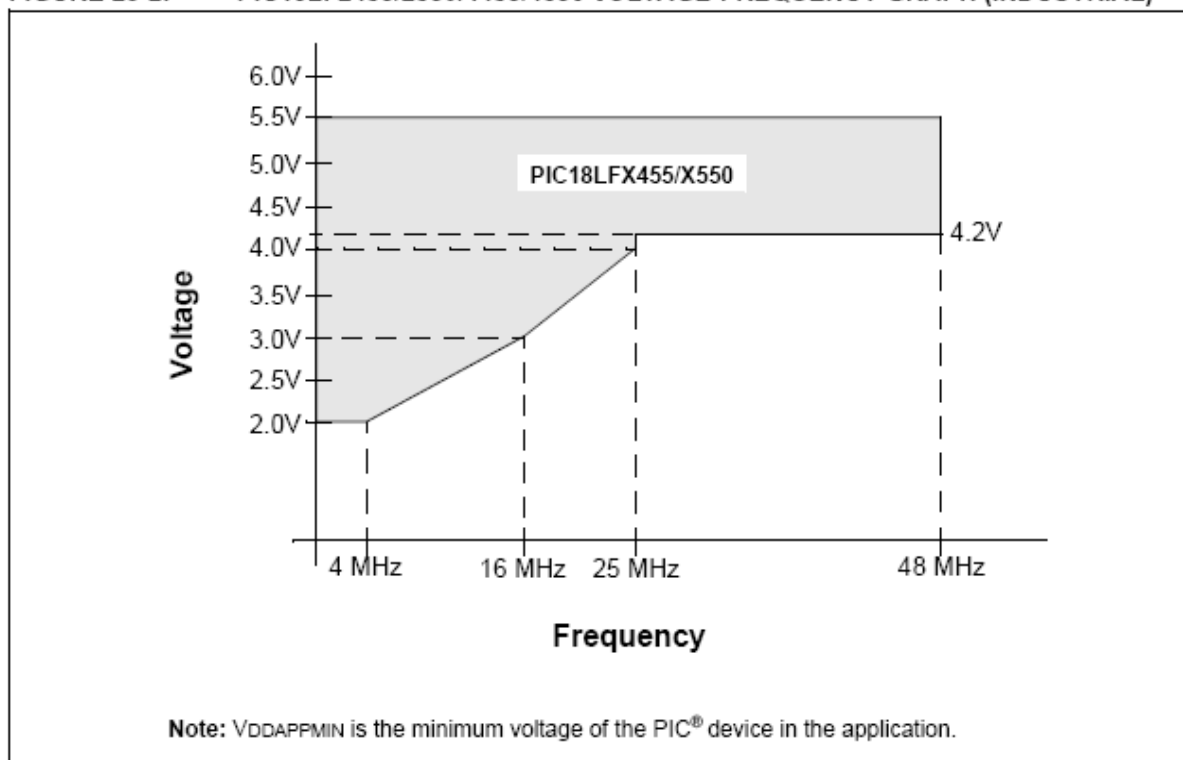


FIGURE 28-2: PIC18LF2455/2550/4455/4550 VOLTAGE-FREQUENCY GRAPH (INDUSTRIAL)



WiFly GSX Transceiver



RN-131G

www.rovingnetworks.com

rn-131-ds v2.3 5/28/2009

"WiFly GSX" 802.11G Module

Features

- Qualified 2.4GHz IEEE 802.11b/g transceiver
- High throughput, 1Mbps sustained data rate with TCP/IP and WPA2
- Ultra-low power - 4uA sleep, 40mA Rx, 210mA Tx (max)
- Small, compact surface mount module
- On board ceramic chip antenna and U.FL connector for external antenna
- 8 Mbit flash memory and 128 KB RAM
- UART hardware interface
- 10 general purpose digital I/O
- 8 analog sensor interfaces
- Real-time clock for wakeup and time stamping
- Accepts 3.3V regulated or 2-3V battery
- Supports Adhoc connections
- On board ECOS -OS, TCP/IP stacks
- Wi-Fi Alliance certified for WPA2-PSK
- FCC / CE/ ICS certified and RoHS compliant.



