

**An-Najah National University**

**Faculty of Graduate Studies**

# **Developing Wireless Sensor Network for Traffic Monitoring**

**By**

**Mohammad Fadi Abdel-Haq**

**Supervisor**

**Dr. Adnan Salman**

**This Thesis is Submitted in Partial Fulfillment of the Requirements for  
The Degree of Master of Advanced Computing, Faculty of Graduate  
Studies, An-Najah National University - Nablus, Palestine.**

**2019**

# Developing Wireless Sensor Network for Traffic Monitoring

By  
Mohammad Fadi Abdel-Haq

This Thesis was Defended Successfully on 16 / 10 / 2019 and approved by:

## Defense Committee Members

## Signature

- Dr. Adnan Salman / Supervisor
- Dr. Imad Saada / External Examiner
- Dr. Baker Abdelhaq / Internal Examiner

  
.....

  
.....

  
.....

## **Dedication**

First and foremost, I would like to give thanks and praise to the Almighty God for His grace and blessings throughout the entire project. Without Him, this was nothing.

This work is dedicated

“To my late father, who has taught me many lessons in life and has gave me valuable educational opportunities, unfortunately didn't stay in this world long enough to see the success of his son.”

“To my mother, her prayers and endless love have made me stronger to face all the problems in this thesis.”

“To my wife, Fatoon, for all of her love and support she has given during the thesis work, even though most of that work occurred on weekends, nights, and all times which supposed to be for my family.”

“To my wonderful kids: Mussab, Leen, and Besan, for bearing with me and my mood swings. I hope that one day they can read this thesis and understand why I spent so much time in front of my Laptop.”

“To my beloved brothers and my sister who encourage and support me.”

## **Acknowledgement**

First and foremost, I offer my sincerest gratitude to my adviser Dr. Adnan Salman for encouragement, guidance and support from the initial to the final level. Without his guidance and persistent help this project would not have been possible.

Special thanks to faculty members working in Computer science, and Mathematics departments for their help and guidance.

Special thanks to Rawdah Technical Community College represented by its Dean Mr. Saleh Abdulhadi and his deputy Mr. Zahran Hassounah for their support and facilities.

## الإقرار

أنا الموقع أدناه مقدم الرسالة التي تحمل العنوان:

### **Developing Wireless Sensor Network for Traffic Monitoring**

أقر بأن ما اشتملت عليه هذه الرسالة إنما هو نتاج جهدي الخاص، باستثناء ما تمت الإشارة إليه حيثما ورد، وأن هذه الرسالة ككل أو من جزء منها لم يقدم من قبل لنيل أية درجة أو بحث علمي أو بحثي لدى أية مؤسسة تعليمية أو بحثية أخرى.

### **Declaration**

The work provided in this thesis, unless otherwise referenced, is the researcher's own work, and has not been submitted elsewhere for any other degree or qualification.

**Student's name:**

اسم الطالب:

**Signature:**

التوقيع:

**Date:**

التاريخ:

## Table of Contents

Title		Page
Dedication		III
Acknowledgement		IV
Declaration		V
Table of Content		VI
List of figures		VIII
List of Tables		X
List of Symbols and Abbreviations		XI
Abstract		XII
<b>Chapter One: Introduction</b>		<b>1</b>
1.	Introduction	1
2.	Objective	4
<b>Chapter Two: Background and Literature Review</b>		<b>5</b>
2.1	Review of Traffic Monitoring Technologies	5
2.1.1	Intrusive Technologies	6
2.1.1.a	Inductive loops	6
2.1.1.b	Pneumatic Tube	7
2.1.1.c	Piezoelectric Sensor	8
2.1.2	Nonintrusive Technologies	8
2.1.2.a	Doppler and Radar Microwave Sensors	8
2.1.2.b	Passive infrared	9
2.1.2.c	Video Image Processing	10
2.1.2.d	Acoustic detector	10
2.1.2.e	Ultrasonic system	10
2.1.2.f	Magnetic Sensor	11
2.2	Comparison of Different Technologies	12
2.3	Motivations of using Wireless Sensor Network	13
2.4	Concept of Wireless Sensor Network	14
2.4.1	WSN Architecture and Components	14
2.4.1.1	Sensor Node	15
2.4.1.2	Gateway (Coordinator)	17
2.4.1.3	Base Station	18
2.4.2	WSN Communication Protocol	18
2.4.2.1	Communication protocol (ZigBee)	19
2.4.3	XBee Networking	21
2.4.3.1	XBee Communication Modes	22
2.4.3.2	XBee Operation Modes	25
2.4.3.3	XBee Module Style	26
2.5	Literature Review	29

<b>Chapter Three: System Design and Architecture</b>		33
3.1	Sensing System	34
3.2	Wireless Sensor Network	38
3.2.1	Sensor Node	41
3.2.2	Coordinator Node	42
3.2.3	Base Station Application	44
<b>Chapter Four: System Implementation</b>		46
4.1	Hardware and Software to implement the Wireless Sensor Network	47
4.1.1	Arduino UNO	47
4.1.2	Arduino XBee Shield	50
4.1.3	Arduino IDE	51
4.1.4	X-CTU	53
4.1.5	Power Source	54
4.2	Wireless Sensor Network implementation	54
4.2.1	XBee to XBee Communication Test	55
4.2.2	Arduino to Arduino Communication Test	57
4.2.3	Data Transmission	59
4.3	Program of Arduino End Device (Sensor Node)	65
4.4	Base Station Application	68
4.5	Data Base	72
4.6	Programming Interface	73
4.7	Retrieve Traffic Parameters	77
4.8	Traffic Visualization	79
<b>Chapter Five: Conclusion</b>		82
References		84
Appendix		88
A.	Theory of Anisotropic Magnetic Resistance (AMR) Sensor	88
B.	Code for Arduino Sensor Node (Transmitter)	99
C.	Code for coordinator Node (Receiver)	99
D.	Code of traffic Visualization	99
	الملخص	ب

### List of Figures

<b>Figure Number</b>	<b>Caption</b>	<b>Page</b>
Figure (2.1)	Inductive Loop Technology	6
Figure (2.2)	Pneumatic Tube Technology	7
Figure (2.3)	Piezoelectric Sensor	8
Figure (2.4)	Microwave Radar	9
Figure (2.5)	Example of Earth's magnetic field distortion	11
Figure (2.6)	Basic architecture of WSN	15
Figure (2.7)	Basic Components of WSN node	16
Figure (2.8)	ZigBee Channels	20
Figure (2.9)	ZigBee Node Structure	21
Figure (2.10)	XBee Module	22
Figure (2.11)	Packaging data for delivery	23
Figure (2.12)	API frame structure	24
Figure (2.13)	XBee-Pro Module and Pinouts	26
Figure (2.14)	Microcontroller Interfacing to XBee	27
Figure (3.1)	System Architecture	33
Figure (3.2)	Honeywell's HMC2003 sensor	36
Figure (3.3)	Configuration of sensing system	37
Figure (3.4)	Sensors configuration for estimating the dimension of the vehicle	38
Figure (3.5)	Sensor node data packet	41
Figure (3.6)	Sensor node processes	43
Figure (3.7)	Coordinator Process	44
Figure (4.1)	Sensor Node	48
Figure (4.2)	Arduino UNO	49
Figure (4.3)	Arduino XBee Shield	51
Figure (4.4)	Arduino IDE Interface	53
Figure (4.5)	X-CTU Software Interface	54
Figure (4.6)	Coordinator and End device XCTU configuration	56
Figure (4.7)	Message sent from End device to Coordinator	57
Figure (4.8)	Arduino communication through XBee modules.	58
Figure (4.9)	Coordinator console showing received messages.	59
Figure (4.10)	API frame received in Hex by coordinator	59
Figure (4.11)	The frame message received on the coordinator	60
Figure (4.12)	XBee API frame interpreter used to decode received frames.	61
Figure (4.13)	Coordinator connection with three end devices	64
Figure (4.14)	Main tasks of Base station application (Traffic Monitor Software)	68



Figure (4.15)	Traffic Monitor Software	70
Figure (4.16)	Running Traffic Monitor Software	72
Figure (4.17)	Table of Traffic Monitor Database	73
Figure (4.18)	Table GetAllParameters method	76
Figure (4.19)	data in JSON format	77
Figure (4.20)	statistical retrieved traffic parameters collected from the node 1	78
Figure (4.21)	Traffic parameters collected from the	79
Figure (4.22)	Traffic visualization interface	80
Figure (4.23)	Traffic visualization after it is run	81

**List of Tables**

<b>Table Number</b>	<b>Caption</b>	<b>Page</b>
Table (2-1)	Zigbee stack layers	20
Table (2-2)	XBee 802.15.4 Pin Assignments	28
Table (4-1)	XBee 802.15.4 Pins	50
Table (4-2)	Configuration of XBee modules for XBee to XBee test communication	56
Table (4-3)	Configuration of XBee end devices and coordinator	63

### List of Symbols and Abbreviations

ITS	Intelligent Traffic Systems
WSN	Wireless Sensor Network
AMR	Anisotropic Magnetic Resistance
WMSN	Wireless Magnetic Sensor Network
WLAN	wireless local area network
SN	Sensor Node
ADC	Analog to Digital Converter
IP	Internet Protocol
OSI Model	Open Systems Interconnection Model
IEEE	Institute of Electrical and Electronics Engineers
RF	Radio Frequency
PHY	Physical layer
MAC	media access control
AT	Transparent Mode
API	Application Programming Interface
LR-WPAN	Low Range Wireless Personal Area Networks
XML	Extensible Markup Language
JSON	JavaScript Object Notation
PWM	Pulse Width Modulation
USB	Universal Serial Bus
CPU	Central Processing Unit
IDE	IDE (Integrated Development Environment).
FTDI	Future Technology Devices International
SWSN	Sensor Wireless Nodes
PANID	Personal Area Network Identifier
SaaS	Software-as-a-service
PaaS	Platform-as-a-Service
IaaS	Infrastructure-as-a-Service
MSB	Most Significant Byte
LSB	Least Significant Byte
DSS	Design Support System

**Developing Wireless Sensor Network for Traffic Monitoring****By****Mohammad Fadi Abdel-Haq****Supervisor****Dr. Adnan Salman****Abstract**

Intelligent Traffic Systems (ITS) management play an important role in modern transportation system. An important area in ITS research is the development of traffic monitoring systems that collect traffic information such as vehicle counting, speed measurements and vehicle classification. This information allows developing smart transportation systems that take into consideration the current traffic situation in order to generate smart decisions.

Existing traffic monitoring systems are mainly based on video recognition or inductive loops. These systems have many limitations. Video-based systems do not work well in inclement weather conditions, like heavy rain or snow. Deployment and maintenance of inductive loops need to cut off the road surface and interrupt traffic vehicles. Moreover, both kinds of traffic monitoring systems are not suitable for large scale deployment.

In this thesis, we developed a Wireless Sensor Network (WSN) application for vehicle counting, vehicle classification and speed measurement based on roadside magnetic sensors. Magnetic sensors can be deployed on the roadside to collect traffic data. Then, the data can be communicated using wireless channels to a central computer to be processed

and analyzed. The traffic data will be saved in a database and can be shared with other transportation applications through the use of web services. This approach has many advantages includes, real time traffic monitoring, ensures proper monitoring of the roads, and requires less human intervention. The focus of this thesis is on the development of the wireless sensor network and the traffic data management. Simulated data is used to evaluate the system instead of deploying magnetic sensors.

# **Chapter One**

## **Introduction**

Transportation is one of the most important human activities that have a major impact on economy and social interactions. It facilitates communication between various locations and it has a major influence on the flourishing of trade. On the other hand, it has side effects and negative impact on the environment due to pollution and emission of carbon dioxide.

Traffic congestion causes significant amount of losses includes economic losses due to the traffic delay, increasing consumption of fuel, and increasing the chances of traffic accidents. One main cause of traffic congestion is the number of vehicles on road. The number of vehicles of all types is increasing constantly. More than 200 thousand vehicles are registered in the west bank according to the Palestinian Ministry of Transport of all types [3]. Such an increase led to saturation on many roads and semi-permanent congestion. Also, poor infrastructures of roads may cause congestion and traffic delay which can hinder the economic and social advancements of the area. Another important cause of traffic congestion is the traffic management system. Therefore, a smart traffic management system that can reduce traffic congestion will be invaluable. For example, by controlling the traffic light schedule dynamically based on the traffic flow parameters.

In order to improve the current traffic management systems, accurate real-time information about the traffic flow is necessary. This includes the number of vehicles, their types and speed. This information will allow developing ITS that minimizes total trip time. Furthermore, it will help engineering road networks where total trip time is minimized.

Several technologies have been used to detect traffic flow parameters include: traffic count, vehicle size and weight, and vehicle speed. These technologies can be classified into two main categories, *intrusive* technologies and *non-intrusive* technologies. Intrusive technologies require the installation of the sensors onto or into the road service. This technology includes several devices such as, Bending Plate [16], Pneumatic Road Tube [16], Piezo-Electric Sensor [16], and Inductive Loop [16]. On the other hand, Non-Intrusive technologies do not interfere with the traffic flow either during installation or during operation. Non-intrusive devices include Passive and Active Infrared [16], Passive Magnetic [16], Microwave - Doppler/Radar, Ultrasonic and Passive Acoustic [16], and Video Image Detection [16]. Each of these technologies has its own limitation. For instance, Video image detection requires expensive processors and sensitive to weather condition. Other technologies require high cost or have a small range of applications[10].

No matter which technology is used to measure the traffic flow parameters, this data must be communicated from road sites to a base station efficiently and reliably. Once the data is available in the base station, it can

be processed, analyzed, saved and shared with other users. Also, the data can be made available to other computer applications through the use of web services which allow creating a smart traffic management system. The focus of this thesis is on developing an application that allow gathering the data from the road sites, communicating the data to the base station through wireless channels, storing the data in a database, and making the data available to other users includes human or computer application. This thesis does not focus on measuring the traffic road parameters themselves. Instead, we used a simulated data to evaluate our system.

In recent years, an efficient design of a Wireless Sensor Network [27] (WSN) received significant attention. A Sensor is a device that detects changes in a particular environment conditions, such as pressure, heat, light, earth magnetic field strength, etc. The output of the sensor is an electrical signal that is transmitted to a micro-controller for further processing. The gathered data from the sensors is then communicated using WSN to a base station computer for further processing. Software in the base-station can be used to determine the traffic flow parameters in the road network and triggers actions in the environment based on the collected data. Or, the data can be communicated to the cloud where it can be further processed, saved, and shared.



**Objectives:**

The main objective of this thesis is to design and develop a software application that satisfies the following requirements:

- 1) Communicates the measured traffic flow parameters (number of vehicles, their type and speed) wirelessly from road sites to a base station. A Wireless Sensor Network (WSN) is used to accomplish this goal.
- 2) The system saves this data in a database for further processing and sharing.
- 3) The system provides dashboard for visualizing the traffic flow parameters by human.
- 4) The system offers a low maintenance cost, ensures proper monitoring of the roads, and requires less human involvement.

## **Chapter Two**

### **Background and Literature Review**

In this chapter, we review several techniques used to detect traffic flow parameters includes, traffic count, vehicle size and weight, and vehicle speed. In Section 3, we discuss the main motivations for using the wireless sensor network for traffic monitoring.

#### **2.1 Review of Traffic Monitoring Technologies**

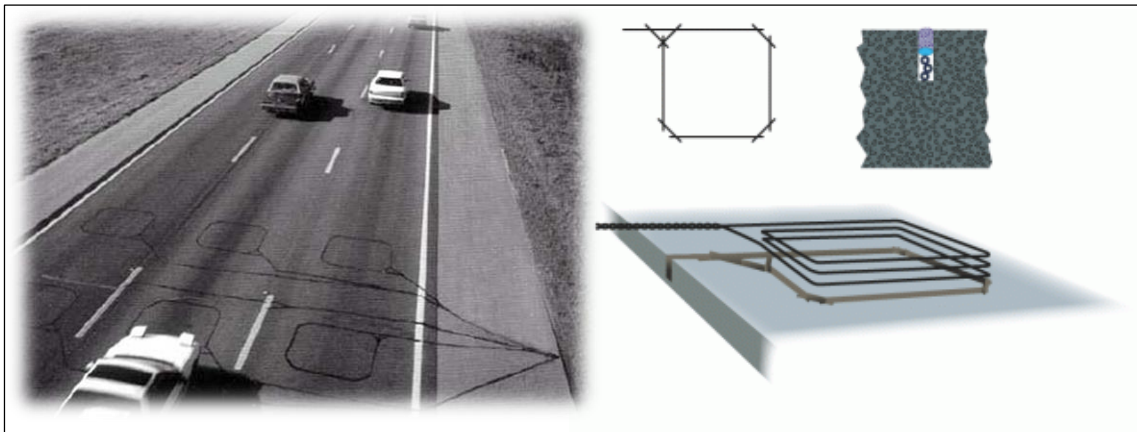
There are several methods used to monitor traffic. One of the simplest and most accurate methods is manually counting vehicles using an electronic hand held counter or records data using a tally sheet. With manual counts, a small sample of data is taken (typically over less than a day) and results are extrapolated for the rest of the year or season. Of course, this approach which is still in use is inefficient and not accurate.

Advanced traffic monitoring technologies can be classified into two categories, intrusive technologies and non-intrusive technologies. Intrusive technologies require the installation of the sensors onto or into the road surface. Nonintrusive sensors are installed above or on the side of roads. In the following subsections we discuss some of these technologies.

## 2.1.1 Intrusive Technologies

### 2.1.1.a Inductive loops

Inductive loops [14] are considered the most common vehicle detection method. An inductive loop consists of wire “coiled” to form a loop that has a circle or rectangle shape that is installed into or under the surface of the road (Fig (2.1)).



**Fig(2.1) : Inductive Loop Technology [19]**

When a vehicle (or any metallic object) drives over the loop, the loop field changes which allows the device to detect the presence of an object (mainly a vehicle). Inductive loops are referred to as presence detectors and in traffic detection are often used to collect vehicle data such as speed and size.

Inductive loop is considered to be the most reliable technology because of its high detection accuracy. However, its biggest disadvantage is that it causes serious traffic disruption during installation and repair. The loop wire is also subjected to stresses of traffic and temperature, making its

failure rate relatively high. Therefore, alternative detectors that can give the same accuracy level with minimum traffic disruption are being actively researched.

#### **2.1.1.b Pneumatic Tube [19]**

In this technology, one or more rubber hoses are stretched across the road and connected at one end to a data logger. The other end of the tube is sealed. When a pair of wheels hits the tube, air pressure in the squashed tube activates the data logger which records the time of the event. A pair of tubes can be stretched across several lanes of traffic.



**Fig (2.2): Pneumatic Tube Technology [19]**

The data logger can establish vehicle direction by recording which tube is crossed first. This has the drawback that if two vehicles cross the tubes at the same time then the direction can't be accurately determined. If two cars are very close to each other when they cross the tubes, the system may see them as one vehicle. Road tubes work well for short duration counts

on lower volume roads. They are not as effective on higher volume, multi-lane highways.



**Fig(2.3) : Piezoelectric Sensor [19]**

### **2.1.1.c Piezoelectric Sensor**

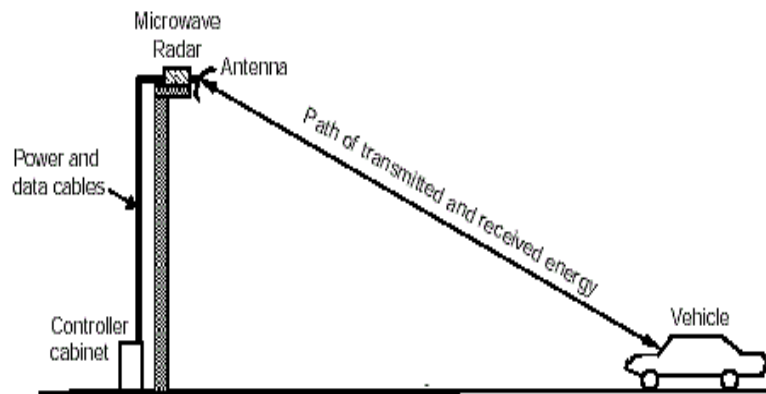
Similar to inductive loop, piezoelectric sensor [19] is installed by embedding it under the pavement. It collects data by converting mechanical energy into electrical energy. When a car drives over the piezoelectric sensor, it squeezes it and generates a voltage. The level of the voltage is proportional to the degree of deformation. When a car moves off, the voltage reverses. This change in voltage can be used to detect and count vehicles.

## **2.1.2 Nonintrusive Technologies**

### **2.1.2.a Doppler and Radar Microwave Sensors**

Doppler microwave detection devices transmit a continuous signal of low-energy microwave radiation at a target area and then analyze the

reflected signal [16]. The detector registers a change in the frequency of waves occurring when the microwave source and the vehicle are in motion relative to one another. This allows the device to detect moving vehicles.



Fig(2.4) : Microwave Radar [8]

Radar is capable of detecting distant objects and determining their position and speed of movement. With vehicle detection, a device directs high frequency radio waves at the roadway to determine the time delay of the return signal, thereby calculating the distance to the detected vehicle. The main advantage of microwave radar is that the system performance is not affected by any weather change. The drawback is that the radar cannot detect motionless vehicle unless an auxiliary device is equipped [18].

#### **2.1.2.b Passive infrared**

Passive infrared devices [18] detect vehicles by measuring the infrared energy radiating from the detection zone. When a vehicle passes the energy radiated changes and the count is increased.

Slow changes in road surface temperature, caused by changing weather conditions, are ignored. Lane coverage is limited to one to two lanes.

#### **2.1.2.c Video Image Processing**

This technology is based on the analysis of images from video camera [18] to extract traffic data. In general, vehicle detection is done by monitoring the changes between successive video frames.

This method has several advantages over other automatic systems. It is cost-effective as it can count in many directions at once. Only one camera is needed for several lanes or exits at a junction. Counts are easily verified simply by watching the video and checking the automated counts. But its performance is affected by many environmental factors, such as weather conditions and lighting conditions.

#### **2.1.2.d Acoustic detector**

This detects vehicles [18] by the sound created as the vehicle passes. The sensor is mounted on a pole pointing down towards the traffic. It can collect counts for one or more travel lanes. Some can communicate their counts wirelessly.

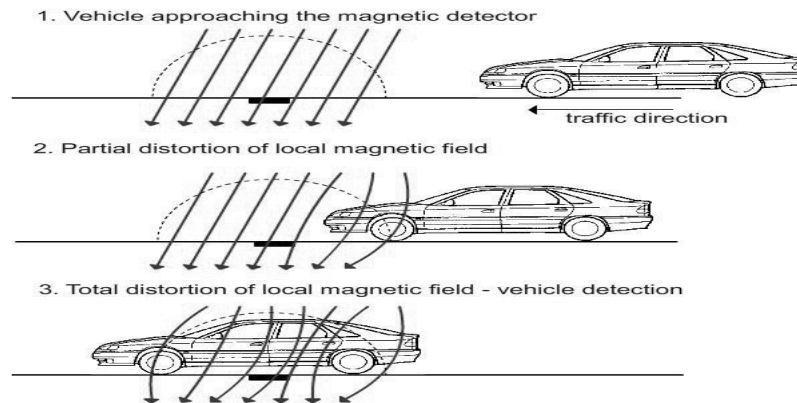
#### **2.1.2.e Ultrasonic system**

Ultrasonic [16] refers to high frequency sound waves that are beyond a human's audible range. Its principle mechanism is similar to that of radar. Sound pulses are transmitted and the reflected pulses are received. The

distance from the receiver to the road or vehicle surfaces is measured according to the wave travel time. If a distance smaller than that to the background road surface is measured, the presence of a vehicle is declared. Speed estimate is obtained by deploying multiple detection zones. Disadvantages include temperature and wind dependence [16].

### 2.1.2.f Magnetic Sensor

This technique detects vehicles by measuring perturbation in the earth's magnetic field [16] caused by the passage of vehicles near the detector. An example of Earth's magnetic field distortion is shown in Fig (2.5)



Fig(2.5) : example of Earth's magnetic field distortion [16]

Based on the level of perturbation, it is possible to determine the size and speed of the passing vehicle. The idea of using the geomagnetism phenomenon in the detection of vehicles is not a new concept on a world-scale. Devices operating on the same principle have been the subjects of research [16]. It should be noted that it is still a novel idea and the implemented systems are effective.



## 2.2 Comparison of Different Technologies

In this section different technologies are compared in terms of their data type availability and system performance. The comparison is based on the results of many experimental evaluation cases summarized in [10, 11, 12, 13 and 14].

Traffic count is available in all the technologies studied. Speed measurement usually requires a dual-detection-zone configuration with synchronized time and fixed separation. Vehicle type classification data is usually obtained by analyzing the detected vehicle lengths, heights, number of axles and spacing.

Inductive loop detector is one of the most accurate count detectors. In [9], it gave an error rate of 0.1-3% for counting vehicles in a one-hour period on the freeway. The system performance may change under the influence of uncontrollable environmental conditions such as temperature, Lighting and high traffic flow [23].

As a result, it is clear to us that the best technology in terms of system performance, data type availability (traffic parameters) and even system cost is the technology which is based on the earth magnetic field.

One type of magnetic field sensors is the Anisotropic Magnetic Resistance (AMR), which is very suitable for use in a sensor node because of its small size and its ability to measure the traffic parameters accurately.

### 2.3 Motivations of using Wireless Sensor Network

The main motivation of developing wireless sensor network for traffic monitoring system is its ability to obtain real-time information about the traffic distribution over a particular region. This information can be used to control light signals in real-time. Also, this data can be saved and shared on the cloud where it can be used in the design of roads.

Other motives are the difference of wireless sensor networks than other popular wireless networks like cellular network and wireless local area network (WLAN) in many characteristics which are described as follows [24]:

***Low Cost.*** Normally hundreds or thousands of sensor nodes are deployed to measure different physical environment parameters. In order to reduce the overall cost of the whole network, the cost of the sensor node must be kept as low as possible.

***Flexibility.*** Wireless sensor networks have a high level of flexibility in their deployment configuration. Since sensor nodes can be placed virtually anywhere on the road as long as they are within communication range, customized configurations can be adopted for different applications and environments. This unique characteristic has advantage over all other technologies.

***Multi-Functional.*** A multi-functions wireless monitoring system can be developed by adding other sensing system to the existing sensor node such as adding temperature sensors to detect ice and snow.

***Communication Capabilities.*** The advantage of wireless communication through radio waves reduces infrastructure cost and maintenance. The network has the property of communicating in short range, with narrow and dynamic bandwidth. The communication channel can be either bidirectional or unidirectional. This feature is extremely useful in enhancing the safety control at intersections, where traffic lights and warning signs can be controlled in advance.

## **2.4 Concept of Wireless Sensor Network**

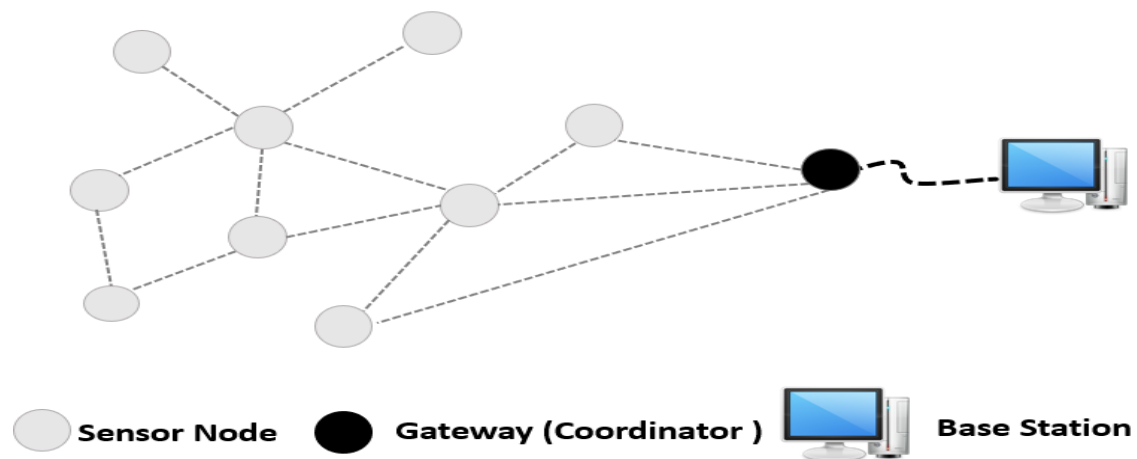
In this section we describe the main concepts of a WSN. This includes wireless sensor network technologies, the communication protocols, and several topologies that can be implemented. In the next chapter we describe our implementation of the WSN.

### **2.4.1 WSN Architecture and Components**

A Wireless Sensor Network (WSN) [20] is a network of small Sensor Nodes (SN) communicating with each other using wireless channels. The goal of these networks is to measure local environmental conditions such as temperature, sound, pressure, etc. Also, it can be used to trigger action in the environments using actuators that are connected to the sensor nodes. WSN

combines distributed sensing elements, actuators, computation, and wireless communication technologies.

A WSN generally consists of a base station that can communicate with a number of wireless sensor nodes via radio channels. Data is collected at the wireless sensor nodes using different sensing elements. Sensor nodes have limited computational capability that allow them to process the collected data and transmit it directly, or using other wireless sensor nodes (routers), to a sink node. The sink node can transmit the data to a base station computer through a gateway, where it can be processed further, saved, and shared with other users and applications. The basic architecture of a WSN is shown below in Figure (2.6) followed by a brief description of its basic elements.

