



Development of a Prototype Wireless Vehicle Speed Monitoring System

Md. Mominul Ahsan BSc Eng

A thesis submitted for the degree of Master of Engineering (MEng)

School of Mechanical and Manufacturing Engineering

Dublin City University, Dublin, Ireland

Supervisors: Professor Saleem Hashmi and Dr. Jennifer McManis

January 2014

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Master of Engineering is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: _____ (Candidate) ID No.: 59119594 Date: _____

Acknowledgement

First, I would like to express my deepest respect and honour to my supervisor, Professor M.S.J Hashmi for his constant encouragement and kind assistance during the whole period of my research. His guidance and motivation techniques helped me to keep stay in the progressive way to reach the final stage of my research. Furthermore, his kind suggestions have helped me to overcome any difficult situation in all stages during this period. I would also like to thank my other supervisor, Dr. Jennifer McManis for her kind support throughout my study. Without their guidance and kind co-operation, it would not have been possible to complete my research work.

I wish to express my cordial thanks to the technicians, Liam Domican and Keith Hickey, for their valuable help and kind understanding for me during the research period. They were always provided me the necessary instruments for experimental set-up and software supports in solving any computing problem. I am really grateful to them for their kind support.

Finally, I would like to acknowledge the support from my parents and brother. They always encouraged me for the higher study. Particularly, my brother always gave me comfort in any difficult situation. His suggestions and encouragement helped me to move ahead in all stages of my life both academically and socially.

Abstract

Globally road accident is considered to be an important issue, which can be reduced by proper vehicle speed monitoring system. More recently, the advancement in wireless sensor technology shows a great promise in designing Intelligent Transportation System (ITS) due to its flexibility and cost-effectiveness for deployment. The aim of this research is to develop a prototype vehicle speed monitoring system using accelerometer-based wireless sensor.

The basic concept of the system is based on the following methodology: developing an experimental system to generate random speed data, which can represent vehicle speed on the road and developing a software to monitor and manage the speed data wirelessly. A wireless sensor attached with a mechanical wheel measures the acceleration vibration of the system, which is equivalent to wheel speed and transmits the data wirelessly to a computer. A software (SpeedManage) has been developed using Java Socket programming codes which converts the vibration data to equivalent speed data and presents these in a Graphical User Interface (GUI). If the detected speed is greater than a set speed limit, the data will be automatically saved in a central database in the form of an electronic report for taking any further action.

The functionality of the system has been simulated in a laboratory environment by setting different speed limits for monitoring single or multiple vehicle speed scenarios through appropriate algorithm and code development. The graphical user interface (GUI) of the software continuously presents the vehicle speeds with time and the overspeeding conditions are indicated. The speed details are also continuously updated on the left hand side of the GUI. The system is also capable of generating an automatic electronic report for a simulated speeding vehicle with vehicle number, speed details, time etc. Therefore, based on the performance of prototype system, it can be concluded that sensor-based vehicle speed monitoring system has great potential for monitoring vehicle speed wirelessly. SpeedManage software should help to effectively, automatically and intelligently monitor vehicle speed.

Glossary

ATM:	Asynchronous Transfer Mode
BAN:	Body Area Network
CDMA:	Code Division Multiple Access
CAN:	Controller Area network
CAM:	Cooperative Awareness Message
CCD:	Charge-coupled Device
DNS:	Domain Name System
DTM:	Dual Transfer Mode
DRGS:	Dynamic Route Guidance System
ECP:	External Check Point
ETM:	Event Triggered Message
ECG:	Electrocardiogram
FDMA:	Frequency Division Multiple Access
FDDI:	Fiber Distributed Data Interface
FTP:	File Transfer Protocol
GPRS:	General Packet Radio Service
GPS:	Global Positioning System
GSM:	Global System for Mobile Communication
GSR:	Galvanic Skin Response
GUI:	Graphical User Interface
HTTP:	Hypertext Transmission Protocol

ITS:	Intelligent Traffic System
ISA:	Intelligent Speed Adaptation
ICP:	Internal Check Point
ITMS:	Intelligent Traffic Microwave Sensors
IP:	Internet Protocol
IPv4:	Internet Protocol Version 4
IPv6:	Internet Protocol Version 6
ISO:	Organization for Standardization
ICT:	Information and communication technology
ICEDB:	Intermittently Connected Data Base
LAN:	Local Area Network
LBA:	Link Based Algorithm
MMS:	Multimedia Messaging Service
MAC:	Multiple Access Control
MAC:	Medium Access Control
MRTG:	Multi-Router Traffic Grapher
NFS:	Network File System
NCP:	Network Control Programme
OSI:	Open System Interconnection
OBU:	On Board Unit
P2P:	Point-to-Point
P2M:	Point-to-Multipoint

PAN:	Personal Area Network
POP3:	Point-of-Presence
PEDAMACS:	Power Efficient and Delay Aware Medium Access Protocol
PSTN:	Publicly Switched Telephone Network
RFID:	Radio Frequency Identification Transponders
SMTP:	Simple Mail Transfer Protocol
SMS:	Short Message Service
SMB:	Server Message Block
SNMP:	Simple Network Management Protocol
SAR:	Synthetic Aperture Radar
SRTM:	Shuttle Radar Topography Mission
SVM:	Support Vector Machine
TCP:	Transmission Control Protocol
TDMA:	Time Division Multiple Access
TMC:	Traffic Message Centre
TMS:	Traffic Monitoring Server
USB:	Universal Serial Bus
UDP:	User Datagram Protocol
UART:	Universal Asynchronous Receiver/Transmitter
VANET:	Vehicle Ad-Hoc Network
V2I:	Vehicle to Infrastructure Communication System
VBA:	Vehicle-Based Algorithm

WSN:	Wireless Sensor Network
WAP:	Wireless Application Protocol
WWW:	World Wide Web
WiMAX:	Worldwide Interoperability for Microwave Access
Wi-Fi:	Wireless Fidelity
WAN:	Wide Area Network
WCDMA:	Wideband Code Division Multiple Access
WLAN:	Wireless Local Area Network
WLT:	Wireless Location Technology

Table of Content

<i>Declaration</i>	2
<i>Acknowledgement</i>	3
<i>Abstract</i>	4
<i>Glossary</i>	5
<i>Table of Content</i>	9
<i>List of Figures</i>	14
<i>List of Tables</i>	18
<i>Chapter 1 - Introduction</i>	19
1.1. INTRODUCTION.....	19
1.2. BACKGROUND OF THE RESEARCH.....	20
1.3. MOTIVATION OF THE RESEARCH.....	23
1.4. RESEARCH AIM AND OBJECTIVES.....	24
1.5. RESEARCH METHODOLOGY	24
1.6. STRUCTURE OF THE THESIS	25
<i>Chapter 2: Wireless Sensor and Networking</i>	27
2.1 SENSOR	27
2.2 SENSOR CLASSIFICATION	27
2.3 SENSING PRINCIPLES OF WIRELESS SENSOR.....	28
2.4 WIRELESS SENSOR NETWORK	29
2.5 HISTORY OF WIRELESS SENSOR NETWORK	29
2.6 ACTIVITY FRAMEWORK OF WIRELESS SENSOR NETWORK	31
2.7 GENERAL TERMINOLOGY USED IN WIRELESS SENSOR NETWORK.....	31
2.7.1 WSN Protocol.....	31
2.7.2 Sensor Node	33
2.7.3 Communication Layers	34
2.8 APPLICATIONS OF WIRELESS SENSORS	35
2.9 ADVANTAGES OF WIRELESS SESNOR.....	36
2.9.1 Safety.....	36
2.9.2 Convenience.....	36

2.9.3 Cost Reduction	37
2.9.4 Accuracy	37
<i>Chapter 3: Literature Survey</i>	38
3.1 INTRODUCTION	38
3.2 INTELLIGENT TRANSPORTATION SYSTEM	39
3.3 ROAD TRAFFIC MONITORING SYSTEMS	39
3.3.1 Camera-Based Monitoring Systems	40
3.3.2 Microwave RADAR-based Systems	41
3.3.3 Laser-Based Systems	42
3.3.4 Ultrasonic-Based Measurement.....	42
3.3.5 Intrusive Sensors and Road Tubes	43
3.3.6 Radio Frequency Identification Transponders	43
3.3.7 Satellite-Based Road Traffic Monitoring.....	44
3.4. WIRELESS SENSOR BASED TRAFFIC MONITORING	44
3.4.1 Techniques for Traffic Monitoring	45
3.4.2 Wireless Sensor inside a Vehicle.....	49
3.4.3 Wireless Sensor on the Road.....	50
3.4.4 Shimmer AWireless Sensor.....	53
3.4.5 Wireless Communication Protocols	55
3.4.6 SCOOT and SCATS Control Systems	56
3.5. ROAD TRAFFIC DATA MANAGEMENT.....	56
3.5.1. Data Collection and Management	56
3.5.2. Expert Systems	57
3.5.3. Data Management Software Development	58
3.6 SUMMARY.....	61
<i>Chapter 4: Experimental Set-Up Development.....</i>	63
4.1 INTRODUCTION	63
4.2 SYSTEM ARCHITECTURE	63
4.3 SENSOR SELECTION.....	64
4.3.1 Shimmer Sensor Device and Its Mechanism	65
4.3.2 Technical Specification of the Shimmer Sensor Device.....	67
4.4 INSTALLATION, CONNECTION AND DATA GENERATION.....	68

4.4.1 Sensor Installation and Configuration	68
4.4.2. Connection Establishment.....	69
4.4.3 Data Generation Test.....	70
4.4.4 Calibration of Shimmer Sensor Device.....	71
4.5 SET UP DEVELOPMENT WITH A TOY CAR.....	74
 4.5.1 Toy Car Set-up Concept.....	74
 4.5.2 Tests with the Toy Car	75
4.6 SET-UP DEVELOPMENT WITH A MECHANICAL WHEEL.....	76
 4.6.1 Mechanical Wheel Set-up Concept.....	76
 4.6.2 Connection Establishment.....	77
 4.6.3 Data Transfer and Storing	78
4.7 WHEEL SPEED MEASUREMENT WITH THE SENSOR.....	79
4.8. EXPERIMENTATION FOR SPEED DATA COLLECTION.....	83
 4.8.1 Experimental Condition	83
 4.8.2 Experimental Procedure.....	84

<i>Chapter 5: Software Development for Intelligent Traffic Speed Monitoring System .</i>	86
5.1. INTRODUCTION	86
5.2 SYSTEM DEVELOPMENT CONCEPT	86
5.3 COMMUNICATION WITH SOCKET PROGRAMMING.....	89
5.4 SOCKET SERVER ALGORITHM FOR SINGLE MOVING VEHICLE	91
 5.4.1 Setting up Data Channel.....	91
 5.4.2 Creating String and Object, and Connecting Client.....	92
 5.4.3 Creating GUI for Displaying Server Data	93
 5.4.4 Reading Data from the .csv File.....	94
 5.4.5 Processing of Vibration Acceleration Data.....	94
 5.4.6 Displaying Data on Server's DOS Window	95
 5.4.7 Displaying Data on Server's GUI	95
 5.4.8 Closing Socket and File Resources	96
5.5 SOCKET CLIENT ALGORITHM FOR SINGLE MOVING VEHICLE	96
 5.5.1 Connection to Server	97
 5.5.2 Client Initialisation	98
 5.5.3 Speed Limit Settings	98
 5.5.4 Establishing Socket Connection.....	98

5.5.5 Creating ‘SpeedManage’ GUI	99
5.5.6 Displaying Data in DOS Window and ‘SpeedManage’ GUI.....	100
5.5.7 Updating the Processed Data on the Client	101
5.5.8 Generating Reports on Client Computer.....	102
5.5.9 Closing the Socket Connection and File Resources	103
5.6 SOCKET SERVER ALGORITHM FOR MULTIPLE MOVING VEHICLES ..	103
 5.6.1 Setting up Data Channel.....	105
 5.6.2 Creation of String and Object, and Connecting Client	105
 5.6.3 Creating GUI for Displaying Server Data	106
 5.6.4 Reading Data From the .csv File.....	106
 5.6.5 Processing of Vibration Acceleration Data.....	107
 5.6.6 Displaying Data on Server’s DOS and GUI windows.....	108
 5.6.7 Checking the Total Number of Data Files	109
 5.6.8 Closing Socket and File Resources	109
5.7 SOCKET CLIENT ALGORITHM FOR MULTIPLE MOVING VEHICLES ...	109
 5.7.1 Connection to Server	110
 5.7.2 Client Initialisation	111
 5.7.3 Speed Limit Settings	111
 5.7.4 Establishing Socket Connection.....	112
 5.7.5 Creating ‘SpeedManage’ GUI	112
 5.7.6 Displaying Data in DOS Window and ‘SpeedManage’ GUI.....	113
 5.7.7 Updating the Processed Data on the Client	114
 5.7.8 Generating Reports on Client Computer.....	115
 5.7.9 Closing the Socket Connection and File Resources	116
<i>Chapter 6: Results and Discussions</i>	117
6.1 INTRODUCTION	117
6.2 SCENARIO DESIGN	117
6.3 SIMULATING A SINGLE VEHICLE SCENARIO	118
 6.3.1 Single Vehicle Running at Lower Speed Limit.....	118
 6.3.2 Vehicle Running at Higher Speed Limit	125
6.4 SIMULATING MULTIPLE VEHICLES SCENARIO.....	128
 6.4.1 Multiple Vehicles Running at Slower Speed Limit	128
 6.4.2 Multiple Vehicles Running at Faster Speed Limit	134

6.5 SUMMARY.....	137
<i>Chapter 7. Conclusions and Suggestion for Future Work.....</i>	138
7.1. CONCLUSIONS	138
7.2. THESIS CONTRIBUTION.....	138
7.3. SUGGESTIONS FOR FUTURE WORK.....	139
(i) LIST OF REFERENCES	141
(ii) APPENDIX.....	151
Appendix - A: Calibration Data of the Shimmer Sensor.....	151
Appendix - B: Determination of Resultant Acceleration.....	154
Appendix - C: Java TM Socket Programming Codes Developed for Single Vehicle	158
Appendix - D: Java TM Socket Programming Codes Developed for Multiple Vehicles...	170

List of Figures

	Page Number
Figure 1.1: Vehicle crash on the road	19
Figure 1.2: A general framework of vehicle speed monitoring	20
Figure 1.3: Use of Intelligent Transport Systems on the M42 in England	21
Figure 1.4: Intelligent transportation system	22
Figure 1.5: Overall research framework	25
Figure 2.1: A Honeywell HMC1022 AMR magnetic sensor	27
Figure 2.2: Steps in sensing process of a sensor	29
Figure 2.3: Wireless sensor network	30
Figure 2.4: Wireless sensor network activities	31
Figure 2.5: Architecture of a sensor node	33
Figure 2.6: A TelosB sensor node	33
Figure 2.7: Layers in the Open System Interconnection (OSI) model	34
Figure 2.8: Applications of wireless sensors	36
Figure 3.1: Literature review framework	39
Figure 3.2: Camera-based traffic monitoring system deployed on a roadway	41
Figure 3.3: Ultrasonic sensor-based traffic monitoring system mounted on an overhead structure	43
Figure 3.4: TerraSAR-X traffic monitoring on the A4 motorway west of Dresden, Germany	44
Figure 3.5: The accelerometer-based traffic monitoring system installed on west Interstate 80 at Pinole, CA, USA	46
Figure 3.6: A Shimmer™ development platform used for structural health monitoring	54
Figure 3.7: The Shimmer™ platform architecture	54
Figure 3.8: The system architecture of CarTel	59
Figure 3.9: Traffic monitoring data management technique	59
Figure 3.10: CarTel portal software architecture	61
Figure 4.1: General system architecture for monitoring vehicle speed	63
Figure 4.2: Prototype system architecture for monitoring vehicle speed	64
Figure 4.3: Shimmer wireless sensor device	65
Figure 4.4: Internal infrastructure of shimmer sensor device	66