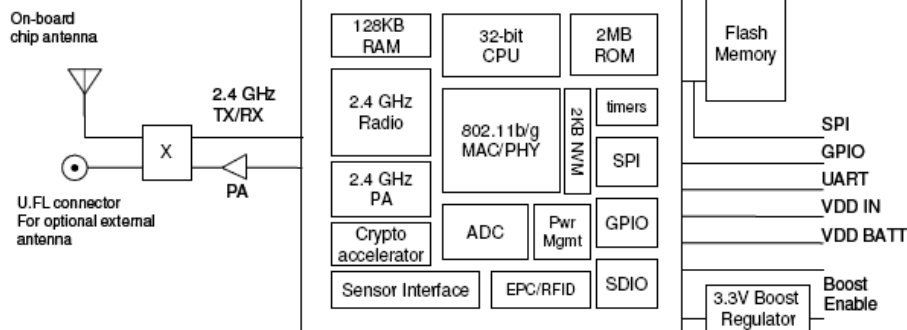
Description

The WiFly GSX module is a stand alone, embedded wireless 802.11 networking module. Because of its small form factor and extremely low power consumption, the RN-131G is perfect for mobile wireless applications such as asset monitoring, GPS tracking and battery sensors. The WiFly GSX module incorporates a 2.4GHz radio, processor, TCP/IP stack, real-time clock, crypto accelerator, power management and analog sensor interfaces. This complete solution is preloaded with software to simplify integration and minimizes development of your application. In the simplest configuration the hardware only requires four connections (PWR, TX, RX, GND) to create a wireless data connection. Additionally, the sensor interface provides temperature, audio, motion, acceleration and other analog data without requiring additional hardware. The WiFly GSX module is programmed and controlled with a simple ASCII command language. Once the WiFly GSX is setup it can scan to find an access point, associate, authenticate and connect over any Wifi network.

Applications

- Wireless audio
- Remote equipment monitoring
- Telemetry
- Security
- Industrial sensors and controls
- Home Automation
- Medical devices

Block Diagram



Overview

- Host Data Rate up to 1 Mbps for UART
- Intelligent, built-in power management with programmable wakeup
- Can be powered from regulated 3.3-3.7V source or 2.0-3.0V batteries
- Real time clock for time stamping, auto-sleep and auto-wakeup
- Configuration over UART using simple ASCII commands
- Web Server or Telnet configuration over WiFi
- Over the air firmware upgrade (FTP)
- Memory 128 KB RAM, 2MB ROM, 2 KB battery-backed memory, 8 Mbit Flash.
- Secure WiFi authentication WEP-128, WPA-PSK (TKIP), WPA2-PSK, EAP-TLS for WPA1 & WPA2 Enterprise
- Built in networking applications DHCP, UDP, DNS, ARP, ICMP
- 802.11 power save and roaming functions

Environmental Conditions

Parameter	Value
Temperature Range (Operating)	-40 °C ~ 85 °C
Temperature Range (Storage)	-40 °C ~ 85 °C
Relative Humidity (Operating)	≤90%
Relative Humidity (Storage)	≤90%

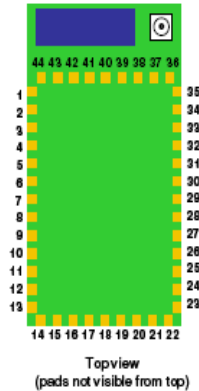
Electrical Characteristics

Supply Voltage	Min	Typ.	Max.	Unit
Supply Voltage VDD	3.0	3.3	3.7	VDC
Supply Voltage (VBATT option)	2.0	3.0	3.3	VDC
Power consumption				
Sleep		4		uA
Standby (doze)	-	15	-	mA
Connected (idle, RX)		40		mA
Connected (TX)		140	212	mA

Radio Characteristics

Parameter	Specifications
Frequency	2402 ~ 2480MHz
Modulation	DSSS(CCK-11, CCK-5.5, DQPSK-2, DBPSK-1)
Channel intervals	5MHz
Channels	1 - 14
Transmission rate (over the air)	1 – 11Mbps for 802.11b / 6 – 54Mbps for 802.11g
Receive sensitivity	-85dBm typ.
Output level (Class1)	+18dBm
Maximum RF input to U.FL connector	10 dBm