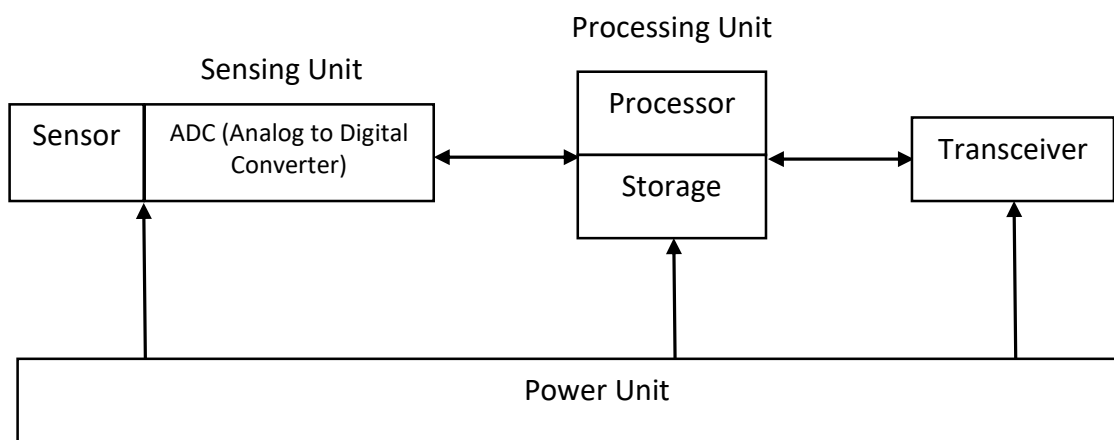


**Fig (2.6): Basic architecture of WSN**

#### **2.4.1.1 Sensor Nodes**

A sensor node, the basic element of a WSN, is capable of gathering data from different kinds of attached sensors. It has the ability of performing

a limited amount of processing on this data. Also, it is capable of communicating with other connected nodes in the network. A sensor node consists of four basic components: a processing unit (microcontroller), a transceiver for wireless communications, a power source, and one or more sensors. These components are integrated on a single or multiple board. Each component is described below and shown in Fig (2.7).



**Fig (2.7): Basic Components of WSN node [22]**

- **Sensing Unit:**

The main functionality of the sensing unit is to measure physical data from the target area. The analog voltage or signal generated by the sensor corresponds to the observed phenomenon. The signal is then digitized by an Analog to Digital Converter (ADC) and sent to the Processor for further processing or analysis [22].

- **Processing Unit:**

The processing unit, which is usually associated with a small storage unit, manages the procedures that make the sensor node collaborate with

other nodes to carry out the assigned sensing tasks, performing power management functions [25], and capable of performing a limited amount of processing the data.

- **Transceiver Unit:**

This unit is used to exchange data between individual wireless nodes and between wireless nodes and the base station. It consists of a radio receiver and a radio transmitter. The main task is to convert a bit/byte stream coming from a processing unit to and from radio signals.

- **Power Unit:**

The power supply is necessary to provide the power for the wireless nodes to work. Since power is limited, the amount of processing the data and the data communication should be considered carefully.

#### **2.4.1.2 Gateway (Coordinator):**

A gateway is a network element that can connect two different-types of networks with different communication protocols. It is basically a protocol translator that can communicate packets between two systems that have different communication protocols and different architecture [7]. In a WSN, the gateway acts as a bridge between the WSN that uses a particular communication protocol such as ZigBee [7] and another network type such as IP-based networks. It is an interface between the application platform and the wireless nodes. Information received from the sensor nodes is translated

by the gateway and forwarded to the application. The application runs on a base station (local computer).

On the other hand, when a command is issued by the application program to a wireless node, the gateway relays the information to the wireless sensor network. A gateway can perform protocol conversion to enable the wireless network to work with other industry standards or non-standard network protocols.

#### **2.4.1.3 Base station**

The base station is the center of the wireless sensor network. It is basically a computer connected to a gateway that has more computational power, more energy and communication resources compared to the sensor nodes. The base station runs an application that analyzes the received data from the sensor nodes, performs appropriate computation, and then displays the information on a user screen. In many applications the base station acts as a bridge between sensor nodes and the end user where it forwards the data to a server which is typically in a cloud.

#### **2.4.2 WSN Communication Protocols**

Wireless sensor nodes are highly limited in resources includes a limited amount of power, limited computational and storage capacities, short communication range, and low bandwidth. Therefore, unlike conventional wireless network, a WSN has its own design that takes these resource constraints into consideration. The specification of a particular design of a

WSN depends on the application itself. This includes the size of the network, the deployment scheme, and the network topology.

The communication protocols used in WSN enable the communication between the sensor nodes themselves and between the sensor nodes and the base station. It consists of the five OSI model [7] for packet switching: application, transport, network, link, and physical layer. The WSN performance and energy consumption are depend on the protocols used in each layer.

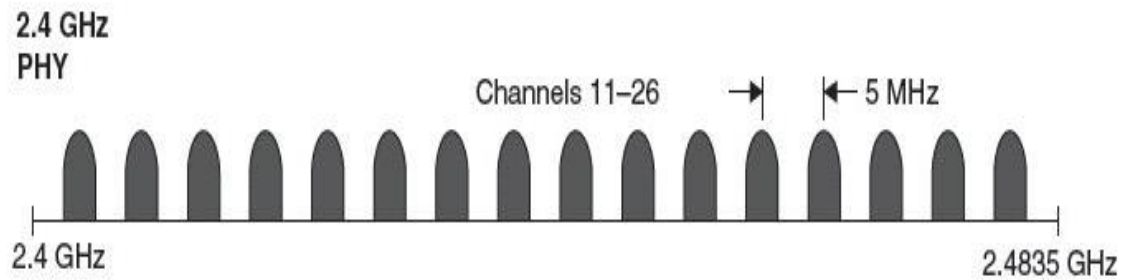
One of the main standards for WSN communication is the IEEE 802.15.4 ZigBee standard described below.

#### **2.4.2.1 Communication protocol (ZigBee)**

The ZigBee stack consists of several layers including the PHY, MAC, Network, Application Support Sub layer (APS), and ZigBee Device Objects (ZDO) layers. The ZigBee stack profile is shown in the table (2-1) [6].

ZigBee utilizes direct-sequence spread spectrum modulation and operates on a fixed channel. The 802.15.4 PHY defines 16 operating channels in the 2.4 GHz frequency band. The figure 4.8 shows ZigBee RF channels, note normally ZigBee RF channels are counted from 11, up to 26 in the 2.4 Ghz band.





**Fig (2.8): ZigBee Channels**

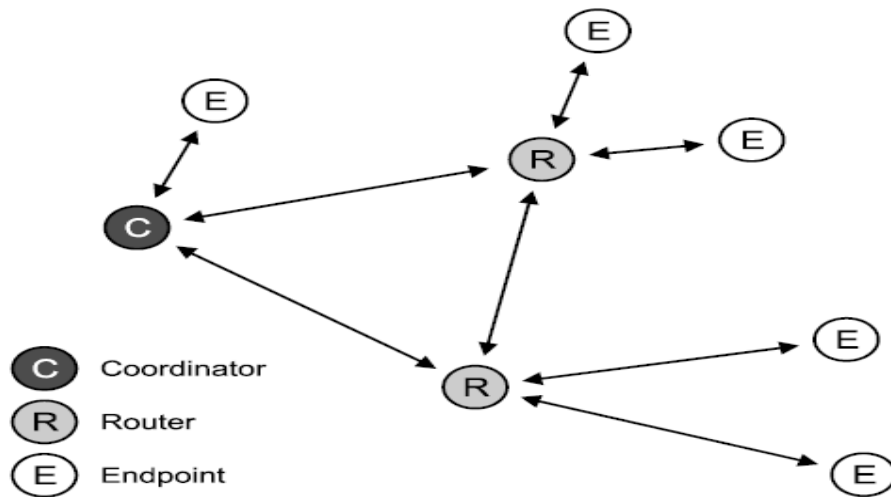
**Table (2-1): ZigBee stack layers**

<b>Zigbee Layer</b>	<b>Description</b>
PHY	Defines the physical operation of the ZigBee device including receive sensitivity, channel rejection, output power, number of channels, chip modulation, and transmission rate specifications. Most ZigBee applications operate on the 2.4 GHz ISM band at a 250kbps data rate.
MAC	Manages RF data transactions between neighboring devices (point to point). The MAC includes services such as transmission retry and acknowledgment management.
Network	Adds routing capabilities that allows RF data packets to traverse multiple devices (multiple "hops") to route data from source to destination (peer to peer).
APS	Application layer that defines various addressing objects including profiles, clusters, and endpoints.
ZDO	Application layer that provides device and service discovery features and advanced network management capabilities.

There are three different types of ZigBee devices as shown in figure 2.9:

- 1. Coordinator node.** The main function of the coordinator is to establish and maintain the network by assigning addresses to join devices and assisting with route building.
- 2. Routers.** These nodes route data between end nodes that cannot communicate directly to each other due to the long distances between them.

- 3. Endpoints.** These nodes collect data using sensors attached to them and control devices that are attached to them. They are typically connected to controllers, sensors and other devices for network interfacing.



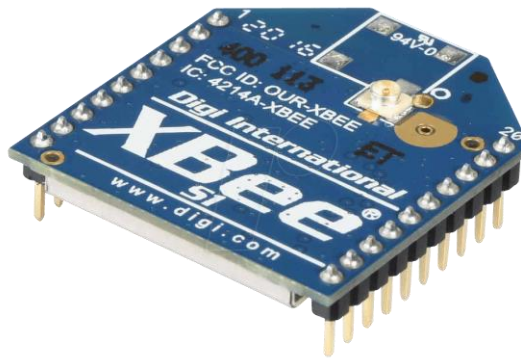
**Fig (2.9): ZigBee Node Structure**

### 2.4.3 XBee Networking

The XBee RF modules from Digi International is wireless transceiver. XBee uses a fully implemented protocol for data communications that provide features required for robust network communications in a wireless sensor network (WSN). Features such as addressing, acknowledgements, and retries ensure safe delivery of data to the intended node.

An XBee module has two addressing options: a fixed 64-bit serial number (MAC address) which cannot be changed, and a 16-bit assignable address that allows over 64,000 addresses on a network. For error checking and acknowledgements, the XBee modules use a checksum to ensure that the

communicated data contains no errors. Acknowledgements are sent to the transmitting node to indicate proper reception. Up to 3 retries are performed by default if acknowledgements are not received.

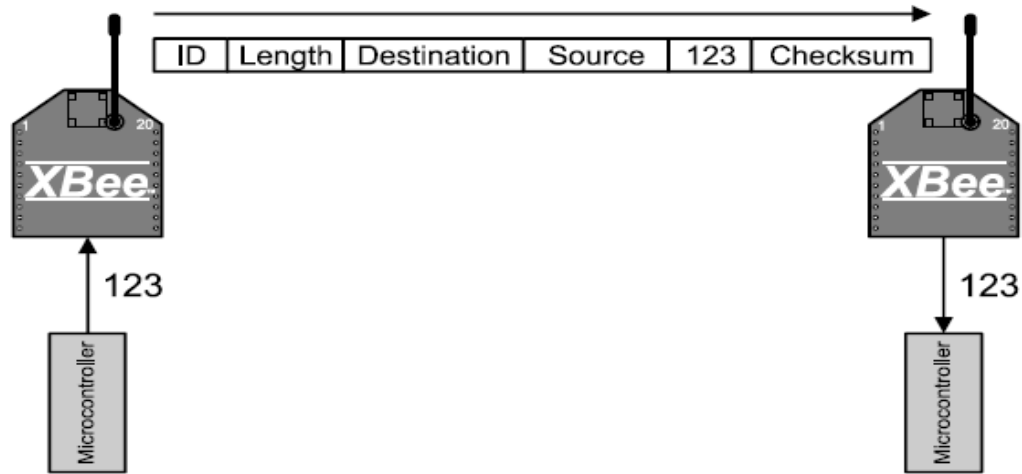


**Fig (2.10): XBee Module**

#### **2.4.3.1 XBee Communication Modes**

XBee supports two types of communication modes, the Transparent Mode (AT mode) and the Application Programming Interface (API) mode for sending and receiving data. In AT mode, the message itself is sent to the module and received by the controller. The protocol links between the two modules is transparent to the end user and it appears to be a nearly direct serial link between the nodes as illustrated in Figure (2.11). This mode allows simple transmission and reception of serial data. AT commands are typically used to configure the XBee, but the process requires placing the XBee module into command mode, then sending AT codes for configuration, and finally exiting the command mode. The transmission and reception are the raw data and the message itself. The message passed between the nodes

encapsulates the required information such as addressing and error checking bytes.

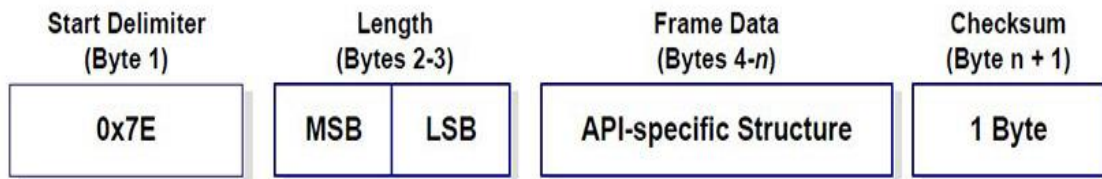


**Fig (2.11): Packaging data for delivery**

In API mode, the programmer packages the data with the required information, such as, the destination address, the type of packet, and the checksum value. The receiving node receives the data that includes the source address, the type of packet, the signal strength, and the checksum value. The advantages of using this mode includes the ability of building packets that include customized data, such as destination address and the receiving node can pull from the packet this information. Even though, the API mode requires programming intensive, it allows greater flexibility and increases reliability. The sender and receiver are not required to be in the same mode. Data may be sent in API Mode and received in AT mode or vice-versa. The mode defines the communications link between the PC or controller and the XBee modem, and not between XBee modules. Data

between XBee modules is always sent using the IEEE 802.15.4 LR-WPAN protocol.

When the API mode is enabled, the data frame structure is defined as follows:



**Fig (2.12): API frame structure**

- **Start delimiter.** The first byte of a frame consists of a special sequence of bits which indicate the beginning of a data frame. Its value is always 0x7E. This allows for easy detection of a new incoming frame.
- **Length.** These two bytes specify the total number of data bytes included in the frame. This excludes the start delimiter, the length, and the checksum.
- **Frame data.** Composed of the API identifier and the API identifier-specific data. The content of the specific data depends on the API identifier (also called API frame type).
- **Checksum.** This is the last byte of the frame. It helps test data integrity and is calculated by taking the hash sum of all the API frame bytes

that came before it, excluding the first three bytes (start, delimiter and length).

#### **2.4.3.2 XBee Operation Modes**

When not receiving or transmitting data, the RF module is in the idle mode. The module shifts into the other modes of operation under the following conditions:

**Transmit Mode.** When the serial data is received and is ready for packetization, the RF module will transmit the data. The destination address determines which node will receive the data. When data is transmitted from one node to another, a network-level acknowledgement is transmitted back across the established route to the source node. This acknowledgement packet indicates to the source node that the data packet was received by the destination node. If a network acknowledgement is not received, the source node will re-transmit the data.

**Receive Mode.** If a valid RF packet is received, the data is transferred to the serial transmit buffer.

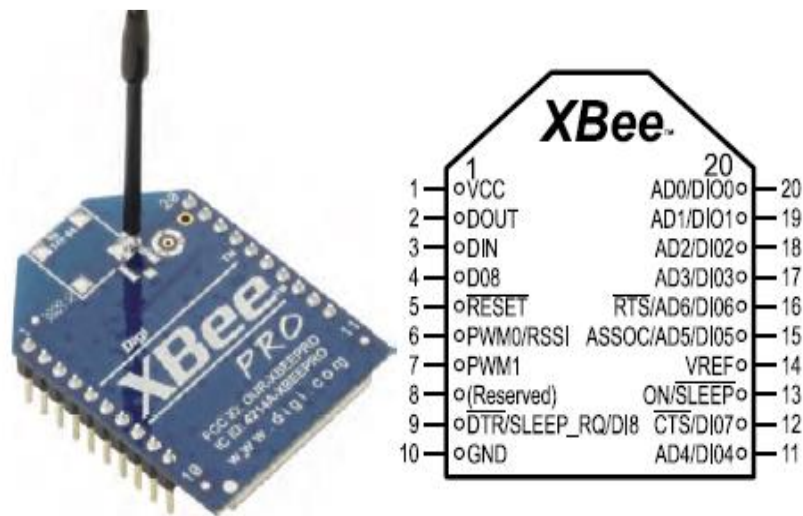
**Sleep Mode.** Sleep modes allow the RF module to enter a state of low power consumption when not in use. The XBee RF modules support both pin sleep (sleep mode entered on pin transition) and cyclic sleep (module sleeps for a fixed time).

### 2.4.3.3 XBee Module Styles

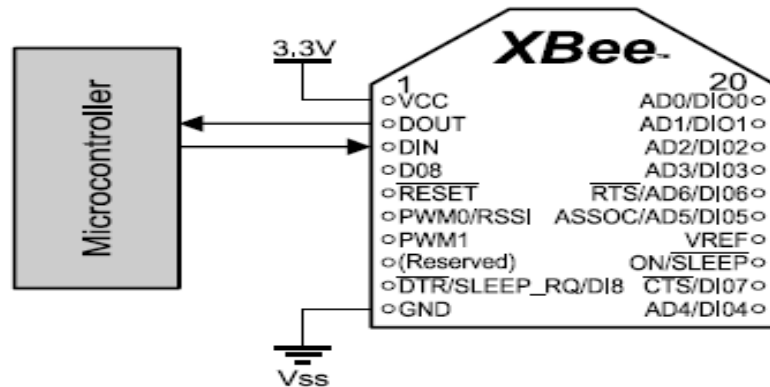
The XBee module comes in several versions but all have similar pinouts as shown in Figure (2.13). Differences between XBee versions include the power output, antenna style, operating frequency and networking abilities. The XBee is a 20-pin DIP module and available in two major versions XBee and XBee-Pro.

Figure (2.14) illustrates a typical microcontroller connection to the XBee [12].

Table (2-2) is a brief discussion of the pins and their functions on the XBee [12].



**Fig (2.13): XBee-Pro Module and Pinouts**



**Fig (2.14): Microcontroller Interfacing to XBee**

The 802.15.4 style of XBee (commonly called Series1) allows point-to-point networking and point-to-multipoint (one node to all nodes) networking. They use the IEEE 802.15.4 data link protocol to move data directly between 2 or more devices. All nodes in the network use same firmware version.

There are two styles of mesh networking protocols available: ZigBee and DigiMesh. Devices can be programmed with either protocol in either AT or API versions depending on the function of the device.

In summary, the XBee is a feature-rich RF module for use on a wireless sensor network where the IEEE 802.15.4 protocol reduces the work of the programming for ensuring data communications. Also, the XBee has many other features for use in a WSN beyond its networking ability.



**Table 2-2: XBee 802.15.4 Pin Assignments**

Pin	Name	Type	Function
1	VCC	P	2.8 V to 3.4 V
2	DOUT	O	Serial data output from XBee (received data)
3	DIN	I	Serial data input to XBee (data to transmit)
4	DO8	O	Digital data output 8
5	RESET	I	Reset module (low)
6	PWM0/ RSSI	O O	Pulse Width Modulated output Received Signal Strength Indication as PWM signal
7	PWM1	O	Pulse Width Modulated output
8	(Reserved)		
9	DTR SLEEP_RQ DI8	I I I	Data Terminal Ready: handshaking for firmware updates (low) Sleep Request: A high places XBee in sleep mode when configured Digital Output 8
10	GND	G	Ground (Vss)
11	AD4 DIO4	A IO	Analog to Digital Input 4 Digital Input/output 4
12	CTS DIO7	O IO	Clear to Send output for controller handshaking (low) Digital Input/Output 7
13	ON/SLEEP	O	Digital output, status indication: High = Awake, Low = Sleep
14	VREF	A	Analog to Digital reference voltage
15	ASSOC AD5 DIO5	O A IO	Associated indication when joining a network Analog to Digital Input 5 Digital Input/Output 5
16	RTS AD6 IO6	I A IO	Ready to Send Handshaking input (Low) Analog to Digital Input 6 Digital Input/Output 6
17- 20	AD3-AD0 DIO3- DIO0	A IO	Analog to Digital Input 3 to 0 Digital Input/Output 3 to 0

Pin Type: P = Power, G = Ground, I = Input, O = Output, A = Analog Input

In general, some environmental conditions are measured by the sensor nodes deployed with a spatial density and at a sampling rate specified by the application.

The signals are processed by the microprocessor of the sensor node to get useful information, then output of the sensor node is transmitted to the access point either through a direct communication or through other sensor nodes. Eventually, data from all sensor nodes is gathered through the access point to the base station for further analysis. Software running on the base station analyses the collected data to extract the desired information about the traffic parameters (detection, speed and classification of detected vehicle). This information will be changed to some meaningful format (XML or JSON) and will be available to the end user or some other system.

In this thesis, we designed our system to use the anisotropic magneto-resistive (AMR) magnetic sensors as a sensing device. However, we didn't use the sensor in the work of this thesis. This sensor is capable of measuring 1/10000 of the earth magnetic field which makes it capable of detecting small perturbation in the earth magnetic field caused by passing vehicles (appendix A provides more theoretical details about the AMR Sensor).

The microprocessor to be used in the sensor node is the Arduino board. And the XBee modules will be used for radio communication with the access points. This module has an outdoor line-of-sight range of 100 m, required power of 150 mW, and a radio frequency data rate of 250 kb/s.

## **2.5 Literature Review**

Vehicle detection has been an active research topic for decades and a full review of all the methods is out of the scope of this thesis. In this section, we summarize some related papers to the proposed system.

In 2018, T. Thamaraimanalan, S.P. Vivekk, G. Satheeshkumar and P.Saravanan [2], introduced an android intelligent irrigation system. The system was implemented based on the idea of remote monitoring without human intervention. The project is based on an intelligent microcontroller to control irrigation system irrigation system controller. The implemented to (remotely irrigate a field by an operator) was to design an irrigation controller through a mobile app to promote remote practices and use of fuzzy logic and neural networks supported by the hardware nodes present in the fields and several parameters that are to be selected on the basis of the stage of the crop. The Implemented consisted of soil moisture to measure the amount of moisture in the soil and with the temperature sensor, it is interfaced with an Arduino Microcontroller. According to the sensor readings thus obtained on the mobile app, the controller will evaluate the decision and the Motor will be activated through a mobile phone and the water will be delivered from the pump through a relay.

The key findings from this paper are that use of manual/automated method for delivering water with the aid of an android app.

In year 2010, S. Kaewkamnerd, J. Chinrungrueng, R.Pongthornseri, and S. Dumnin [11], introduced Vehicle classification based on magnetic sensor that is the system consists of a low power microprocessor together with AMR magnetic sensors and an RF transceiver. Vehicle classification tree based on above extraction features and it focuses on low computational

feature extraction and classification processes suitable for implementing on microcontroller.

In May 2010, W. Zhang, G. Tan, H. Shi, and M. Lin [13] introduced real time vehicle surveillance, utilize the advances in wireless sensor networks to develop a magnetic signature and length estimation-based vehicle classification methodology with binary proximity magnetic sensor networks and intelligent neuron classifier. In this algorithm, use of low cost and highly sensitive magnetic sensors to measure the magnetic field distortion when vehicle crosses the sensors and detect vehicle via an adaptive threshold.

The vehicle length is estimated with the geometrical characteristics of the proximity sensor networks, and finally identifies vehicle type from an intelligent neural network classifier.

In 2008, S. Jeng and S. Ritchie studied real-time vehicle classification using inductive loop signature data present a method for vehicle identification based on analyzing the inductive signatures in the frequency domain instead of working in the time domain. Transform domain will be used for vehicle classification by means of a simple threshold-based method. However, the accuracy rate is not 100% and can vary from 40% to 100%, depending on the amount of “problematic” data present in the sensor readings and the class of vehicle under consideration [14].

Vehicle detection and classification based on feature extraction from camera systems have been developed by many researchers [24], [19]. The research in [19] presented model based and fuzzy-logic approaches to improve the reliability of such systems.

An evaluation of three commercial camera-based vehicle detection systems is presented in [10] under adverse weather conditions of snow, fog, and rain. The results therein show that the performance of such systems deteriorates under adverse weather, particularly under snow conditions in both daytime and night time. Increases in false activations by up to 90% and in missed calls by up to 50% were shown to occur in adverse weather.

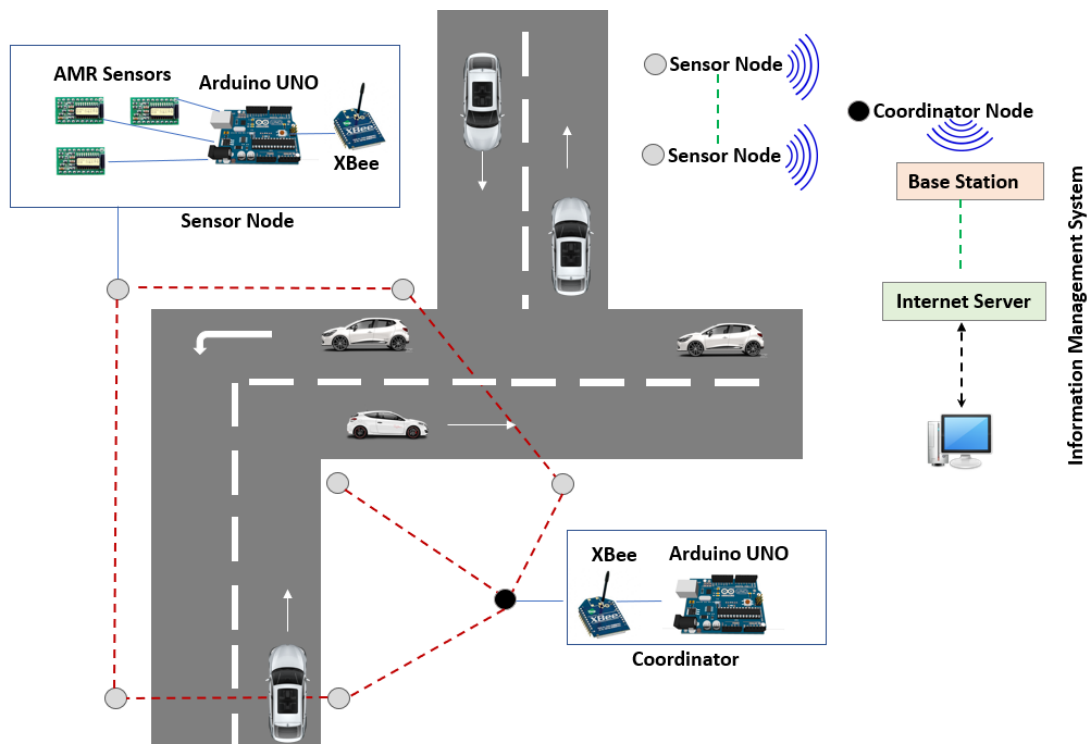
In 2008, D. Nan, T. Guozhen, M. Honglian, L. Mingwen, and S. Yao [15], introduced Low-power vehicle speed estimation algorithm based on WSN. A three nodes model to capture the vehicle speed based on the two nodes detection model using the Magnetic sensitive signal with WSN. In the model, the Collecting Node A and B were mainly detected the vehicle and transmitted the detection information to the third node, the Detecting Node. The Detecting Node was the key node of the mode. The speed calculation and the command of the whole system were executed on it [15].

The focus of this thesis is not on the particular technology that is used in counting and measuring the traffic parameters on the road. But the focus is on the automation of gathering the traffic parameters from a particular location on the road in a storage place where it can be processed and shared by other users or application.

## Chapter Three

### System Design and Architecture

In this chapter, we show the design and architecture of the traffic monitoring system. The design consists of software and hardware components. The hardware components consist of a WSN, magnetic sensors, and a base station. The software component includes a database to store the traffic parameters, a dashboard to visualize the traffic parameters (detection, speed and classification of detected vehicle), and a web service interface to allow accessing the traffic data from other application. Figure (3.1) illustrates the architectural design of the traffic monitoring system.



**Figure (3.1): System Architecture**

As shown in Figure (3.1), the WSN consists of four main elements: sensor nodes, a gateway, base station and an internet server. Sensor nodes gather the data from their attached sensors and process it to extract traffic parameters (count, shape, and speed). Then, through wireless links, they communicate this data through a gateway (coordinator node) to the base station. The gateway is connected to a base station computer which has unlimited computational resources, enhanced radio communication and unlimited power supply. The application running on the base station gathers the information from the WSN and communicates this information to a web server on the cloud. The web server saves the data in a database.

In the following subsections, we further describe these elements and the main technologies that we used to implement them.