	Page Number
Figure 4.5: Shimmer docking station with controls	67
Figure 4.6: Shimmer sensor device mounted on the Shimmer dock	67
Figure 4.7: Configuring the shimmer sensor	69
Figure 4.8: Shimmer sensor connected via Bluetooth with or without the docking station	69
Figure 4.9: ShimmerConnect front panel ready for data collection	70
Figure 4.10: Graph generation from vibration acceleration data generated by the sensor via Bluetooth	70
Figure 4.11: Orientations of Shimmer sensor during calibration	72
Figure 4.12: Shimmer sensor attached with a remote controlled toy car	75
Figure 4.13: Shimmer sensor is generating vibration acceleration data while moving with car	76
Figure 4.14: Schematic diagram of a mechanical wheel based set-up	77
Figure 4.15: Photograph of mechanical wheel based set-up	77
Figure 4.16: Connection establishment between sensor and server computer.	78
Figure 4.17: Real-time vibration acceleration plot for the wheel rotating at 20 rpm	78
Figure 4.18: Shimmer sensor calibration process	80
Figure 4.19: The resultant vibration acceleration vs. different rotational speeds (rpm) of the wheel	81
Figure 4.20: Vibration acceleration graphs in X, Y and Z directions at different rotational speeds of the mechanical wheel	82
Figure 4.21: Experimental set-up for speed monitoring using wireless sensor	84
Figure 5.1: System concept for developing intelligent traffic monitoring software	87
Figure 5.2: A general algorithm for developing traffic monitoring system	88
Figure 5.3: Communication between client and server	89
Figure 5.4: Socket in server and client	90
Figure 5.5: Steps of transferring data from server to client using socket programming	90
Figure 5.6: Socket Server algorithm for a single moving vehicle	92
Figure 5.7: Socket client algorithm for a single moving vehicle	97
Figure 5.8: Socket server algorithm for multiple moving vehicles	104

	Page Number
Figure 5.9: Socket client algorithm for multiple moving vehicles	110
Figure 6.1: Road traffic scenarios considered for demonstration	117
Figure 6.2: Connection established between a server computer and a client computer	118
Figure 6.3: Selection of speed limit in the client computer for the scenario of single vehicle running at slower speed limit	119
Figure 6.4: Set speed limit for a single vehicle running at slower speed limit	119
Figure 6.5: Display of vehicle speed data in a DOS window of the server computer for single vehicle running at slower speed limit	120
Figure 6.6: Display of latest speed data with vehicle details in a server dialog box for a single vehicle running at slower speed limit	121
Figure 6.7: Display of vehicle speed data in a DOS window of the client computer for a single vehicle running at slower speed limit	122
Figure 6.8: Display of vehicle speed plot on a GUI in the client computer for a single vehicle running at slower speed limit	123
Figure 6.9: Example of speeding report in pdf format for a single vehicle running at slower speed limit	125
Figure 6.10: Selection of speed limit in the client computer for the scenario of single vehicle running at faster speed limit	126
Figure 6.11: Display of latest vehicle speed data with vehicle details in a server dialog box for a single vehicle running at faster speed limit	126
Figure 6.12: Display of a vehicle speed plot on a GUI in the client computer for a single vehicle running at faster speed limit	127
Figure 6.13: Selection of speed limit in the client computer for the scenario of multiple vehicles running at slower speed limit	128
Figure 6.14: Display of vehicle speed data in a DOS window of the server computer for multiple vehicles running at slower speed limit	129
Figure 6.15: Display of latest vehicle details in a server dialog box for multiple vehicles running inside at slower speed limit	130
Figure 6.16: Display of vehicle speed data in a DOS window of the client computer for multiple vehicles running at slower speed limit	131
Figure 6.17: Display of a vehicle speed plot on a GUI in the client computer for multiple vehicles running at slower speed limit	132

Figure 6.18: Example of speeding report in pdf format for multiple vehicles running at slower speed limit	133
Figure 6.19: Selection of speed limit in the client computer for the scenario of multiple vehicles running at faster speed limit	134
Figure 6.20: Display of latest vehicle speed data with vehicle details in a server dialog box for multiple vehicles running at faster speed limit	135
Figure 6.21: Display of a vehicle speed plot on a GUI in the client computer for multiple vehicles running at faster speed limit	136
Figure 7.1: Illustration of vehicle speed measurement by vibration based wireless sensors	139
Figure B.1: The rotating rigid body considering the Shimmer sensor as an integral part	154
Figure B.2: The angular velocity of a rigid body ‘A’	155
Figure B.3: The motion of the reference frame xyz with respect to the inertial reference frame XYZ	156
Figure B.4: The resultant acceleration with respect to the reference frame	156
Figure B.5: Resultant acceleration \mathbf{a}_{rel}	157

List of Tables

	Page Number
Table 2.1: Sensor classification	28
Table 4.1: Technical specifications of the Shimmer sensor device	68
Table 4.2: Technical specifications of the Accelerometer	68
Table 4.3: Vibration accelerometer data saved in Tabular format in Excel	71
Table 4.4: Summary of sensor calibration data	74
Table 4.5: Vibration acceleration data obtained from Shimmer sensor at 20 rpm wheel speed	79
Table 4.6: The average resultant vibration acceleration of the wheel at different rpm	80
Table 4.7: Comparison between actual and measured wheel speeds	82
Table 4.8: Experimental conditions for data collection	84
Table 6.1: Speeding report in tabular format for a single vehicle running at slower speed limit	124
Table 6.2: Speeding report in tabular format for multiple vehicles running at slower speed limit	133
Table 6.3: Report for over speed vehicles details in .csv format running at faster speed limit	135
Table A.1: X-axis Shimmer sensor data along the axis of the positive gravitational acceleration ('+g') ['X plus up.csv' file]	151
Table A.2: X-axis Shimmer sensor data along the axis of the negative gravitational acceleration ('-g') ['X plus down.csv' file]	152

Chapter 1 - Introduction

1.1. INTRODUCTION

Human errors, in particular driver errors, are the causes for the most of the road accidents. It is reported that over 80% of all major crashes on Irish roads are caused from inconsiderate driving [1]. The main causes of accidents are vehicle over-speeding, driving after consuming alcohol, distraction during driving, non-adherence to traffic signals, non-wearing of seat-belts and safety gears, tailgating, poor lane discipline etc. However, in most of the countries speeding or excessive vehicle speed on the road is considered to be the single biggest factor for road accidents contributing to fatal injuries or even death and financial costs to society (Figure 1.1). In Ireland, over 40% of fatal collisions are caused by excessive or inappropriate speed [1]. Increasing traffic congestion and street accident due to over speeding of vehicle are critical issues to solve for the smart cities. Thus, vehicle speed monitoring and controlling is one of the important issues in order to maintain a safe road.



Figure 1.1: Vehicle crash on the road [2]

Different systems (Road-based and Vehicle-based) are used for monitoring vehicle speed. However, managing speed data in an efficient and intelligent way is an on-going issue in transportation system. This research proposes a wireless sensor based framework for vehicle speed monitoring and controlling (Figure 1.2). In general, a

sensor (accelerometer-based) and other auxiliary devices are used to detect a vehicle moving on the road. The vehicle speed is calculated by the sensor and cross checked with a speed limit set in a particular road. If the detected speed is greater than the speed limit, then the device sends the data to a central server wirelessly and generates a report indicating with time road and vehicle details so that the report can be sent to the offender for subsequent actions.

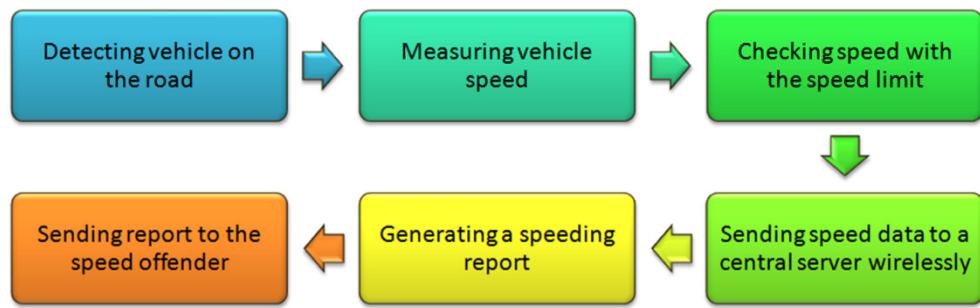


Figure 1.2: A general framework of vehicle speed monitoring

1.2. BACKGROUND OF THE RESEARCH

Intelligent Transportation System (ITS) is an advanced technology that can be used to minimise road accidents. Freeway and arterial management, emergency management, parking management, real time traffic network conditions and vehicle speed monitoring are controlled by the deployment of ITS [3]. Figure 1.3 shows an example of ITS system used in the UK highway. However, the road accidents are still forecasted as the headline news in the public media.

A number of sensor-based systems have been developed to use on the road and inside the vehicle for vehicle speed detection, control and monitoring purposes. Such as speed camera, induction loop system, ultrasonic sensors, micro-loop probes, pneumatic road tubes and GPS in order to count number of vehicles, monitor vehicle speed, control traffic signal and to divert traffic intelligently. However, the existing systems suffer from a number of problems including high equipment cost, complex installation, high maintenance cost and dependency on weather condition.



Figure 1.3: Use of Intelligent Transport Systems on the M42 in England [4]

Furthermore, the government and various funding agencies are trying to control the road traffic system by spending a huge amount of money. The road transport authority and researchers are working diversely to control the vehicle speed both in urban and rural areas.

The urban mobility report shows that the benefit of ITS is \$5.6 billion savings for the 85 urban areas in USA by implementing the ITS technologies. However, if the systems are used on all major roads then the benefit would be \$10.2 billion in 2003 [3]. The goal of ITS is to solve problems in traffic system such as vehicles, road and transport infrastructure. In addition, weather and road conditions, traffic disorders, emergency services, route planning, traffic safety and travel information are offered by the ITS [5].

There are two communication patterns in ITS based on network circumstances: Cooperative awareness message (CAM) and Event triggered message (ETM). In the CAM system, vehicles transmit very short message containing the vehicle's position, velocity etc. On the other hand, in some cases extra messages are transmitted due to safety of vehicle and driver are called ETM. Furthermore, there are few classes and applications in ITS, such as Co-operative road safety, co-operative traffic efficiency, co-operative local services and global internet services [6]. Dynamic route guidance system (DRGS) is another part of ITS that provides real time traffic information for route guidance. It contains traffic message centre (TMC) and radio receiver. Figure 1.4 presents a model for Intelligent Transportation System.

Intelligent Transportation System

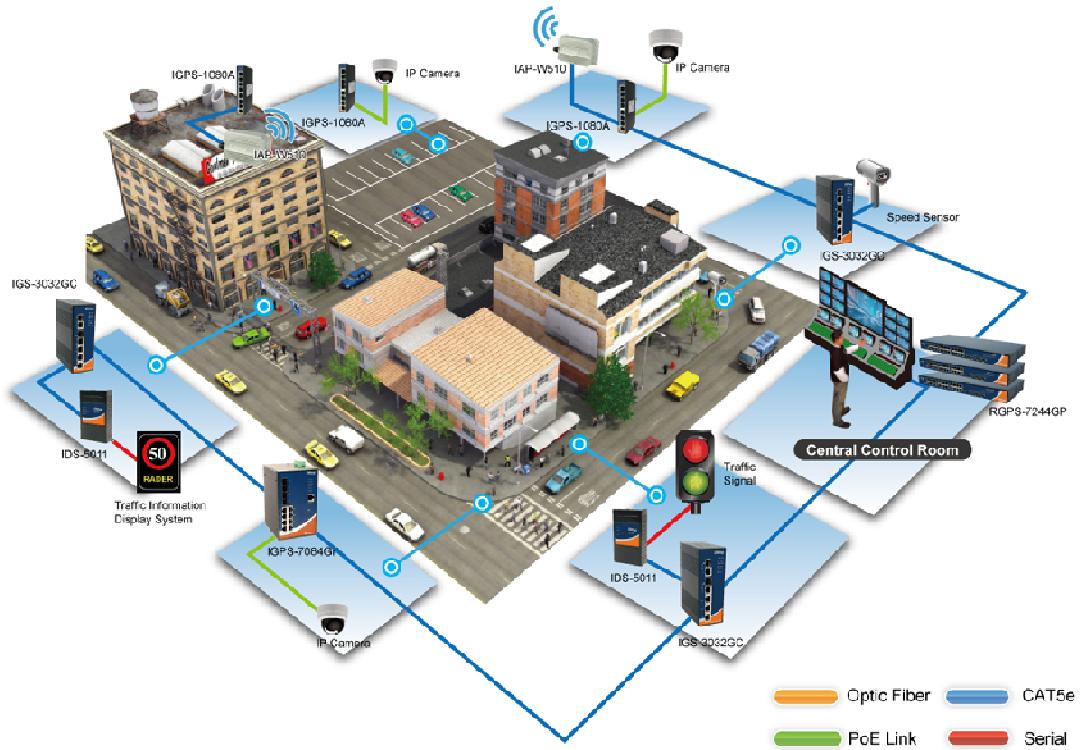


Figure 1.4: Intelligent transportation system [7]

Intelligent speed adaptation (ISA) is one of the important part of ITS for collecting vehicle speed information and, controlling and monitoring vehicle speed. The ISA system is used to warn driver to prevent exceeding the speed limit as well as increase the consciousness in driver's mind. ISA can reduce crash risk, fuel consumption and emissions to atmosphere [8]. Recently, the geospatial position of a vehicle are identified comparing with current position and speed by digital road map. Different types of ISA system are being used for vehicle such as informative, advisory, recording and intervening. In addition, “pay as you go” system has made a great advancement in ISA where the drivers are given discount by the insurance provider based on driving behaviour. ISA runs by set-up and navigation modes. The drivers and vehicle details are measured by two nodes while the vehicle violates speed limit [9]. However, this system faces difficulty in functioning properly in different weather conditions.

More recently it has been suggested that Wireless sensor network (WSN) has wider scope in ITS for monitoring and controlling traffic wirelessly. It is used in public infrastructure such as deployment of sensors on the intersections and curved roads. In the event of speeding sensor detects the vehicle details and sends the data to an analysis

station and warning messages are sent to the vehicle. On the other hand, vehicle Ad-Hoc network (VANET) system communicates with nearby vehicles through wireless sensor network [10]. Additionally, real time traffic information is provided by sensor nodes, which are small and cheap. The dynamic data driven application system in WSN provides high accuracy of accident and congestion propagation in the road network system. On the other hand, the acoustic and magnetic sensors can be used in WSN to detect vehicle. WSN can also be considered as distributed road information provider due to its ability to gather real time road traffic information and the traffic data characteristics among the neighbouring roads.

1.3. MOTIVATION OF THE RESEARCH

Road accident is an increasing phenomenon due to inconsiderate driving on the road. Unmonitored road and traffic is one of the issues seeking urgent attention of the authorities. Some roads have no speed camera and appropriate monitoring system for identifying the over-speed vehicles. Therefore, vehicles passing through these roads crossing the maximum speed limits can cause fatal accidents. Although many systems have been developed in order to minimise the accidents, still further work is necessary to improve the situation. The application of WSN in road traffic monitoring systems can reduce road traffic accidents through wireless sensing and communication. Although there are few devices working as wireless communication systems, the operation of these systems is not in complete wireless mode. Furthermore, the existing systems are not fully automated in terms of speed detection and monitoring of vehicle detection, speed measurement, monitoring and data communication.

Therefore, in order to reduce accidents through monitoring vehicle speeds, I propose a new system (SpeedManage) to communicate the vehicle speed data wirelessly using TCP/IP communication from a sensor placed on the vehicle to a remotely placed server computer. Java®-based socket programming technique can be used to communicate between the server and client computers. The system can collect speeding data in a server computer and transfers to the client computer. An automated detailed report is generated and saved in the client computer for a speeding vehicle. Therefore, the whole system can work in an intelligent an automated way.

1.4. RESEARCH AIM AND OBJECTIVES

The aim of the research is to develop a model system based on wireless sensor for detecting vehicle, measuring and monitoring speed and managing the relevant speed data.

In order to reach the aim, the following objectives have been established:

1. To develop an experimental set-up incorporating a wireless sensor device, a variable speed mechanical wheel assembly and a laptop in order to generate random speed data.
2. To design and develop a wireless vehicle speed monitoring system capable of detecting a wide range of simulated vehicle speeds.
3. To simulate real-life traffic monitoring scenarios using the experimental set-up and monitoring system in a laboratory environment and to generate reports for speeding vehicle automatically in the system.