Pin Description



Pin	Name	Description	Default
1	SENSOR-6	Sensor interface, analog input to module, 1.2V	No connect
2	SENSOR-4	Sensor interface, Analog input to module, 1.2V	No connect
3	SENSOR-5	Sensor interface, Analog input to module, 1.2V	No connect
4	SENSOR-7	Analog input to module, 1.2V	No connect
5	RESET	Module reset, Active Low, reference to VDD-BATT, 160 min. pulse	Pull up
6	EPC-ANT-A	EPC port, RFID antenna A	No connect
7	EPC-ANT-B	EPC port, RFID antenna B	No connect
8	SUPERCAP	Balance center pin voltage on stacked super capacitors, Analog 3.3V	No connect
9	FORCE_AWAKE	Force the module to wakeup, input to module, 31us min. pulse	
10	GPIO-13	UART RTS flow control, 8mA drive, 3.3V tolerant	
11	GPIO-12	UART CTS flow control, 8mA drive, 3.3V tolerant	
12	UART-RX	RX to the module, 8mA drive, 3.3V tolerant	
13	UART-TX	TX from the module, 8mA drive, 3.3V tolerant	
14	SPI-MOSI	SPI master data out (Contact Roving Networks for details)	No connect
15	SPI-CLK	SPI clock, (Contact Roving Networks for details)	No connect
16	SPI-MISO	SPI master data in (Contact Roving Networks for details)	No connect
17	3.3V-REG-OUT	boost regulator control output, connect to 3.3V-REG-IN to enable	No connect
18	3.3V-REG-IN	boost regulator control input, connect to 3.3V-REG-OUT to enable	GND to disable
19	GND	Ground	
20	VDD-BATT	Battery input, 2.0-3.3V with boost regulator in use, 3.0-3.7V otherwise	
21	VDD-IN	3.3 to 3.7 voltage, do not connect when boost regulator is in use	
22	DMA-TX	Debug port	No connect
23	DMA-RX	Debug port	No connect
24	GPIO-9	Restore factory resets, 8mA drive, 3.3V tolerant	
25	GPIO-8	GPIO, 24mA drive, 3.3V tolerant	Weak pull down
26	GPIO-7	GPIO, 24mA drive, 3.3V tolerant	Weak pull down
27	GPIO-6	Association STATUS, 24mA drive, 3.3V tolerant	LED, Weak pull down
28	GPIO-5	Data transfer STATUS, 24mA drive, 3.3V tolerant	LED, Weak pull down
29	GPIO-4	Connection STATUS, 24mA drive, 3.3V tolerant	LED, Weak pull down
30	SENSOR-1	Sensor interface, analog input to module, 1.2V	
31	SENSOR-2	Sensor interface, analog input to module, 1.2V	
32	SENSOR-3	Sensor interface, analog input to module, 1.2V	
33	SENSE-PWR	Voltage output from module to power external sensors, 1.2-3.3V	
34	SENSOR-0	Wakeup from external condition	
35	NO CONNECT		No connect
36-44	GND	Must be connected for proper antenna performance	

5. **Keep out areas.** When designing your PCB avoid exposed trace and via beneath the module.

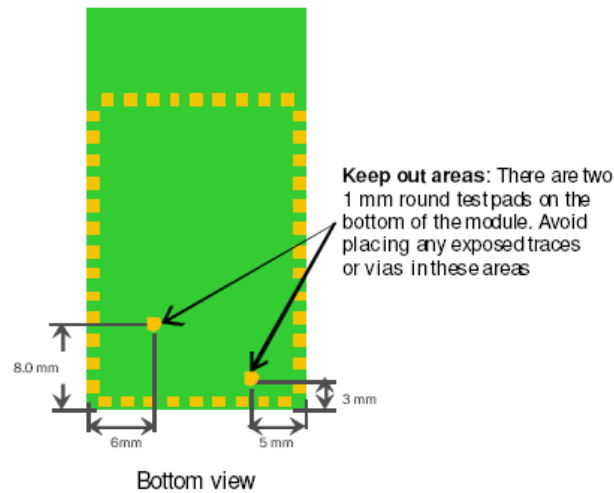
6. **Powering the module.** The WiFly module can be powered from either 3.0VDC batteries or 3.3VDC regulated power.