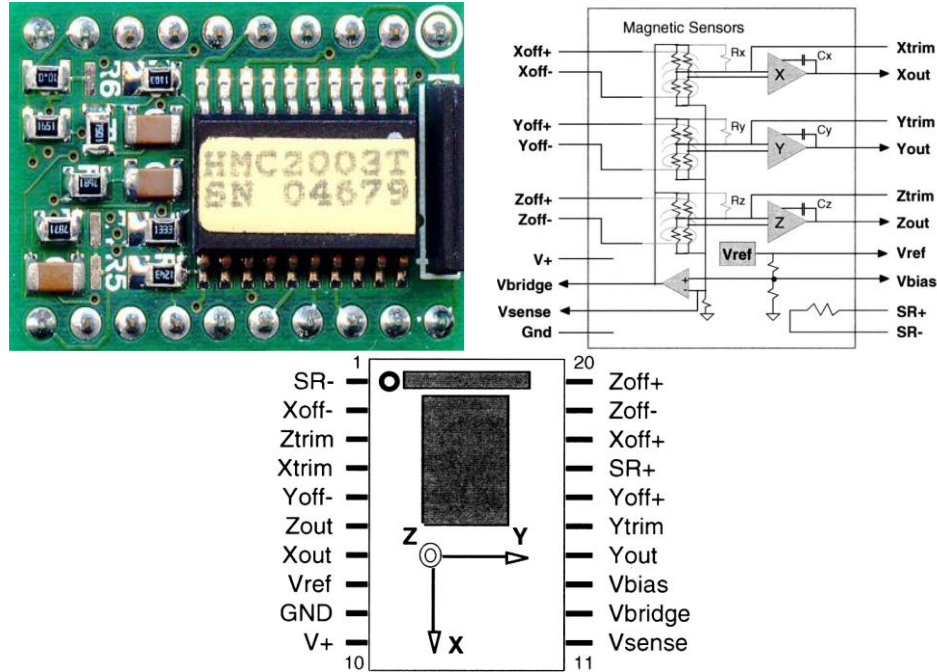
### **3.1 Sensing system**

Our approach in this thesis considers the use of magnetic sensors to measure the traffic parameters. In this approach, the earth magnetic field at a particular location on the side of the road is measured using a magnetic sensor. Once a vehicle passed by the sensor, the earth magnetic field in that location is perturbed. The perturbation in the earth magnetic field can be detected. Consequently, the number of vehicles passed by the sensor, the speed of the passing vehicle, and the size of the vehicle can be inferred from the perturbed signals measured by multiple sensors as described at the end of this section.

The design of our system considers the use of the AMR Honeywell HMC2003 [8] magneto resistive sensor shown in Fig 3.2 that will be used in our future work. The HMC2003 is a highly-sensitive, three-axis magnetic sensor assembly used to measure low magnetic field strengths. It is a combination of Honeywell's most sensitive sensors HMC1001 and HMC1002. HMC2003 provides maximum user flexibility due to its analog interface with critical nodes available for pin interface. HMC2003 is a three-axis magnetic sensor uses three permalloy magneto-resistive sensors to measure the strength and direction of an incident magnetic field. This sensor can measure the field along the length, width, and height (X, Y, Z axis) from the hybrid. It has a reference voltage of 2.5 Volt, with a power supply range of 6-15 Volts.

HMC20003 is configured in a 20-pin layout as shown in Fig 3.2. Its dynamic range is from less than 40 micro gauss to  $\pm 2$  gauss. It is integrated with magnetically coupled straps to eliminate unwanted magnetic fields. These sensors can measure the changes in the field accurately even if the target is away from the sensor. HMC2003 sensors can be used to measure the presence, magnitude, and direction of a magnetic field. They can sense changes in the magnetic field in the presence of ferromagnetic objects. They can also be used to measure the earth's field for navigation and compassing purposes.





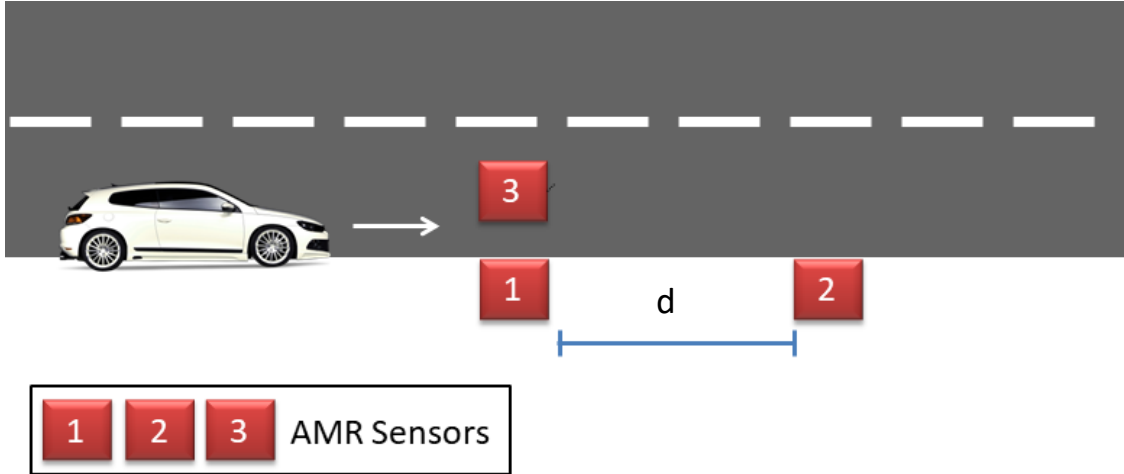
**Fig (3.2): (left) Honeywell's HMC2003 sensor**

(middle) Block Diagram

(right) Pin Diagram

Due to its high sensitivity, even at very low field ranges, the HMC2003 sensor is chosen over other AMR sensors sold by other manufacturers to be integrated in our system.

Figure (3.3) shows the configuration of the proposed sensing system which includes three three-axis AMR sensors placed on the side of the road to detect passing vehicles for the purpose of counting them and estimating their speed and size [5].



**Figure (3.3): Configuration of sensing system. Sensor 2 is placed longitudinally from sensor 1. Sensor 3 is placed vertically above sensor 1.**

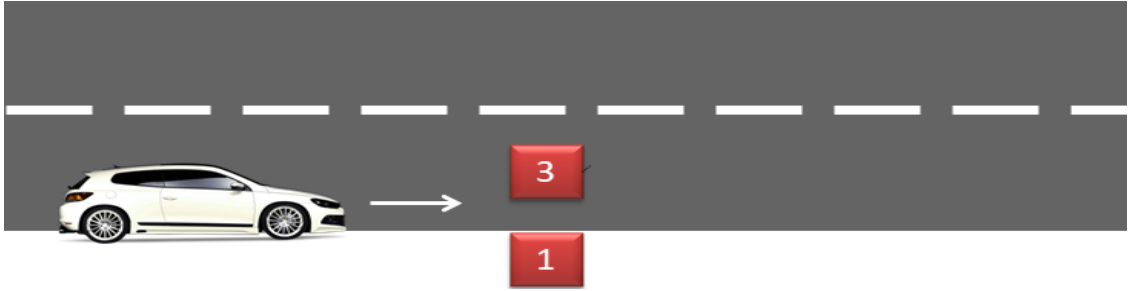
- ***Vehicles detection and counting.*** The signals along the z-axis have almost the same patterns for a large variety of vehicles. Hence, the magnetic readings of the z-axis of sensor 1 can be used for detecting and counting the passing vehicles.
- ***Speed estimation.*** The speed of the passing vehicle can be estimated by using two longitudinally spaced magnetic sensors (sensor 1 and sensor 2) as shown in Figure (3.3). Assuming the distance between the two sensors is  $d$ , the speed of the vehicle can be estimated from the detection time of the two sensors. If the detection time of sensors 1 and 2 are  $T_1$  and  $T_2$ , the speed of vehicle can be calculated as:

$$v = \frac{d_{1-2}}{T_2 - T_1},$$

where  $d_{1-2}$  is the distance between sensors 1 and 2

- ***Vehicles Classification.*** The proposed classification approach is based on using the size of magnetic disturbance. Multiple magnetic sensors can be placed at different location from the road. Then a correlation analysis or machine learning classification approach can be used to provide an estimate of the vehicle size.

In Figure (3.4), we show a possible placement of a second sensor (sensor 3) vertically above sensor 1. The magnetic signals along the z-axis have patterns for different types of vehicles. Therefore, magnetic readings of the z-axis from multiple sensors can be used to give an estimate of the vehicle size.



**Figure (3.4): Sensors configuration for estimating the dimension of the vehicle**

Each node placed in a particular location gathers the magnetic field disturbance information resulting from vehicles passing from multiple attached sensors. This information can be transferred to coordinator node using radios signals.

### **3.2 Wireless Sensor Network**

An important part of the system architecture is a WSN as shown in Figure (3.1). A WSN is a group of wireless sensor nodes distributed at

different locations by the road side. Multiple magnetic sensors can be attached to a sensor node to sense perturbation in the earth magnetic field. Then, these perturbation signals can be used to determine the traffic parameters. Once a sensor node identified a passing vehicle, it transfers this information to the master node. The master node is a special node that acts as a sink where information from all wireless nodes are transferred to this node. The master node (Coordinator) is connected to the base station computer. The base station computer is connected to the web application, a shared place for processing, managing the data, and saving the data in a database.

The Wireless Sensor Network is developed based on the open source hardware platform Arduino Uno and XBee radios for wireless communications.

Arduino Uno microcontroller is an open-source computing platform based on a simple microcontroller board, and a development environment for writing software, the Arduino Software (IDE) which is based on processing language.

Arduino can be used to develop interactive projects, taking inputs from a variety of sensors, and controlling a variety of lights, motors, and other physical outputs. Arduino projects can be stand-alone, or they can communicate with software running on the computer. Arduino can read inputs from a sensor or a message and produce an output that can activate an action in the environment such as turning a motor on or off.

Wireless nodes communicate with each other wirelessly through XBee radio modules. XBee modules simply send data and receive data on the provided frequency. This helps in dealing with issues such as media access rules, data delivery verification, error checking, and in multi-node networks (which node will accept and use the data).

Our choice to use Arduino is due to its flexibility to be customized and extended. It offers a variety of digital and analog inputs includes serial interface, and digital and PWM outputs. It is also easy to use and implement, since it can be connected to a computer via USB port and communicates with a computer using standard serial protocol.

The intended design of the system is to include the HMC2003 magnetic sensor as described in previous section.

In order to allow the XBee module to enter sleep modes and save power, it needs to be configured properly. The coordinator XBee is set to be awake at all time and simply just waiting for data. The XBee of sensor node are set to be on cyclic sleep mode. This allows the XBee to sleep for a set amount of time.

In order to reliably send data between the coordinator and the sensor nodes, a custom packet is created. The first byte in the packet is the header byte with a fixed value of 0xFF, the next field is the node ID, followed by the data bytes which includes the signals along x, y and z axis, followed by the last byte which is a checksum byte as shown in figure (3.5).



**Fig (3.5): Sensor node data packet**

### **3.2.1 Sensor Node**

Each sensor node reads the analog values of the three components of the magnetic field and sends them to its Arduino microprocessor. These values can be used to detect if a vehicle is passed. If these values indicate a passing vehicle, the current time and the digital values of the three components are sent to the coordinator nodes using the XBee library **send** function. The function assembles the messages into an API frames. The following steps can be used by sensor node:

**Step 1:** Start

**Step 2:** Header byte and Id is assigned to sensor node

**Step 3:** Sense magnetic field in three axes (x, y and z)

**Step 4:** Send Analog values of sensed magnetic field in three axes (x, y and z).

To Arduino

**Step 5:** Does the value of z-axes indicate a vehicle movement?

Yes: go to step 6

No: go to step 9

**Step 6:** Set detection time

**Step 7:** Inbuilt ADC in Arduino converts the values of x, y and z to digital value

**Step 8:** Calculate checksum

**Step 8:** Send Header byte, Id, detection time, digital value values of (x, y and z) and checksum to coordinator

**Step 9:** End.

### **3.2.2 Coordinator Node**

The coordinator node collects the data from the sensor nodes and makes some computations on them. Then it saves the results to be accessed later as a Web service. The following steps are the performed by coordinator node:

**Step 1:** Start

**Step 2:** Receive Header byte, Id, detection time, digital values of (x, y and z) and checksum

**Step 6:** Calculate checksum

**Step 3:** Does calculated checksum value = received checksum value?

Yes: go to step 4

Yes: go to step 9

**Step 4:** Add one to counter

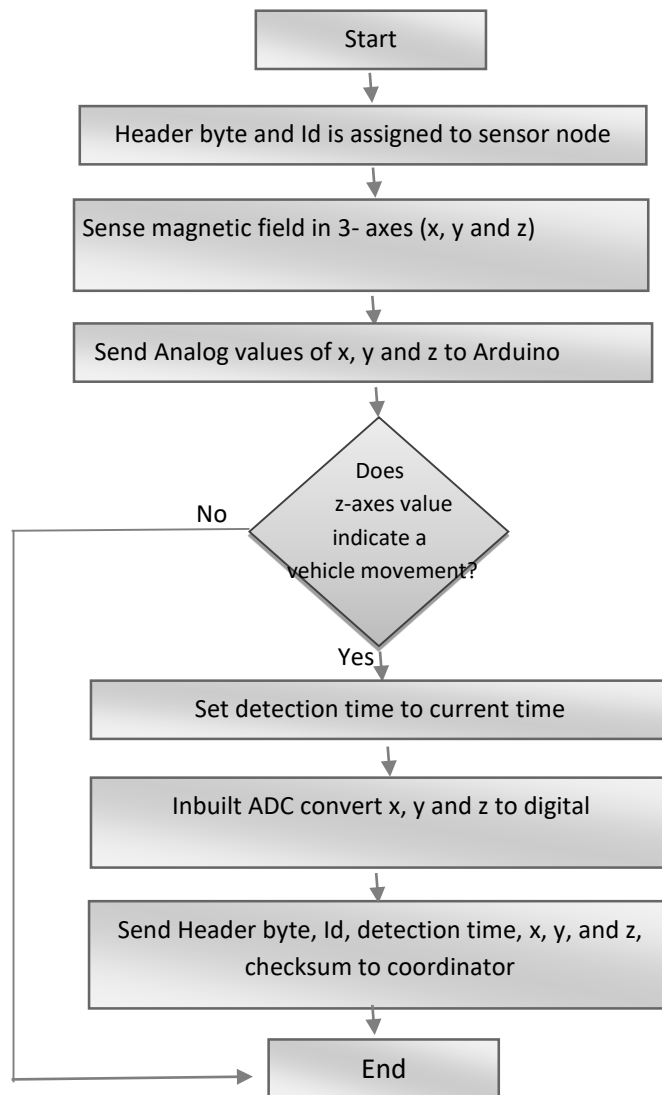
**Step 5:** calculate vehicle speed from

$$v = \frac{\text{distance between sensor node1 and sensor node2}}{\text{Received detection time from sensor2} - \text{Received detection time from sensor1}}$$

**Step 6:** calculate the difference between the digital values of (x, y and z) received from sensor1 and sensor3 to determine the class of vehicle

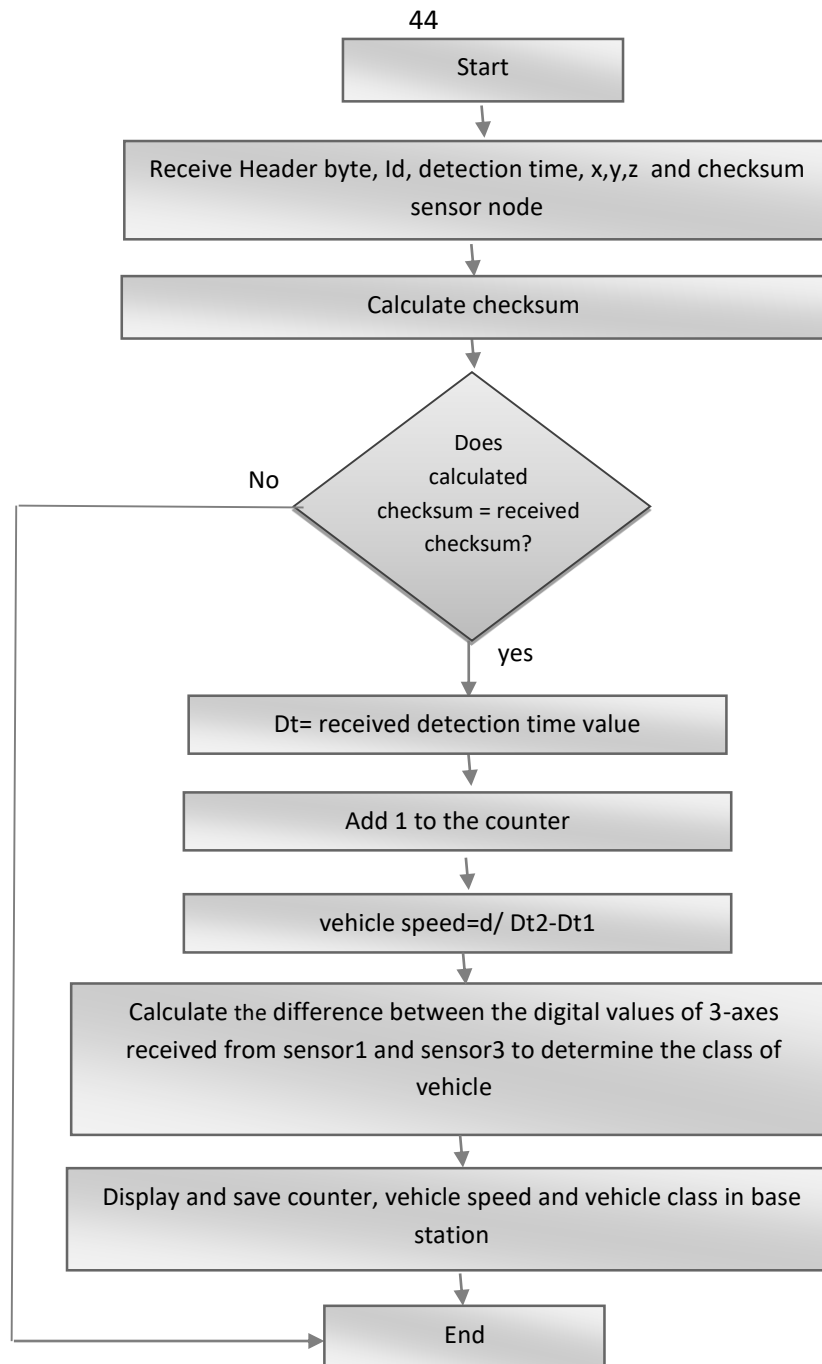
**Step 8:** Save counter, vehicle speed and vehicle class in base station

**Step 9:** End.



**Fig (3.6): Sensor node processes**





**Fig (3.7): Coordinator Process**

### 3.2.3 Base Station Application

The base-station application listens to the serial port. Once the application reads a header byte, it knows it is the beginning of a new packet and it disassembles the packet and does the appropriate calculations to

determine source node ID and the received data. The application also shows a simple visual interface and displays the received data. It also tags the data with the real time and save it in database. The application allows the search for the movement of vehicles within a specified date and time and displays the results as statistical charts. The database consists of a single table with the attributes (sensor ID (transmitter), vehicle classification, detection date, detection time, and vehicle speed).

## Chapter Four

### System Implementation and Results

In this chapter, we describe the implementation of our system. This includes, the implementation of the WSN, and the base station application and its interfaces as mentioned in the previous chapter.

The implementation of a wireless sensor networks should meet several requirements include low power consumption, contain reliable sensors interface, transmit small amounts of data, and easy to deploy. Additional requirements are:

- **Radiolocation.** The system should be able to identify the location of any wireless sensor node.
- **Single Destination points for data.** All sensors send their data to one central point, the base station. This will simplify the requirements of the routing algorithm.
- **Ease of maintenance.** Maintaining the system should be achieved centrally, therefore the base station must be capable of determining if maintenance is required.
- **Easy access to sensor data.** The base station stores all sensor data and accessing this data can be achieved through this application.
- **Mechanism to support dynamic environment.** The working conditions in road monitoring systems are not static.

## **4.1 Hardware and Software to implement the Wireless Sensor Network**

In this section, we show the components of the system, how it could be used in the traffic monitoring, and the corresponding hardware and software specifications of the prototypes that we developed, as well as the communication protocols.

The main components of a Wireless Sensor Node are: the microcontroller, Radio Frequency transceiver (RF), power source, and several attached sensors. There are several technologies that can be used to implement the WSN. The main difference between these technologies is the tradeoff between the ease of use and the cost. Other difference is the communication protocol that they support. In this thesis, we used the XBee modules for the short-range low-power wireless communication. For the microcontroller, we used the Arduino Uno. For the environment sensors, we considered to use the HMC2003 magnetic sensor in the future work. The XBee modules and the HMC2003 magnetic sensors are discussed in Chapter [24]. These and other electronics parts that we used are discussed further below in more details.

### **4.1.1 Arduino Uno**

Arduino is an open source programmable circuit board that can be used for building electronics projects. This board contains a microcontroller which is able to be programmed to sense and control objects in the physical world.

There are many versions of the Arduino boards introduced in the market include Arduino Uno, Arduino Due, Arduino Leonardo, Arduino Mega. In this thesis we used the Arduino Uno.



**Fig (4.1): Sensor Node**

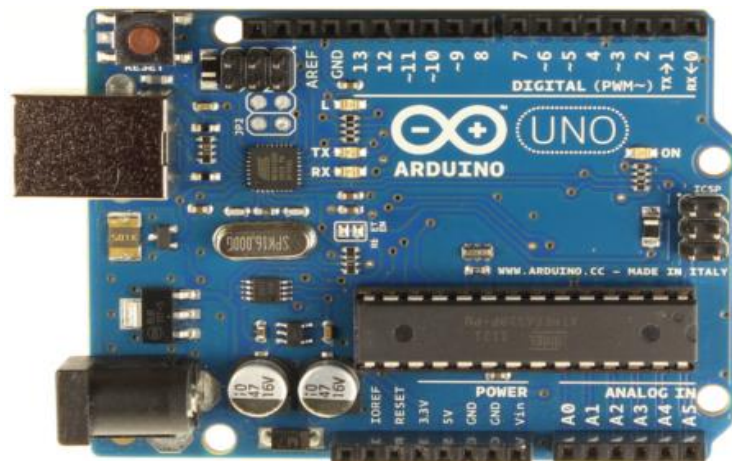
Some of the key characteristics why these microcontrollers are particularly suitable to WSNs systems are: their flexibility in connecting with other devices (like sensors), their low power consumption, and their built-in memory.

Arduino microcontroller is the option chosen which covers all requirements. It allows the designers to control and sense the external electronic devices in the real world. Therefore, it is considered the core of a wireless sensor node. It collects data from the sensors, processes this data, decides when and where to send it, and receives data from other sensor nodes. It has to execute various programs, ranging from signal processing and communication protocols to application programs.

Arduino consists of a circuit board, which can be programmed using the Arduino IDE (Integrated Development Environment). The Arduino IDE is used to write and upload programs to the physical board. The board is able to read analog or digital input signal from different sensors and turn it into an output. [1]

Unlike most programmable circuit boards, Arduino does not need an extra piece of hardware in order to load the code onto the board. It simply uses the USB port. The Arduino IDE supports using special rules of code structuring making it easier to program.

Arduino Uno (as shown in Fig 4.2) comes with USB interface, 6 analog input pins, 14 I/O digital ports that allow external connections with circuit on board. These pins provide the flexibility and ease of use to the external devices that can be connected through these pins. There is no interface required to connect the devices to the board. Simply, plug the external device into the pins of the board that are laid out on the board in the form of the header. The description of each pin is shown in Table (4-1) [1].



**Fig (4.2): Arduino UNO**

**Table 4-1: XBee 802.15.4 Pins**

Category	Pin Name	Details
Power	Vin, 3.3V, 5V, GND	Vin: Input voltage to Arduino when using an external power source. 5V: Regulated power supply used to power microcontroller and other components on the board. 3.3V: 3.3V supply generated by on-board voltage regulator. Maximum current draw is 50mA. GND: ground pins.
Reset	Reset	Resets the microcontroller.
Analog Pins	A0 – A5	Used to provide analog input in the range of 0-5V
Input/Output Pins	Digital Pins 0 - 13	Can be used as input or output pins.
Serial	0(Rx), 1(Tx)	Used to receive and transmit TTL serial data.
External Interrupts	2, 3	To trigger an interrupt.
PWM	3, 5, 6, 9, 11	Provides 8-bit PWM output.
SPI	10 (SS), 11 (MOSI), 12 (MISO) and 13 (SCK)	Used for SPI communication.
Inbuilt LED	13	To turn on the inbuilt LED.
TWI	A4 (SDA), A5 (SCA)	Used for TWI communication.
AREF	AREF	To provide reference voltage for input voltage.

### 4.1.2 Arduino XBee Shield

The Arduino shield allows the Arduino board to communicate wirelessly using ZigBee. It can be used as a SERIAL/USB replacement or it can be placed into a command mode and configure it for a variety of broadcast and mesh networking options. The shields break out each of the XBee's pins to a through-hole solder pad. It also provides female pin headers for use of digital pins 2 to 7 and the analog inputs, which are covered by the shield (digital pins 8 to 13 are not obstructed by the shield).

The XBee shield has two jumpers. These determine how the XBee's serial communication connects to the serial communication between the microcontroller and FTDI USB to-serial chip on the Arduino board.

With jumpers in the XBee, the DOUT pin of the XBee module is connected to the RX pin of the microcontroller; and the DIN is connected to TX (DOUT and DIN pins are used to send and receive data which are explained in Section 4.1.1).

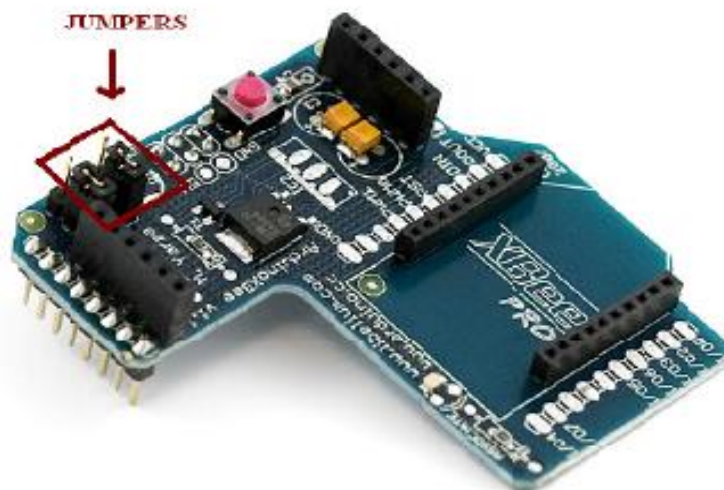


Fig (4.3): Arduino XBee Shield

### 4.1.3 Arduino IDE

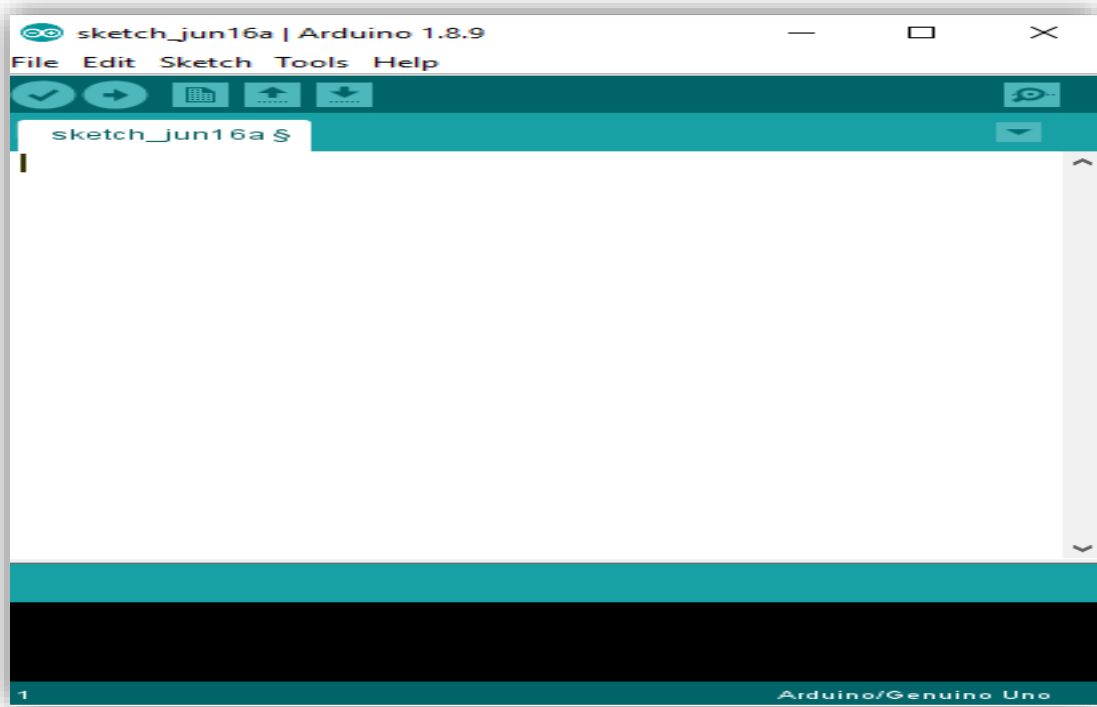
The Arduino integrated development environment (IDE) is an official software introduced by Arduino.cc, that is mainly used for editing, compiling



and uploading the code in the Arduino Device. Almost all Arduino modules are compatible with this software that is an open source and is readily available to install and start compiling the code on the go. IDE contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions, and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and is also capable of compiling and uploading programs to the board.

The Arduino programming language is based on a very simple hardware programming language called processing, which is similar to the C language. The Arduino IDE comes with a software library called "Wiring" from the original Wiring project [1] which makes many common input/output operations much easier.