1.5. RESEARCH METHODOLOGY

Figure 1.5 displays three aspects of the work to be undertaken in this research. Current literature related to traffic monitoring system, wireless sensor network and its application in ITS has been studied from different research databases to get the necessary knowledge in the research area, current trend and research gap. An experimental set-up has been designed and commissioned to generate continuous speed data, which can be measured, recorded and transferred to a computer wirelessly. The main instruments used in the experimental set-up include a commercial wireless sensor (Shimmer), a mechanical wheel, tachometer coupled with a motor, client computer and a server computer. The wireless sensor indirectly measures the wheel speed through sensing the vibration acceleration of the wheel and communicates the speed data to the server computer wirelessly. A ‘SpeedManage’ software has been developed using Java Socket programming language for detecting, calculating and monitoring speeds. The functioning of the monitoring system has been simulated for different range of speeds. Furthermore, the system has been developed to automatically generate report comprising vehicle details, speed and time when the speed is found to be more than a pre-set speed limit.

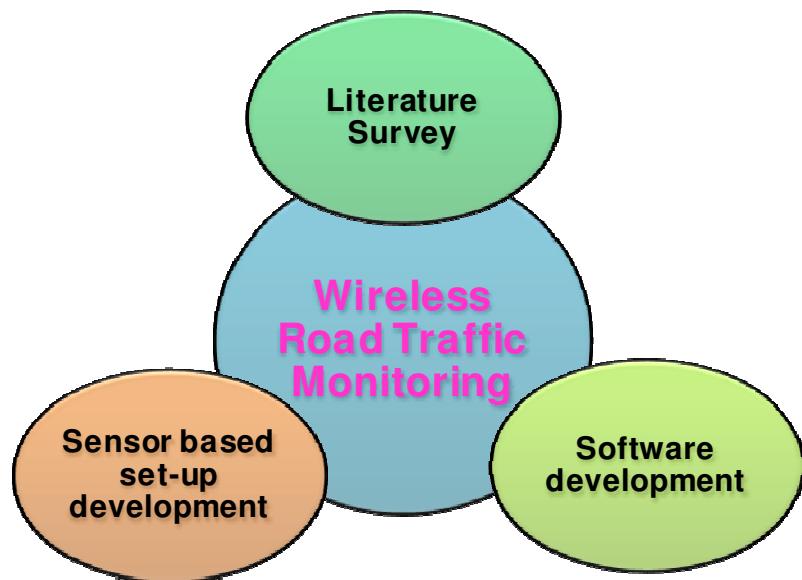


Figure 1.5: Overall research framework

1.6. STRUCTURE OF THE THESIS

The whole project work has been presented in this thesis comprising of seven chapters.

Chapter 1-Introduction: This chapter introduces the present work and sets out the aims and objectives of the research.

Chapter 2-Wireless Sensor and Networking: This chapter introduces basic wireless sensor and its components, wireless sensor network (WSN), its advantages and brief description of wireless protocol.

Chapter 3-Literature Survey: This chapter describes the overview of recent vehicle monitoring systems using wireless sensor, contribution of programming based approaches, road traffic data management techniques and other approaches. The chapter has also finds the gap in the current research related to the application of WSN in ITS.

Chapter 4- Experimental Set-Up Development: The chapter describes the model experimental set-up, details of wireless sensor and its calibration procedure and the data generation techniques from the shimmer sensor.

Chapter 5- Software Development for Intelligent Traffic Speed Monitoring System: This chapter presents the details of the software development with example codes.

Chapter 6-Results and Discussions: A graphical user interface maintains connection between server computer and client computer for displaying the vehicle speeds scenarios. The chapter also shows the output of the vehicle speed measurement in different scenarios and also calculates the speed and detects the over speeds as report format in the client computer and saves.

Chapter 7-Conclusions and Suggestion for Future Work: The chapter describes the outcomes of the research and indicates some future works to progress further by adding new features and technologies.

Appendix: Some experimental data and programming code have been added in this section.

Chapter 2: Wireless Sensor and Networking

2.1 SENSOR

Sensor is hardware device that has the ability to sense, monitor and measure physical condition [11]. A sensor can perform on-board processing and communication. It also has the ability to store information. There are two types of sensors: passive sensor and active sensor [12]. Passive sensors are self-powered where the measurement is performed without notation of direction. Temperature sensor is an example of passive sensor. Active sensor is a group of sensors that can generate shockwaves by explosions. Radar and seismic sensor are the examples of active sensors. Figure 2.1 displays a magnetic sensor.

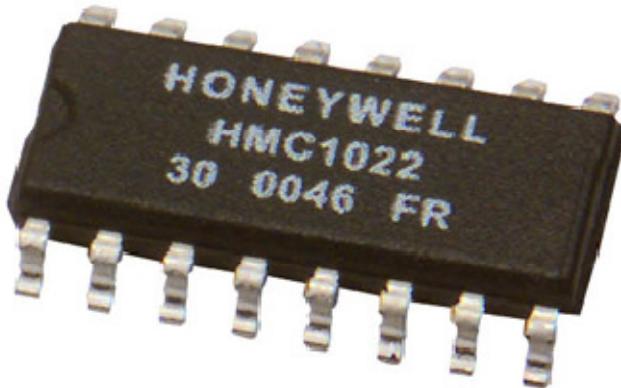


Figure 2.1: A Honeywell HMC1022 AMR magnetic sensor [13]

2.2 SENSOR CLASSIFICATION

There are different types of sensors such as acoustic, magnetic sensors, photonic, vibration, humidity, temperature, traffic surveillance, alarm and motion sensors. Sensors can be classified according to the sensing areas as shown in Table 2.1.

Table 2.1: Sensor classification [14]

Sensing area	Sensor examples
Mechanical	Strain gauges, tactile sensors, capacitive diaphragms,
Motion, vibration	Accelerometers, gyroscopes, photo sensors
Position	GPS, ultrasound-based sensors, infrared-based sensors, inclinometers
Optical	Photodiodes, phototransistors, infrared sensors, CCD sensors
Acoustic	Piezoelectric resonators, microphones
Electromagnetic	Hall-effect sensors, magnetometers
Humidity	Capacitive and resistive sensors, hygrometers, MEMS-based humidity sensors
Temperature	Thermistors, thermocouples
Pressure	Pressure gauges, barometers, ionization gauges

2.3 SENSING PRINCIPLES OF WIRELESS SENSOR

A sensor is an object, which can perform sensing task. Human body can capture optical information, sounds and smell from the environment using their eyes, ears and nose. They do not need to touch the monitored object for gathering information. These are the examples of remote sensors. From the technical point of view, a sensor can translate physical parameters into the form of signals for analysing and measuring. Figure 2.2 shows the steps of data acquisition and actuation process by a sensor. A sensor acts as transducer [14], which can convert the physical phenomena into electrical or some other form of energy. Furthermore, the resulting signals are processed during signal conditioning stage. The analog-to-digital converter transforms the analog signal into a digital signal. Finally, an actuator can control the flow of signal directly in the physical world by further processing, converting and conditioning.

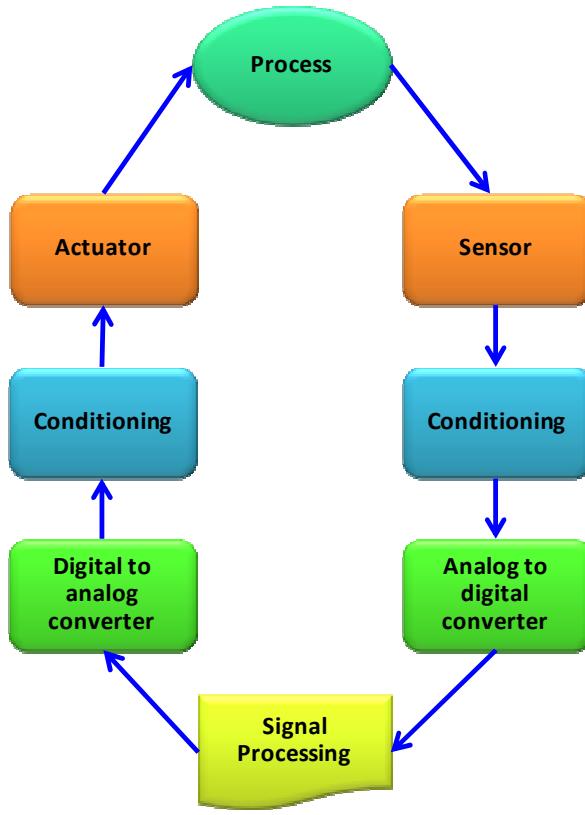


Figure 2.2: Steps in sensing process of a sensor (adapted from [14])

2.4 WIRELESS SENSOR NETWORK

Wireless sensor network [14] contains processing station, central base station and sensor nodes. Sensors nodes are employed to collect data, the processing centre can process the data and transfer to the central data base station. Figure 2.3 displays a general framework of wireless sensor network.

2.5 HISTORY OF WIRELESS SENSOR NETWORK

Maxwell and Hertz [15] first started wireless communication based on the idea of transmission and electromagnetic waves. Tesla tested by transmitting the information using the wave after very short time of Maxwell and Hertz. In 1889, Marconi made wireless communication from a boat to Isle of Wight in English Channel and in 1909, the use of radio was popular. The Defence Advanced Research Projects Agency (DARPA) organised Distributed Sensor Nets Workshop basis on networking technologies and distributed algorithms in 1978. In the early 1980s, DARPA introduced sensor networks program considering sensor information technology (SensIT).

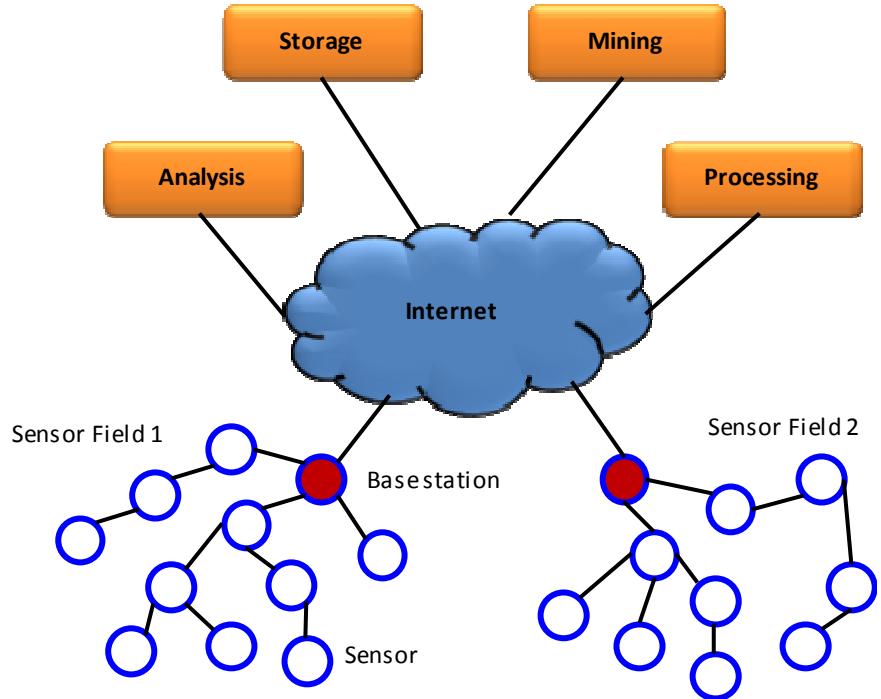


Figure 2.3: Wireless sensor network (adapted from [14])

Furthermore, the concept of wireless integrated network sensors had been developed in collaboration with the Rockwell science centre, the University of California at Los Angeles. Afterwards in 1996, low power wireless integrated micro sensor was developed based on a CMOS chip, interface circuits, integrating multiple sensors, digital signal processing circuits, wireless radio and microcontroller on to a chip. In 1999, motes had been developed as a small sensor by the smart dust project at the University of California at Berkeley. This system can be integrated into tiny devices. Furthermore, the Berkeley wireless research centre focused on development of low power sensor devices by the Pico radio project those can power themselves from solar or vibrational energy. MIT had developed a micro-adaptive multi domain power aware sensors project that can scale dynamic voltage to reduce power requirements at the software level using the techniques to restructure data processing algorithm. Over the last decade, a number of commercial efforts have further developed the sensor system. Many companies such as Crossbow, Sensoria, Worldsens, Dust networks and Ember Corporation provided the opportunity to purchase ready-made sensor devices for programming, maintenance and sensor data visualisation [14].

2.6 ACTIVITY FRAMEWORK OF WIRELESS SENSOR NETWORK

A wireless sensor network consists of distributed sensors to monitor both physical and environmental conditions of different types, such as temperature, sound, pressure, and vibration. Figure 2.4 describes a data acquisition system in the first part and data distribution system in the second part. In data acquisition part, the base station collects data wirelessly from different environment through wireless data collection network. Later, the base station controller sends data to the management centre for storing and analysing. On the other hand, the wireless data are distributed using wireless local area network (WLAN) through Wi-Fi, Bluetooth, and cellular network.

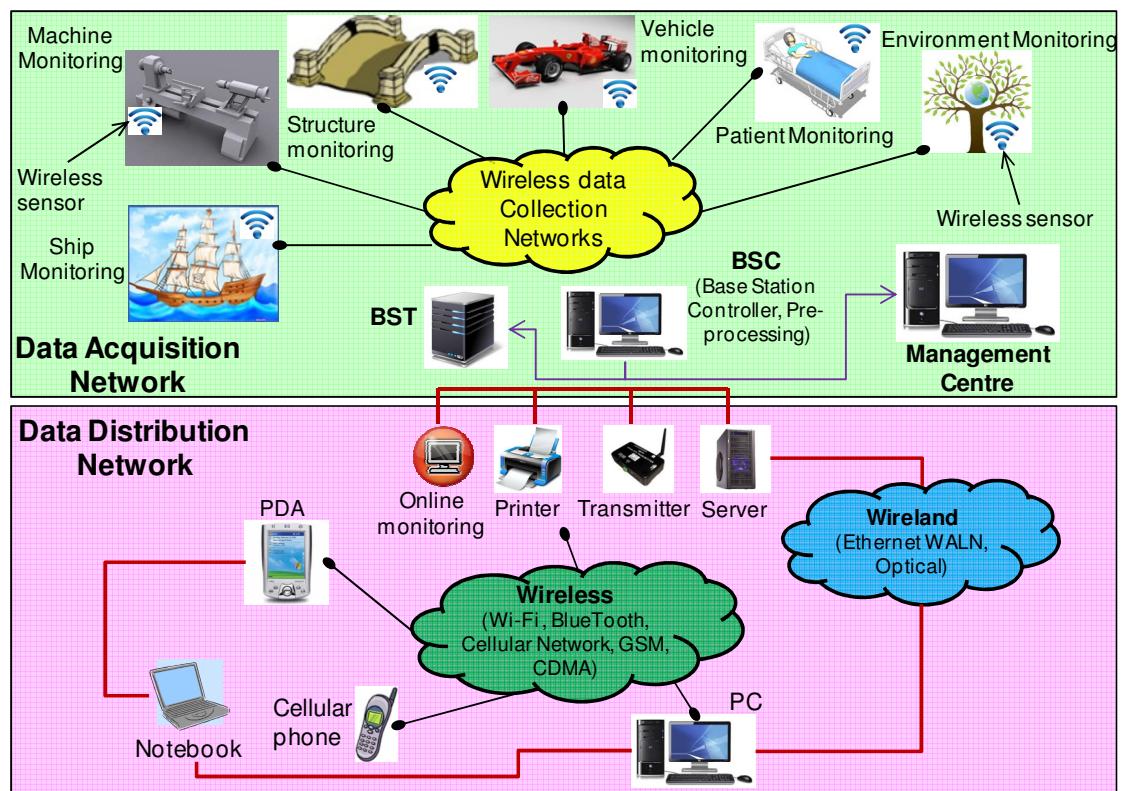


Figure 2.4: Wireless sensor network activities [16]

2.7 GENERAL TERMINOLOGY USED IN WIRELESS SENSOR NETWORK

For basic understanding, a few general terms related to Wireless sensor network (WSN) has been presented in the following sections.

2.7.1 WSN Protocol

The IEEE 802.15.4 standard has been developed to operate 868 MHz, 915 MHz, and 2.45 GHz frequency bands with supporting data rate of 20, 40, and 250kbps [17]. This

standard contains two types of topology nodes: star, which is similar to Bluetooth and peer-to-peer, which allows the communication freely within the devices. All the communications are maintained through Personal Area Network (PAN) co-ordinator. Residential, industrial, environment monitoring control and automation applications are conducted by wireless sensor network using the IEEE 802.15.4. It is used for low cost of deployment, low power consumption, and low complexity. Mostly the standard is used for low rate wireless personal area networks and short range communication for maximising battery life. The formation of star and peer-to-peer topology is allowed by the standard to communicate between network devices. It supports the protocols for physical layer and data link layer. The MAC layer can validate frames, delivery, network interface, network synchronisation, secure service, and device association. Devices are interacted with each other through wireless network shows different layers of IEEE 802.15.4.