3.0VDC battery power

- Apply power to pin 20 (VDD-BATT)
- Short pin 17 (3.3V-REG-OUT) to pin 18 (3.3V-REG-IN)

3.3 VDC power

- Apply power to pin 20(VDD-BATT) and pin 21 (VDD-IN)
- Connect pin 18 (3.3V-REG-IN) to ground and leave pin 17 (3.3V-REG-OUT) unconnected.



7. **Sensor Interfaces.** Inputs must not exceed 1.2V. Sensitivity saturates at 400 mV.

8. **Adhoc mode and Restoring Factory Settings.** Adhoc mode is controlled through GPIO-9. It is a good idea to connect pin 24, GPIO-9 to a switch or jumper connected to a pull up. When GPIO-9 is driven high at power up the module will be in Adhoc mode. If GPIO-9 is then toggled low 5 times, the initial factory default configuration will be RESTORED. This is useful for cases where the module is mis-configured and is no long responding.

Compliance Information

- FCC Certified for us in the United States and CE approved for use in Europe and other countries.
- Environmentally friendly RoHS compliant

Ordering Information

Part Number	Description
RN-131G	With chip antenna
RN-131G-EVAL	Development Kit for the RN-131G (Includes the RN-131G module)
RN-UFL-SMA6	6 inch cable with U.FL connector on one end and SMA on the other
For other configurations, contact Roving Networks directly.	

Visit <http://www.rovingnetworks.com> for current pricing and a list of distributors carrying our products.

Copyright © 2009 Roving Networks. All rights reserved.

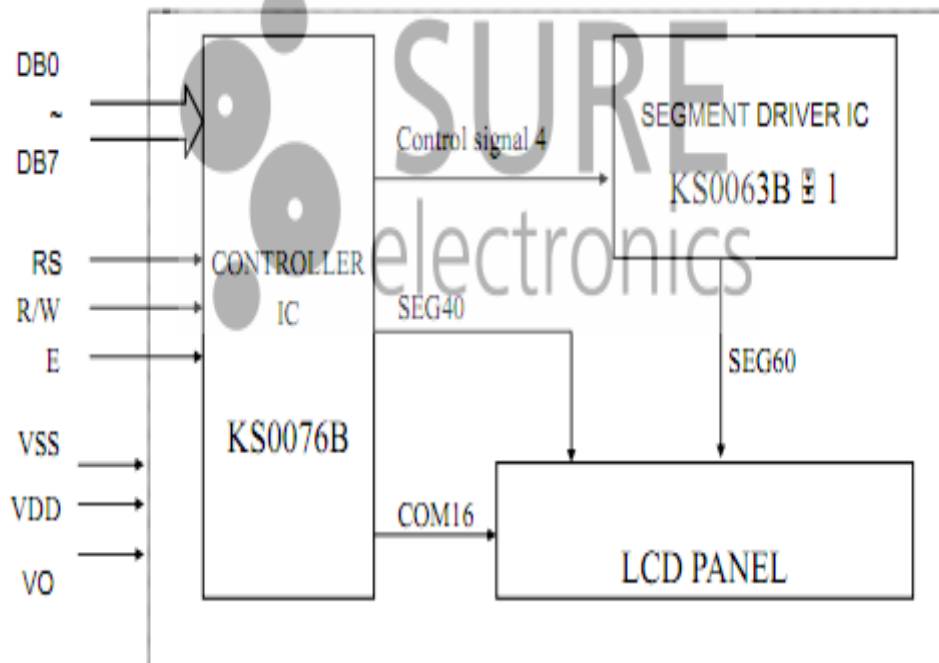
Roving Networks reserves the right to make corrections, modifications, and other changes to its products, documentation and services at any time. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

PID F/C Dual Digital Temperature Controller SSR

4. Electrical characteristics (VDD = +5V±10%, VSS = 0V, Ta = 25°C)

Parameter	Symbol	Condition	Min	Typ	Max	Unit
Supply voltage for logic	VDD	---	4.5	5.0	5.5	V
Supply current for logic	IDD	---	---	0.92	3	mA
Operating voltage for LCD	VDD - VO	0°C	4.2	4.8	5.4	V
		25°C	3.6	4.4	5.1	V
		50°C	3.1	3.6	4.2	V
Input voltage 'H' level	VIH	---	VDD - 2.2	---	VDD	V
Input voltage 'L' level	VIL	---	0	---	0.8	V