The interface of Arduino IDE is shown in figure (4.4)



**Fig (4.4): Arduino IDE Interface**

#### **4.1.4 X-CTU**

X-CTU is free software provided from Digi which allows reading and setting the XBee module parameters, such as the Node Identification, destination address, PAN Id, operating channel, or applying configuration changes, etc. It offers the following features:

- Integrated terminal window.
- Provides a range of test tools.
- Displays the Received Signal Strength Indicator (RSSI).
- Displays both ASCII and Hexadecimal characters in terminal window.

- Compose test packets for test transmissions.
- Automatically detect module type.

The interface X-CTU is shown in figure (4.5)

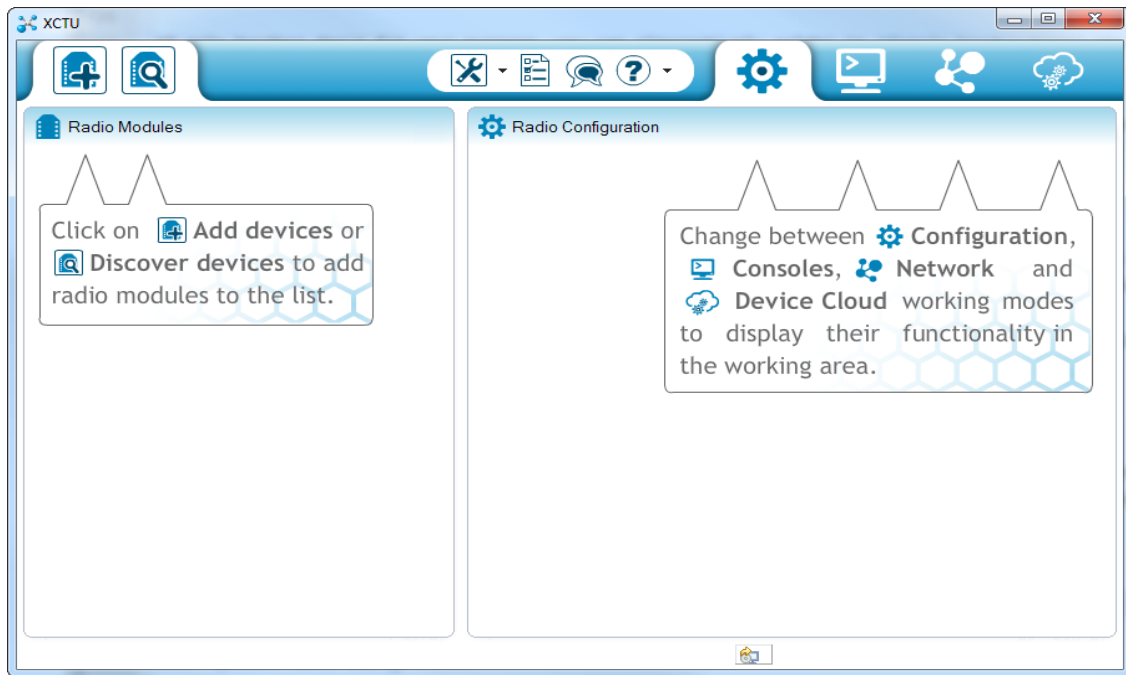


Fig (4.5): X-CTU Software Interface

#### 4.1.5 Power Source

The nodes should work autonomously. This means that there should be a power management stage to ensure such a condition. We can use battery or solar panel to accomplish this.

### 4.2 Wireless Sensor Network implementation

In order to implement the network, two communication tests were performed. First, we tested the communication using radios between two XBee modules. A “simple Zigbee chat session” as FALUDI calls it in [21].

The other test is the communication between two Arduino boards where the first board sends a message through the serial to its XBee module. The XBee module transmits the message using radio signal to the XBee in the receiver node. Consequently, the XBee in the receiver node forwards the message through the serial port to the Arduino board in the receiver node.

#### 4.2.1 XBee to XBee Communication Test

The goal of this test is to transmit real-time text messages from one XBee module to another. For this test, I used:

- X-CTU software.
- Two (2) XBee modules.
- Two (2) XBee explorer USB modules.

The configuration of the XBee modules involves setting the Channel, PAN ID, and Address values as follows:

**Channel.** XBee modules must be on the same channel to communicate with each other.

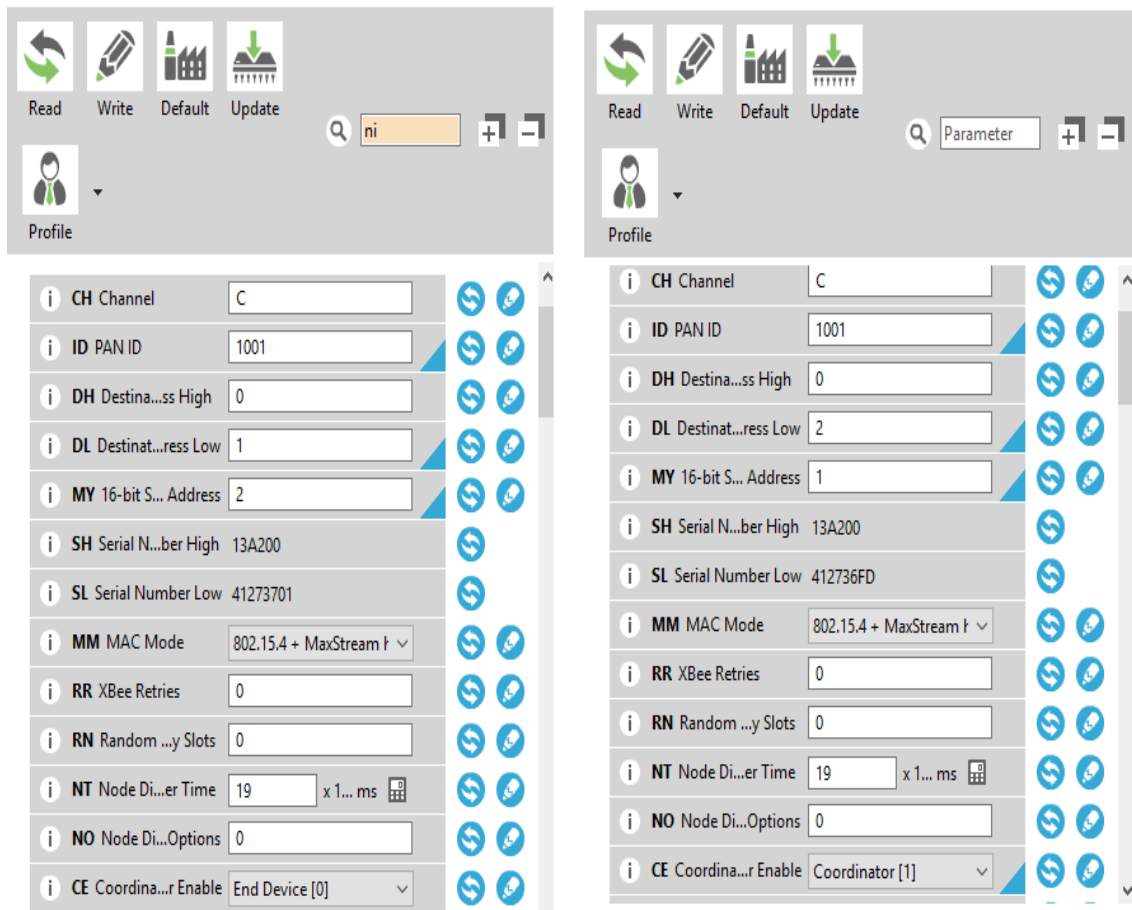
**PAN ID.** XBee modules must share the same PAN ID to communicate with each other (available values are between 0 and 0xFFFF).

**Addressing.** Each XBee has a source address and a destination address. An XBee's destination address specifies to which address it can send data. Using the X-CTU software, one XBee was configured as a coordinator and named

“Coordinator”. The other XBee was set as an end device and labeled “EndDevice”. Other Parameters is configured as shown in Figure (4.6).

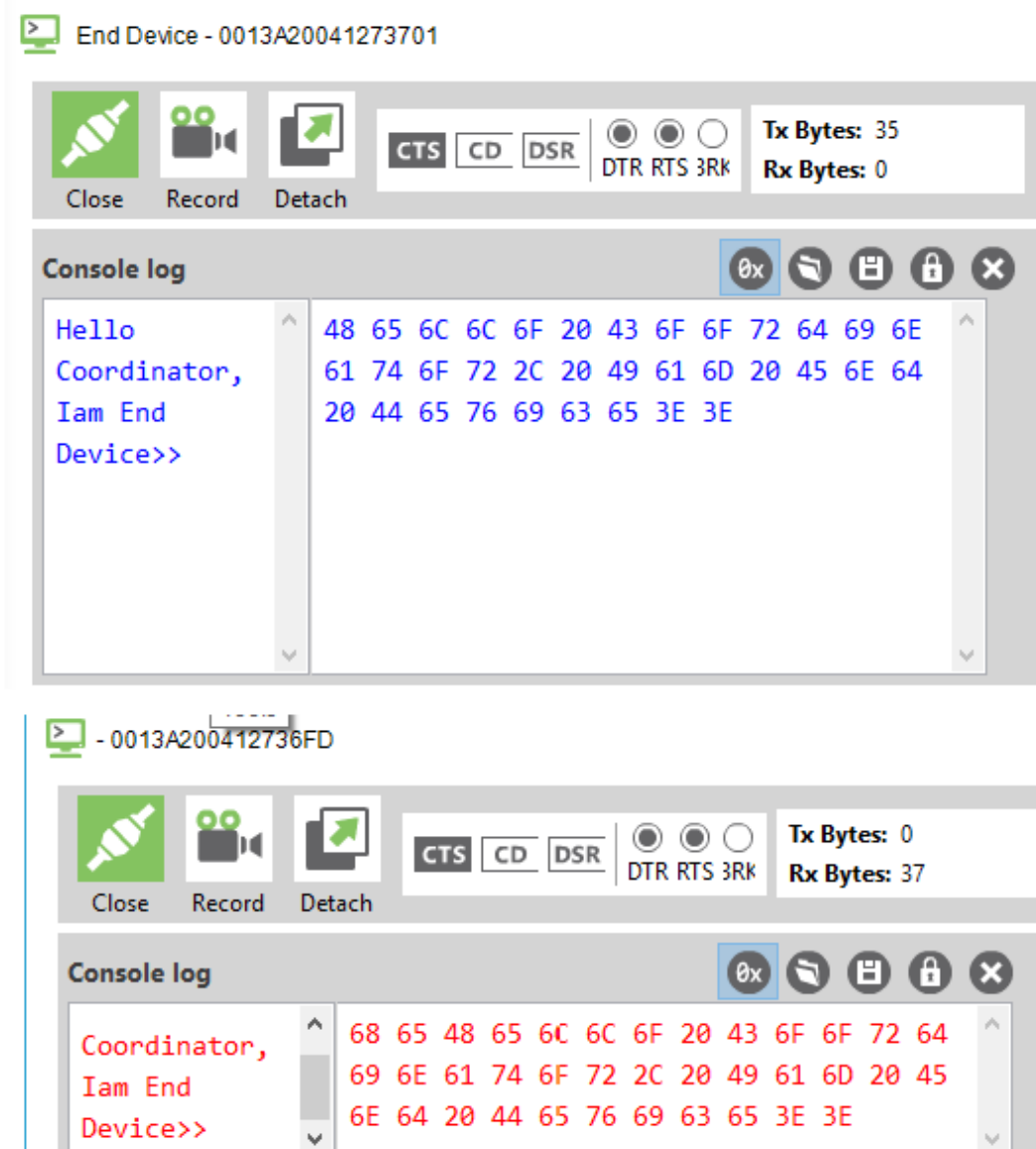
**Table 4-2: Configuration of XBee modules for XBee to XBee test communication**

	channel number (CH)	PAN ID (ID)	Destination High (DH)	Destination Low (DL)	MY Address
Coordinator	C	1001	0	2	1
End Device	C	1001	0	1	2



**Fig (4.6): Coordinator and End device XCTU configuration**

Next, I proceeded to open a console window in the X-CTU and created the message that was sent from the end device to the coordinator. Fig. 4.7:



**Fig (4.7): Message sent from End device to Coordinator**

Finally, as expected, the communication between the XBee modules works fine.

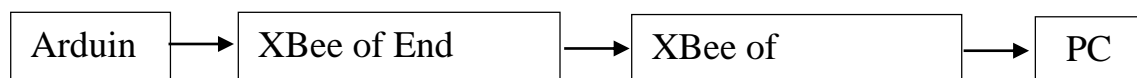
#### **4.2.2 Arduino to Arduino Communication Test**

A simple sketch with the message "Hello from End device" was programmed on the Arduino board (Figure (4.8) shows the script at the

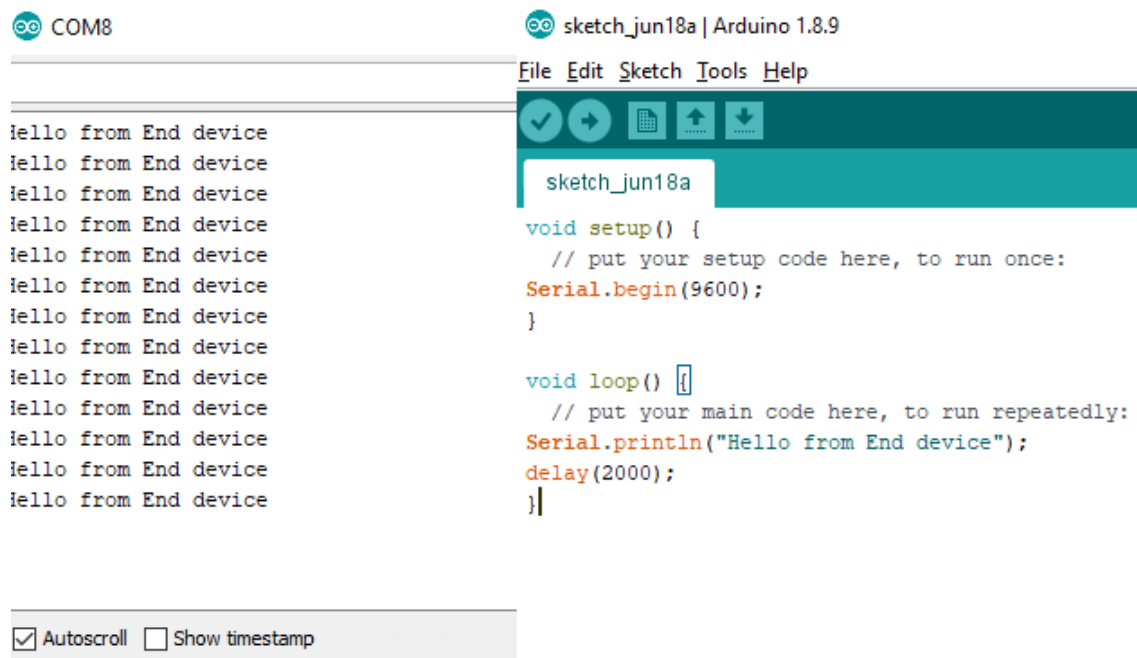
Arduino IDE and on the Terminal). The message that end device is transmitting to the coordinator every two seconds.

The configured XBee of end device is placed on the XBee shield, and the shield with XBee is placed on the Arduino board. It's important to change the jumper's position from USB to XBEE. The configured XBee of Coordinator is placed on the XBee explorer USB, and the XBee explorer USB XBee is connected to PC.

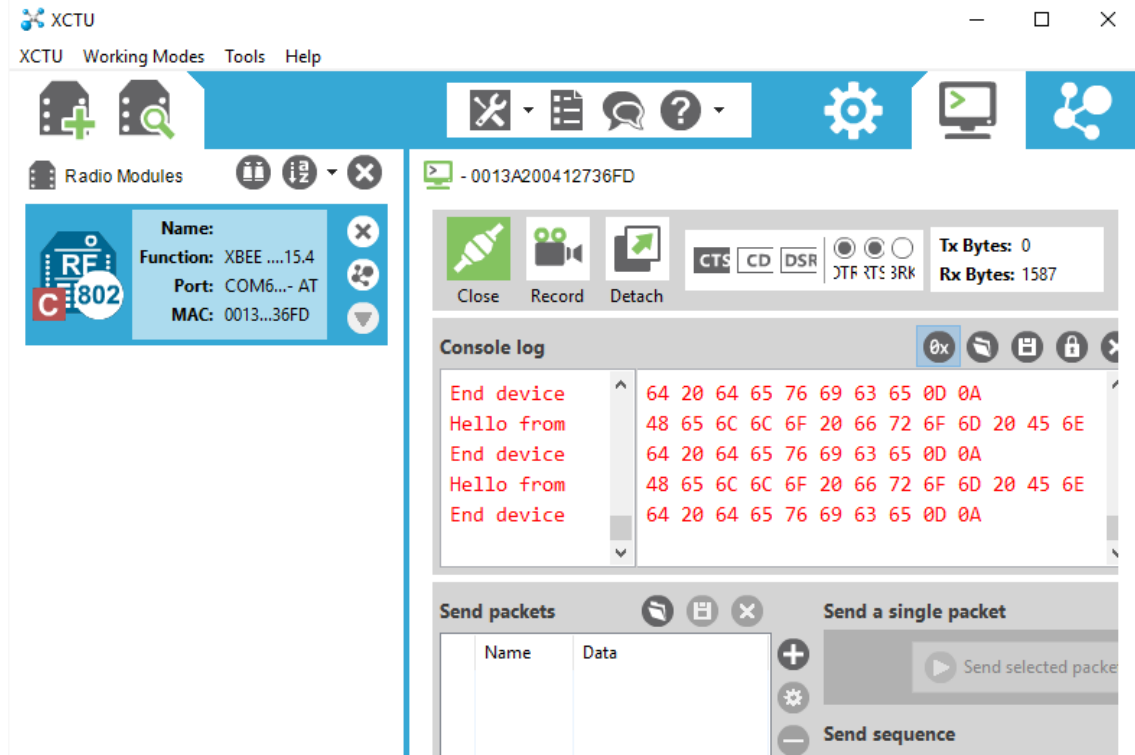
The Communication path is:



Now, on the coordinators side, Fig. 4.9 shows that it actually received a frame.



**Fig (4.8): Arduino communication through XBee modules.**



**Fig (4.9): Coordinator console showing received messages.**

### 4.2.3 Data Transmission

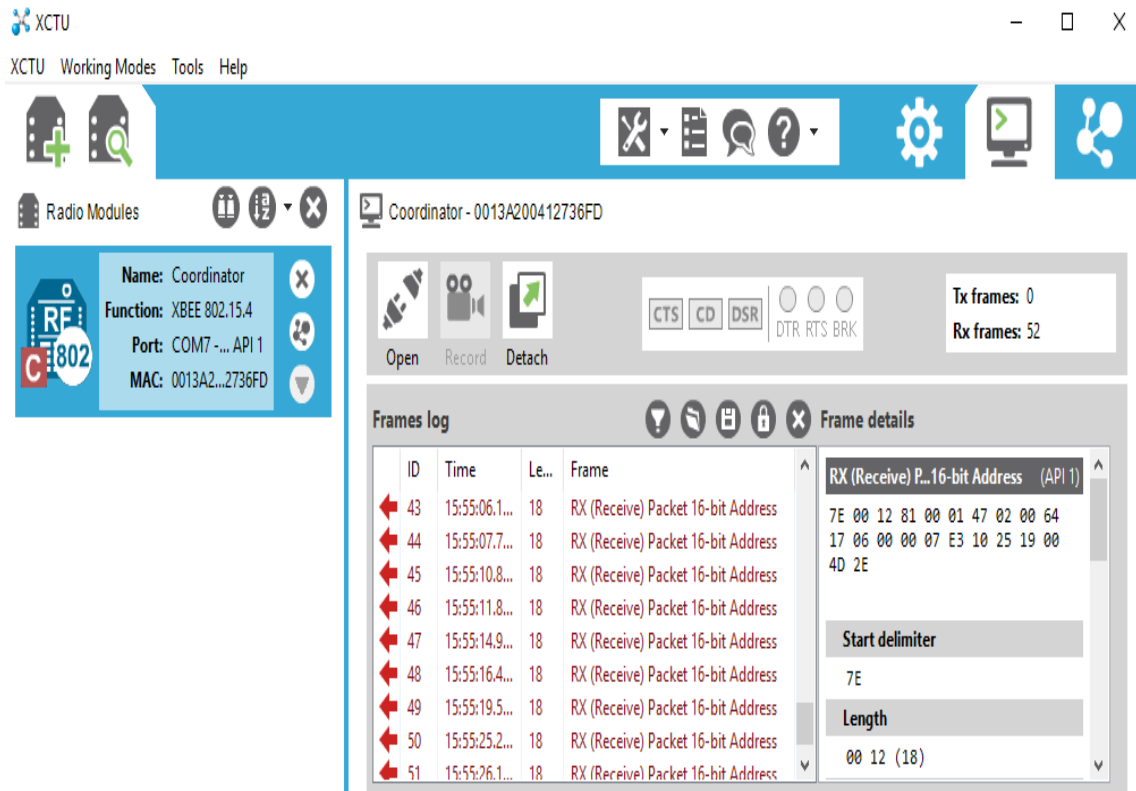
Each end node (Sensor Node) will generate a random vehicle speed and size at current time, and send them to the coordinator. The Coordinator will receive an API frame like the one shown on Figure (4.10).

```
7E 00 12 81 00 01 47 02 00 64 17 06 00 00 07 E3 10 25 19 00 4D 2E
```

**Fig (4.10): API frame received in Hex by coordinator.**

The frame message received on the coordinator sent by the end device has 18 bytes, which includes control headers, payload, delimiter and checksum.





**Fig (4.11): The frame message received on the coordinator**

using the X-CTU frame interpreter I got the following info from the frame as shown in Figure (4.12)

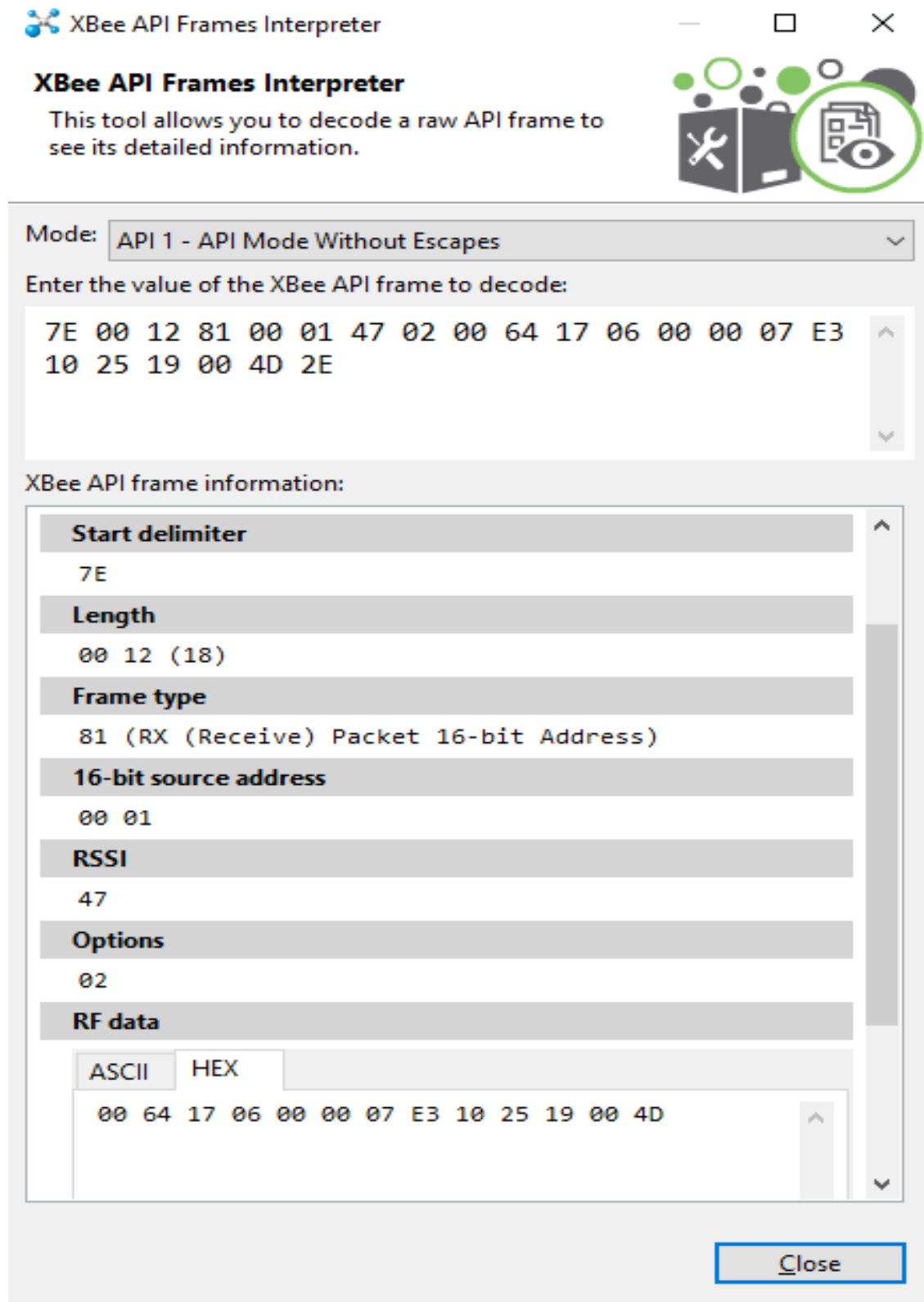


Fig (4.12) : XBee API frame interpreter used to decode received frames.

Each field of the API frame is described below:

- Start delimiter (7E): It indicate that a new frame has arrived.
- Length (18): refers to the number of bytes between length and checksum fields.
- Frame type (81): is the code for a received packet (Packet 16-bit Address).
- 16-bit source address (0001): is the sender network address (Node Id).
- RSSI (47): refers to the strength of signal.
- Receive options (02): indicates that packet has been acknowledged
- RF data: the message sent from the end device and received by the coordinator.
- Checksum (2E): To test data integrity, the device calculates and verifies a checksum on non-escaped data.

To calculate the checksum of an API frame, we do the following:

1. Add all bytes of the packet, except the start delimiter 0x7E and the length.
2. Keep only the lowest 8 bits from the result.
3. Subtract this quantity from 0xFF.

To verify the checksum of an API frame we:

1. Add all bytes including the checksum without including the delimiter and length.

2. If the checksum is correct, the last two digits on the far right of the sum equal 0xFF.

After every stage of the system (variable measurement and data transmission) has been studied, we proceeded to complete our system as below.

We decided to simulate the sensor data by creating a function that generates random data as an alternative to the magnetic sensor reading since the focus of this thesis is on building the wireless network and the data communication and not the application of the magnetic sensor.

Each end device node (Sensor Node) generates a random data (vehicle speed and size), then it sends these data to the coordinator. So, we want multiple Arduino to send data to a single master node (Coordinator).

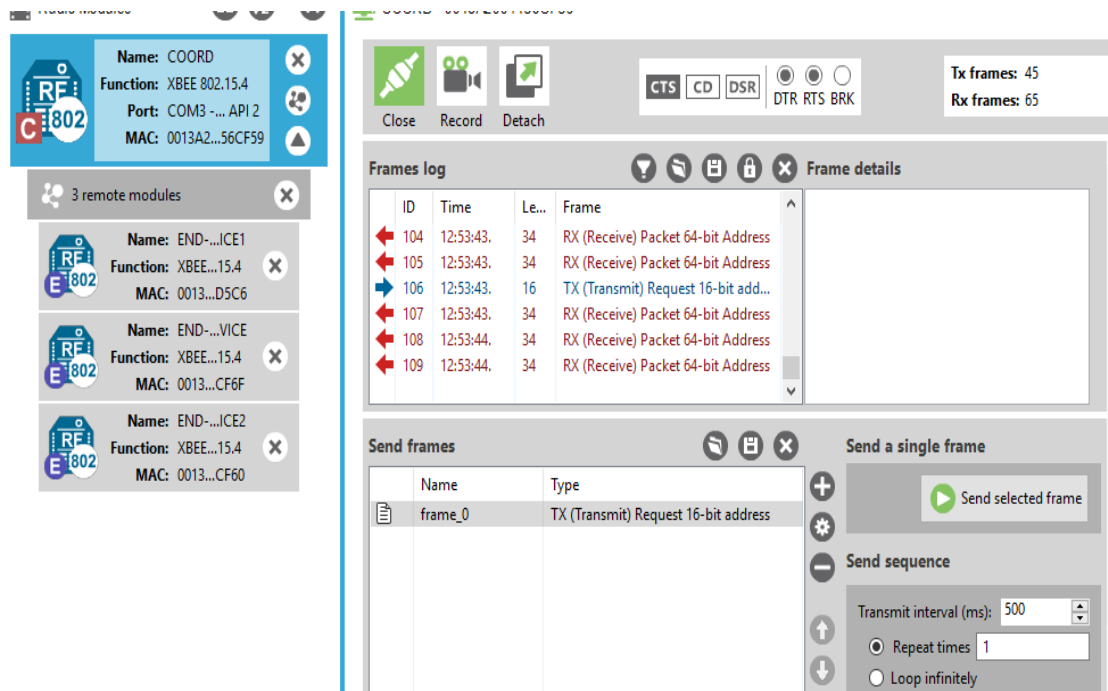
The configurations of XBee end devices and coordinator is as shown in the Table (4-3).