ISA100.11a [18] is a wireless network technology standard that has been developed by the international society of automation and officially released in September, 2009. ISA100.11a is used for simple, flexible and scalable security functionality with star and mesh network topologies. This standard is developed for low data rate wireless monitoring and processing applications. This standard uses 2.4 GHz radio. It maintains contact with other different wireless devices in a network system. Robustness, infrastructure, scalability, interoperability and low power energy consumption are defined by this standard.

ZigBee [14] was initially used on a low cost communication technology for low data rates and low power consumption. Afterwards, ZigBee is the commercial name for the IEEE 802.15.4 technology. ZigBee is a simple wireless communication technology used in embedded applications. A mesh network can be consisted by ZigBee used in many devices. There are three types of ZigBee devices such as ZigBee coordinator, ZigBee router, and ZigBee end device. A very little power on cell battery is used by ZigBee device. Moreover, bridge network, store information and initiate network information are also conducted by ZigBee coordinator. Sensors, actuators and controller are the main parts of ZigBee end device those can create data communication within a router.

2.7.2 Sensor Node

The wireless sensor node [14] is an important part of wireless sensor network. There are multiple applications are conducted using wireless sensor node. The wireless sensor node is consisted by controller, transceiver, external memory and other sensors interfaced with analog to digital converters. The sensor nodes can sense object and communicate within a certain distance for transferring and processing data by its inbuilt transceiver. The basic architecture of a sensor is schematically presented in Figure 2.5. There are different types of sensor node used in different applications such as Mica, Mica2, Micaz, BTNode2, BTNode3, iMote, ZebraNet, TelosB etc. Figure 2.6 shows a commercial TelosB sensor node.

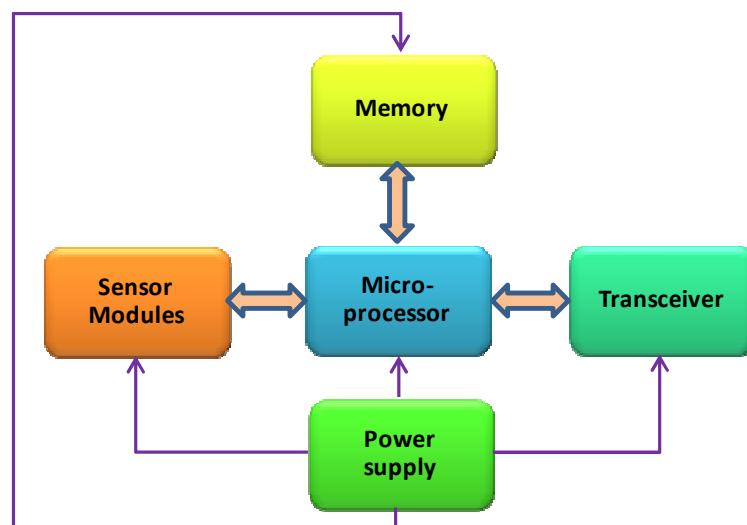


Figure 2.5: Architecture of a sensor node



Figure 2.6: A TelosB sensor node [13]

2.7.3 Communication Layers

Different communications layers are used in wireless sensor network with different standards and protocols for establishing connections within devices in a wireless sensor network [19, 20]. The open system interconnection (OSI) reference model has been established by the International Organization for Standardization (ISO) with different communication layers (Figure 2.7).

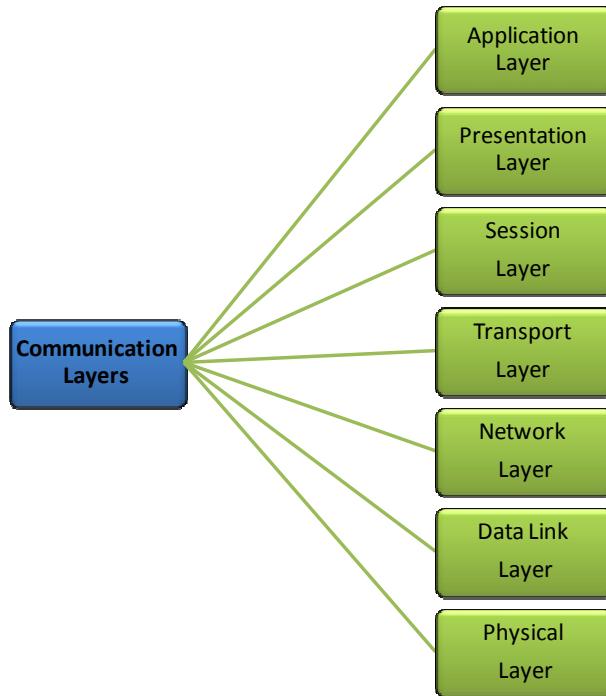


Figure 2.7: Layers in the Open System Interconnection (OSI) model [19, 20]

Application layer is used for application services. The user authentication, resource checking etc. are performed by different protocol such as HTTP, FTP, SMTP, SNMP and POP3. The distributed information services are also provided in this layer. The presentation layer mainly manages the data translation. All the incoming data are transferred and represented the outgoing data in different formats. The standards are employed to handle communication between sender and receiver through computers in the session layer. The communication is maintained by terminating sessions. TCP and UDP protocols are used in transport layer for completing data transfer successfully between the end points. Data flow control and end-to-end error recovery are managed by this layer. Furthermore, the successful packet transmissions are also confirmed by this layer. The standard of network layer manages the network connections through routing and terminating within nodes in the sensor network. IPv4, IPv6 etc. are used in

this layer for reliable connections of devices in a network system. The data link layer ensures the reliability of physical connection of data. The standard of data link layer can share and access the data through medium access control (MAC). Ethernet, Wi-Fi (IEEE 802.11) and Bluetooth are used as standards and protocol in the layer. Ethernet, Wi-Fi, Bluetooth, WiMAX etc. are used in physical layer. Physical layer controls data transmission stream within a wireless sensor network. The physical layer maintains mechanical, electrical and other procedural characteristics. The voltage signal durations, frequencies and data transmissions are controlled at this stage.

2.8 APPLICATIONS OF WIRELESS SENSORS

Wireless sensors have found many applications in several areas as shown in Figure 2.8 [11, 21-26]. For example, active volcanoes can be monitored using wireless sensors where collected data are sent to a central data base station by a radio system. The observation of the useful characters of the animal species in biological research is another application of wireless sensors. Wireless sensors are widely used to monitor behaviour of storm [11]. The application for the wireless sensors in healthcare includes pulse oxygen saturation sensors, blood pressure sensors, blood flow sensors, blood oxygen level sensor for measuring cardiovascular exertion, body and skin temperature sensors, electrocardiogram, respiration sensors, electroencephalogram and electromyogram for muscle activities [11]. Military surveillance is another application of the wireless sensors [24]. Environmental monitoring, flood and water level monitoring, seabird habitat monitoring, agricultural monitoring using unmanned aerial vehicles and traffic monitoring are also considered to be the applications of wireless sensors.

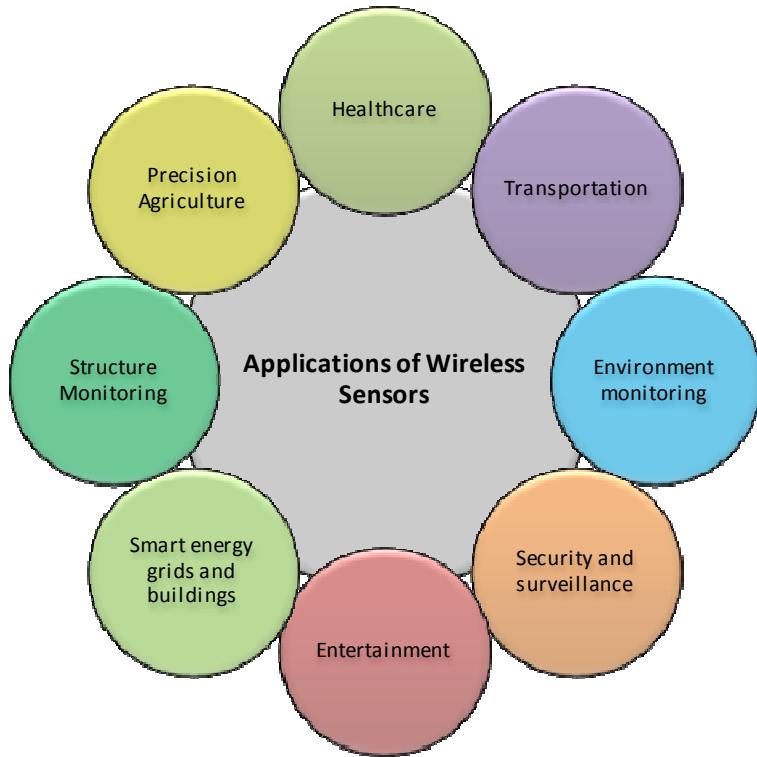


Figure 2.8: Applications of wireless sensors

2.9 ADVANTAGES OF WIRELESS SESNOR

The demand of using wireless sensor is increasing day by day. The main advantages [27-29] of wireless sensor in different real life applications are presented in the following sections.

2.9.1 Safety

Wireless sensors are successfully used in different environments conditions such as high temperature and very low temperature in different industrial and residential areas. Operators can monitor the processes in using wireless sensor in different activities in hazardous environment and confirm the safe operations performed. There is no data loss risk to access and obtain data from the collection point in the system.

2.9.2 Convenience

It is convenient for the engineers and operators to monitor and access different sub stations from one station. Thus, it saves time and lot of monitoring tasks can be performed by less human interaction. This is how recently a number of large factories,

companies, stations, air traffic etc. are using the wireless technologies for centralised monitoring and controlling.

2.9.3 Cost Reduction

The controlling process using wireless sensor is cost effective by avoiding extension wire and other unnecessary accessories. Thus, the companies are benefitted by eliminating more workers to reduce the companies' budget for workers. As a result, company authorities can extend their business efforts in other areas.

2.9.4 Accuracy

The system developed by the sensor can provide very precise estimation of performing high-level task within a very short time rather than manual operations. The operations also reduced time as it can perform and handle lot of data for transferring, storing and analysing purposes remotely.

Chapter 3: Literature Survey

3.1 INTRODUCTION

Monitoring road traffic particularly vehicle speeds is an utmost priority in today's society as the number of vehicles on the roadways is growing and the traffic is becoming very congested. On an average, one person dies by vehicle crash in every minute. Nearly 10 million people are injured each year by vehicle accidents. The hospital bill, damaged property and other costs are adding up to 1%-3% of the world gross domestic product [30]. Over speed of a vehicle is one of the most important reasons for increasing traffic accidents. Recent statistics shows that 25.1% accidents are caused by vehicular over speed [31]. Traffic planners and law enforcement agencies are paying much attention on the road traffic monitoring systems and methods for an increased safety policy, incident management and road traffic reporting. In general, these monitoring systems are involved for counting, classifying and/ or estimating speeds of the moving vehicles on the road. Therefore, there is a need to provide an easy-to-use and more flexible road traffic management system for monitoring the speed of a moving vehicle on the roadways.

In order to improve the road safety, it is important to know about existing road traffic monitoring systems. Although speed camera is commonly used for road traffic monitoring, a variety of sensors have been employed for automatically monitoring the speed of vehicles [32]. This chapter starts with a very basic idea of intelligent transportation system (ITS) and its major areas. Subsequently, different road traffic monitoring system has been presented briefly. Extensive literature related to wireless sensor based systems has been reviewed to gain knowledge about the existing systems. Another major part of the literature review was focused on road traffic data management systems. A general framework for the current literature review has been presented in Figure 3.1.

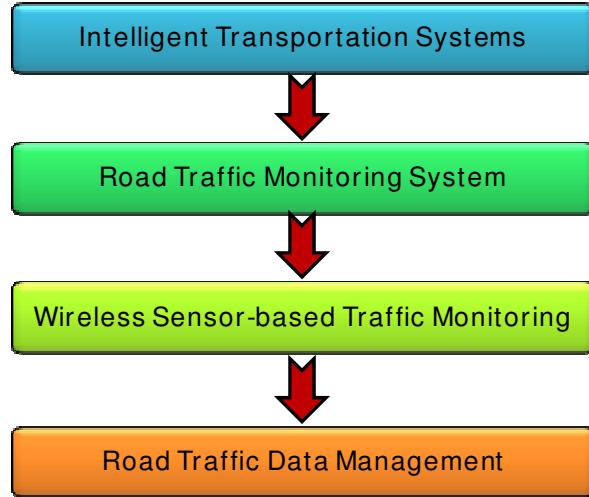


Figure 3.1: Literature review framework

3.2 INTELLIGENT TRANSPORTATION SYSTEM

The concept Intelligent Transportation System (ITS) is to improve safety, efficiency and service, and traffic situation by applying advanced technologies of sensing, detecting, computing, communicating, controlling and managing in all kinds of transportations. Traffic information collection is one of the vital parts of intelligent transportation system (ITS) associated with road design, traffic design, traffic management, control, and implementation [33]. General traffic information collection system is based on inductive loop detector, microwave detection, video detection, infrared detection etc. Dynamic Route Guidance System (DRGS) is one of the important parts of ITS that can supply real-time traffic information and vehicle parameters [34]. Traffic Message Centre (TMC) and on board information receiver are the main elements of DRGS [35]. In the context of this research work, the discussion will be focused on the road traffic monitoring systems with particular emphasis on the wireless systems.

3.3 ROAD TRAFFIC MONITORING SYSTEMS

The oldest road traffic monitoring method involves counting of the number of passing vehicles by survey personnel stationed at traffic intersections [36]. Later, use of automatic measurement methods including ultrasonic and magnetic detector came into action. Furthermore, the automated methods have been developed using pattern recognition from image signals obtained by cameras. The automated techniques help in monitoring and controlling the road traffic by identifying the types of the vehicles, number of vehicles, traffic density at a particular location, obstruction of the traffic etc.

Conventional traffic monitoring systems use direct speed measuring devices. These systems include video camera-based systems, radar counters, Doppler radar, magnetic detector-based measurement techniques, ultrasonic-based measurement techniques, acoustic detectors, inductive pick-up loops etc. [37]. In addition to the primary cost and maintenance cost of these devices, they have technical disadvantages. Doppler radar can readily provide outputs, which can be received and processed by computers. However, Doppler radar may not provide accurate data [38]. Inductive pick-up loops are installed in highway surfaces. These are connected to a central processor unit. While the cost of these detectors is not too high, the installation and maintenance costs of the detectors are expensive because of the involvement of a great amount of cabling, or dedication of a substantial transmission spectrum. The main disadvantage is that inductive pick-up loops usually do not provide speed data directly. Conventional traffic monitoring devices like infrared detectors, acoustic detectors, cameras, radar counters can help in identifying the presence of vehicles and provide data from which vehicle speeds can be roughly established [39]. However, these systems do not provide a mechanism for collating, storing and processing individual vehicle speed data in server computers. Some of the conventional traffic monitoring systems and vehicle speed measuring techniques has been presented in the following sections.

3.3.1 Camera-Based Monitoring Systems

Camera based systems use video image processing techniques in order to identify the moving vehicles and traffic flow parameters. Images are grabbed using video cameras [40-43]. A series of image processing algorithms are used on the raw images in order to digitise them. Vehicle speed data are extracted with image processing technique. The cost of installation and maintenance of the video cameras are high. The performance of the camera-based traffic monitoring systems is unsatisfactory in foggy weather conditions or poor visibility, or other environmental conditions, e.g. reflection coming from the wet pavements, shadows etc. A typical camera-based traffic monitoring system deployed on a roadway is presented in Figure 3.2 [13].

The speeds of the vehicles passing within a time interval are recorded by grabbing images of the passing vehicles using a digital television camera connected to a computer [44]. Using this system, a vehicle's speed is computed from the relationship between the number of frames and the two positions of the vehicle at a specific time interval.

TRIP II system has been reported for obtaining the vehicle speed on the roads by employing image processing technique in traffic monitoring [45]. This technique automatically detects and measures the speed of the moving vehicles passing through the camera thereby allowing the system to collect the information on the vehicles, determine the speeds of the vehicles and other associated information. It has been estimated that a detection accuracy of better than 99% can be achieved through image processing technique.