5. Block Diagram



PID F/C Dual Digital Temperature Controller SSR

6. Interface pinout

Pin NO.	Symbol	Level	Description
1	VSS	0V	Ground
2	VDD	5.0V	Supply voltage for logic
3	VO	---	Input voltage for LCD
4	RS	H/L	H : Data signal, L : Instruction signal
5	R/W	H/L	H : Read mode, L : Write mode
6	E	H, H → L	Enable signal for KS0076
7	DB0	H/L	Data bit 0
8	DB1	H/L	Data bit 1
9	DB2	H/L	Data bit 2
10	DB3	H/L	Data bit 3
11	DB4	H/L	Data bit 4
12	DB5	H/L	Data bit 5
13	DB6	H/L	Data bit 6
14	DB7	H/L	Data bit 7
15	NC	---	No connection
16	NC	---	No connection

Appendix B:
Codes

Code 1 for ultrasonic sensor:

```
#include <p18f4550.h>
#include<adc.h>
#include<timers.h>
#include<string.h>
#include<delays.h>
#include<usart.h>

#pragma config FOSC = INTOSC_HS
#pragma config WDT = OFF
#pragma config LVP=OFF

int i=0;// count number of interrupt
int t=0;//count the time in second that the vehicle is under the sensor

#pragma interrupt aa
void aa(void)
{
    PIR1bits.TMR1IF=0;
    i++;
    if(i==50)
    {
        t++;
        i=0;
    }
}

#pragma code high_vector=0x08
void high_vector (void)
{ _asm goto aa _endasm }
#pragma code

void main(void)
{
    int c=0; //number of vehicle
    int d=0; // distance between sensor and any thing under it
    int ttot=0; //total time

    char ar1[2]="AM ";
    char ar2[2]="AH ";
    char conn1[21]="open 169.254.1.1 2000";//array to connection
    char clo[5]="close"; // array to close connection
    char conn2[21]="open 169.254.1.3 2000";// array for other connection
```



```

OpenUSART( USART_TX_INT_OFF & USART_RX_INT_OFF &
USART_ASYNCH_MODE & USART_EIGHT_BIT & USART_CONT_RX &
USART_BRGH_LOW, 12 );

```

```

OpenADC(ADC_FOSC_64 & ADC_RIGHT_JUST & ADC_2_TAD , ADC_CH0 &
ADC_INT_OFF & ADC_REF_VDD_VSS , ADC_2ANA); // convert analoge signal
into digital

```

```

OpenTimer1(TIMER_INT_ON & T1_16BIT_RW & T1_SOURCE_INT &
T1_PS_1_8 ); // in open timer 1 there is an error in timers.h ?? use 8bit in declaration
to use 16bit in timer 1

```

```

RCONbits.IPEN=1;           // IPEN=1 enable interrupts
INTCON = 0b10000000;      // Enables all high priority interrupts
PIE1=1;                   // Enables the TMR1 overflow interrupt
IPR1=1;                   // TMR1 Overflow Interrupt Priority is high
ADCON1=15;
OSCCON=OSCCON | 0b01111110;

```

```

while(1)
{
    ConvertADC();
    while (BusyADC());
    d= ReadADC();
    if(d<10) //distance from sensor and street ground

        {
            while(d<10)
            {

                WriteTimer1(0x8ACF); //Set start value of timer,set interrupt at every
20 ms

                ConvertADC();
                while(BusyADC());
                d= ReadADC();

                if(d>10)
                {

                    c++;
                    ttot=t+ttot;
                    t=0;
                }
            }
        }
}