**Table 4-3: Configuration of XBee end devices and coordinator**

	Coordinator	End Device1	End Device2	End Device3
channel number (CH)	C	C	C	C
PAN ID (ID)	1001	1001	1001	1001
Destination High (DH)	0	0	0	0
Destination Low (DL)	1	4	4	4
MY Address	4	1	2	3
Node Identifier (NI)	Coordinator	Sensor Node1	Sensor Node2	Sensor Node3
Coordinator Enable (CE)	Coordinator	End Device	End Device	End Device

After the configuration process, all XBee modules can exchange the data using the XCTU terminal.

Figure (4.13) illustrates connection example between one coordinator and three other end devices, the figure is the coordinator XCTU terminal:



**Fig (4.13): Coordinator connection with three end devices**

The Arduino executes a sketch that makes the operation described above and sends the data value via the serial to the XBee radio. The XBee radio end device sends this information to the coordinator node. The coordinator takes care of receiving the data, processes it properly, and saves it in a database for further processing and analysis.

The nodes were deployed within enough space to send the data and to make sure that nodes were communicating with the coordinator correctly.

### 4.3 Program of Arduino End Device (Sensor Node):

Appendix B at the end of this thesis, is the c-programing language code loaded to the Arduino end device node.

The main tasks of transmitter Arduino node are:

1. Define a byte array (13 bits) to hold traffic parameters.

```
uint8_t payload [13] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }
```

2. Define 16-bit addressing: typically, the coordinator 0xFFFF

```
Tx16Request tx = Tx16Request(0xFFFF, payload,
sizeof(payload));
```

3. set sleep and wake up mode

```
pinMode(XBee_wake, INPUT); // put pin in a high impedance state
digitalWrite(XBee_wake, HIGH);

sleepTime = 898000; // time in ms (max sleep time is 49.7 days)
sleep.pwrDownMode(); //set sleep mode

sleep.sleepDelay(sleepTime); //sleep for: sleepTime
```

4. Define an integer array as an Assumption of vehicle size

```
int randsize[3]={80, 100, 200};
```

5. Generate normally distributed random number based on Average and Standard Deviation to obtain assumed vehicle speed and size.

```
int GenerateVehicleSize()

{int pickedindex=randn(0.91, 0.41);

    return randsize[pickedindex];}
```

```
int GenerateVehicleSpeed()

{    return randn(55, 17.8); }
```

```
int GenerateVehicleSpeed()

{    return randn(55, 17.8); }
```

```
double randn (double mu, double sigma){
    double U1, U2, W, mult, randnum;
    static double X1, X2;
    static int call = 0;
    if (call == 1) {
        call = !call;
        randnum = mu + sigma * (double) X2;
        if (randnum < 20 || randnum > 100)
            return mu;
        else
            return randnum;}
    do{
        // Get two random numbers
        U1 = -1 + ((double) rand () / RAND_MAX) * 2;
        U2 = -1 + ((double) rand () / RAND_MAX) * 2;
        // Radius of circle
        W = pow (U1, 2) + pow (U2, 2);}
```

```

while (W >= 1); //If outside unit circle, then reject number
mult = sqrt ((-2 * log (W)) / W);
X1 = U1 * mult;
X2 = U2 * mult;
call = !call;
randnum = mu + sigma * (double) X1;
if (randnum < 20 || randnum > 100)
    return mu;
else
    return randnum;
}

```

6. Get the current date and time and consider them as detection date and time using “RTCLib”

```

RTC_Millis RTC;

DateTime now =RTC.now();

```

7. Insert the traffic Parameters to *payload* array

```

// Converting int to hex could also used highByte() lowByte()
payload[0] = v_size >> 8 & 0xff;
payload[1] = v_size & 0xff;
payload[2] = now.day();
payload[3] = now.month();
u.j = now.year();
payload[4] = u.b[3];
payload[5] = u.b[2];
payload[6] = u.b[1];
payload[7] = u.b[0];

```



```

payload[8] = now.hour();
payload[9] = now.minute();
payload[10] = now.second();
payload[11] = v_speed >> 8 & 0xff;
payload[12] = v_speed & 0xff;

```

## 8. Send Tx request

```

xbee.send(tx);

```

After sending a tx request, we expect a status response, wait up to 5 seconds for the status response.

## 4.4 Base Station Application

The application was programmed using (Visual Basic.Net).

Figure (4.14) illustrates the main tasks of application (Traffic Monitor Software).

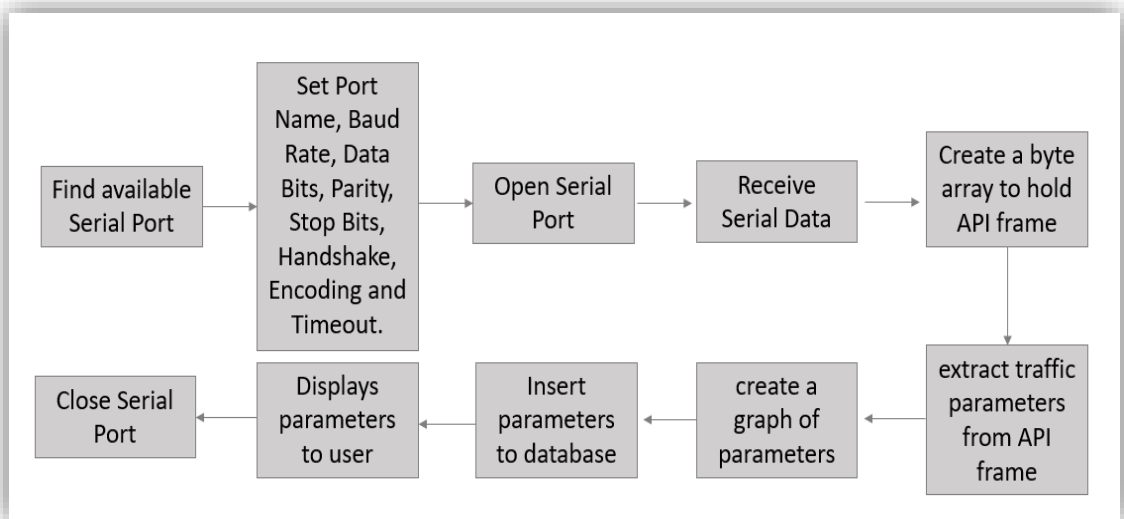


Fig (4.14): Main tasks of Base station application (Traffic Monitor Software)

The XBee coordinator connected with serial USB port with computer using the USB Adapter.

The software is designed to simply listen to the serial port. Once the software sees a header byte '0xFFFF', it knows it is the beginning of a packet and then it purses the packet and determines the ID and the data received from the end device node.

Another component of the application is a simple visual interface. The interface displays the received data. The application then stamps the data with the real time and saves it in database for further processing and sharing.

Figure (4.15) illustrates the main interface of the base station application. The application uses a single interface. The interface displays the received traffic parameters from end device in a list box.

The interface includes:

- A drop-down list that contains the available COM ports
- A drop-down list that contains the Baud Rates
- A connect/disconnect button.
- A list box where the received API frame is added.
- Text boxes where the components of received API frame is displayed after converted the traffic parameters to Hex.
- Data grid view where the received traffic parameters is added.

- Chart where the received traffic parameters is drawn.

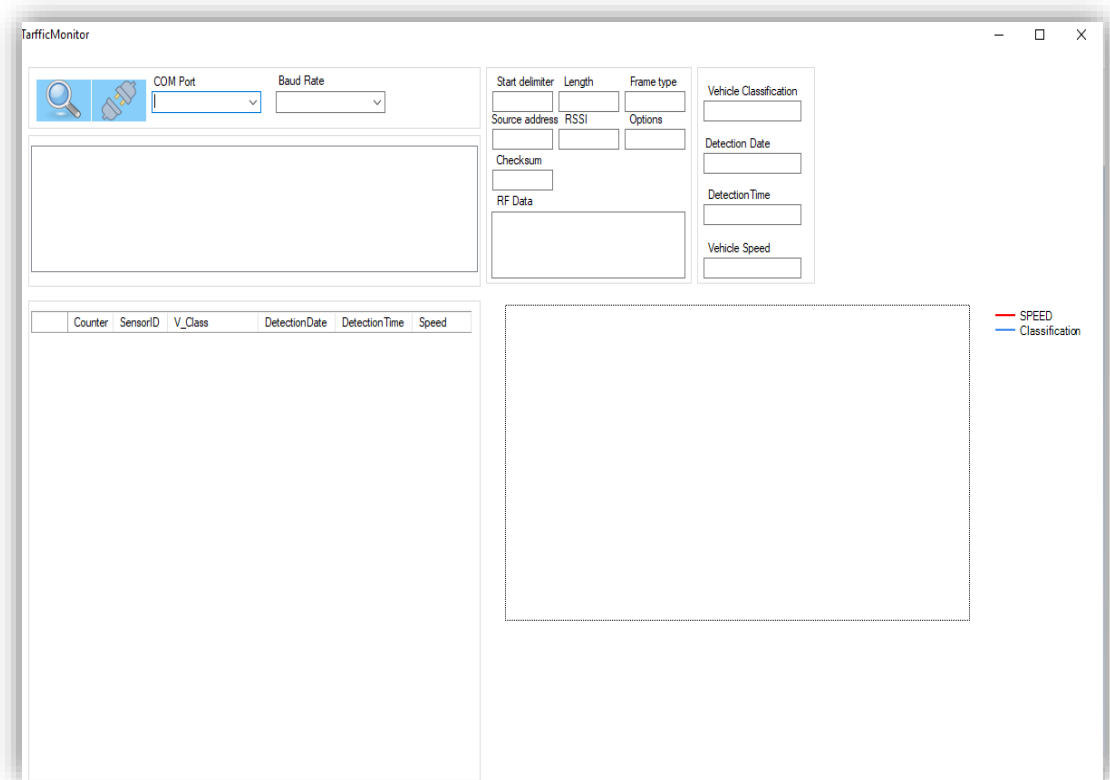




Fig (4.15): Traffic Monitor Software

As shown in figure 4.14 the main tasks of Base station application:

1. Find available COM ports in Pc and add it to the dropdown list. The function responsible for that is (FindSerialPorts()).
2. Set the properties of selected serial port and then open it. The properties of serial port are Port Name, Baud Rate, Data Bits, Parity, Stop Bits, Handshake, Encoding and Timeout. The function responsible for that is (OpenPort()) .
3. Close an opened serial port. The function responsible for that is (ClosePort()).

4. Read received serial data, extract traffic parameters from it and then add these parameters to database.

When user click on the search button (  ), the function (FindSerialPorts()) populates the COM port drop down list with the available COM ports. The program then waits for the user to pick one and select baud rate.

When user click on the connect button (  ), the first thing the routine does is determine if the serial port is open. The serial port is closed if it is open, then the serial port properties are set, the serial port is opened and the timer is started.

A timer is used to check for incoming data. The timer is set to trigger every 500 ms or half a second and when triggered it calls the RcvdTimer\_Tick() routine.

RcvdTimer\_Tick() calls a function(ReceiveSerialData()) that creates a byte array to hold the API frame that is read through the serial port then this API frame is returned as an array of bytes.

There are other routines that add received API frame to the list box, extract traffic parameters from the frame, create a graph of the parameters, save the parameters to database and displays them to user.

When user click on any received API frame in the list box, the function (extractdata()) will disassemble the selected frame and display its parts to the user in hex format.

The routine (InsertData()) will add the traffic parameters to the data grid view and insert them to the database.

Figure (4.16) demonstrate the execution of the application.

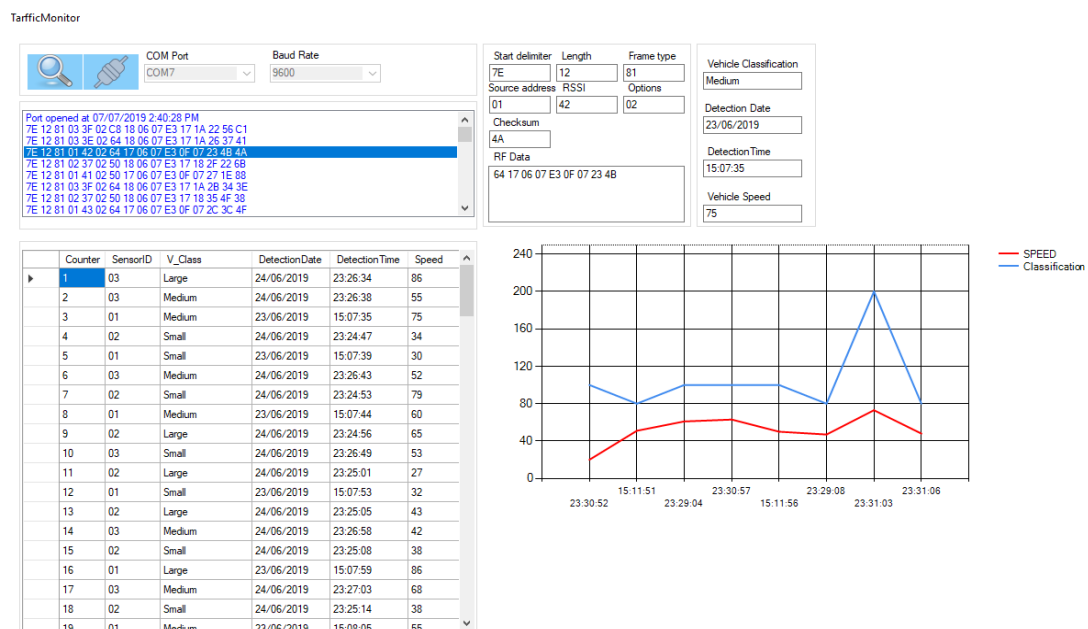


Fig (4.16): Running Traffic Monitor Software

The code for the Base station application is found in APPENDIX B.

The code for web service of saved database is found in APPENDIX C.

## 4.5Data Base

The database of Traffic Monitor Software consists of single table with the following attributes:

- Sensor ID (transmitter).
- Vehicle Class.
- Detection Date.
- Detection Time.
- Vehicle speed.

tbl_Traffic		
SenderId		
Calssification		
DetectionDate		
DetectionTime		
Speed		

Column Name	Data Type	Allow Nulls
SenderId	nvarchar(2)	<input checked="" type="checkbox"/>
Calssification	nvarchar(6)	<input checked="" type="checkbox"/>
DetectionDate	date	<input checked="" type="checkbox"/>
DetectionTime	time(7)	<input checked="" type="checkbox"/>
Speed	int	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Fig (4.17): Table of Traffic Monitor Database

Microsoft SQL server was used to implement the data base, it is a relational database management system developed by Microsoft. As a database server, it is a software product with the primary function of storing and retrieving data as requested by other software applications—which may run either on the same computer or on another computer across a network (including the Internet).

#### 4.6 Programming Interface

Also, we provided a programming interface for our system where other application can connect to the system and retrieve information about

the traffic parameters. The programming interface consists of one class called Vehicle.cs described below.

### **1. Create a class file named “Vehicle.cs”.**

We have defined five entities using get and set properties within the Vehicle class. These entities correspond to the columns of the database table.

```
namespace TrafficService
{
    public class Vehicle
    {
        public String sensorId { get; set; }
        public string vclass { get; set; }
        public int vspeed { get; set; }
        public string detectionDate { get; set; }
        public string detectionTime { get; set; }
    }
}
```

### **2. create method named “GetAllParameters()”**

We have used one strongly typed list of objects using Vehicle.cs class to manipulate, search and sort the lists and I have used the connection string of database.

```
List<Vehicle> listTraffic = new List<Vehicle>();
String constring =
ConfigurationManager.ConnectionStrings["TrafficPr"].ConnectionStrin
g;
```

- Then, all ADO.NET objects and properties are mentioned with their vast roles like *SqlConnection* , *SqlCommand* , *SqlDataReader* etc.

```
SqlConnection cn    = new SqlConnection(constring);
SqlCommand cmd = new SqlCommand(strdtails, cn);
SqlDataReader  rdr ;
```

- use SQL statement to read all field of the table.

```
"SELECT SenderId, Calssification, DetectionDate, DetectionTime,
Speed FROM TrafficDB.dbo.tbl_Traffic"
```

- Mention all the fields using object of Vehicle class and add them to List class object “listTraffic“

```
rdr = cmd.ExecuteReader();
while (rdr.Read()){
    Vehicle v = new Vehicle();
    v.sensorId = rdr["SenderId"].ToString();
    v.vclass = rdr["Calssification"].ToString();
    v.detectionDate = rdr["DetectionDate"].ToString();
    v.detectionTime = rdr["DetectionTime"].ToString();
    v.vspeed = Convert.ToInt32(rdr["Speed"]);
    listTraffic.Add(v);}
```

- Convert the listTraffic object values into JSON string.

```
JavaScriptSerializer js = new JavaScriptSerializer();
Context.Response.Write(js.Serialize(listTraffic));
```



- Table's data in JSON format will be like :

```
[{"sensorId":"01","vclass":"Small","vspeed":36,"detectionDate":"23/06/2019","detectionTime":"15:48:28"},
{"sensorId":"01","vclass":"Large","vspeed":74,"detectionDate":"23/06/2019","detectionTime":"15:48:32"},
{"sensorId":"01","vclass":"Medium","vspeed":46,"detectionDate":"23/06/2019","detectionTime":"15:48:3"},
{"sensorId":"01","vclass":"Small","vspeed":55,"detectionDate":"23/06/2019","detectionTime":"15:48:38"},
{"sensorId":"01","vclass":"Large","vspeed":66,"detectionDate":"23/06/2019","detectionTime":"15:48:38"}]
```



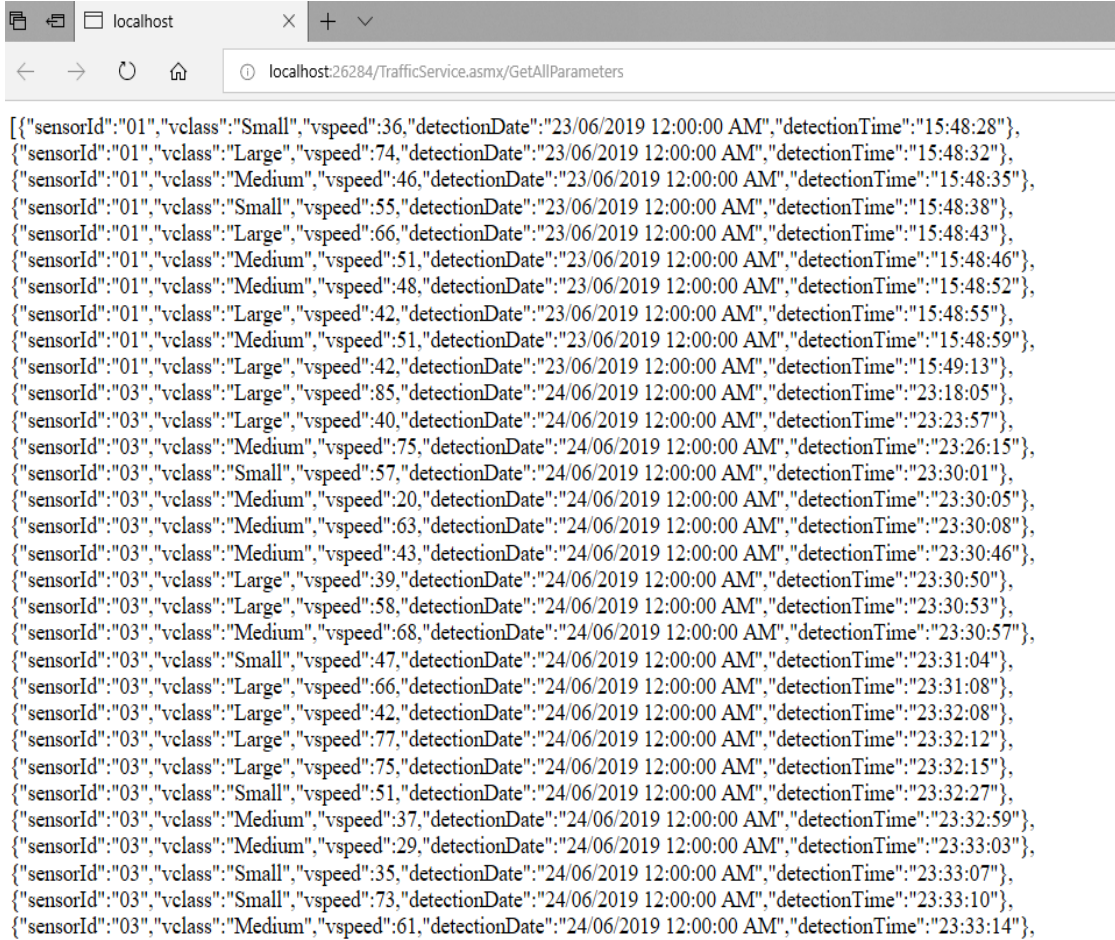
```

TrafficWebService.TrafficService
GetAllParameters()

public class TrafficService : System.Web.Services.WebService
{
    [WebMethod]
    public void GetAllParameters()
    {
        List<Vehicle> listTraffic = new List<Vehicle>();
        String constring = ConfigurationManager.ConnectionStrings["TrafficPr"].ConnectionString;
        SqlConnection cn = new SqlConnection(constring);
        cn.Open();
        String strdtails = "SELECT SenderId, Calssification, DetectionDate, DetectionTime, Speed FROM TrafficDB.dbo.tbl_Traffic";
        SqlCommand cmd = new SqlCommand(strdtails, cn);
        SqlDataReader rdr;
        rdr = cmd.ExecuteReader();
        while (rdr.Read())
        {
            Vehicle v = new Vehicle();
            v.sensorId = rdr["SenderId"].ToString();
            v.vclass = rdr["Calssification"].ToString();
            v.detectionDate = rdr["DetectionDate"].ToString();
            v.detectionTime = rdr["DetectionTime"].ToString();
            v.vspeed = Convert.ToInt32(rdr["Speed"]);
            listTraffic.Add(v);
        }
        JavaScriptSerializer js = new JavaScriptSerializer();
        Context.Response.Write(js.Serialize(listTraffic));
    }
}

```

**Fig (4.18): Table GetAllParameters method**



```
[{"sensorId":"01","vclass":"Small","vspeed":36,"detectionDate":"23/06/2019 12:00:00 AM","detectionTime":"15:48:28"},
{"sensorId":"01","vclass":"Large","vspeed":74,"detectionDate":"23/06/2019 12:00:00 AM","detectionTime":"15:48:32"},
{"sensorId":"01","vclass":"Medium","vspeed":46,"detectionDate":"23/06/2019 12:00:00 AM","detectionTime":"15:48:35"},
{"sensorId":"01","vclass":"Small","vspeed":55,"detectionDate":"23/06/2019 12:00:00 AM","detectionTime":"15:48:38"},
{"sensorId":"01","vclass":"Large","vspeed":66,"detectionDate":"23/06/2019 12:00:00 AM","detectionTime":"15:48:43"},
{"sensorId":"01","vclass":"Medium","vspeed":51,"detectionDate":"23/06/2019 12:00:00 AM","detectionTime":"15:48:46"},
{"sensorId":"01","vclass":"Medium","vspeed":48,"detectionDate":"23/06/2019 12:00:00 AM","detectionTime":"15:48:52"},
{"sensorId":"01","vclass":"Large","vspeed":42,"detectionDate":"23/06/2019 12:00:00 AM","detectionTime":"15:48:55"},
{"sensorId":"01","vclass":"Medium","vspeed":51,"detectionDate":"23/06/2019 12:00:00 AM","detectionTime":"15:48:59"},
{"sensorId":"01","vclass":"Large","vspeed":42,"detectionDate":"23/06/2019 12:00:00 AM","detectionTime":"15:49:13"},
{"sensorId":"03","vclass":"Large","vspeed":85,"detectionDate":"24/06/2019 12:00:00 AM","detectionTime":"23:18:05"},
{"sensorId":"03","vclass":"Large","vspeed":40,"detectionDate":"24/06/2019 12:00:00 AM","detectionTime":"23:23:57"},
{"sensorId":"03","vclass":"Medium","vspeed":75,"detectionDate":"24/06/2019 12:00:00 AM","detectionTime":"23:26:15"},
{"sensorId":"03","vclass":"Small","vspeed":57,"detectionDate":"24/06/2019 12:00:00 AM","detectionTime":"23:30:01"},
{"sensorId":"03","vclass":"Medium","vspeed":20,"detectionDate":"24/06/2019 12:00:00 AM","detectionTime":"23:30:05"},
{"sensorId":"03","vclass":"Medium","vspeed":63,"detectionDate":"24/06/2019 12:00:00 AM","detectionTime":"23:30:08"},
{"sensorId":"03","vclass":"Medium","vspeed":43,"detectionDate":"24/06/2019 12:00:00 AM","detectionTime":"23:30:46"},
{"sensorId":"03","vclass":"Large","vspeed":39,"detectionDate":"24/06/2019 12:00:00 AM","detectionTime":"23:30:50"},
{"sensorId":"03","vclass":"Large","vspeed":58,"detectionDate":"24/06/2019 12:00:00 AM","detectionTime":"23:30:53"},
{"sensorId":"03","vclass":"Medium","vspeed":68,"detectionDate":"24/06/2019 12:00:00 AM","detectionTime":"23:30:57"},
{"sensorId":"03","vclass":"Small","vspeed":47,"detectionDate":"24/06/2019 12:00:00 AM","detectionTime":"23:31:04"},
{"sensorId":"03","vclass":"Large","vspeed":66,"detectionDate":"24/06/2019 12:00:00 AM","detectionTime":"23:31:08"},
{"sensorId":"03","vclass":"Large","vspeed":42,"detectionDate":"24/06/2019 12:00:00 AM","detectionTime":"23:32:08"},
{"sensorId":"03","vclass":"Large","vspeed":77,"detectionDate":"24/06/2019 12:00:00 AM","detectionTime":"23:32:12"},
{"sensorId":"03","vclass":"Large","vspeed":75,"detectionDate":"24/06/2019 12:00:00 AM","detectionTime":"23:32:15"},
{"sensorId":"03","vclass":"Small","vspeed":51,"detectionDate":"24/06/2019 12:00:00 AM","detectionTime":"23:32:27"},
{"sensorId":"03","vclass":"Medium","vspeed":37,"detectionDate":"24/06/2019 12:00:00 AM","detectionTime":"23:32:59"},
{"sensorId":"03","vclass":"Medium","vspeed":29,"detectionDate":"24/06/2019 12:00:00 AM","detectionTime":"23:33:03"},
{"sensorId":"03","vclass":"Small","vspeed":35,"detectionDate":"24/06/2019 12:00:00 AM","detectionTime":"23:33:07"},
{"sensorId":"03","vclass":"Small","vspeed":73,"detectionDate":"24/06/2019 12:00:00 AM","detectionTime":"23:33:10"},
{"sensorId":"03","vclass":"Medium","vspeed":61,"detectionDate":"24/06/2019 12:00:00 AM","detectionTime":"23:33:14"}]
```

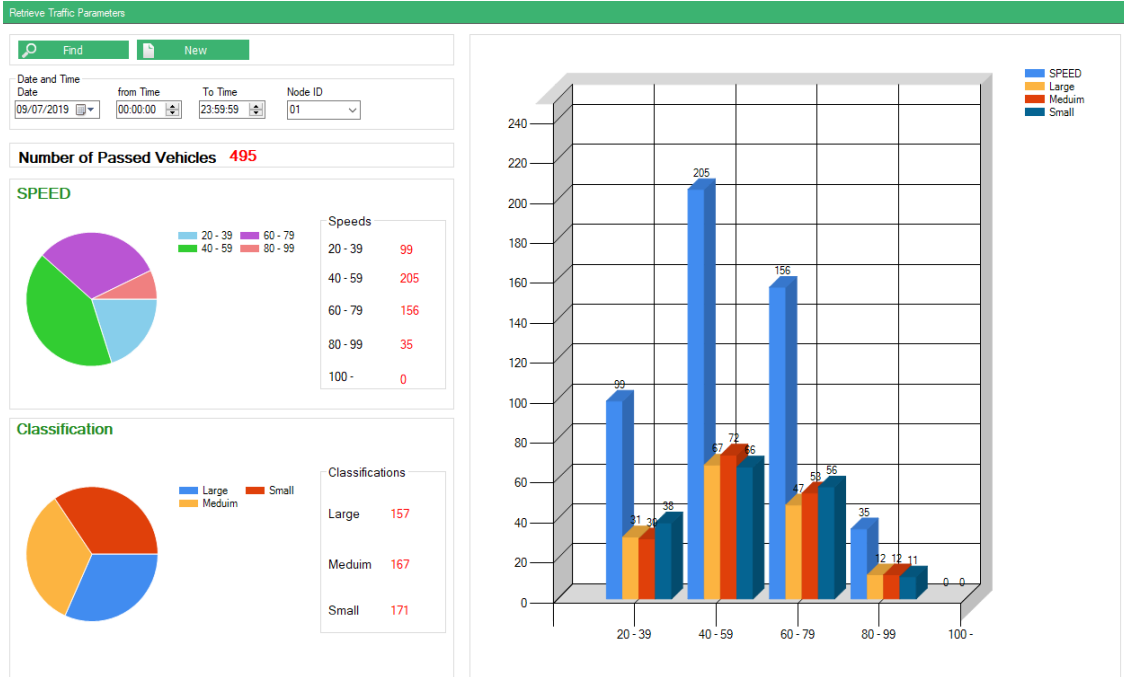
**Fig (4.19): data in JSON format**

## 4.7 Retrieve Traffic Parameters

I have created a simple Windows application that retrieves counters of traffic parameters from the database and displays them to the user.