Figure 3.2: Camera-based traffic monitoring system deployed on a roadway [13]

3.3.2 Microwave RADAR-based Systems

Microwave radars are used to estimate the speeds of the moving vehicles on roads [46]. Typical microwave radar transmits electromagnetic energy at a constant frequency and uses the Doppler principle considering the difference in frequency between the transmitted and received signals, which is proportional to the speeds of the moving vehicles [25]. Another type of microwave radar transmits a frequency-modulated continuous wave with varied time-dependent transmitted frequency. The speeds of the vehicles are estimated using the microwave radars. The radars estimate the time taken by a vehicle to move between two positions. However, they are not able to estimate the speeds of the moving vehicles accurately.

A microwave radar apparatus operated in a non-coherent mode has been reported for observing road traffic located at Logan airport in Boston [47]. The apparatus comprises of a conventional rotating antenna and a pulsed magnetron transmitter. The vehicle speeds are obtained by tracking individual vehicles over successive antenna scans for a specific period of time. However, the speed determination technique is not suitable in congested traffic conditions or when one vehicle tries to cross the other speeding vehicle on the road.

Weissman [35] describes a method and system for remotely monitoring road traffic using one or more microwave radar systems located at elevated positions. The technology used in this system is the pulse-Doppler waveforms (i.e., electromagnetic energy), which provide range-velocity maps of traffic over a number of selectable roadway sections. The maps are used to estimate the speed and associated useful traffic information (e.g., density, volume of traffic flow, discontinuities in the traffic flow etc.).

3.3.3 Laser-Based Systems

Traffic monitoring system includes laser-based systems, which are used for counting, classifying and measuring speeds of the moving vehicles on the roads. One advantage of these systems is that these do not need installation work on the floor of the roads/highways; rather, they are installed on overhead positions of the roads/highways [13]. Laser-based systems are durable and reliable for the measurement of the speeds of the moving vehicles. The main disadvantage of the system is that the systems require a pre-structured traffic on the roads.

3.3.4 Ultrasonic-Based Measurement

Ultrasonic-based sensors are used for detecting the vehicles and speeds of those moving on the roads. Ultrasonic sound waves are used for this purpose. Typically, these sensors are mounted on an overhead structure on the roads (Figure 3.3) and transmit sound waves at 25 KHz to 50 KHz. It has both transmitters and receivers of sound waves [13]. According to the principles of ultrasonic-based detectors, a part of the transmitted energy is reflected back from the vehicles to the receivers. The difference between the transmission and reception of the sound waves is calculated, the vehicles are detected and their speeds are estimated. One disadvantage of the ultrasonic-based sensor technology for measuring the speeds of the moving vehicles is that the technology is

expensive. Another disadvantage of this technology is it is very sensitive to noise and environmental conditions of the roads.



Figure 3.3: Ultrasonic sensor-based traffic monitoring system mounted on an overhead structure [13]

3.3.5 Intrusive Sensors and Road Tubes

Most of the traffic control systems use intrusive sensors, including inducting loop detectors, micro-loop probes and pneumatic road tubes for their high accuracy of vehicle detection ($> 97\%$) [48]. Moreover, above-ground sensors like video image processing, microwave radar, laser radar, passive infrared, ultrasonic and passive acoustic array are being used. All of these sensors include high equipment cost. The accuracy of these sensors depends on the environment conditions.

3.3.6 Radio Frequency Identification Transponders

Radio Frequency Identification Transponders (RFID) are used in order to obtain individual travel time based on vehicle re-identification [49]. Moreover, License Plate Recognition (LPR) systems based on image processing techniques are used for measuring the vehicle travel time when a vehicle crosses two sensors. Furthermore, global positioning system devices are used for obtaining the position and speed of the vehicles with high accuracy for gathering traffic information. There are some existing

speed adaptation systems such as RFID-Radio Beacons, which works to transmit data to receiver in the vehicle with local speed limits, school zones, variable speed limit or traffic warnings. This system is used for slow (or low-speed) vehicles running only in the restricted zones.

3.3.7 Satellite-Based Road Traffic Monitoring

A satellite-based traffic monitoring system is reported in [50-52]. A radar Earth observation satellite, known as ‘TerraSAR-X’, is equipped with a “Dual Receive Antenna Mode” (DRA-Mode) allowing recording of two radar channels. Synthetic Aperture Radar (SAR) aids in estimating the vehicle speeds moving on the roads. This measurement is very precise. In 2000, the first speed estimation of the vehicles moving on roads from the Earth orbit was conducted on the A9 motorway north of Munich using the German X-SAR radar onboard the Shuttle Radar Topography Mission (SRTM) (Figure 3.4).



Figure 3.4: TerraSAR-X traffic monitoring on the A4 motorway west of Dresden, Germany [50]

3.4. WIRELESS SENSOR BASED TRAFFIC MONITORING

A range of different traffic monitoring and subsequent control approaches are used nowadays. Some of the systems use sensors embedded on the roadways and the systems are dependent on connected cameras and basic weather sensors [53]. The electronic traffic monitoring systems use a number of sensors. The associated installation and maintenance cost of these systems is very high. Constant maintenance is another issue of these systems.

More recently wireless sensors have found application in ITS to significantly improve the efficiency of existing transportation systems. Wireless sensors have the capability of integrating detecting, computing, and wireless communication with low cost and easy installation [54, 55]. The following sections reviews the wireless sensor employed in road traffic monitoring.

3.4.1 Techniques for Traffic Monitoring

3.4.1.1. Accelerometer sensor

Another road traffic monitoring system is known as TrafficSense [56]. TrafficSense uses smartphones to establish this idea where a microphone, camera, GPS, accelerometer can be used with phones as traffic sensing function [11]. The system is able to detect the audio data and send it to a base station for further processing. It sends the collected data to a server. Computing, communication and sensing features are present in this system. TrafficSense uses the accelerometer on a phone to sense road and traffic conditions.

An automated vehicle classification system has been proposed by Ma et al. [57]. The reported system uses wireless accelerometers and magnetometers wherein the accelerometers detect vehicle axle and the magnetometers report vehicle arrivals and departures and determine speeds of the moving vehicles. This system is installed on Interstate 80 at Pinole, CA, USA, and tested under different traffic conditions. High accuracy is obtained in determining the speeds of the vehicles using this system. Different views of the traffic monitoring system installed on west Interstate 80 at Pinole, CA, USA is presented in Figure 3.5.

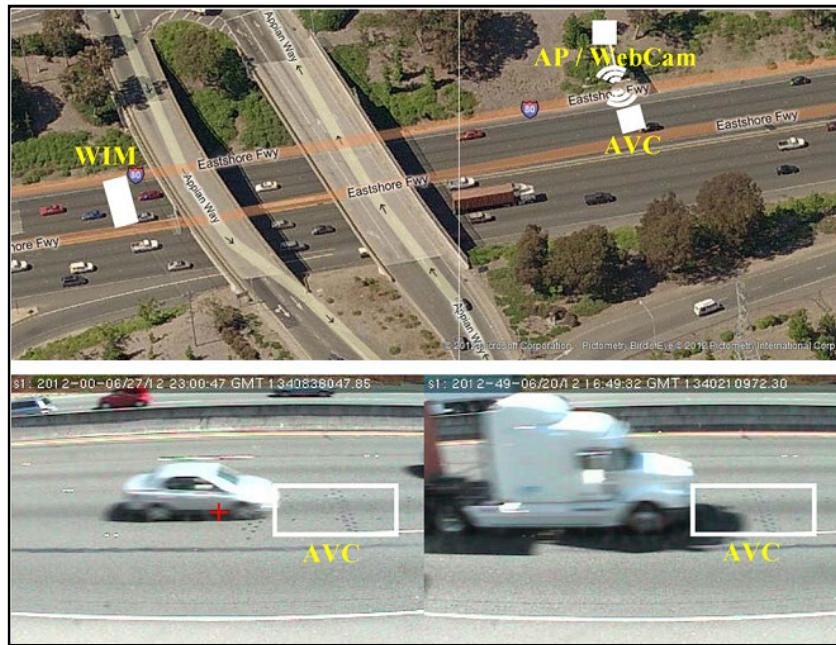


Figure 3.5: The accelerometer-based traffic monitoring system installed on west Interstate 80 at Pinole, CA, USA [57]

A traffic monitoring system, on the roads of Bangalore, India has been reported in [58] where the system deploys sensing components like accelerometer, microphone, GSM radio, and GPS sensors. Although the principal thrust of the reported system is on road condition monitoring, the system also evaluates traffic speed using GSM cellular tower information.

3.4.1.2. Magnetic sensor

An alternative traffic monitoring arrangement, known as ‘Traffic-Dot’, using wireless sensor networks has been reported by Coleri et al. [37]. The monitoring arrangement enables the wireless sensor network, particularly at the sensor nodes, to collect traffic information for measuring the vehicles’ speeds with a node pair. The collected information is then forwarded to the access point over radio. The components of the ‘Traffic-Dot’ system are a built-in processor, a radio, a magnetometer, a battery, and a cover for protecting the system from the moving vehicles. The magnetic sensor data estimate the speeds of the moving vehicles.

SensysTM is a wireless vehicle detection system [11, 59] that uses pavement-mounted magneto-resistive sensors. They are used in order to detect the movement of the moving vehicles, particularly the speeds of the vehicles. These wireless sensors transmit

collected data by radio technology based device to a nearby Sensys™ access point. These data are forwarded to remote traffic management controllers.

Hisao and Takashi [60] have disclosed a traffic flow management system, which uses a magnetic sensor for detecting vehicles. A number of magnetic sensors are installed on the road. The principal aim of the system is to detect the magnetic characteristics of a moving vehicle. The difference between a pair of maxima and minima in the magnetic data during a fixed period of time is calculated and compared with a pre-set value. The speed of the vehicle is determined based on the magnitude of the phase difference of the magnetic data.

3.4.1.3. Ultrasonic sensor

A traffic monitoring system proposed by Mastuo [61] measures the speed of moving vehicles on the road. The system also calculates the measurement errors and data transmission sampling errors. Transmission sampling errors are increased as the vehicle speed increases.

Chen et al. [62] have reported the use of a microphone array for detecting the sound waves generated by the vehicles moving on the roads. The sound waves are digitised further and processed by a computer program. These digitised sound waves are translated to measure the speed of the vehicles.

3.4.1.4. Loop detector

Single-loop detectors are one type of the commonly used vehicle speed detectors. In the recent times, out-of-pavement detectors have replaced loop detectors by emulating the operation of single-loop detectors. Single-loop detectors are helpful in determining the vehicle speed on the road [63].

Intelligent transportation systems are used to estimate freeway traffic speeds on urban roads in real-time environment. Bachmann et al. [64] have reported an effective speed estimation technique by loop detectors based on a multi-sensor data fusion. An automated method of probe vehicle data collection is included in this speed monitoring system. The system is capable of collecting vehicle data wirelessly, including speeds of the vehicles, using the vehicles' Bluetooth-enabled devices travelling past the roadside Bluetooth receivers.

3.4.1.5. GPS

An intelligent vehicle monitoring system has been reported by Zhang et al. [65]. The proposed system communicates wirelessly between a moving vehicle and the control centre by using GPRS network. The vehicle mobile component includes a GPS receiver, a GPRS module and a main control module. A single microchip controller is used in the main control module. Motorola G20 is used as GPRS module and GPS-25LVS is used as GPS receiver to communicate with single chip microcontroller through UART as communication network. Overall, the main control module processes the positioning data received by the GPS receiver and transmits the data through the GPRS module. Serial interface, SIM interface and power interface are used in this system. G20 module is a GSM/GPRS modem that supports EGSM 900/1800MHz and GSM 850/1900 MHz. Serial interface of this modem fits V24 protocol and supports voice, data, fax and short message. GPS-25LVS OEM board is a 12-channel single chip receiver can trace all visible GPS satellite and process signals. Physical links, point-to-point links and TCP/UDP links are used here for GPRS data transfer [65, 66].

Karthikeyan et al. [67] report how a GPS transceiver, receiver can collect data and from the moving vehicles on highway and compared between vehicles' speeds and the pre-set speed limit on the road. This system contains ISA system with a wireless transceiver. The tollgate in highway has transceiver that can communicate with vehicle's transceiver. The vehicle equipped with the system arrives near the tollgate, which broadcasts data. The data is saved in vehicle module as a table of information of coordinates and speed limit. After storing the data in the memory of the system, the GPS receiver tracks its location and compares its coordinates with the received data. GPS compares between the speed limit listed in the table and speed limit from that point. The system alerts the driver by sound, displaying speed limit and indicates the local speed limit. Matlab[®] platform is used for image data verification.

3.4.1.6. Web-based technology

Luciani [53] has proposed a traffic monitoring system and method using one fixed transceiver located near the road that is in communication with a moving transceiver located in the vehicle running on the road. The two transceivers use Bluetooth[®] wireless technology. Raw or processed data of the vehicle are sent to a Traffic Monitoring Server (TMS) using web-based technology. The system calculates the speed of the moving

transceiver based on time difference. These calculations are performed at the wireless network level. The calculated speed of the moving transceiver is compared with a reference speed on the road.

Lapidot and Silverberg [68] have disclosed a traffic monitoring system for monitoring moving vehicles. The system comprises of a mobile communication network that helps in receiving data, transmitting to the desired place and storing the collected data of the vehicles. This is a kind of web-based technology, which uses a web server. The computer computes average traffic speed based on the vehicle locations and its travel time to cover the locations. The system helps to evaluate the traffic parameters such as vehicle travel time, vehicle speed, traffic density and traffic flow.

3.4.2 Wireless Sensor inside a Vehicle

A speed control technique for heavy-duty vehicles under constant vibration acceleration has been reported by Endo et al. [69]. This control technique is conducted by simulation, considering saturation and delay characteristics in accelerating heavy-duty vehicle where desired speed and distance are calculated. This system is complex to understand clearly, as it does not show experimental set-up.

Traffic monitoring using Wireless Location Technology (WLT) is another example of monitoring speed of vehicles on the roadway. WLT-based traffic monitoring facilitates in collecting traffic information, particularly the speeds of the moving vehicles [70]. In this system, the locations of wireless devices in the vehicles are monitored in order to obtain vehicle speeds in the traffic flow. The communication of the wireless device placed in the vehicle with a series of sensors installed on the roadways is utilised to derive the speed of the vehicle. Fontaine and Smith [71] have reported the efficacy of WLT-based monitoring systems under different configurations.

Herrera et al. [72] have assessed the feasibility of a traffic monitoring system using GPS-enabled mobile phones near Union City, California. This system performs sampling and data collection using Nokia N95 with GPS chipset thereby obtaining position data and speed of the vehicles. A prototype system architecture is implemented for testing the sampling strategies of the collected data. This system provides high accuracy in position and velocity measurements using the GPS chipsets.

Sridharamurthy et al. [73] have proposed a traffic violation detection technique for vehicular ad hoc networks comprising a sensor device, a digital map, and a GPS based system. A Vehicle to Infrastructure (V2I) communication system is used in the reported technique. Accelerometer is used for enabling the V2I communication. The principal objective of this technique is to detect the vehicles disobeying the present speed limits. In the devised technique, the vehicle sends a message in a packet format to the closest infrastructure node about its ID, location, speed and direction. The infrastructure node is equipped with a GPS receiver and micro-controller.

Ghadiri et al. [74] have elucidated the effectiveness of an Intelligent Speed Adaptation (ISA) system in Penang, Malaysia. An ISA system has been set up in vehicles that support driver compliance with the pre-set speed limit on different roads. The pilot system was organised by two trips a day for a two-week period between 27th September and 12th October 2010 under real traffic conditions along an 18.5 km long test corridor in Penang, Malaysia. The test was implemented on highway with regular speed limits of 50, 60 and 110 km/hr. A vocal message and visual warning was embedded in the ISA device to inform the driver when he reached 10 km/hr close by the introduce speed limit. The collected data shows by random spot speed registered in the device and generates by itself. Data were logged and downloaded after each trip and during the trial.

Agerholm [75] has reported an ISA system where On Board Unit (OBU), comprising GPS/GPRS unit with a memory card stored digital map with speed limit, is installed in the moving vehicles. A visual display shows the speed limit and penalty points and a GPS antenna. The GPS receiver in OBU calculates a position every second and matches with that of the speed limit indicated on the speed map. If the speed of the vehicle exceeds more than 5 km/hr, OBU gives a verbal warning to the driver and repeats every sixth second until the speed of the vehicle is below 5 km/hr. Next and subsequent warnings generate penalty points for the driver. OBU is able to calculate the right speed of the vehicle and its position. The ISA system is a complicated set-up, and the installation and maintenance costs are comparatively high.