```

```

if(c==4)
{

if(ttot>10&&ttot<30)//congestion is medium
{
for(i=0;i<=21;i++)
{
while(BusyUSART()==1);
putcUSART(conn1[i]);// scan for IP
}

for (i=0;i<=2;i++)
{
while(BusyUSART()==1);
putcUSART(ar1[i]);// send alert to vehicle
}

for(i=0;i<=5;i++)
{while(BusyUSART()==1);
putcUSART(clo[i]); //close connection
}

}
else if(ttot>30)//congestion is high
{
for(i=0;i<=21;i++)
{
while(BusyUSART()==1);// scan for IP
putcUSART(conn1[i]);
}

for (i=0;i<=2;i++)
{
while(BusyUSART()==1);
putcUSART(ar2[i]);// send alert to vehicle
}

for(i=0;i<=5;i++)
{
while(BusyUSART()==1);
putcUSART(clo[i]);
}
}

c=0;}

}

}

```

Code 2 for LCD:

```
#include <p18f4550.h>
#include<adc.h>
#include<timers.h>
#include<string.h>
#include<delays.h>
#include<usart.h>

#pragma config FOSC = INTOSC_HS
#pragma config WDT = OFF
#pragma config LVP=OFF

int i=0;// count number of interrupt
int t=0;//count the time in second that the vehicle is under the sensor

#pragma interrupt aa
void aa(void)
{
    PIR1bits.TMR1IF=0;
    i++;
    if(i==50)
    {
        t++;
        i=0;
    }
}

#pragma code high_vector=0x08
void high_vector (void)
{ _asm goto aa _endasm }
#pragma code

void main(void)
{
    int c=0; //number of vehicle
    int d=0; // distance between sensor and any thing under it
    int ttot=0; //total time

    char ar1[2]="AM ";
    char ar2[2]="AH ";
    char conn1[21]="open 169.254.1.1 2000";//array to connection
    char clo[5]="close"; // array to close connection
    char conn2[21]="open 169.254.1.3 2000";// array for other connection
```

```
OpenUSART( USART_TX_INT_OFF & USART_RX_INT_OFF &
USART_ASYNCH_MODE & USART_EIGHT_BIT & USART_CONT_RX &
USART_BRGH_LOW, 12 );
```

```
OpenADC(ADC_FOSC_64 & ADC_RIGHT_JUST & ADC_2_TAD , ADC_CH0 &
ADC_INT_OFF & ADC_REF_VDD_VSS , ADC_2ANA);// convert analoge signal
into digital
```

```
OpenTimer1(TIMER_INT_ON & T1_16BIT_RW & T1_SOURCE_INT &
T1_PS_1_8 );// in open timer 1 there is an error in timers.h ?? use 8bit in declaration
to use 16bit in timer 1
```

```
RCONbits.IPEN=1;          // IPEN=1 enable interrupts
INTCON = 0b10000000;      // Enables all high priority interrupts
PIE1=1;                  // Enables the TMR1 overflow interrupt
IPR1=1;                  // TMR1 Overflow Interrupt Priority is high
ADCON1=15;
OSCCON=OSCCON | 0b01111110;
```

```
while(1)
{
    ConvertADC();
    while (BusyADC());
    d= ReadADC();
    if(d<10) //distance from sensor and street ground

        {
            while(d<10)
            {

                WriteTimer1(0x8ACF); //Set start value of timer,set interrupt at every
20 ms

                ConvertADC();
                while(BusyADC());
                d= ReadADC();

                if(d>10)
                {

                    c++;
                    ttot=t+ttot;
                    t=0;
                }
            }
        }

    if(c==4)
    {
```

```

if(ttot>10&&ttot<30)//congestion is medium
{
    for(i=0;i<=21;i++)
    {
        while(BusyUSART()==1);
        putcUSART(conn1[i]);// scan for IP
    }

    for (i=0;i<=2;i++)
    {
        while(BusyUSART()==1);
        putcUSART(ar1[i]);// send alert to vehicle
    }

    for(i=0;i<=5;i++)
    {while(BusyUSART()==1);
    putcUSART(clo[i]); //close connection
    }

}
else if(ttot>30)//congestion is high
{
    for(i=0;i<=21;i++)
    {
        while(BusyUSART()==1);// scan for IP
        putcUSART(conn1[i]);
    }

    for (i=0;i<=2;i++)
    {
        while(BusyUSART()==1);
        putcUSART(ar2[i]);// send alert to vehicle
    }

    for(i=0;i<=5;i++)
    {
        while(BusyUSART()==1);
        putcUSART(clo[i]);
    }
}
c=0;}

}

}

```