The application retrieves the counters of traffic parameters from the database by a given date and two times. The retrieved data is a counter of traffic parameters collected from specific nodes are displayed to the user as statistical data in the form of pie charts.

Figure (4.20) illustrates the application with statistical retrieved counters of traffic parameters collected from the node 1.



**Fig (4.20): statistical retrieved traffic parameters collected from the node 1**

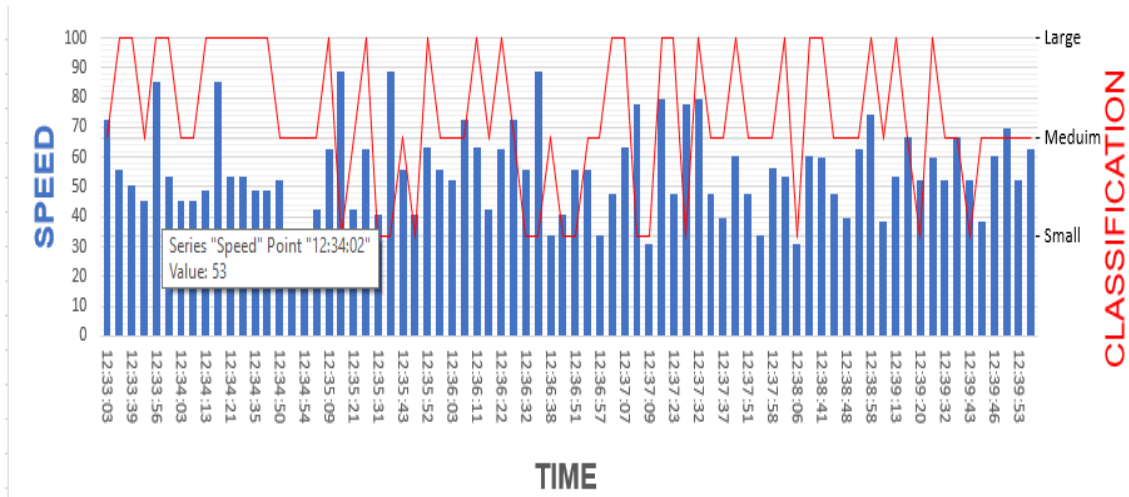
Each pie chart displays the number of vehicles in two categories (speeds and classifications).

Speed category shows the number of vehicles for five speeds (20-39, 40-59, 60-79, 80-99 and more than 100 km/hour).

Classification category shows the number of vehicles for three classification (small, medium and large).

The column chart displays the number of vehicles with all speeds and classifications in one chart.

Figure (4.21) illustrates the speeds and classifications with detection times collected from all nodes.



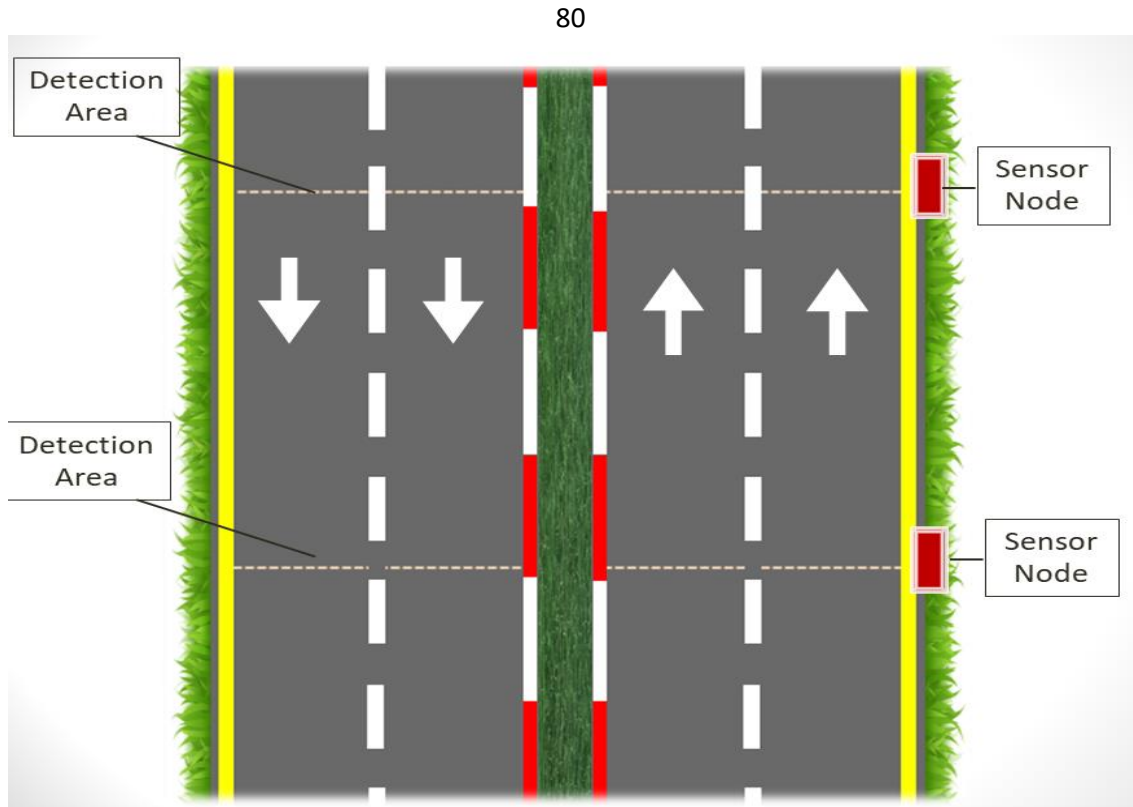
**Fig (4.21): Traffic parameters collected from the nodes**

The code for statistical retrieved traffic parameters collected from the nodes is found in APPENDIX D.

## 4.8 Traffic Visualization

The working of the design is evaluated by the acquisition of a baseline data set. The baseline data set provides a data package for testing of the sensors performance and functionality which include sensed magnetic field in 3-axes (x, y and z) for each sensor node.

The data package for each sensor node contains vehicle classification, vehicle speed and direction. To obtain and apply them we created an application for traffic visualization on road. The application divided the road into two directions (left and right) and placed two sensors on the side of the road with a distance from each other as shown in Figure (4.22).



**Fig (4.22): Traffic visualization interface**

We assumed that the vehicles have three Classifications (large, medium and small) with sizes ( $\{50, 100\}$ ,  $\{80, 200\}$ ,  $\{40, 80\}$  inch).

The application selects from these values randomly and assigns it to a vehicle, then moves it at random speed by changing the coordinates of its location using loop).

When the vehicle reaches the area detection for both sensor nodes, the both detection times are obtained.

The speed of the vehicle is then calculated from the equation:

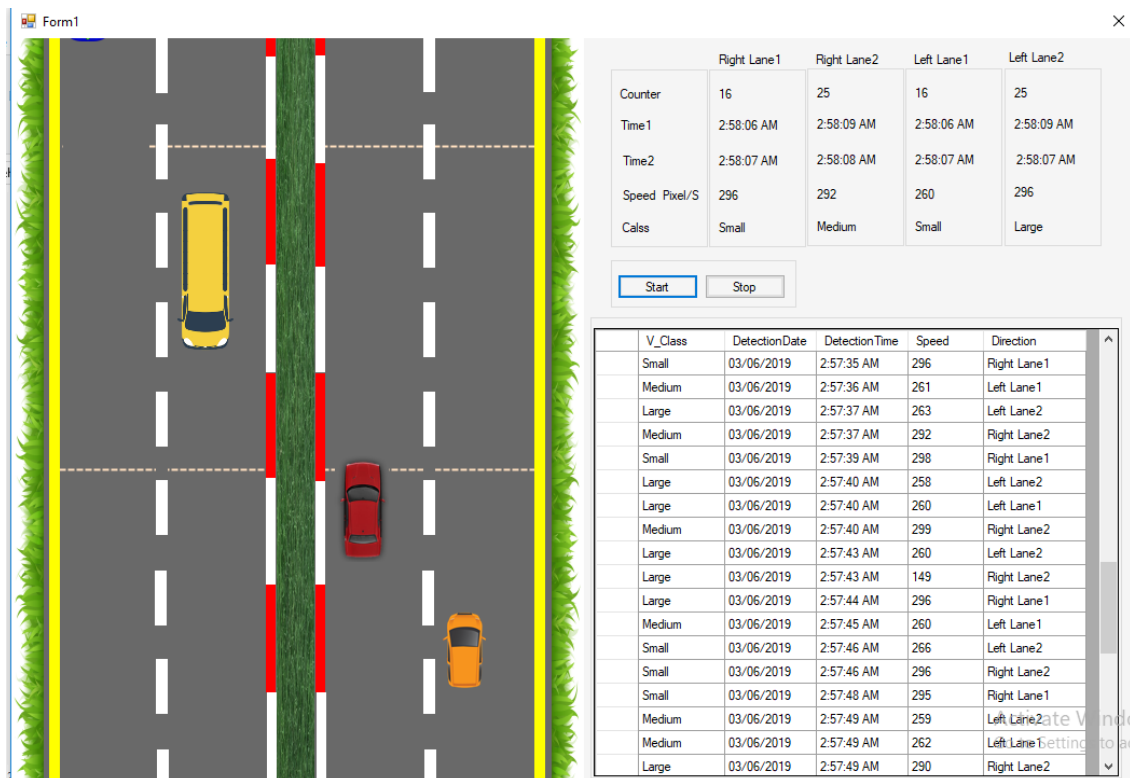
$$\frac{\text{distance between sensor node1 and sensor node2}}{\text{Second detection time} - \text{First detction time}}$$

Therefore, the dataset obtained from vehicle traffic using this simulator are including (vehicles counter, detection date, detection time of sensor node1, detection time of sensor node2, vehicle speed, vehicle classification, and vehicle direction).

The Figure (4.23) shows the application after it is run.

The code of application is found in APPENDIX D

The data obtained from the simulator of vehicle traffic are saved to database table to be accessible through the Web.



**Fig (4.23): Traffic visualization after it is run**

## **Chapter Five**

### **Conclusion**

The body of this thesis consists of two main subjects, the first describes the design of a simple magneto-resistive sensor, the other focuses on how to build a wireless sensor network.

In the beginning of the thesis, a detailed theory of Anisotropic Magneto-Resistive sensors (AMR) is described to provide a better understanding of the underlying principles of the sensors used in the design. Having introduced the theory, the details of hardware equipment employed in the design are described in the hardware to Implement chapter.

The details of different components available and the requirement of each component in the design is explained. Next, the Wireless Sensor Network technologies was reviewed, and among a broad amount of options to implement based on criteria of easy implementation, documentation, distance range, mesh networking, then it was decided to use XBee modules. Then we decided to work with Arduino hardware/software, because it is easy to implement, the IDE is programmed in C language an even more important, there are lots of tutorials and projects available by the Arduino community online. Following that, communication tests were performed-on a first hand, it was made a XBee to XBee test, and second, an Arduino to Arduino test via wireless, aided by the XBee- in order to ensure that data sent by nodes

was effectively received by the coordinator for further processing or analysis.

This project can be considered as a small step for creating a traffic monitor station, or also to create a control and monitoring system for crops water irrigation.

The base station applications were implemented that allow to save, retrieve and display data within graphical interfaces and statistical charts.

There is still a lot to work in the field of Wireless Sensor Network technology, since it has an enormous potential to improve outdoor control systems and quality of life of citizens within a called smart city for instance: the development of low power communication hardware, low-power microcontrollers, MEMS based sensors and actuators, and energy-scavenging devices is necessary to enhance the potential and the performance of sensor networks. Also, this project offers flexibility in hardware selection along with the ability to expand the number of sensors used.



## References

- [1] Arduino.cc (2019). ***Arduino – Introduction***. [online] Available from:  
<https://www.arduino.cc/en/guide/introduction>
  
- [2] T.Thamaraimanalan, S.P.Vivekk, G.Satheeshkumar and P.Saravanan (2018). ***Smart Garden Monitoring System Using IOT***. **Asian Journal of Applied Science and Technology (AJAST) (Open Access Quarterly International Journal)**, vol. 2, pp. 186-192.
  
- [3] Rana Abu Qarea (2017), ***Annual Statistical Report, Palestinian Ministry of Transport***, Ramallah, Palestine. available from:  
[http://www.mot.gov.ps/wp-content/uploads/Portals/\\_Rainbow/Documents/Stats/Annual\\_Repoert2017.pdf](http://www.mot.gov.ps/wp-content/uploads/Portals/_Rainbow/Documents/Stats/Annual_Repoert2017.pdf)
  
- [4] Lisa Jogschies (2015), ***Recent Developments of Magnetoresistive Sensors for Industrial Applications***, **Sensors**. 15. 28665-28689. 10.3390/s151128665.
  
- [5] Saber Taghvaeeyan and Rajesh Rajamani (2014). ***Portable Roadside Sensors for Vehicle Counting, Classification, and Speed Measurement***. **IEEE Transactions on Intelligent Transportation Systems**, vol. 15, pp. 73-83.
  
- [6] FOSIAO LLC (2014). ***Zigbee RF Channels***. [online] Available from: <https://fosiao.com/content/zigbee-and-wifi-rf-channels>
  
- [7] Robert Faludi (2011). ***Building Wireless Sensor Networks: with ZigBee, XBee, Arduino, and Processing*** (1st ed). USA, CA: O'Reilly Media, Inc. [online] Available from:  
<https://ab-log.ru/files/File/books/WirelessSensorNetwork.pdf>

- [8] Honeywell (2011). *3-Axis Magnetic Sensor Hybrid HMC2003: Datasheet*. [online] Available from:  
[https://neurophysics.ucsd.edu/Manuals/Honeywell/HMC\\_2003.pdf](https://neurophysics.ucsd.edu/Manuals/Honeywell/HMC_2003.pdf)
- [9] Erik Minge, Scott Petersen, and Jerry Kotzenmacher (2011). *Evaluation of Non-Intrusive Technologies for Traffic Detection*. *Transportation Research Record*, vol. 2256, pp. 95–103.
- [10] J. Medina, M. Chitturi, and R. Benekohal (2010). *Effects of fog, snow, and rain on video detection systems at intersections*. *Transportation Letters*, vol. 2, no. 1, pp. 1–12.
- [11] S. Kaewkamnerd, J. Chinrungrueng, R. Pongthornseri, and S. Dumnin (2010). *Vehicle classification based on magnetic sensor signal*. *IEEE International Conference on Information and Automation*, pp. 935–939.
- [12] Martin Hebel and George Bricker (2010). *Getting Started with XBee RF Modules (1<sup>st</sup> ed) - A Tutorial for BASIC Stamp and Propeller Microcontrollers*. [online] Available from:  
<https://www.jameco.com/Jameco/Products/ProdDS/2171225QuickGuide.pdf>
- [13] W. Zhang, G. Tan, H. Shi, and M. Lin (2010). *A distributed threshold algorithm for vehicle classification based on binary proximity sensors and intelligent neuron classifier*. *Journal of Information Science and Engineering*, vol. 26, no. 3, pp. 769–783.
- [14] S. Jeng and S. Ritchie (2008). *Real-time vehicle classification using inductive loop signature data*. *Transportation Research Record*, vol. 2086, pp. 8–22, 2008
- [15] D. Nan, T. Guozhen, M. Honglian, L. Mingwen, and S. Yao (2008). *Low-power vehicle speed estimation algorithm based on WSN*. in *Proc. 11th International IEEE Conference on Intelligent Transportation Systems*, pp. 1015–1020.

- [16] S. Y. Cheung and P. Varaiya (2007). *Traffic Surveillance by Wireless Sensor Networks: Final Report for PATH TO 5301*. University of California, Berkely. [online]. Available from: <https://pdfs.semanticscholar.org/959b/65e302c8ab778834eb4f87edc0a5714baa0d.pdf>
- [17] Andreas Willig and Holger Karl (2007). *Protocols and Architectures for Wireless Sensor Networks* (1st ed). New York, NY, USA: Wiley-Interscience.
- [18] L. E. Y. Mimbela and L. A. Klein (2007). *A summary of vehicle detection and surveillance technologies used in intelligent transportation systems*. UC Berkeley Transportation Library. [online]. Available from: <https://www.fhwa.dot.gov/policyinformation/pubs/vdstits2007/vdstits2007.pdf>
- [19] H. Cheng, H. Du, L. Hu, and C. Glazier (2005). *Vehicle detection and classification using model-based and fuzzy logic approaches*. Transportation Research Record, vol. 1935, pp. 154–162.
- [20] Nirupama Bulusu, Sanjay Jha (2005), *Wireless sensor networks*, Boston, MA: Artech House.
- [21] D. Puccinelli y M. Haenggi (2005). *Wireless Sensor Networks: Applications and Challenges of Ubiquitous Sensing*. IEEE Circuits and Systems Magazine, pp. 19-29.
- [22] I. KEMAPECH, I. DUNCAN and A. MILLER (2005). *A Survey of Wireless Sensor Networks Technology*. In Proceedings of the 6th annual post graduate symposium on the Convergence of telecommunications, networking, and broadcasting.
- [23] Dr. Peter T. Martin (2003), *Detector Technology Evaluation*, University of Utah, MPC Report No. 03-154. [Online]. Retrieved from <https://www.ugpti.org/resources/reports/downloads/mpc03-154.pdf>

- [24] S. Gupte, O. Masoud, R. F. K. Martin, and N. P. Papanikolopoulos (2002). ***Detection and classification of vehicles.*** **IEEE Transactions on Intelligent Transportation Systems**, vol. 3, no. 1, pp. 37–47.
- [25] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci (2002). ***Wireless sensor networks: a survey***, **Computer Networks**, vol. 38, pp. 393-422.
- [26] Michael Caruso, Tamara Bratland, Carl Smith and Robert Schneider (1998). ***A new perspective on magnetic field sensing.*** **Sensors** (Peterborough, NH). 15.
- [27] Tarun Agarwal. ***wireless sensor network and their applications*** [Online]. Available from:  
<https://www.elprocus.com/introduction-to-wireless-sensor-networks-types-and-applications/>

## APPENDIX

### A. Theory of Anisotropic Magnetic Resistance (AMR) Sensor

The earth has a natural magnetic field of 25 to 65 microtesla (0.25 to 0.65 Gauss) at the earth's surface as shown in Figure (A.1). Since almost all vehicles have significant amounts of ferrous metals in their chassis, they cause a disturbance in the earth's magnetic field around them. If this disturbance can be detected using sensing technology, it will provide a way to detect passing vehicles and an estimate of their size and speed.

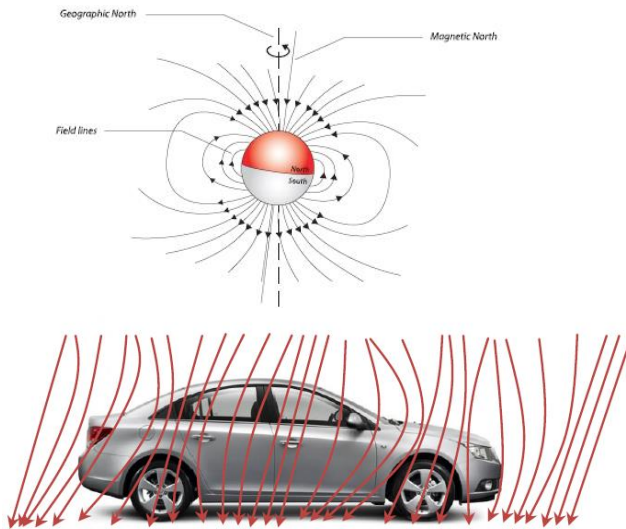
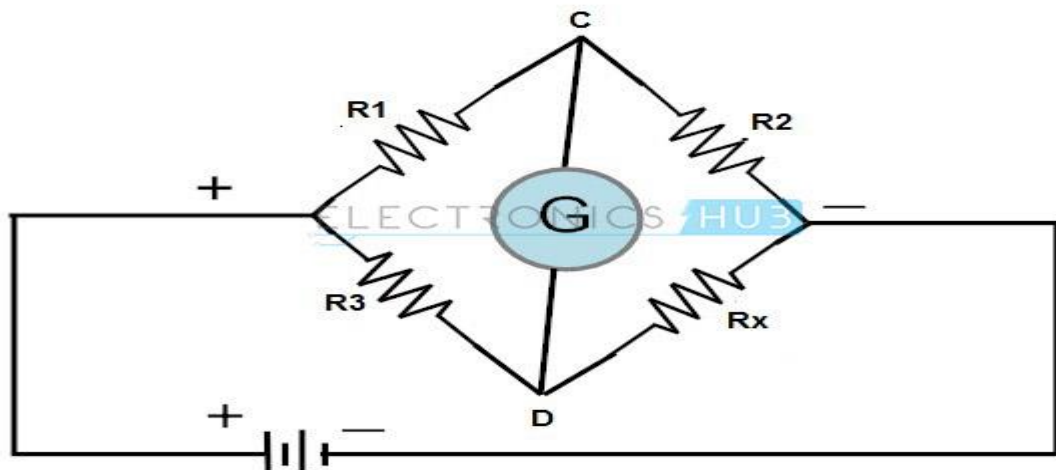


Figure (A.1): The earth magnetic field lines (left), the perturbation of the earth magnetic caused by presence of a vehicle (right).

Recent advancement in magneto resistance technology allows detecting such small disturbance in the earth magnetic field. For instance, the 3-Axis Magnetic Sensor Hybrid HMC2003 from Honeywell is a high sensitivity magnetic sensor that can measure the direction and strength of a low incident magnetic field. This sensor is highly sensitive to magnetic fields

along the three axis (X, Y, and Z). It can detect fields as low as 40 microgauss and up to  $\pm 2$  Gauss. The sensor provides analog outputs for each axis. Further, the sensor has a high bandwidth which allows measuring passing vehicles or even planes at high speed.

AMR sensor is designed in Wheatstone bridge configurations to measure magnetic fields by measuring the small change in resistance of this bridge. This is used to convert a resistance change to a voltage change of a transducer. A Wheatstone bridge consists of four resistors that are connected with the supply source and indicating instruments as shown in figure A.2.



**Figure (A.2): Wheatstone bridge**

This bridge is used to find the unknown resistance very precisely by comparing it with a known value of resistances. In this bridge null or balanced condition is used to find the resistance.

For this bridge balanced condition voltage at points C and D must be equal. Hence, no current flows through the galvanometer. For getting the balanced condition one of the resistors must be variable.

From the figure,

The voltage at point  $D = V \times \frac{R_x}{R_3 + R_x}$ , and the voltage at point  $C = V \times \frac{R_2}{R_1 + R_2}$

The voltage (V) across galvanometer or between C and D is

$$V_{CD} = V \times \frac{R_x}{R_3 + R_x} - V \times \frac{R_2}{R_1 + R_2}$$

When the bridge is balanced  $V_{CD} = 0$

So,

$$V \times \frac{R_x}{R_3 + R_x} = V \times \frac{R_2}{R_1 + R_2}$$

$$\frac{R_x}{R_3 + R_x} = \frac{R_2}{R_1 + R_2}$$

$$R_x R_1 + R_x R_2 = R_2 R_3 + R_2 R_x$$

$$R_x R_1 = R_2 R_3$$

$$R_2 / R_1 = R_x / R_3$$

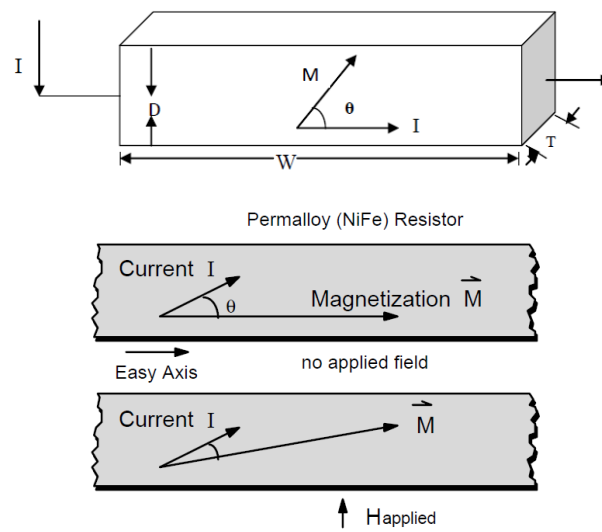
This is the condition to balance the bridge. And for finding the unknown value of resistance

$$R_x = R_3 \times \frac{R_2}{R_1}$$

From the above equation  $R_4$  or  $R_x$  can be computed from the known value of resistance  $R_3$  and the ratio of  $R_2/R_1$ . Therefore, most of the cases  $R_2$  and  $R_1$  values are fixed and the  $R_3$  value is variable so that null value is achieved and the bridge gets balanced.

### 1. Magnetic Resistance

The anisotropic magnetoresistive (AMR) effect was first described in 1857 by William Thomson [4]. Thomson observed that the resistivity of ferromagnetic materials depends on the angle between the direction of electric current and the orientation of magnetization. AMR effect can be explained with reference to figure A.3.



**Figure (A.3): Principle of Magneto- Resistive Effect**

In a thin magnetic film, the angle between magnetization ( $M$ ) and sensing current ( $I$ ) is  $\theta$ , then the electrical resistance can be defined as,

$$R = R_0 + \Delta R \cos^2 \theta$$



In the above equation  $R_0$  is the fixed part and  $\Delta R$  is the maximum value of variable resistance.

AMR effect takes place in ferrous materials. The transducer of magneto-resistive elements is in the form of a Wheatstone bridge. Resistance of all four resistor elements in the bridge is  $R$ , and supply voltage is  $V_b$ . The supply voltage causes a current to flow in the resistors. Due to the applied field  $H$ , the magnetization in two opposition resistors rotates in the direction of current and in the other two resistors it rotates away from the direction of current.

An increase in the resistance ( $R$ ) of the two resistors with magnetization in the direction of current is observed. The other two resistors have a decrease in the resistance ( $R$ ). The output will be proportional to applied field  $H$ ,  $\Delta V = S V_b$ .

So, the film resistance is greatest when the current flows parallel to the  $M$  and least when the current is perpendicular to  $M$ .

AMR sensors undergo a resistance change of 2-3% in the presence of magnetic field. The figure A.4 [4] shows the AMR circuit.

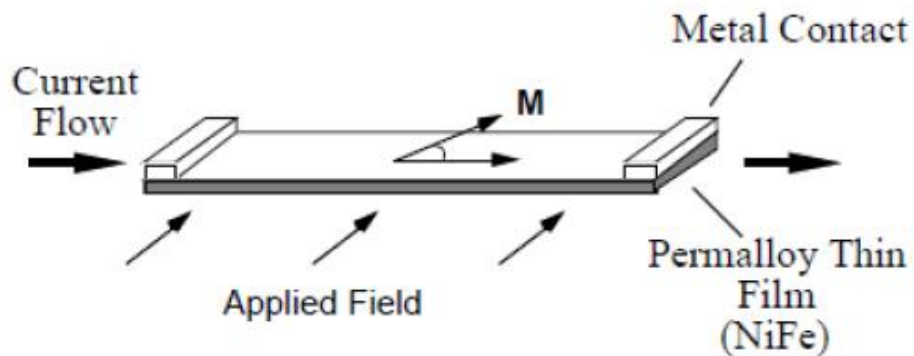


Fig (A.4): Operation of AMR sensor

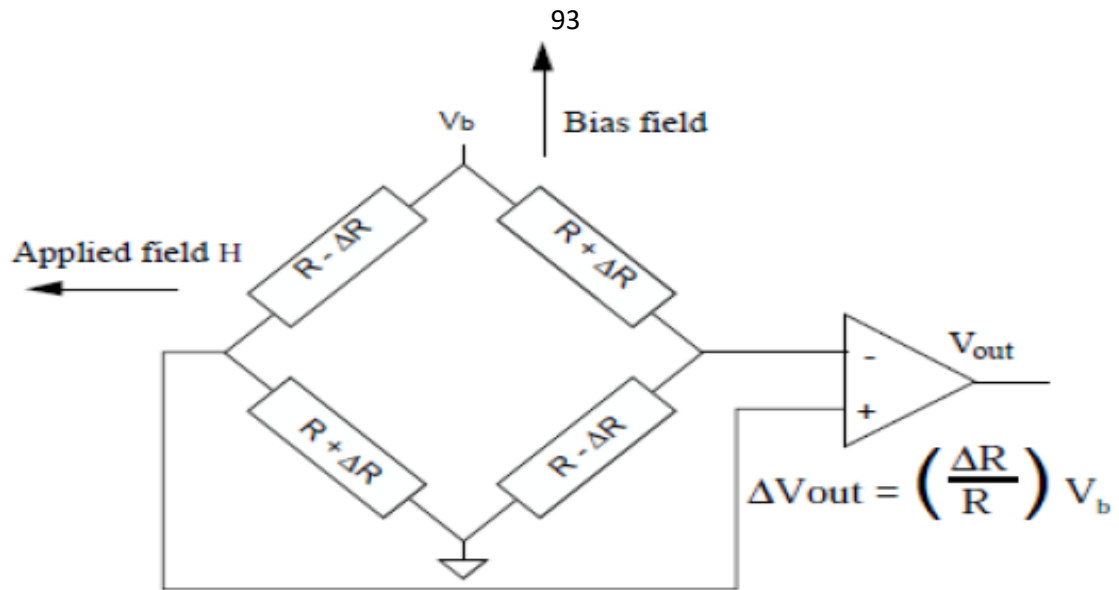


Fig (A.5): Transducer of MR elements

AMR sensors are made of a nickel-iron (Permalloy) thin film deposited on silicon wafer in a resistor bridge pattern. The deposition of Permalloy film on the wafer is done in a strong magnetic field. This strong magnetic field sets the orientation of magnetization vector in the direction preferred. The magnetization vector is parallel to the length of the resistor and can be set to either left or right in the film. The magnetic range of AMR sensors lies in the range of the Earth's magnetic field. AMR sensors can measure both linear and angular position displacement in the Earth's magnetic field.