3.4.3 Wireless Sensor on the Road

A traffic monitoring system for installation on the roadways has been proposed by Megalingam et al. [76]. The system includes a wireless sensor network, with both

software and hardware, to simulate the road traffic monitoring in a laboratory environment. The principal goal of the system is to monitor and generate reports for speeding vehicles during over speeds and accidents. A motor generates the vehicle speed controlled by a micro-controller. If any interrupted signal is generated a data acquisition card receives the interrupted signals and sends to a mote. The mote transmits these data and sends to a router through the wireless sensor network. The router transmits the data to another mote connected to a sensor board and simultaneously sends it to a gateway which is built by the sensor board and mote attached by a connector. The gateway sends the received information to a server through a USB interface for further analysis. Cygwin software (Unix-like environment and command-line interface for windows) is used to configure three motes. Two motes are used for sender and receiver and the third mote can capture any radio packet data (TOS). Xsniffer software monitors data transmission and displays as packet and store on excel file, which is processed by Matlab® for over speed detection and reporting. The cost of the system under real-world scenario has not been given [31].

Megalingam et al. [22] have also proposed a smart traffic controller for efficient traffic routing and tracking over-speed vehicles using wireless sensor network. An ITS system is proposed through software and hardware implementation using different algorithms for detecting vehicle speed and reporting the over-speed vehicles through wireless sensor network. The system is implemented in a laboratory environment. A motor is introduced for generating speed, which is controlled by a micro-controller when an interrupted signal is generated. A data acquisition card receives the interrupted signals and sends them to a mote. The mote transmits these data and sends to a router through a wireless sensor network. The router transmits the data to another mote connected with a sensor board and sends it to a gateway which is built by the sensor board and mote attached by a connector. The gateway sends received information to a server through a USB interface for analysis. The motes are configured in Cygwin (Unix-like environment and command-line interface for windows). The message read by Xniffer are stored on excel file processed by GUI using Matlab® for over speed detection and reporting. This data file enables the traffic police to keep records of the offenders. The cost of implementation of the system in real-world environment is unknown.

CitySense [77] is an urban-scale wireless networking traffic monitoring test bed containing around 100 Linux-based embedded computers fitted with a range of radios

and numerous sensors. The system is mounted on buildings and streetlights across the city of Cambridge, USA. In this system, nodes of the network can be monitored and are directly programmable by the end users. The cost of this wireless sensor-based traffic monitoring is comparatively high.

A traffic monitoring system using a wireless sensor network and an access point has been described by Coleri et al [37]. A group of nodes, sensors, processor, a radio and a battery are included in this wireless sensor network. Traffic information, such as vehicle number, speed and length of vehicle, is generated and sent to access point over radio. The traffic management centre collects the information from each access point. The hardware used in this system is Traffic-Dot.

Atluri et al. [78] have identified suitable configuration and working capacity of a transmission based optical sensor by evaluating the number, presence and speed of individual vehicle. A system has been developed with laser transmitters and receivers. Traffic parameters and speed of the vehicles are measured by this sensor.

A vehicle speed detection system using image-processing technique has been reported by Pornpanomchai et al. [79]. The speed of a moving vehicle is measured from the known distance and time from a video scene. The limitation of this system depends on a few factors, such as size of video scene, size of vehicle, marking point, stability of brightness level and number of colours from the input video.

Zengqiang et al. [31] have reported a method of detecting the over speed vehicles with a wireless traffic monitoring system based on GPRS. This system uses communication, internet, image processing and other advanced technologies. When speed exceeds the threshold value, two images of the vehicle are monitored on screen. The first image is for recognise the vehicle number plate and the second image is to record freeway scene. Every monitoring station has its unique IP address and the data package consists of speed, peccancy time, paccancy place serial number, compressed image of establishing shot etc., which are sent to a remote monitoring centre through GPRS network and GGSN gateway. Afterwards, the packets are partitioned and stored in the server of a control centre. The speed monitoring station includes two piezoelectric sensors, a light sensor, two CCD cameras, an auxiliary lamp, a circuit board for signal regulating, a control main board and wireless modem (Motorola G20). Although this system has high reliability and low cost, it is unfit for constructing lineate monitoring network.

3.4.4 Shimmer AWireless Sensor

ShimmerTM sensors are wireless in nature. The sensor devices have been employed in different fields such as kinematic [80, 81], physiological [82, 83] and ambient sensing applications [84, 85].

3.4.4.1 Disease diagnosis

A shimmer wireless sensor has been used as a non-invasive technique of monitoring and diagnosing diseases symptoms in a patient's body [86]. The ShimmerTM baseboard has platform expandability feature as it contains both internal and external expansion connectors. Shimmer's internal expansion feature allows the user to connect daughter boards depending on its applications. The usage may be in electrocardiogram (ECG), kinematics (gyroscope, magnetometer) and Galvanic Skin Response (GSR) to the baseboard. ShimmerTM can be connected to a programming dock through its external expansion.

ShimmerTM accelerometer is used to build a system for calculating the severity of symptoms and motor complications in patients suffering from Parkinson disease [87]. The work has developed a Support Vector Machine (SVM) classifier for estimating the severity of tremor, bradykinesia and dyskinesia from the collected accelerometer data. The SVM-based reports are sent for video recordings of the patients with a number of standardised motor tasks. These video recordings are analysed for the severity of Parkinson symptoms. Results display estimation of clinical scores based on compatible features with ShimmerTM platform.

3.4.4.2 Patient Monitoring

Extensive applications of the ShimmerTM sensors are found in the field of biomedical engineering. For example, the sensor is used for patients' vital sign monitoring such as blood pressure and blood flow, core temperature, ECG and carbon dioxide concentration [88]. The vital signs of the patients are grabbed with shimmer sensor motes implemented in TinyOS using wireless body area networks by an inter- and intra-BAN communication protocols. A base station and mobile terminal are required in order to communicate the data.

3.4.4.3 Structure Monitoring

Ultrasonic structural health monitoring has been conducted using Shimmer™ wireless sensors. The monitoring mechanism combines solar and wind energy harvesters for optimising energy through a task-scheduling algorithm with dynamic voltage and frequency scaling [34]. Figure 3.6 shows a Shimmer™ development platform consisting of three independent boards used for this purpose. Figure 3.7 presents the Shimmer platform architecture.

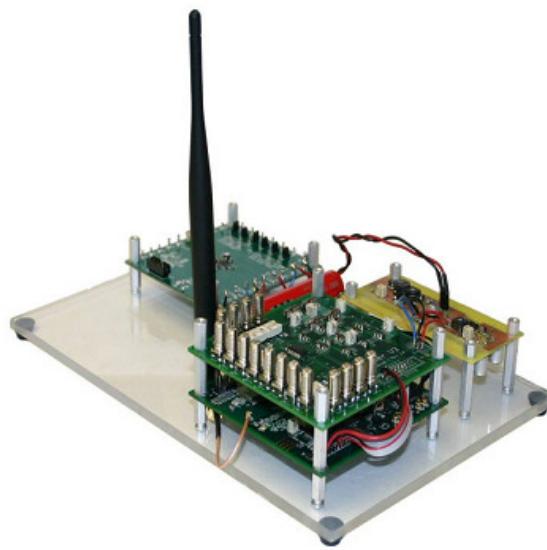


Figure 3.6: A Shimmer™ development platform used for structural health monitoring [34]

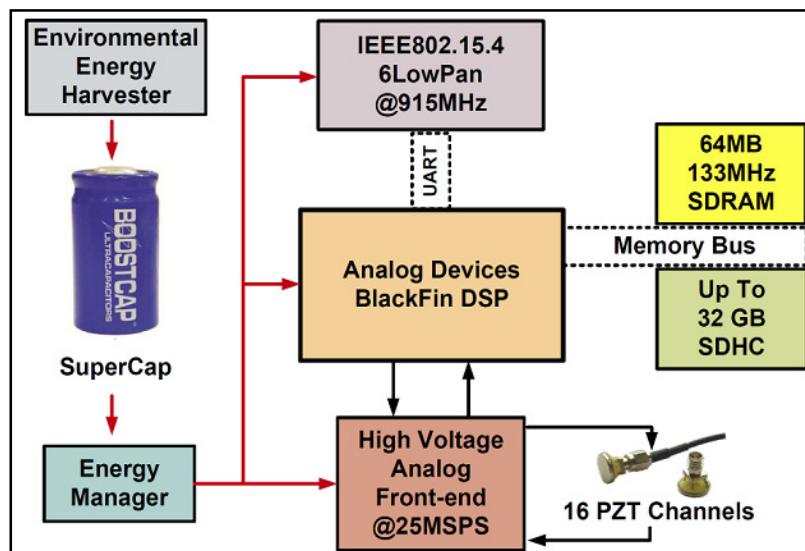


Figure 3.7: The Shimmer™ platform architecture [34]

3.4.4.4 Environment Monitoring

Shimmer™ wireless sensor has also been applied for monitoring maritime [89]. The system can interface multiple environmental sensors with a software interface called ‘BioMobius’ using Graphical User Interface (GUI). Fibre optic sensor is integrated to Shimmer™ for creating linkages between maritime colours, algae pigmentation, scattering and absorption for prototyping, real time kinematic motions and physiological sensing purposes.

3.4.5 Wireless Communication Protocols

The wireless sensor network-based traffic monitoring system is used in [37] wherein a communication protocol, known as “PEDAMACS” (Power Efficient and Delay Aware Medium Access Protocol for Sensor Networks) is utilised in this traffic monitoring arrangement. The system, including the communication protocol, ensures the timely arrival of vehicle data from the sensor nodes to the access point. The communication protocol of the wireless sensor network determines how the radios are operated. Battery lifetime is also determined by the protocol. PEDAMACS protocol is a Time Division Multiple Access (TDMA) scheme that falls under the category of the existing MAC protocols. The PEDAMACS protocol has four phases, such as the topology learning phase, the topology collection phase, the scheduling phase and the adjustment phase.

A wireless traffic monitoring system, known as ‘TrafficView’, is proposed in [90]. In this system, each moving vehicle is allowed to broadcast a small packet of a few kilobytes to communicate wirelessly with the receiver. This is made possible with a MAC layer protocol, i.e., IEEE 802.11b protocol. This protocol limits the size of the payload that is sent on the network channel to a maximum size (which is 2312 bytes).

Huang and Miller have proposed an electronic traffic signalling protocol framework [91]. Benefits of wireless traffic monitoring system and the potential problem of omnidirectional property of wireless monitoring are discussed. The potential inconsistency threats caused by the drift in GPS data is disclosed in order to provide a guideline for intelligent transportation systems framework.

ZigBee is a set of communication protocol that is required for transmitting data from a tollgate to vehicle [67]. It is a wireless-based device operated in 868MHz, 915 MHz and 2.4 GHz frequency bands. There are packet losses after transmission of data. While the

data size increases with respect to time in indoor operations, the signal strength decreases with distance resulting in a packet loss in outdoor operations. Data transmission experiments are conducted in indoor, outdoor and dynamic environments. Within the indoor environment, the data transmission time and subsequent packet loss is quite predictable. However, the data transmission and packet loss is unpredictable in the outdoor environment.

3.4.6 SCOOT and SCATS Control Systems

The two main traffic control systems are used are: (i) Split Cycle Offset Optimization Technique (SCOOT) and (ii) Sydney Coordinated Adaptive Traffic System (SCATS) [61].

3.5. ROAD TRAFFIC DATA MANAGEMENT

3.5.1. Data Collection and Management

Data management technique in traffic monitoring is one of the vital parts of intelligent traffic system. Inductive loop detector can acquire data related to traffic flow, usage of the roadways and speed of the vehicles for continuous monitoring of the traffic and simultaneously display the information through display device for the benefits of the vehicle drivers [92]. Microwave vehicle detector can detect multilane, vehicle flow, occupancy, vehicle speed, vehicle length etc. Infrared ray-based traffic monitoring systems scans the vehicle and converts the optical signals to digital signals for collecting the vehicle data. Online traffic monitoring system is an electronic system that collects data over time relating to a location with built-in sensors. Thus, for data management purpose the electronic online traffic monitoring has replaced chart recorders in different applications [93].

A traffic monitoring system and method has been reported by Roberts [94]. In this system, data related to traffic is collated and sent using a short messaging system message via a wireless network. In this system, Publicly Switched Telephone Network (PSTN) is used for obtaining a connection to a central computer. Data collection device is embedded in a remote traffic-monitoring unit. Traffic count data and other related data (e.g., location) are collected. Monitored traffic count data are routed from the remote traffic-monitoring unit to a central computer. The control information is routed

from the central computer to the remote traffic-monitoring unit. The traffic count data is stored and further processed at a remote location.

Mizunuma and Masaki [95] have reported a vehicular traffic monitoring system with specific vehicles in a long distance high-speed traffic. In order to know the speed and position of individual vehicle, communication with each of the vehicles on the highway is made possible using a channel for communication. In this system, a data processor through the principal network is used for storing vehicle speed information for the passing vehicles as detected by the sensor. However, a series of mobile transceivers mounted on respective vehicles and a video camera installed on the highway controlled by the data processor aid to obtain the parameters, including speed of the moving vehicles. Although this system is feasible, it involves huge cost. Furthermore, the data storage capacity of the computer associated in this system should be very high as it stores the images of the moving vehicles.

Arnold et al. [32] have developed a system for managing road traffic by measuring the speed of a moving vehicle on the road with different sensor configurations. A single transducer side fire configured sensor is used in order to determine the speed of the moving vehicle. Distance measurements, phase measurements, or Doppler shift measurements are the methods used for measuring the speed of the vehicles passing through the roads.

Jorgenson et al. [96] have reported a speed-monitoring program at Indiana, USA. The collected data is managed at the speed monitoring stations, which are distributed across the highway classes based on spatial distribution, crash distribution and daily vehicle-miles travelled. Statistical models are utilised in order to locate the mean speed and compare the same with that of the daily speed distributions.

3.5.2. Expert Systems

Genetic algorithm, fuzzy logic control, artificial neural networks, queuing network etc. are used in traffic monitoring systems [97]. An intelligent traffic monitoring expert system with RFID technology is reported by Wen [98]. The monitoring system collects traffic data using an RFID reader, a personal computer, a pair of infrared sensors and a high-speed database server. RFID helps the system to obtain and calculate average speed of the vehicles and average flow information on each road. A communication program is used in collating the collected data and exchanges them with nearby server

computers dynamically. A multi-agent system based on type-2 fuzzy decision module is also used in urban traffic management system [99].

3.5.3. Data Management Software Development

Programming interface with traffic monitoring devices has been reported in the literature. An example-based learning algorithm for moving vehicle detection using video image has been presented by Zhou et al. [100]. Principal component analysis is utilised in the computer program. A classifier based on a support vector machine has been designed to classify the vehicle. The algorithm is able to process more than 15 frames per second and the maximum image size for this is (384×288) pixels.

CarTel network nodes rely on opportunistic wireless (e.g., Wi-Fi, Bluetooth) connectivity—to the internet, or to “data mules” e.g., mobile phone flash memories, or USB keys, for communicating with the portal [101]. Recently, CarTel has been deployed on a small scale in Boston and Seattle. Figure 3.8 illustrates the system architecture integrating the different components of CarTel. In this architecture, the moving vehicles collect data, and log them to their local Intermittently Connected Data Base (ICEDB). Once the wireless network connectivity is available, the collected data is transmitted to the portal via a software called CafNet (carry-and-forward network). The portal is available to the users for further monitoring of the moving vehicles.

The information collected from the vehicles by the traffic monitoring system is managed by the base stations in numerous ways. The collected information contains updated traffic information and traffic parameters. Floating data contains information on time and location provided by positioning systems and the information collected from various vehicle sensors [102]. Figure 3.9 shows a data management technique using extended Floating Car Data (xFCD) service.

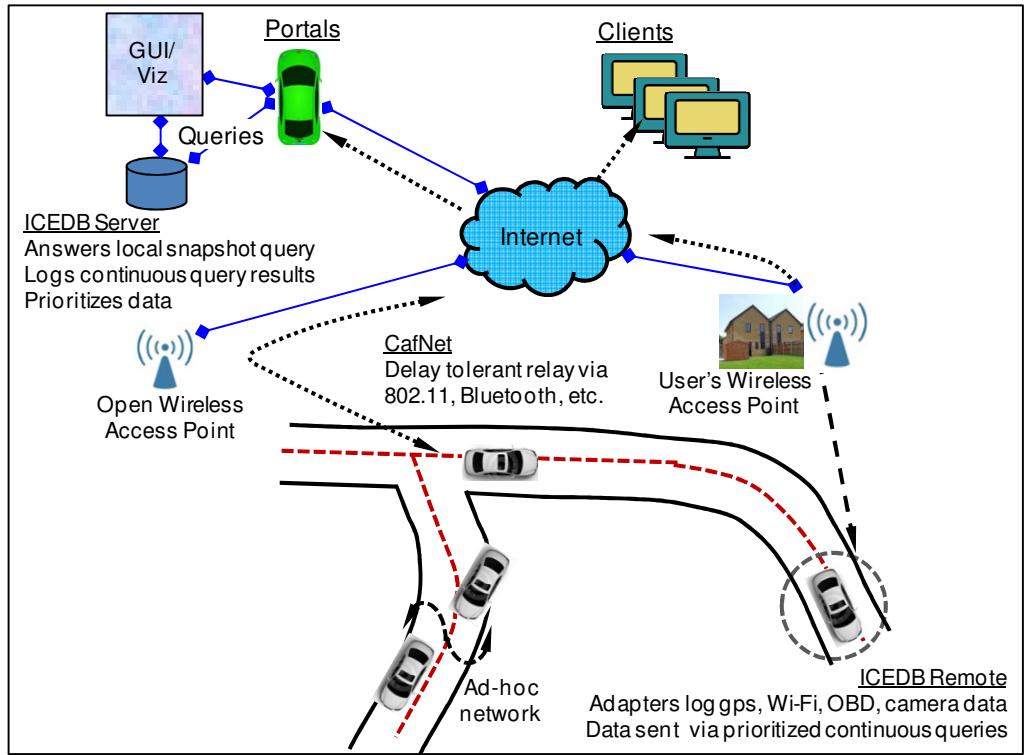


Figure 3.8: The system architecture of CarTel (Adapted from [101])

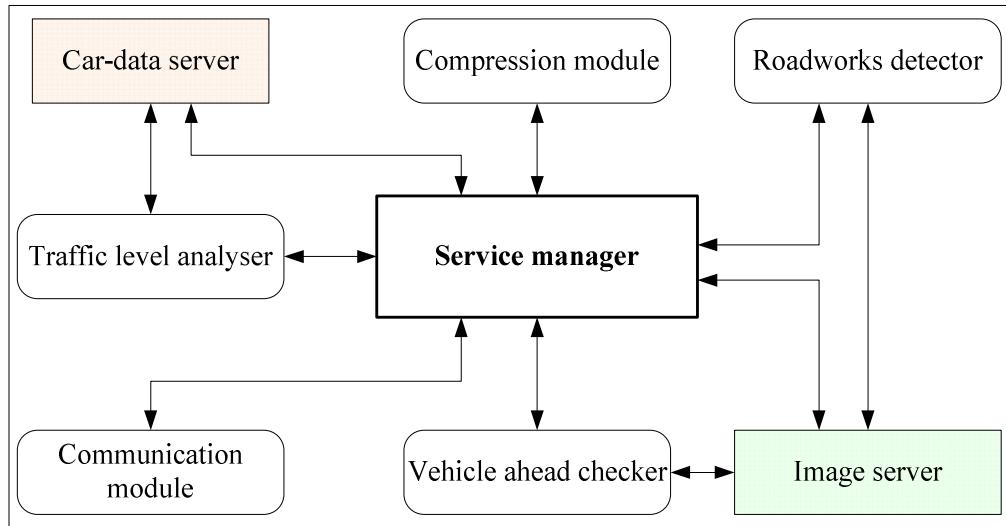


Figure 3.9: Traffic monitoring data management technique [102]

The service manager is the black box of data management and it controls the data collection. Car-data server and image server provide inputs to the service manager. Traffic level analyser, roadwork detector and vehicle ahead checker are activated only when the strategic decision-making requires extended data or data collection based on events. The collected data may be either visual data taken by a video camera/CCD camera or data from the on-board electronic control unit. Compression module

compresses the visual data, i.e. images, before sending them to the service manager. Communication module manages the communication with the service manager. A variety of software interface is provided in this data management technique.

Traffic monitoring system based on mobile sensor networks coupled with programming interface has proven helpful in determining the speeds of moving vehicles on the road. The speeds of the moving vehicles are computed and at the same time, the performance of this technique is monitored [103] using two traffic status estimation algorithms: the Link-Based Algorithm (LBA) and the Vehicle-Based Algorithm (VBA). These algorithms compute the real-time mean speed for individual vehicles on every road segment in Shanghai, China.

A wireless traffic monitoring system with programming interface has been designed by Nadeem et al. [90]. The developed system is known as ‘TrafficView’ that provides the driver with a dynamic view of the road traffic. The system uses a portable computer augmented with two slots of PCMCIA sleeve, Global Positioning System (GPS), 802.11b wireless network card, DSP-100 2-port RS-232 serial PCMCIA card, and an OBDI-II interface. The wireless card helps to establish the network connectivity, which in turn allows the moving vehicle to send and receive information. The ‘TrafficView’ software on the node of the network periodically updates the status of the vehicle, i.e., speed of the vehicle, using the OBDI-II interface. The speed of a moving vehicle is computed as a weighted average, while the broadcast times are computed as the minimum value.

The CarTel wireless traffic monitoring system has programming interface known as ‘CafNet’. Figure 3.10 shows the CarTel portal software architecture [101]. The collected data is stored in a relational database. Continuous query related to the moving vehicles can be made via CafNet software.

A transmission-based optical sensor system having laser transmitters and receivers has been described by Atluri et al. [78]. Simplicity in design of the system facilitates traffic management in roadwork zones. The system is implemented in Seneca Creek Road in Clemson, South Carolina. Vehicle data on the road is collected for two hours with an average of 336 vehicles per hour. The speeds of the vehicles moving on the road are measured using an integrated software known as SAS, which allows the users to

perform a number of statistical tests. This software enables the users to examine the data management using statistical techniques and to conclude on the traffic monitoring.

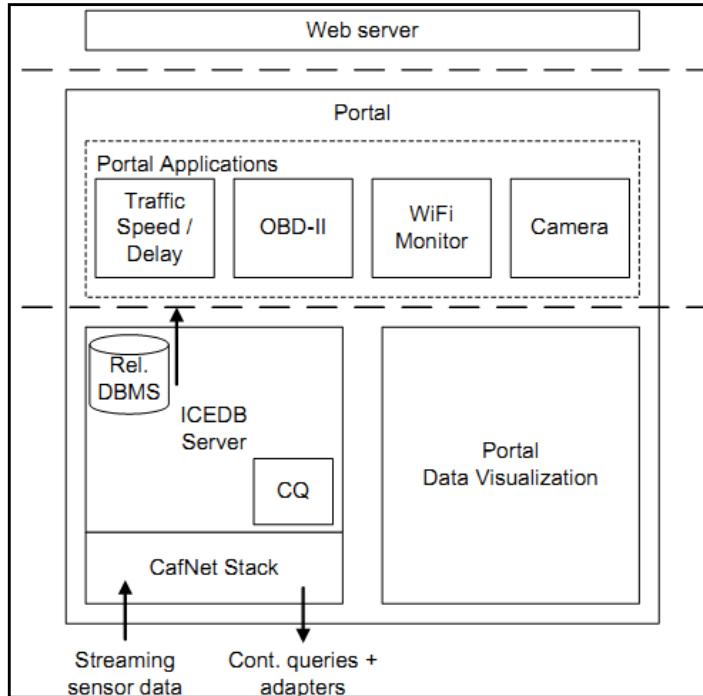


Figure 3.10: CarTel portal software architecture [101]

Hong et al. have designed and implemented a Web-based network traffic monitoring and reporting system [104]. The use of web technology and the Multi-Router Traffic Grapher (MRTG) is a simple, uniform, universal tool and supports HTTP 1.0 or higher versions. A single web page of the web-based monitoring system can accommodate four different plots such as daily, weekly, monthly and yearly, in order to facilitate the authorities for observing and analysing the traffic trends.

3.6 SUMMARY

A critical analysis of the available literature reveals that there are several shortcomings of the existing road traffic monitoring systems and vehicle speed monitoring approaches. Some of the existing vehicle speed monitoring systems has high equipment cost, complex installation processes, high maintenance cost and dependency on weather condition. Other systems have difficulties in managing the collected data. The difficulties at the user end of the client computers for accessing and further processing is cumbersome in some of the prevailing systems. Literature also shows that no integrated system is available for facilitating vehicle detection, speed measurement and monitoring, and collating and managing vehicle speed data. Shimmer sensor has not

been used for traffic monitoring. Furthermore, no study has considered the use of Java socket programming to interface the management of data related to vehicle speed.

The flexibility of the speed monitoring approach is an important issue in this regard. In order to obviate the shortcomings of the existing road traffic monitoring systems, a ShimmerTM sensor based experimental set-up is designed within a laboratory environment. Simulation of the real-world vehicle movement scenarios using the experimental set-up in the laboratory environment is also targeted. Ease of communication through TCP/IP and Java®-based socket programming between server and client computers is another issue that has been addressed in this research. Finally, this research proposes GUI-based ‘SpeedManage’ software for managing the vehicle speed data in a number of ways.

Chapter 4: Experimental Set-Up Development

4.1 INTRODUCTION

In order to test the concept of automatic road traffic monitoring using wireless sensor, a prototype experimental set-up has been developed in a laboratory. The set-up is developed with a wireless sensor device, a variable speed mechanical wheel and a laptop. The mechanical wheel acts as a vehicle. The wireless sensor is used to detect wheel movement by sensing vibration, measure the wheel speed, and transfer the speed data to the server. This chapter presents the prototype set-up development in detail and the procedure for speed data generation.

4.2 SYSTEM ARCHITECTURE

The general system architecture for real life traffic monitoring system has been presented in Figure 4.1. In this system, a sensor detects the vehicle running on a road and measures the speed of the vehicle. The processing unit also compares the measured speed with a pre-set speed limit on that particular road. If the vehicle speed is greater than the speed limit, the speed data is sent to a Central Control Centre through a wireless sensor network. The central control Centre collates and manages the speeding data to impose the penalty for the speed offenders.

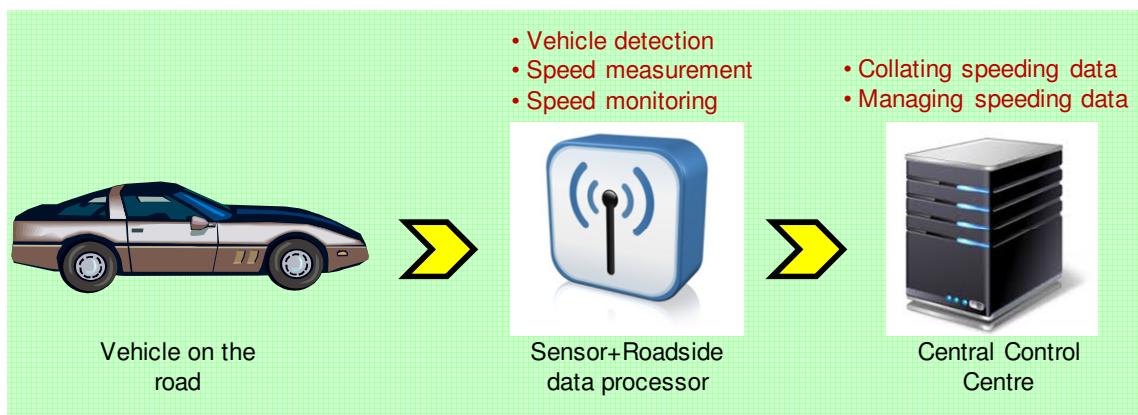


Figure 4.1: General system architecture for monitoring vehicle speed

In order to test the functionality of the general traffic monitoring system, a prototype system has been developed in a laboratory. The prototype system architecture has been presented in Figure 4.2. A mechanical wheel has been considered as a vehicle. The speed of the wheel can be varied randomly and measured by an integrated tachometer.

A wireless sensor is attached to the wheel to measure the vibration of the wheel, which will be proportionate to the wheel speed. A relationship is established between the wheel vibration and speed through calibration technique. This relationship can be used for measuring wheel speed (rpm) from the vibration data. The vibration data are sent to a client wirelessly and converted them to rotational wheel speed, which is again converted to linear speed to make it a realistic vehicle speed. If the speed is greater than a set speed limit, the data is sent to a remote server. In the server, a report is generated based on the speeding data to impose penalty for the offenders.

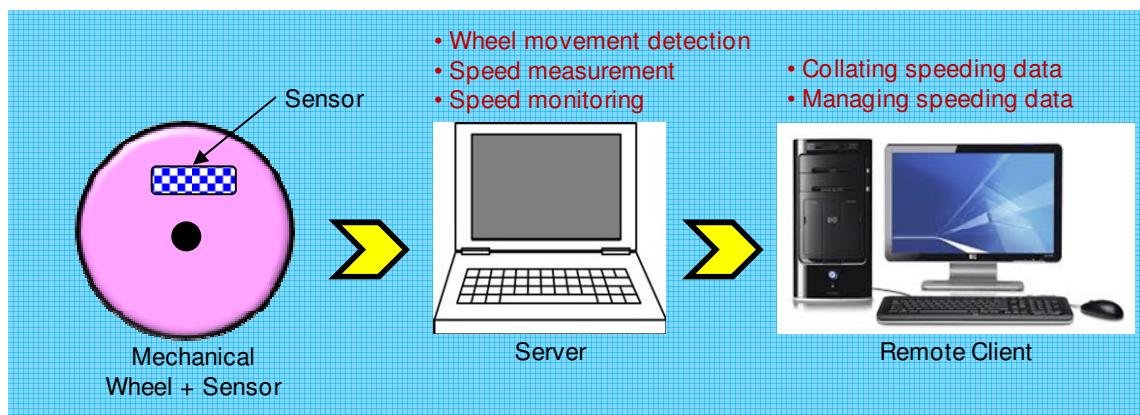


Figure 4.2: Prototype system architecture for monitoring vehicle speed

In this chapter, the discussion is only confined to the steps for developing prototype experimental set-up in order to generate random speed data by the wireless sensor. However, the management of the speed data will be described in Chapter 5.

4.3 SENSOR SELECTION

A commercial wireless sensor device (Shimmer) as shown in Figure 4.3 has been selected for developing the wireless system within a laboratory environment. This sensor device enables the user to transfer a large volume of data to the server. Shimmer sensor device generates vibration acceleration data. This small sensor device can capture and communicate data in real time. The applications of this sensor device include wireless magnetometer, accelerometer, gyroscope, GPS etc. It provides a range of functions including basic motion sensing, data capture, processing and wireless transmission of sensed data [105]. Shimmer is designed for wearable and remote sensing applications and is highly flexible and adaptable.



Figure 4.3: Shimmer wireless sensor device [105]

4.3.1 Shimmer Sensor Device and Its Mechanism

The Shimmer device is designed for wearable sensing applications and has an on-board microcontroller. The device has the feature of wireless communication via Bluetooth or low power radio. The device stores data locally to a micro SD card [105]. The internal infrastructure of the shimmer sensor device is illustrated in Figure 4.4. The device has an integrated accelerometer for motion sensing, activity monitoring and inertia measurement applications. The device acts as a baseboard for a full range of Shimmer wireless sensor modules and can be connected to a range of sensors, e.g., Gyro, Magnetometer, GPS/temperature, or strain gauge modules. It is used for a specific application, with configurable sensitivity, sampling rate, transmission rate and frequency, communication protocols, and packet formats. The benefit of this device is that it can be easily integrated and can interact with existing systems and technologies.