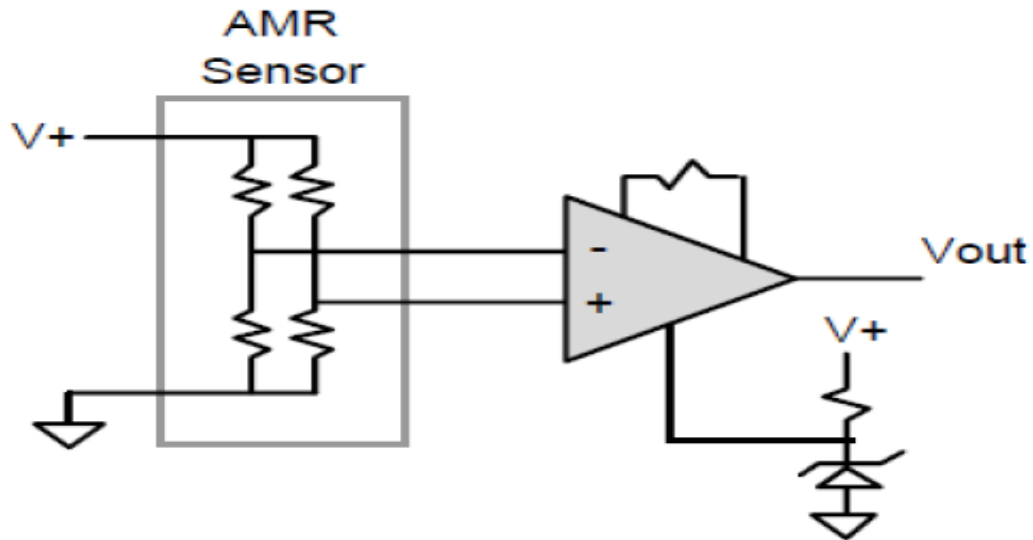


Fig (A.6): AMR sensor circuit

## 2. Resetting the Magnetization

A concern for any magnetic sensor made of ferromagnetic material is the exposure to a disturbing magnetic field. For AMR sensors, this disturbing field actually breaks down the magnetization alignment in the Permalloy film that is critical to the sensor operation. The direction and magnitude of vector  $M$  is essential to repeatable, low noise, and low hysteresis output signals. The top film in Figure A.7 illustrates the AMR film when exposed to a disturbing magnetic field. The Permalloy strip is broken up into random oriented magnetic domains that degrades the sensor operation [26] shown in Figure A.3

To recover the magnetic state, a strong magnetic field must be applied along the length of the Permalloy film to restore  $M$ . Within tens of nanoseconds the random domains will line up along the easy axis as shown in the lower film of Figure A.7. Now the  $M$  vector will stay in this state for years as long as there is no magnetic disturbing field present.

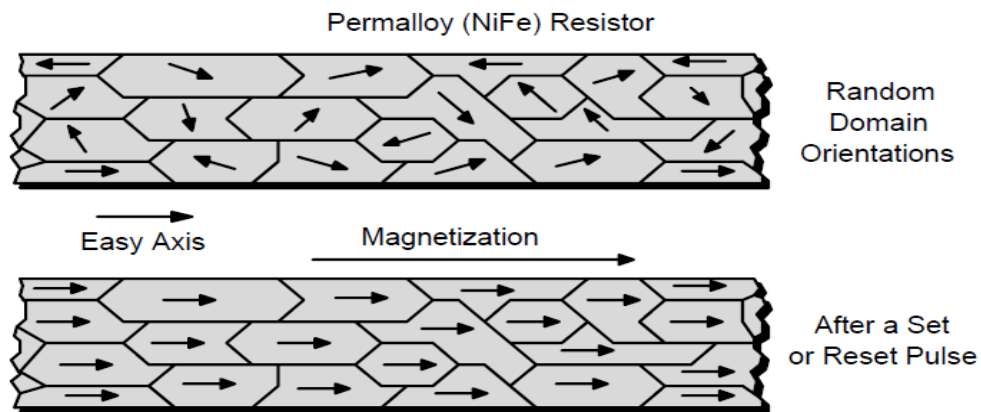


Fig (A.7): Magnetic Domain Orientation in AMR Thin Films

A common method used to realign these domains is to use external coil around the Wheatstone bridge resistors. Switching a high current pulse through the coil (Figure A.8) will create a large magnetic field of 60-100 gauss and restore the  $M$  vector [26].

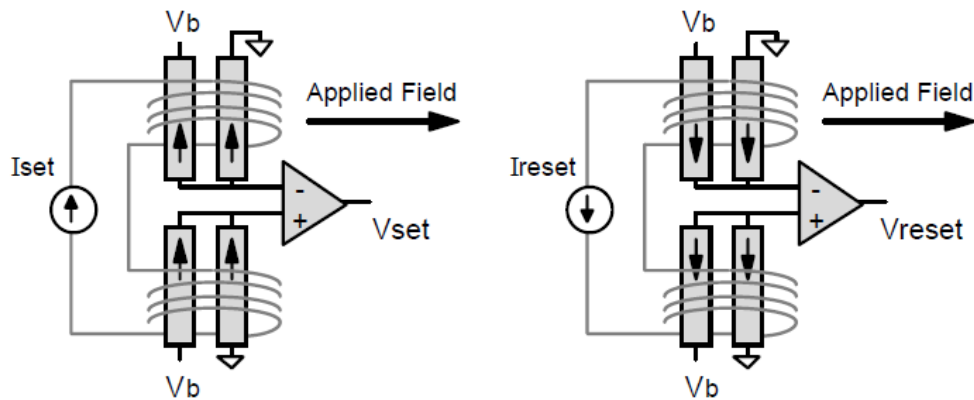


Fig (A.8): Set and Reset Flipping Circuits

This process is referred to as flipping the magnetic domains with a set pulse. This flipping action will also take place for a pulse in the opposite direction through this external coil.

In this case, the reset pulse, the domains will all point in the opposite direction along the easy axis. The KMZ-10A AMR sensor from Philips requires an external coil around the package to create the set and reset fields.

Honeywell's family of AMR sensor uses on chip strap that replaces the external coil to set and reset field effects.

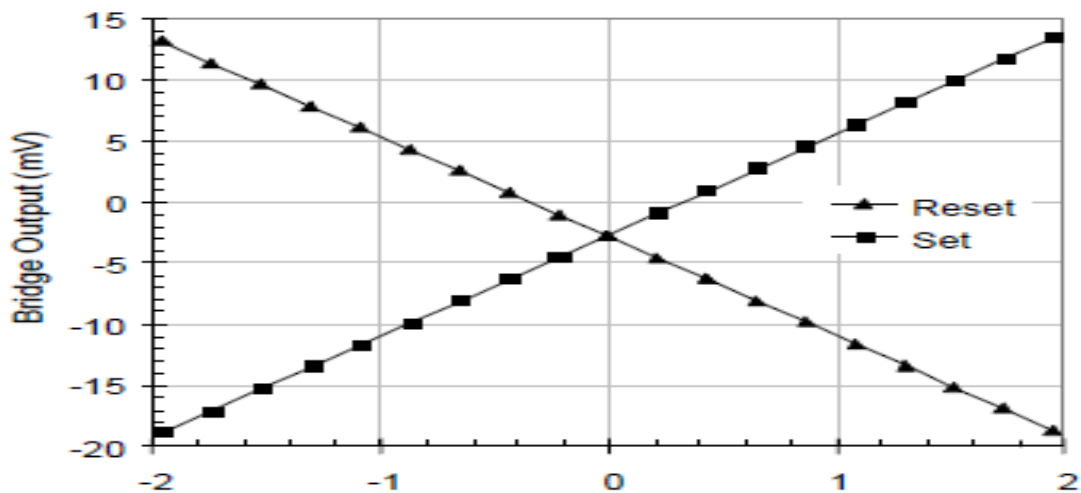


Fig (A.9): Set and Reset Output Transfer Curves

There are undesirable effects inherent in the sensor may interfere with magnetic field sensing such as bridge offset voltages and temperature effects. The transfer curves for a sensor after it has been set, and then reset, shows an inversion of the gain slope and a common crossover point on the bridge output axis (the zero-field bridge offset voltage). For the sensor in Figure (A.9), the bridge offset is around  $-3$  mV. This is due to the resistor mismatch during the manufacture process. This offset voltage is usually not desirable and can be reduced, or eliminated, using one of four techniques described below.

## 1. Manual offset trim

It is by adding a parallel trim resistor across one leg of the bridge to force both outputs to the same voltage. Addition of a resistor should be done in a zero magnetic field environment so that it does not get affected by the Earth's magnetic field. The drawback of this method is labor intensive since each sensor may require a different value trim resistor.

## 2. Offset strap

It is by using a coil to create a field in the sensitive axis direction. When a current is passed through this coil, a field which is equal to the bridge offset voltage is created and it will nullify the bridge offset.

Honeywell's family of AMR sensors uses on-chip offset strap to accomplish offset adjustment. The offset current must be determined in a zero-gauss environment and requires a constant dc source.

## 3. Set / Reset with microprocessor

This method uses numerical subtraction to measure the field  $H_{applied}$ , first activate a set pulse, then take a reading and store it as  $V_{set}$ . Again, activate a reset pulse and store the reading as  $V_{reset}$ . The expressions for these two readings, and their difference, are:

$$V_{set} = S \times H_{applied} + V_{os} \quad (1)$$

$$V_{reset} = -S \times H_{applied} + V_{os} \quad (2)$$

$$V_{set} - V_{reset} = 2 \times S \times H_{applied} \quad (3)$$

In equation (3) there is no  $V_{os}$  term.

The benefit of offset cancellation using this method is that any temperature drift of the bridge offset, including the amplifier, is eliminated

This is a powerful and easy to implement if the readings are controlled by a microprocessor.

#### **4. Electronic feedback**

A feedback signal is generated to null the bridge offset voltage. In this method, the output of the bridge is made to switch between  $V_{set}$  and  $V_{reset}$  by applying set and reset pulses.

This process modulates the output signal to a higher band. The amplifier#1 outputs a negative dc signal compared to the bridge offset and this signal is fed back to the input of amplifier#2. This process will eliminate the offset voltage. In the third stage the output which is free from bridge offset is demodulated.

Any of the above methods can be used to solve the issue bridge offset voltage in the AMR sensors.

## **B. Code for Arduino Sensor Node (Transmitter)**

*/\**

*XBee-Arduino library*

*<https://github.com/andrewrapp/xbee-arduino/blob/master/XBee.h>*

*\*/*

*#include <XBee.h>*

*/\**

*For Date and time using just software,*

*based on millis() & timer*

*\*/*

*#include <RTClib.h>*

*/\**

*The Program is for Series 1 XBee*

*Sends a TX16 or TX64 request with the values of Traffic parameters and checks the status response for success*

*Note: In my testing it took about 15 seconds for the XBee to start reporting success, so I've added a startup delay*

*\*/*



```

#include <Sleep_n0m1.h>

/* Sleep Mode Library */

XBee xbee = XBee();

RTC_Millis RTC;

unsigned long start = millis();

Sleep sleep;

unsigned long sleepTime; //How long you want the arduino to sleep.

// allocate 13 bytes for to hold traffic parameteres

uint8_t payload[] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };

// // union to convert integer year of date to byte string

union {

    int32_t j;

    byte b[4];

} u;

// Assumptions of vehicle sizes

int randsize[3]={80, 100, 200};

int v_size;

```

*//vehicle speed*

*int v\_speed;*

*// 16-bit addressing: typically the coordinator 0xFFFF*

*Tx16Request tx = Tx16Request(0xFFFF, payload, sizeof(payload));*

*TxStatusResponse txStatus = TxStatusResponse();*

*// flash TX indicator*

*int statusLed = 11;*

*int errorLed = 12;*

*void flashLed(int pin, int times, int wait)*

*{*

*for (int i = 0; i < times; i++)*

*{*

*digitalWrite(pin, HIGH);*

*delay(wait);*

*digitalWrite(pin, LOW);*

*if (i + 1 < times)*

*{*

```

    delay(wait);

}

}

}

// Function to generate normal distributed random number

//based on Average and Standard Deviation

double randn (double mu, double sigma)

{

    double U1, U2, W, mult, randnum;

    static double X1, X2;

    static int call = 0;

    if (call == 1)

    {

        call = !call;

        randnum = mu + sigma * (double) X2;

        if (randnum < 20 || randnum > 100)

            return mu;

```

```

    else

        return randnum;

    }

do

{

    // Get two random numbers

    U1 = -1 + ((double) rand () / RAND_MAX) * 2;

    U2 = -1 + ((double) rand () / RAND_MAX) * 2;

    // Radius of circle

    W = pow (U1, 2) + pow (U2, 2);

}

while (W >= 1); //If outside unit circle, then reject number

mult = sqrt ((-2 * log (W)) / W);

X1 = U1 * mult;

X2 = U2 * mult;

call = !call;

randnum = mu + sigma * (double) X1;

```

```

    if (randnum < 20 || randnum > 100)

        return mu;

    else

        return randnum;

}

// Generate normal distributed random Vehicle Size

int GenerateVehicleSize()

{

    // int pickedindex=randn(0.91, 0.41);

    int pickedindex=random(3);

    return randsize[pickedindex];

}

// Generate normal distributed random Vehicle speed

int GenerateVehicleSpeed()

{

    return randn(55, 17.8);

}

```

```

void setup()

{

    pinMode(statusLed, OUTPUT);

    pinMode(errorLed, OUTPUT);

    Serial.begin(9600);

    xbee.setSerial(Serial);

    RTC.begin(DateTime(__DATE__, __TIME__));

    sleepTime = 898000; //set sleep time in ms, max sleep time is 49.7 days

}

void loop()

{

    // start transmitting after a startup delay. Note: this will rollover to 0 eventually so
    not best way to handle

    if (millis() - start > 15000)

    {

        DateTime now =RTC.now();

        v_size = GenerateVehicleSize(); // random speed
    }

```

```
v_speed= GenerateVehicleSpeed(); // random speed
```

```
payload[0] = v_size >> 8 & 0xff; // Converting int to hex could also used
```

```
highByte() lowByte()
```

```
payload[1] = v_size & 0xff;
```

```
payload[2]=now.day();
```

```
payload[3]=now.month();
```

```
u.j=now.year();
```

```
payload[4] = u.b[3];
```

```
payload[5] = u.b[2];
```

```
payload[6] = u.b[1];
```

```
payload[7] = u.b[0];
```

```
payload[8]=now.hour();
```

```
payload[9]=now.minute();
```

```
payload[10]=now.second();
```

```
payload[11] = v_speed >> 8 & 0xff;
```

```
payload[12] = v_speed & 0xff;
```

```
xbee.send(tx);
```

```
// flash TX indicator

flashLed(statusLed, 1, 100);

sleep.pwrDownMode(); //set sleep mode

sleep.sleepDelay(sleepTime); //sleep for: sleepTime

}

// after sending a tx request, we expect a status response

// wait up to 5 seconds for the status response

if (xbee.readPacket(5000))

{

    // got a response!

    if (xbee.getResponse().getApiId() == TX_STATUS_RESPONSE)

    {

        xbee.getResponse().getTxStatusResponse(txStatus);

        // get the delivery status, the fifth byte

        if (txStatus.getStatus() == SUCCESS)

        {

            // success. time to celebrate
```



```

    flashLed(statusLed, 5, 50);

}

else

{

    // the remote XBee did not receive our packet. is it powered on?

    flashLed(errorLed, 3, 500);}}

else if (xbee.getResponse().isError())

{

    flashLed(errorLed, 3, 500);

}

else

{

    // local XBee did not provide a timely TX Status Response. Radio is not
configured properly or connected

    flashLed(errorLed, 2, 50);

}

delay(random(1000, 20000));

```

```
}

```

### C. Code for coordinator Node (Receiver)

```
Imports System.IO.Ports
Imports System.Text
Imports System.Data.SqlClient
Public Class frmTrafficMonitor
    Dim receivedData As String = ""
    Private Function FindSerialPorts() As List(Of String)
        Dim lstport As New List(Of String)
        For Each port As String In SerialPort.GetPortNames
            lstport.Add(port)
        Next
        Return lstport
    End Function
    Public Function OpenPort(ByVal sprt As SerialPort, ByVal Prtname As String, ByVal baudrate As
Integer) As String
        Dim statusmsg As String = Nothing
        Try
            'first check if the port is already open
            'if its open then close it
            If sprt.IsOpen = True Then
                sprt.Close()
            End If
            'set the properties of our SerialPort Object
            sprt.PortName = Prtname
            sprt.BaudRate = baudrate
            'BaudRate
            sprt.DataBits = 8
            'DataBits
            sprt.Parity = Parity.None
            ' Parity
            sprt.StopBits = StopBits.One
            'StopBits
            sprt.Handshake = Handshake.None
            'Handshake
            sprt.Encoding = System.Text.Encoding.Default
            'Encoding
            sprt.ReadTimeout = 10000
            'Timeout
            'now open the port
            sprt.Open()
            'display message
            statusmsg = "Port opened at " + DateTime.Now + "" + Environment.NewLine + ""
        Catch ex As Exception
            statusmsg = ex.Message.ToString
        End Try
        Return statusmsg
    End Function
    Public Function ClosePort(ByVal sprt As SerialPort)
        Dim statusmsg As String = Nothing
        If sprt.IsOpen Then
            statusmsg = "Port closed at " + DateTime.Now + "" + Environment.NewLine + ""
            sprt.Close()
        End If
        Return statusmsg
    End Function
    Private Function extractdata(ByVal frm As String, ByVal dttype As String)
```

```

Dim rtnvalue As String = Nothing
Dim tem() As String = frm.Split(" "c)
Dim frame As New List(Of String)
For Each s As String In tem
    If s <> "" Then
        frame.Add(s)
    End If
Next
Try
    Select Case dtype
        Case "Start delimiter"
            rtnvalue = frame.Item(0).ToString
        Case "Length"
            rtnvalue = frame.Item(1).ToString
        Case "Frame type"
            rtnvalue = frame.Item(2).ToString
        Case "sensorid"
            rtnvalue = frame.Item(3).ToString
        Case "RSSI"
            rtnvalue = frame.Item(4)
        Case "Options"
            rtnvalue = frame.Item(5)
        Case "RF Data"
            For i = 6 To frame.Count - 2
                rtnvalue = rtnvalue + " " + frame.Item(i)
            Next
        Case "v_class"
            rtnvalue = Convert.ToInt64(frame.Item(6), 16).ToString

            Case "detecciondate"
                rtnvalue = ZeroPadding(Convert.ToInt64(frame.Item(7), 16).ToString) + "/" +
ZeroPadding(Convert.ToInt64(frame.Item(8), 16).ToString) + "/" +
ZeroPadding(Convert.ToInt64(frame.Item(9) + frame.Item(10), 16).ToString)

                Case "detectiontime"
                    rtnvalue = ZeroPadding(Convert.ToInt64(frame.Item(11), 16).ToString) + ":" +
ZeroPadding(Convert.ToInt64(frame.Item(12), 16).ToString) + ":" +
ZeroPadding(Convert.ToInt64(frame.Item(13), 16).ToString)
                Case "v_speed"
                    rtnvalue = Convert.ToInt64(frame.Item(14), 16).ToString
                Case "Checksum"
                    rtnvalue = frame.Item(frame.Count - 1)
            End Select
    Catch ex As Exception
    End Try
    Return rtnvalue
End Function
Private Function ZeroPadding(ByVal strValue As String) As String
    If Integer.Parse(strValue) <= 9 Then
        Return "0" + strValue
    Else
        Return strValue
    End If
End Function
Private Sub InsertData(ByVal recdata As String)
    Try
        Dim counter As Integer = 1
        Dim sensorid As String = extractdata(recdata, "sensorid")
        Dim v_class As String = getVehicleClassification(extractdata(recdata, "v_class"))
        Dim detecciondate As String = extractdata(recdata, "detecciondate")
        Dim detectiontime As String = extractdata(recdata, "detectiontime")
        Dim v_speed As String = extractdata(recdata, "v_speed")
    
```

```

If DataGridView1.RowCount > 0 Then
    Dim rowcount As Integer = DataGridView1.RowCount - 1
    counter = DataGridView1.Rows(rowcount).Cells(0).Value
    counter = counter + 1
End If
DataGridView1.Rows.Add(New String() { Trim(counter.ToString), Trim(sensorid.ToString),
Trim(v_class.ToString), Trim(deteciondate.ToString), Trim(detectiontime.ToString),
Trim(v_speed.ToString)})
    Dim DetectionDatestr As Date = Date.Parse(deteciondate.ToString)
    Dim sqlDetectionDate As String = DetectionDatestr.ToString("yyyy-MM-dd")
    Dim DetectionTimestr As Date = DateTime.Parse(detectiontime.ToString)
    Dim sqlDetectionTime As String = DetectionTimestr.ToString("h:mm:ss tt")
    Dim constring As String = "Data Source=.\SQLEXPRESS;Initial Catalog=TrafficDB;User
ID=sa;Password=104123"
    Dim cn As SqlConnection = New SqlConnection(constring)
    cn.Open()
    Dim strsql As String = "INSERT INTO TrafficDB.dbo.tbl_Traffic (SenderId, Calssification,
DetectionDate, DetectionTime, Speed ) VALUES (N'" + sensorid + "', N'" + v_class + "',
CONVERT(DATETIME, '" + sqlDetectionDate + "', 101), CONVERT(DATETIME, '" + sqlDetectionTime
+ "', 108), '" + v_speed + "')"
    Dim cmd As SqlCommand = New SqlCommand(strsql, cn)
    cmd.ExecuteNonQuery()
    cn.Close()
Catch ex As Exception
    MsgBox(ex.Message)
End Try
End Sub
Function ReceiveSerialData() As String
    Dim bytes As Integer = TrafficSerialPort.BytesToRead
    'create a byte array to hold the awaiting data
    Dim comBuffer As Byte() = New Byte(bytes - 1) {}
    'read the data and store it
    TrafficSerialPort.Read(comBuffer, 0, bytes)
    'return the data to
    Dim Incoming As String = Disassembleframe(comBuffer)
Try
    If Incoming Is Nothing Then
        Return "nothing"
    Else
        Return Incoming
    End If
Catch ex As TimeoutException
    Return "Error: Serial Port read timed out."
End Try

End Function
Private Function Disassembleframe(ByVal comByte As Byte()) As String
    'create a new StringBuilder object
    Dim builder As New StringBuilder(comByte.Length * 3)
    Dim dbyte() As String = Nothing
    Dim i As Integer = 0
    'loop through each byte in the array
    For Each data As Byte In comByte
        builder.Append(Convert.ToString(data, 16).PadLeft(2, "0").PadRight(3, " "))
        'convert the byte to a string and add to the stringbuilder
    Next
    'return the converted value
    Return builder.ToString().ToUpper()
End Function

Private Function getVehicleClassification(ByVal vsize As String)

```

```

Dim resultclass As String = Nothing
Select Case vsize
    Case "80"
        resultclass = "Small"
    Case "100"
        resultclass = "Medium"
    Case "200"
        resultclass = "Large"
End Select
Return resultclass
End Function
Private Sub frmTrafficMonitor_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    RcvdTimer.Enabled = False
End Sub
Private Sub btnFindPort_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
btnFindPort.Click
    Dim ports As List(Of String) = FindSerialPorts()
    btnFindPort.Cursor = Cursors.WaitCursor
    Cmbports.Items.Clear()
    For Each prt As String In ports
        Cmbports.Items.Add(prt)
    Next
    If Cmbports.Items.Count > 0 Then
        Cmbports.SelectedIndex = 0
        CmbBrate.SelectedItem = "9600"
        btnFindPort.Cursor = Cursors.Hand
    Else
        MsgBox("COM Port not detected")
    End If
End Sub

Private Sub btnconnect_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
btnconnect.Click
    Dim message As String = Nothing
    If CmbBrate.Items.Count > 0 AndAlso Cmbports.Items.Count > 0 Then
        Try
            btnconnect.Visible = False
            btndisconnect.Visible = True
            ClosePort(TrafficSerialPort)
            message = OpenPort(TrafficSerialPort, Cmbports.SelectedItem, Val(CmbBrate.SelectedItem))
            lstRecievedFrame.Items.Add(message)
            CmbBrate.Enabled = False
            Cmbports.Enabled = False
            RcvdTimer.Enabled = True
        Catch ex As Exception
        End Try
    End If
End Sub

Private Sub btndisconnect_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btndisconnect.Click
    Try
        Dim message As String = ClosePort(TrafficSerialPort)
        lstRecievedFrame.Items.Add(message)
        btnconnect.Visible = True
        btndisconnect.Visible = False
        CmbBrate.Enabled = True
        Cmbports.Enabled = True
        Cmbports.SelectedItem = Nothing
        Cmbports.Items.Clear()
        RcvdTimer.Enabled = False
    End Try
End Sub

```

```

    Catch ex As Exception
    End Try
End Sub

Private Sub RcvdTimer_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
RcvdTimer.Tick
    receivedData = ReceiveSerialData()
    If Trim(receivedData.StartsWith("7E")) AndAlso Trim(receivedData).Length = 47 Then
        lstRcievedFrame.Items.Add(receivedData)
        InsertData(receivedData)

        Dim speed = extractdata(receivedData, "v_speed")
        Dim classification = extractdata(receivedData, "v_class")
        Dim dttime = extractdata(receivedData, "detectiontime")
        Chart1.Series("SPEED").Points.AddXY(dttime, speed)
        Chart1.Series("Classification").Points.AddXY(dttime, classification)

        If Chart1.Series(0).Points.Count = 9 Then
            Chart1.Series(0).Points.RemoveAt(0)
        End If
        If Chart1.Series(1).Points.Count = 9 Then
            Chart1.Series(1).Points.RemoveAt(0)
        End If
        Chart1.ChartAreas(0).AxisY.Maximum = 250
    End If
End Sub

Private Sub lstRcievedFrame_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles lstRcievedFrame.SelectedIndexChanged

    Try
        Dim str As String = Trim(lstRcievedFrame.SelectedItem.ToString)
        If str.StartsWith("7E") Then
            txtstartdl.Text = extractdata(str, "Start delimiter")
            txtlength.Text = extractdata(str, "Length")
            txtfrmtype.Text = extractdata(str, "Frame type")
            txtsourceadd.Text = extractdata(str, "sensorid")
            txttrssi.Text = extractdata(str, "RSSI")
            txtoption.Text = extractdata(str, "Options")
            txtRFdata.Text = extractdata(str, "RF Data")
            txtChecksum.Text = extractdata(str, "Checksum")
            txtvehicleclass.Text = getVehicleClassification(extractdata(str, "v_class"))
            txtdetctdate.Text = extractdata(str, "deteciondate")
            txttime.Text = extractdata(str, "detectiontime")
            txtspeed.Text = extractdata(str, "v_speed")
        End If
    Catch ex As Exception
    End Try
End Sub
End Class

```

```

Imports System.Data.SqlClient
Imports System.Windows.Forms.DataVisualization
Imports System.Windows.Forms.DataVisualization.Charting
Public Class Form1

```

```

Dim constring As String = "Data Source=.\SQLEXPRESS;Initial Catalog=TrafficDB;Integrated
Security=SSPI"
Dim cn As SqlConnection = New SqlConnection(constring)
Private Sub button13_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
button13.Click
    Application.Exit()
End Sub
Private Function CalculateNumberofVehicles()
    Dim v_num As Integer = 0
    Dim sqlstr As String = "SELECT Count(SenderId) AS Counter FROM TrafficDB.dbo.tbl_Traffic
WHERE (DetectionDate = " + getDetectionDate() + ") AND (CAST(DetectionTime AS TIME) BETWEEN
" + getDetectionTime() + ") AND (SenderId=" + cmbNodeId.SelectedItem.ToString + ")"
    Try
        cn.Open()
        Dim cmd As New SqlCommand(sqlstr, cn)
        Dim myreader As SqlDataReader = cmd.ExecuteReader()
        While myreader.Read()
            v_num = myreader("Counter").ToString()
        End While
    Catch ex As Exception
        MsgBox(ex.Message)
    Finally
        cn.Close()
    End Try
    Return v_num
End Function
Private Function getDetectionDate()
    Dim str As String = Nothing
    Dim dt As Date = pikdate.Value
    str = "CONVERT(DATETIME, " + getsqldate(dt) + ", 102)"
    Return str
End Function
Private Function getDetectionTime()
    Return "" + pikfromtime.Text.ToString() + " and " + piktotime.Text.ToString() + ""
End Function
Private Function getsqldate(ByVal dd As Date)
    Dim day As String = dd.Day
    Dim month As String = dd.Month
    Dim year As String = dd.Year
    Dim d As String = year + "-" + month + "-" + day
    Return d
End Function
Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
    InitializeControls()
End Sub

Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button2.Click
    If Chart1.Series(0).Points.Count = 0 Then
        lblV_passed.Text = CalculateNumberofVehicles()
        Label27.Text = getclassificationcounter("Large")
        Label25.Text = getclassificationcounter("Medium")
        Label23.Text = getclassificationcounter("Small")

        Label17.Text = getspeedcounter("20", "39")
        Label15.Text = getspeedcounter("40", "59")
        Label9.Text = getspeedcounter("60", "79")
        Label19.Text = getspeedcounter("80", "99")
        Label21.Text = getspeedcounter("100", "999")
        drawpiechart()

        DrawColChart()
    End If
End Sub

```

```

        GroupBox9.Visible = True

    End If
End Sub
Private Sub drawpiechart()
    Chart1.Series("SPEED").Points.AddXY(Label16.Text, Label17.Text)
    Chart1.Series("SPEED").Points.AddXY(Label14.Text, Label15.Text)
    Chart1.Series("SPEED").Points.AddXY(Label7.Text, Label9.Text)
    Chart1.Series("SPEED").Points.AddXY(Label18.Text, Label19.Text)

    Chart2.Series("Classification").Points.AddXY(Label26.Text, Label27.Text)
    Chart2.Series("Classification").Points.AddXY(Label24.Text, Label25.Text)
    Chart2.Series("Classification").Points.AddXY(Label22.Text, Label23.Text)

End Sub
Private Function getclassificationcounter(ByVal cls As String)
    Dim rvalue As Integer = 0
    Dim sqlstr As String = "SELECT Count(SenderId) AS Counter FROM TrafficDB.dbo.tbl_Traffic
WHERE (DetectionDate = " + getDetectionDate() + ") AND (CAST(DetectionTime AS TIME) BETWEEN
" + getDetectionTime() + ") AND (Calssification LIKE N%" + cls + "%') AND ( SenderId=" +
cmbNodeId.SelectedItem.ToString + ")"
    Try
        cn.Open()
        Dim cmd As New SqlCommand(sqlstr, cn)
        Dim myreader As SqlDataReader = cmd.ExecuteReader()
        While myreader.Read()
            rvalue = myreader("Counter").ToString()
        End While
    Catch ex As Exception
        MsgBox(ex.Message)
    Finally
        cn.Close()
    End Try
    Return rvalue
End Function
Private Function getspeedcounter(ByVal minspeed As String, ByVal maxspeed As String)
    Dim rvalue As Integer = 0
    Dim sqlstr As String = "SELECT Count(SenderId) AS Counter FROM TrafficDB.dbo.tbl_Traffic
WHERE (DetectionDate = " + getDetectionDate() + ") AND (CAST(DetectionTime AS TIME) BETWEEN
" + getDetectionTime() + ") AND (Speed BETWEEN " + minspeed + " AND " + maxspeed + ") AND (
SenderId=" + cmbNodeId.SelectedItem.ToString + ")"
    Try
        cn.Open()
        Dim cmd As New SqlCommand(sqlstr, cn)
        Dim myreader As SqlDataReader = cmd.ExecuteReader()
        While myreader.Read()
            rvalue = myreader("Counter").ToString()
        End While
    Catch ex As Exception
        MsgBox(ex.Message)
    Finally
        cn.Close()
    End Try
    Return rvalue
End Function
Private Function getspeed_with_sizecounter(ByVal minspeed As String, ByVal maxspeed As String,
ByVal cls As String)

    Dim rvalue As Integer = 0
    Dim sqlstr As String = "SELECT Count(SenderId) AS Counter FROM TrafficDB.dbo.tbl_Traffic
WHERE (DetectionDate = " + getDetectionDate() + ") AND (CAST(DetectionTime AS TIME) BETWEEN

```



```

" + getDetectionTime() + ") AND (Speed BETWEEN " + minspeed + " AND " + maxspeed + ") AND
(Calssification LIKE "" + cls + ") AND ( SenderId="" + cmbNodeId.SelectedItem.ToString + ")
Try
    cn.Open()
    Dim cmd As New SqlCommand(sqlstr, cn)
    Dim myreader As SqlDataReader = cmd.ExecuteReader()
    While myreader.Read()
        rvalue = myreader("Counter").ToString()
    End While
Catch ex As Exception
    MsgBox(ex.Message)
Finally
    cn.Close()
End Try
Return rvalue
End Function
Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button3.Click
    InitializeControls()
End Sub

Private Sub DrawColChart()

    Dim Spd20_39 As Integer = getspeedcounter("20", "39")
    Dim Spd40_59 As Integer = getspeedcounter("40", "59")
    Dim Spd60_79 As Integer = getspeedcounter("60", "79")
    Dim Spd80_99 As Integer = getspeedcounter("80", "99")
    Dim Spd100_999 As Integer = getspeedcounter("100", "999")

    Dim Spd20_39_Large As Integer = getspeed_with_sizecounter("20", "39", "Large")
    Dim Spd40_59_Large As Integer = getspeed_with_sizecounter("40", "59", "Large")
    Dim Spd60_79_Large As Integer = getspeed_with_sizecounter("60", "79", "Large")
    Dim Spd80_99_Large As Integer = getspeed_with_sizecounter("80", "99", "Large")
    Dim Spd100_999_Large As Integer = getspeed_with_sizecounter("100", "999", "Large")

    Dim Spd20_39_Medium As Integer = getspeed_with_sizecounter("20", "39", "Medium")
    Dim Spd40_59_Medium As Integer = getspeed_with_sizecounter("40", "59", "Medium")
    Dim Spd60_79_Medium As Integer = getspeed_with_sizecounter("60", "79", "Medium")
    Dim Spd80_99_Medium As Integer = getspeed_with_sizecounter("80", "99", "Medium")
    Dim Spd100_999_Medium As Integer = getspeed_with_sizecounter("100", "999", "Medium")

    Dim Spd20_39_Small As Integer = getspeed_with_sizecounter("20", "39", "Small")
    Dim Spd40_59_Small As Integer = getspeed_with_sizecounter("40", "59", "Small")
    Dim Spd60_79_Small As Integer = getspeed_with_sizecounter("60", "79", "Small")
    Dim Spd80_99_Small As Integer = getspeed_with_sizecounter("80", "99", "Small")
    Dim Spd100_999_Small As Integer = getspeed_with_sizecounter("100", "999", "Small")

    Chart4.ChartAreas(0).AxisX.LabelStyle.Enabled = True
    Chart4.Series("SPEED").Points.AddXY("20 - 39", Spd20_39)
    Chart4.Series("Large").Points.AddXY("20 - 39", Spd20_39_Large)
    Chart4.Series("Meduim").Points.AddXY("20 - 39", Spd20_39_Medium)
    Chart4.Series("Small").Points.AddXY("20 - 39", Spd20_39_Small)

    Chart4.Series("SPEED").Points.AddXY("40 - 59", Spd40_59)
    Chart4.Series("Large").Points.AddXY("40 - 59", Spd40_59_Large)
    Chart4.Series("Meduim").Points.AddXY("40 - 59", Spd40_59_Medium)
    Chart4.Series("Small").Points.AddXY("40 - 59", Spd40_59_Small)

    Chart4.Series("SPEED").Points.AddXY("60 - 79", Spd60_79)
    Chart4.Series("Large").Points.AddXY("60 - 79", Spd60_79_Large)
    Chart4.Series("Meduim").Points.AddXY("60 - 79", Spd60_79_Medium)
    Chart4.Series("Small").Points.AddXY("60 - 79", Spd60_79_Small)

```

```

Chart4.Series("SPEED").Points.AddXY("80 - 99", Spd80_99)
Chart4.Series("Large").Points.AddXY("80 - 99", Spd80_99_Large)
Chart4.Series("Meduim").Points.AddXY("80 - 99", Spd80_99_Medium)
Chart4.Series("Small").Points.AddXY("80 - 99", Spd80_99_Small)

Chart4.Series("SPEED").Points.AddXY("100 - ", Spd100_999)
Chart4.Series("Large").Points.AddXY("100 - ", Spd100_999_Large)
Chart4.Series("Meduim").Points.AddXY("100 - ", Spd100_999_Medium)
Chart4.Series("Small").Points.AddXY("100 - ", Spd100_999_Small)

End Sub

Private Sub InitializeControls()
    pikdate.Value = Now.Date
    pikfromtime.Text = "00:00:00"
    piktotime.Text = "23:59:59"
    lblV_passed.Text = Nothing
    Chart1.Series(0).Points.Clear()
    Chart2.Series(0).Points.Clear()
    Chart4.Series(0).Points.Clear()
    Chart4.Series(1).Points.Clear()
    Chart4.Series(2).Points.Clear()
    Chart4.Series(3).Points.Clear()

    Label27.Text = Nothing
    Label25.Text = Nothing
    Label23.Text = Nothing

    Label17.Text = Nothing
    Label15.Text = Nothing
    Label9.Text = Nothing
    Label19.Text = Nothing
    Label21.Text = Nothing
    cmbNodeId.SelectedItem = "01"

End Sub

End Class

```

## D. Code of traffic visualization

```

Imports System.Threading
Imports System.Data.SqlClient
Public Class Form1
    Dim imgs(16) As Bitmap
    Dim retimgs(16) As Bitmap
    Dim car1 As PictureBox
    Dim car2 As PictureBox
    Dim car3 As PictureBox
    Dim car4 As PictureBox
    Dim thread1 As System.Threading.Thread
    Dim thread2 As System.Threading.Thread
    Dim thread3 As System.Threading.Thread
    Dim thread4 As System.Threading.Thread

    Dim count1 As Integer = 0
    Dim count2 As Integer = 0

```

```

Dim count3 As Integer = 0
Dim count4 As Integer = 0

Dim yLocation1 As Integer = 0
Dim yLocation2 As Integer = 0
Dim yLocation3 As Integer = 0
Dim yLocation4 As Integer = 0

Dim yLocation11 As Integer = 0
Dim yLocation22 As Integer = 0
Dim yLocation33 As Integer = 0
Dim yLocation44 As Integer = 0

Private counted1 As Boolean = False
Private counted2 As Boolean = False
Private counted3 As Boolean = False
Private counted4 As Boolean = False

Private settime1 As Boolean = False
Private settime2 As Boolean = False
Private settime3 As Boolean = False
Private settime4 As Boolean = False

Private adddb1 As Boolean = False
Private adddb2 As Boolean = False
Private adddb3 As Boolean = False
Private adddb4 As Boolean = False

Private V_Calss1 As String = Nothing
Private V_Calss2 As String = Nothing
Private V_Calss3 As String = Nothing
Private V_Calss4 As String = Nothing

Private detectiontime1 As Date = Nothing
Private detectiontime2 As Date = Nothing
Private detectiontime3 As Date = Nothing
Private detectiontime4 As Date = Nothing

Private detectiontime21 As Date = Nothing
Private detectiontime22 As Date = Nothing
Private detectiontime23 As Date = Nothing
Private detectiontime24 As Date = Nothing

Delegate Sub UpdateProgressHandler(ByVal TotalPages As Integer, ByVal CC As String)

Private Shared random As Random = New Random()

Private Function RandomNumber(ByVal min As Integer, ByVal max As Integer)
    Return random.Next(min, max)
End Function

Private Sub sotreImgList()
    imgs(0) = vehicle_Counter.My.Resources.C01
    imgs(1) = vehicle_Counter.My.Resources.C02
    imgs(2) = vehicle_Counter.My.Resources.C03
    imgs(3) = vehicle_Counter.My.Resources.C04
    imgs(4) = vehicle_Counter.My.Resources.C05
    imgs(5) = vehicle_Counter.My.Resources.C06
    imgs(6) = vehicle_Counter.My.Resources.C07
    imgs(7) = vehicle_Counter.My.Resources.C08
    imgs(8) = vehicle_Counter.My.Resources.C09

```

```

    imgs(9) = vehicle_Counter.My.Resources.C10
    imgs(10) = vehicle_Counter.My.Resources.C11
    imgs(11) = vehicle_Counter.My.Resources.C12
    imgs(12) = vehicle_Counter.My.Resources.C13
    imgs(13) = vehicle_Counter.My.Resources.C14
    imgs(14) = vehicle_Counter.My.Resources.C15
    imgs(15) = vehicle_Counter.My.Resources.C16
    imgs(16) = vehicle_Counter.My.Resources.C17
    retimgs(0) = vehicle_Counter.My.Resources.R01
    retimgs(1) = vehicle_Counter.My.Resources.R02
    retimgs(2) = vehicle_Counter.My.Resources.R03
    retimgs(3) = vehicle_Counter.My.Resources.R04
    retimgs(4) = vehicle_Counter.My.Resources.R05
    retimgs(5) = vehicle_Counter.My.Resources.R06
    retimgs(6) = vehicle_Counter.My.Resources.R07
    retimgs(7) = vehicle_Counter.My.Resources.R08
    retimgs(8) = vehicle_Counter.My.Resources.R09
    retimgs(9) = vehicle_Counter.My.Resources.R10
    retimgs(10) = vehicle_Counter.My.Resources.R11
    retimgs(11) = vehicle_Counter.My.Resources.R12
    retimgs(12) = vehicle_Counter.My.Resources.R13
    retimgs(13) = vehicle_Counter.My.Resources.R14
    retimgs(14) = vehicle_Counter.My.Resources.R15
    retimgs(15) = vehicle_Counter.My.Resources.R16
    retimgs(16) = vehicle_Counter.My.Resources.R17
End Sub
Private Function getImgsize(ByVal index As Integer)
    Dim imgsize As Size = New Size(50, 100)
    If index <= 5 Then
        imgsize = New Size(80, 200)
    End If

    If index >= 13 Then
        imgsize = New Size(40, 80)
    End If
    Return imgsize
End Function
Private Function classification(ByVal hiegt As Integer)
    Dim cls As String = Nothing
    Select Case hiegt
        Case 100
            cls = "Medium"
        Case 200
            cls = "Large"
        Case 80
            cls = "Small"
    End Select
    Return cls
End Function
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    Control.CheckForIllegalCrossThreadCalls = False
    sotreImgList()
End Sub
Private Sub LoadCarLane1()
    Dim randimg As Integer = RandomNumber(0, 16)
    car1 = New PictureBox
    car1.Image = imgs(randimg)
    car1.Size = getImgsize(randimg)
    car1.BringToFront()
    car1.Location = New Point(400, 700)
    car1.SizeMode = PictureBoxSizeMode.Normal

```

```

    car1.Visible = True
    Me.Controls.Add(car1)
End Sub

Private Sub LoadCarLane2()
    Dim randimg As Integer = RandomNumber(0, 16)
    car2 = New PictureBox
    car2.Image = imgs(randimg)
    car2.Size = getImgsize(randimg)
    car2.BringToFront()
    car2.Location = New Point(300, 700)
    car2.SizeMode = PictureBoxSizeMode.Normal
    car2.Visible = True
    Me.Controls.Add(car2)
End Sub

Private Sub LoadCarLane3()
    Dim randimg As Integer = RandomNumber(0, 16)
    car3 = New PictureBox
    car3.Image = retimgs(randimg)
    car3.Size = getImgsize(randimg)
    car3.BringToFront()
    car3.Location = New Point(48, -300)
    car3.SizeMode = PictureBoxSizeMode.Normal
    car3.Visible = True
    Me.Controls.Add(car3)
End Sub

Private Sub LoadCarLane4()
    Dim randimg As Integer = RandomNumber(0, 16)
    car4 = New PictureBox
    car4.Image = imgs(randimg)
    car4.Size = getImgsize(randimg)
    car4.BringToFront()
    car4.Location = New Point(150, -300)
    car4.SizeMode = PictureBoxSizeMode.Normal
    car4.Visible = True
    Me.Controls.Add(car4)
End Sub

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
    Try
        Me.Controls.Remove(car1)
        Me.Controls.Remove(car2)
        Me.Controls.Remove(car3)
        Me.Controls.Remove(car4)
    Catch ex As Exception
    End Try

    LoadCarLane1()
    Timer1.Enabled = True
    LoadCarLane2()
    Timer2.Enabled = True
    LoadCarLane3()
    Timer3.Enabled = True
    LoadCarLane4()
    Timer4.Enabled = True
End Sub

Private Sub Timer1_Tick(ByVal sender As Object, ByVal e As System.EventArgs) Handles Timer1.Tick
    thread1 = New Threading.Thread(AddressOf movecar1)

```

```

thread1.Start()

If car1.Location.Y < 425 AndAlso counted1 = False Then
    yLocation1 = car1.Location.Y
    count1 = count1 + 1
    V_Calss1 = classification(car1.Height)
    detectiontime1 = TimeOfDay
    counted1 = True
End If
If car1.Location.Y < 130 AndAlso settime1 = False Then
    yLocation11 = car1.Location.Y
    detectiontime21 = TimeOfDay
    settime1 = True
End If

tblfstlane.Text = CInt(count1)
lblclass1.Text = V_Calss1
lbltime1.Text = detectiontime1.ToString("h:mm:ss tt")
lbltime21.Text = detectiontime21.ToString("h:mm:ss tt")
Dim timediff As TimeSpan = (detectiontime21 - detectiontime1)
Dim vspeed As Double
If timediff.TotalSeconds >= 0 Then
    vspeed = (yLocation1 - yLocation11) / timediff.TotalSeconds
    lblspeed1.Text = vspeed

    If adddb1 = False AndAlso V_Calss1 <> Nothing AndAlso counted1 = True Then
        DataGridView1.Rows.Add(New String() {V_Calss1, Now.Date.ToString("dd/MM/yyyy"),
detectiontime1.ToString("h:mm:ss tt"), vspeed, "Right Lane1"})
        adddb1 = True
    End If
End If
tblfstlane.Refresh()
End Sub

Private Sub movecar1()
    car1.Location = New Point(car1.Location.X, car1.Location.Y - RandomNumber(2, 7))
    If car1.Location.Y < -300 Then
        Dim randimg As Integer = RandomNumber(0, 16)
        car1.Size = getImgsize(randimg)
        car1.Location = New Point(car1.Location.X, 700)
        car1.Image = imgs(randimg)
        car1.Location = New Point(car1.Location.X, car1.Location.Y - RandomNumber(2, 7))
        counted1 = False
        settime1 = False
        adddb1 = False
    End If
    Me.ResumeLayout()
End Sub

Private Sub movecar3()

    car3.Location = New Point(car3.Location.X, car3.Location.Y + RandomNumber(2, 7))
    If car3.Location.Y > 700 Then
        Dim randimg As Integer = RandomNumber(0, 16)
        car3.Size = getImgsize(randimg)
        car3.Location = New Point(car3.Location.X, -300)
        car3.Image = retimgs(randimg)
        car3.Location = New Point(car3.Location.X, car3.Location.Y + RandomNumber(2, 7))
        counted3 = False
        settime3 = False
    End If
End Sub

```

```

        adddb3 = False
    End If
    Me.ResumeLayout()

End Sub
Private Sub movecar4()

    car4.Location = New Point(car4.Location.X, car4.Location.Y + RandomNumber(3, 10))
    If car4.Location.Y > 700 Then
        Dim randimg As Integer = RandomNumber(0, 16)
        car4.Size = getImgsize(randimg)
        car4.Location = New Point(car4.Location.X, -300)
        car4.Image = retimgs(randimg)
        car4.Location = New Point(car4.Location.X, car4.Location.Y + RandomNumber(3, 10))
        counted4 = False
        settime4 = False
        adddb4 = False
    End If
    Me.ResumeLayout()

End Sub

Private Sub Timer2_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Timer2.Tick

    thread2 = New Threading.Thread(AddressOf movecar2)
    thread2.Start()
    If car2.Location.Y < 425 AndAlso counted2 = False Then
        yLocation2 = car2.Location.Y
        count2 = count2 + 1
        counted2 = True
        V_Calss2 = classification(car2.Height)
        detectiontime2 = TimeOfDay
    End If

    If car2.Location.Y < 130 AndAlso settime2 = False Then
        yLocation22 = car2.Location.Y
        detectiontime22 = TimeOfDay
        settime2 = True
    End If
    lblscndlane.Text = CInt(count2)
    lblclass2.Text = V_Calss2
    lbltime2.Text = detectiontime2.ToString("h:mm:ss tt")
    lbltime22.Text = detectiontime22.ToString("h:mm:ss tt")
    Dim timediff As TimeSpan = (detectiontime22 - detectiontime2)
    Dim vspeed As Double
    If timediff.TotalSeconds >= 0 Then
        vspeed = (yLocation2 - yLocation22) / timediff.TotalSeconds
        lblspeed2.Text = vspeed
        If adddb2 = False AndAlso V_Calss2 <> Nothing AndAlso counted2 = True Then
            DataGridView1.Rows.Add(New String() {V_Calss2, Now.Date.ToString("dd/MM/yyyy"),
detectiontime2.ToString("h:mm:ss tt"), vspeed, "Right Lane2"})
            adddb2 = True
        End If
    End If
    lblscndlane.Refresh()

End Sub

Private Sub movecar2()

```

```

car2.Location = New Point(car2.Location.X, car2.Location.Y - RandomNumber(3, 10))
If car2.Location.Y < -300 Then
    Dim randimg As Integer = RandomNumber(0, 16)
    car2.Size = getImgsSize(randimg)
    car2.Location = New Point(car2.Location.X, 700)
    car2.Image = imgs(randimg)
    car2.Location = New Point(car2.Location.X, car2.Location.Y - RandomNumber(3, 10))
    counted2 = False
    settime2 = False
    adddb2 = False
End If
Me.ResumeLayout()

End Sub

Private Sub Timer3_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Timer3.Tick
    thread3 = New Threading.Thread(AddressOf movecar3)
    thread3.Start()
    If car3.Location.Y > 40 AndAlso counted3 = False Then
        yLocation3 = car3.Location.Y
        count3 = count3 + 1
        counted3 = True
        V_Calss3 = classification(car3.Height)
        detectiontime3 = TimeOfDay
    End If

    If car3.Location.Y > 300 AndAlso settime3 = False Then
        yLocation33 = car3.Location.Y
        detectiontime23 = TimeOfDay
        settime3 = True
    End If

    lblthirdlane.Text = CInt(count3)
    lblclass3.Text = V_Calss3
    lbltime3.Text = detectiontime3.ToString("h:mm:ss tt")
    lbltime23.Text = detectiontime23.ToString("h:mm:ss tt")
    Dim vspeed As Double
    Dim timediff As TimeSpan = (detectiontime23 - detectiontime3)
    If timediff.TotalSeconds >= 0 Then
        vspeed = (yLocation33 - yLocation3) / timediff.TotalSeconds
        lblspeed3.Text = vspeed
        If adddb3 = False AndAlso V_Calss3 <> Nothing AndAlso counted3 = True Then
            DataGridView1.Rows.Add(New String() {V_Calss3, Now.Date.ToString("dd/MM/yyyy"),
detectiontime3.ToString("h:mm:ss tt"), vspeed, "Left Lane1"})
            adddb3 = True
        End If
    End If

    tblfstlane.Refresh()

End Sub

Private Sub Timer4_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Timer4.Tick
    thread4 = New Threading.Thread(AddressOf movecar4)
    thread4.Start()
    If car4.Location.Y > 40 AndAlso counted4 = False Then
        yLocation4 = car4.Location.Y
        count4 = count4 + 1
        counted4 = True
        V_Calss4 = classification(car4.Height)

```



```

        detectiontime4 = TimeOfDay
    End If

    If car4.Location.Y > 300 AndAlso settime4 = False Then
        yLocation44 = car4.Location.Y
        detectiontime24 = TimeOfDay
        settime4 = True
    End If
    lblfourthlane.Text = CInt(count4)
    lblclass4.Text = V_Calss4
    lbltime4.Text = detectiontime4.ToString("h:mm:ss tt")
    lbltime24.Text = detectiontime24.ToString("h:mm:ss tt")
    Dim timediff As TimeSpan = (detectiontime24 - detectiontime4)
    Dim vspeed As Double
    If timediff.TotalSeconds >= 0 Then
        vspeed = (yLocation44 - yLocation4) / timediff.TotalSeconds
        lblspeed4.Text = vspeed
        If adddb4 = False AndAlso V_Calss4 <> Nothing AndAlso counted4 = True Then
            DataGridView1.Rows.Add(New String() {V_Calss4, Now.Date.ToString("dd/MM/yyyy"),
detectiontime4.ToString("h:mm:ss tt"), vspeed, "Left Lane2"})
            adddb4 = True
        End If
    End If

    lblfourthlane.Refresh()
End Sub

Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button3.Click
    Timer1.Dispose()
    Timer2.Dispose()
    Timer3.Dispose()
    Timer4.Dispose()
    Button1.Text = "Restart"
End Sub

Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button2.Click
    Timer1.Dispose()
    Timer2.Dispose()
    Timer3.Dispose()
    Timer4.Dispose()
    Me.Close()
End Sub

Private Sub saverecords()
    Dim constring As String = "Data Source=DESKTOP-BBKMTI1\SQLEXPRESS;User
ID=sa;Password=104123"
    Dim cn As SqlConnection = New SqlConnection(constring)
    Dim strsql As String = Nothing

    DataGridView1.CurrentCell = DataGridView1(1, 1)
    For Each row As DataGridViewRow In DataGridView1.Rows
        Try
            Dim V_Class As String = row.Cells("V_Class").Value.ToString
            Dim DetectionDatestr As Date = Date.Parse(row.Cells("DetectionDate").Value.ToString)
            Dim DetectionDate As String = DetectionDatestr.ToString("yyyy-MM-dd")
            Dim DetectionTimestr As Date = DateTime.Parse(row.Cells("DetectionTime").Value.ToString)
            Dim DetectionTime As String = DetectionTimestr.ToString("h:mm:ss tt")
            Dim Speed As String = row.Cells("Speed").Value.ToString

```

```

        Dim Direction As String = row.Cells("Direction").Value.ToString
        cn.Open()
        strsql = "INSERT INTO [vehicle Counter].dbo.MainTable(V_Class, DetectionDate,
DetectionTime, Speed, Direction) VALUES ('" + V_Class + "', CONVERT(DATETIME, '" +
DetectionDate + "', 101), CONVERT(DATETIME, '" + DetectionTime + "', 108), '" + Speed + "', '" +
Direction + "')"

        Dim cmd As SqlCommand = New SqlCommand(strsql, cn)
        cmd.ExecuteNonQuery()
        Catch ex As Exception
            MsgBox(ex.Message)
        Finally
            cn.Close()

        End Try
        Next
        DataGridView1.Rows.Clear()
        removealldata(GroupBox2)
        removealldata(GroupBox3)
        removealldata(GroupBox4)
        removealldata(GroupBox5)

        Timer1.Dispose()
        Timer2.Dispose()
        Timer3.Dispose()
        Timer4.Dispose()

    End Sub

    Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button4.Click
        saverecords()
    End Sub
    Private Sub removealldata(ByVal gpbox As GroupBox)
        For Each lbl As Label In gpbox.Controls
            lbl.Text = "0"
        Next
    End Sub
End Class

```

جامعة النجاح الوطنية  
كلية الدراسات العليا

# تطوير شبكة استشعار لاسلكية لرصد حركة المروور

إعداد  
محمد فادي عبد الحق

إشراف  
د. عدنان سلمان

قدمت هذه الأطروحة استكمالاً لمتطلبات الحصول على درجة الماجستير في الحوسبة المتقدمة  
بكلية الدراسات العليا في جامعة النجاح الوطنية في نابلس، فلسطين.

2019

ب

تطوير شبكة استشعار لاسلكية لرصد حركة المرور

إعداد

محمد فادي عبد الحق

إشراف

د. عدنان سلمان

## الملخص

تلعب إدارة أنظمة المرور الذكية (ITS) دورًا مهمًا في نظام النقل الحديث حيث يعتبر تطوير أنظمة مراقبة حركة المرور من الأبحاث الهامة في هذا المجال.

أحد أهم أنظمة المرور الذكية هي تلك الأنظمة التي تعتمد على تطبيقات متقدمة لجمع معلومات عن حالة المرور في الزمن الحقيقي مثل عد المركبات وقياس سرعاتها وتحديد أحجامها من أجل اتخاذ قرارات ذكية.

تعتمد أنظمة مراقبة حركة المرور الحالية بشكل أساسي على كاميرات تصوير الفيديو أو الحلقات الاستقرائية. هذه الأنظمة لها العديد من القيود. فعلى سبيل المثال، أداء الأنظمة القائمة على كاميرات تصوير الفيديو يتأثر بأحوال الطقس كالامطار الغزيرة والثلوج كذلك يحتاج نشر وصيانة الحلقات الاستقرائية إلى حفر سطح الطريق وبالتالي عرقلة لحركة المركبات، علاوة على ذلك، تكلفة كلا النوعين من أنظمة مراقبة حركة المرور عالية وليست مناسبة للنشر على نطاق واسع.

في هذه الرسالة، قمنا بتطوير شبكة استشعار لاسلكية (WSN) لعد المركبات وتحديد حجمها وقياس السرعة على أساس أجهزة الاستشعار المغناطيسية التي توضع على جانب الطريق.

بيانات المرور التي تم جمعها من أجهزة الاستشعار اللاسلكية يتم إرسالها إلى كمبيوتر مركزي لاسلكيا لمعالجتها وتحليلها وحفظها في قاعدة بيانات وكما يمكن مشاركتها مع تطبيقات أخرى من خلال استخدام خدمات الويب.

ج

التطبيقات التي تم تطويرها في الاطروحة توفر مراقبة حركة المرور في الوقت الحقيقي، وتكلفة منخفضة، كما تضمن مراقبة مناسبة للطرق ومشاركة بشرية أقل.

ينصب تركيز هذه الاطروحة على تطوير شبكة استشعار لاسلكية لادارة بيانات المرور حيث تم محاكاة البيانات لتقييم النظام بدلاً من نشر أجهزة استشعار مغناطيسية.