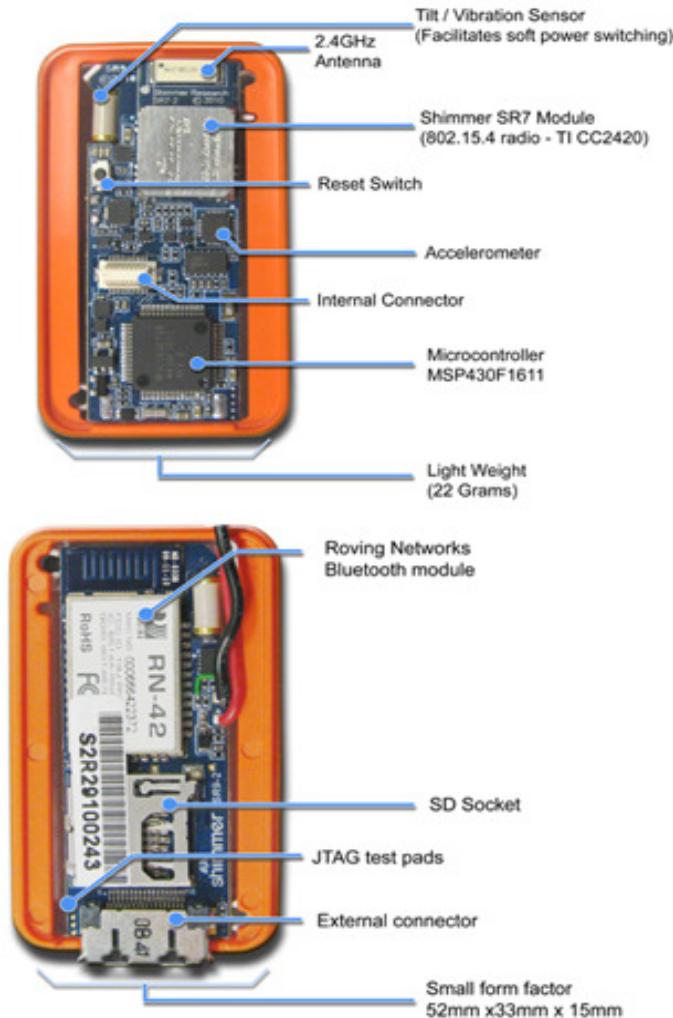


Figure 4.4: Internal infrastructure of shimmer sensor device [105]

A Shimmer docking station is used for programming the Shimmer sensor device. The controls of a Shimmer docking station are also illustrated in Figure 4.5. The dock is connected with via USB cable to any PC or laptop. The driver of the dock gets loaded automatically on the server having windows system. A message in the lower right of screen is displayed identifying the dock when the dock is plugged in to the server. A micro SD card is inserted into a shimmer dock before connecting it to a PC or laptop. The Shimmer dock contains a USB reader, a GPS module, a dual Universal Asynchronous Receiver/Transmitter (UART) and a flash media controller (SMSC USB2640). The power/reset button of the dock aids to switch on power, or reset the Shimmer sensor device, or power-off the Shimmer device. The GPS enable button toggles between GPS and host PC UART connection to Shimmer. The communication interface is NMEA 0183 v3.01and MTK NMEA command protocols with 115,200 bps,

8 data bits, no parity, and 1 stop bit. A Shimmer sensor device mounted on a dock is illustrated in Figure 4.6.

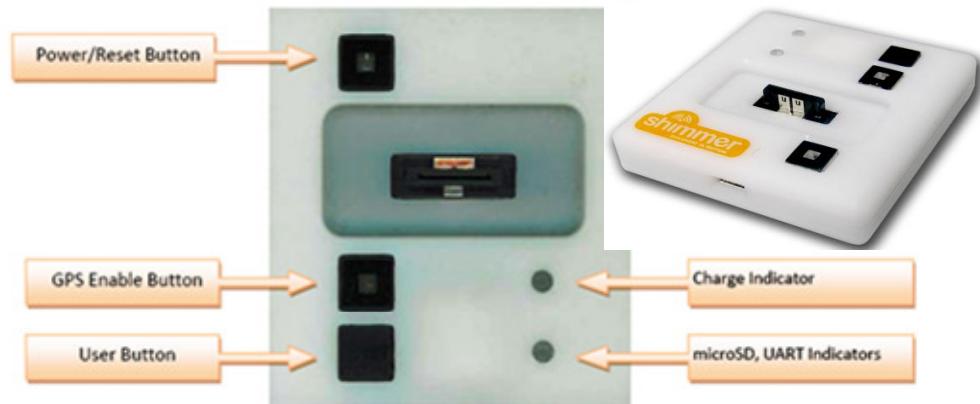


Figure 4.5: Shimmer docking station with controls [105]



Figure 4.6: Shimmer sensor device mounted on the Shimmer dock [105]

4.3.2 Technical Specification of the Shimmer Sensor Device

The Shimmer wireless sensor platform comprises of a Shimmer unit. The unit consists of a sensor device and a Shimmer dock. The technical specifications of the sensor device are provided in Table 4.1. The sensor device contains MSP430 Microcontroller and an accelerometer. The technical specifications of the accelerometer are provided in Table 4.2.

Table 4.1: Technical specifications of the Shimmer sensor device [105]

Features	Description
Size	53mm x 32mm x 25mm
Microcontroller	MSP430F1611
SD Card	Integrated 2 GB micro SD card slot
Radios	Bluetooth – Roving Networks RN-42; and 802.15.4 Radio – TI CC2420
Frequency Band	2.4GHz
Integrated	3 Axis Accelerometer – Free scale MMA7361; and Tilt/Vibration Switch – Signal Quest SQ-SEN-200

Table 4.2: Technical specifications of the Accelerometer [105]

Features	Description
Range	$\pm 1.5g$, $\pm 6g$
Sensitivity	800mV/g at 1.5g
Non-linearity	$\pm 1\%$ FSO
Cross-axis	$\pm 5\%$
Operating temperature	40°C to 85°C
Supply voltage	2.2V to 3.6V

4.4 INSTALLATION, CONNECTION AND DATA GENERATION

4.4.1 Sensor Installation and Configuration

The data collection from the sensor is accomplished by installing a ShimmerConnect software in the computer. In order to make a connection between the sensor and computer, a driver for Shimmer USB dock is also installed. Once the driver and software are installed, the shimmer device has been connected with the Dock, and the Dock is connected with the computer via USB. Shimmer sensor can be configured from tools menu of shimmer connect interface window and configuration in accelerometer mode is shown in Figure 4.7 to generate vibration acceleration data.

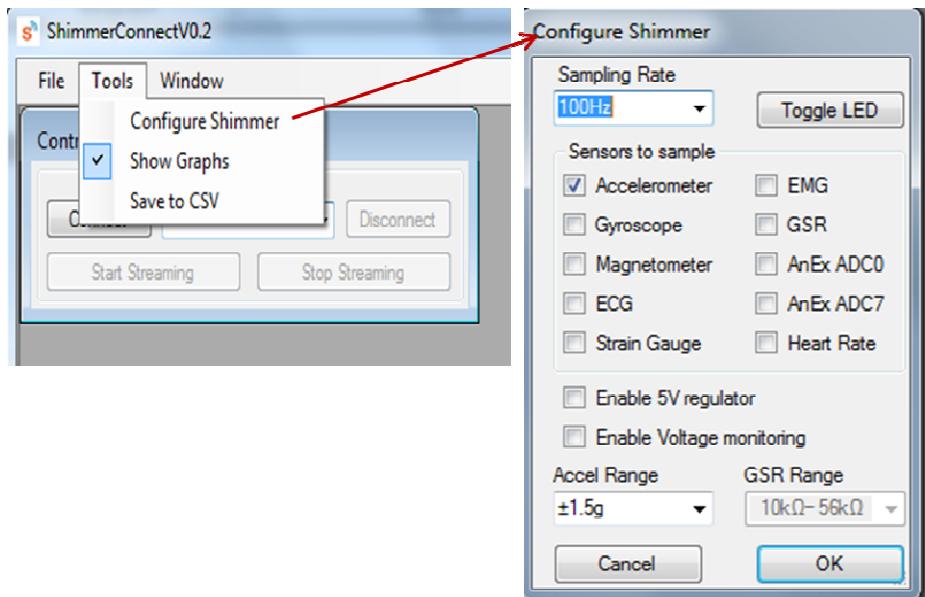


Figure 4.7: Configuring the shimmer sensor

4.4.2. Connection Establishment

The connection between the computer and Shimmer sensor is established by Bluetooth pairing. Figure 4.8 displays the wireless connection between Shimmer sensor and the computer as indicated by the green light signal. This process allows removal of the Shimmer sensor device from the dock because of the established Bluetooth connection between the server and the Shimmer device.



Figure 4.8: Shimmer sensor connected via Bluetooth with or without the docking station

An appropriate port number, sampling rate, sample type and sample range are selected and the front panel of the ShimmerConnect starts to stream vibration acceleration data after clicking the ‘start streaming’ button (Figure 4.9).

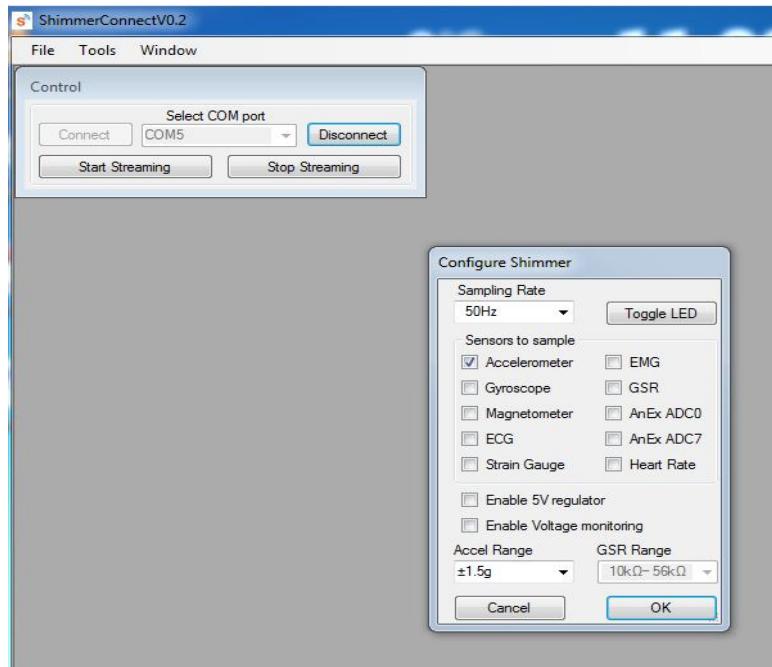


Figure 4.9: ShimmerConnect front panel ready for data collection

4.4.3 Data Generation Test

Data generation test was carried out in order to check that any movement of the device is picked up by the accelerometer sensor and the corresponding data are sent to the computer via BlueTooth connection. The shimmer sensor is moved manually and vibration acceleration data in X, Y and Z directions generated by the sensor is transferred to the computer through Bluetooth® connection. The graph shown in Figure 4.10 displays the vibration acceleration characteristics in different directions. The data for X, Y, and Z directions are also saved in the computer as .csv file format (Table 4.3).

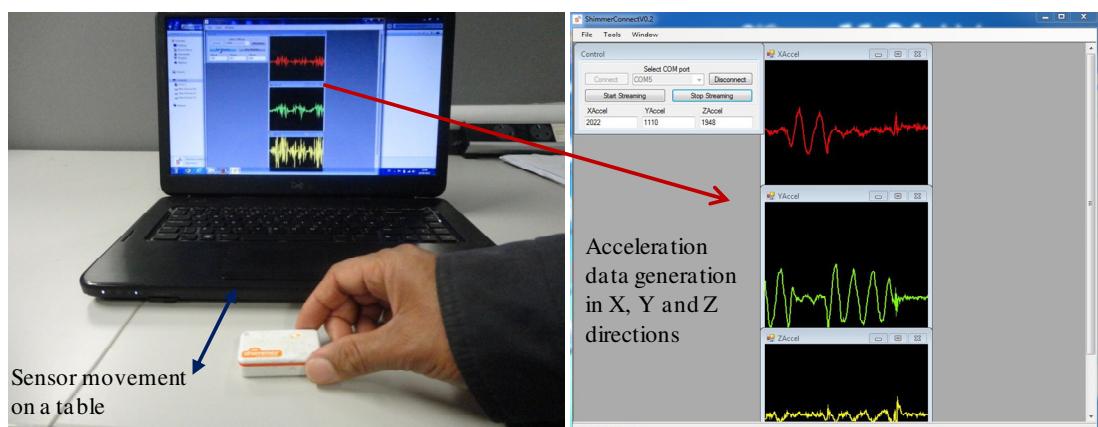


Figure 4.10: Graph generation from vibration acceleration data generated by the sensor

Table 4.3: Vibration Accelerometer data saved in Tabular format in Excel

TimeStamp	XAccel	YAccel	ZAccel
17953	1975	1184	2142
18593	1971	1181	2067
19233	1955	1122	2123
19873	1920	1161	2092
20513	1957	1155	2078
21153	1930	1214	2160
21793	1971	1162	2081
22433	1960	1226	2104
23073	1933	1195	2151
23713	1951	1154	2125
24353	1968	1140	2124
24993	1954	1168	2075
25633	1951	1163	2155
26273	2001	1184	2135
26913	1924	1152	2111
27553	1975	1182	2126

4.4.4 Calibration of Shimmer Sensor Device

Like any other device, the accelerometer integrated in the Shimmer device needs to be calibrated before its use in experimental work in order to get accurate data. The Shimmer sensor device is calibrated with respect to gravitational vibration acceleration ('g'). The X-axis of the Shimmer sensor is placed along the axis of the positive gravitational vibration acceleration ('g') and negative 'g'-axis. The two-directional movement of the Shimmer accelerometer is performed for two seconds and the corresponding data are recorded. These data are saved in two .csv files. The .csv files are named as 'X plus up' and 'X plus down' (Appendix A). Similarly the Y and Z axes of the Shimmer accelerometer are placed along +'g'-axis and -'g'-axis. Figure 4.11 presents the different orientations of the Shimmer sensor during the calibration procedure.



Figure 4.11: Orientations of Shimmer sensor during calibration

As that of X-axis all data are saved in different .csv file names. The recorded data are utilised in the following set of equations in order to obtain the calibrated vibration accelerations in X, Y and Z axes of the Shimmer device.

$$\text{Calibrated acceleration on } X - \text{axis} = X_{\text{accel}} = \frac{X_{\text{Raw}} - X_{\text{Offset}}}{X_{\text{Scale}}} \quad (4.1)$$

$$\text{Calibrated acceleration on } Y - \text{axis} = Y_{\text{accel}} = \frac{Y_{\text{Raw}} - Y_{\text{Offset}}}{Y_{\text{Scale}}} \quad (4.2)$$

$$\text{Calibrated acceleration on } Z - \text{axis} = Z_{\text{accel}} = \frac{Z_{\text{Raw}} - Z_{\text{Offset}}}{Z_{\text{Scale}}} \quad (4.3)$$

In equation (4.1), the offset value (X_Offset) is referred to as the average sensor reading for zero-'g' values in the two opposite directions of the X-axis of the Shimmer sensor. Considering calibration of the Shimmer along X axis, zero-'g' values are referred to those values along both the directions of the other two axes, e.g., Y and Z axes, where the values of 'g' are considered as zero. Similarly, offset values in the Y and Z

directions (Y_Offset and Z_Offset) are the average sensor readings for zero-‘g’ values in the two opposite directions of the Y and Z axes of the Shimmer sensor respectively.

Scale value alongside the X axis of the Shimmer sensor is defined as the average sensor reading for +‘g’ values in the two opposite directions of the X-axis of the sensor. Similarly, scale values in the Y and Z directions of the Shimmer sensor are the average sensor reading for +‘g’ values in the two opposite directions of the Y and Z axes of the sensor respectively.

Eq. 4.1, Eq. 4.2 and Eq. 4.3 contain raw values in the X, Y and Z directions of the Shimmer sensor. The recorded data in the .csv file format during the two-directional movement of the Shimmer accelerometer for any of the axes, e.g., X, Y and Z, are referred as raw values (X_Raw, Y_Raw and Z_Raw). The data provided in the .csv file named as ‘X plus down’ and ‘X plus up’ are considered during the calculation phase of the calibration procedure. The average vibration acceleration along X-axis (for negative ‘g’ value) as obtained from the ‘X plus down’ file is 1747.67. Similarly, the average vibration acceleration alongside X-axis (for +‘g’ value) as obtained from the ‘X plus up’ file is 2268.68. The calculation of the calibrated vibration acceleration alongside X-axis of the Shimmer sensor for +‘g’ value is appended below:

$$X_{Scale\,value} = \frac{2268.68 + (-1747.67)}{2} = 260.51$$

Four zero-‘g’ values are found from the two directional average Y and Z vibration acceleration data of the sensor considering ‘g’. Zero-‘g’ values are found from the .csv file for Y and Z axes vibration acceleration data while their positions are considered on the basis of ‘g’.

The zero-‘g’ value of X-axis vibration acceleration (X_{accel}) found from the .csv file named ‘Y plus down’ is 2005.08. The .csv file named ‘Y plus up’ provides zero-‘g’ value for X_{accel} which is 2006.60. In the similar manner the zero-‘g’ value for X_{accel} are obtained from the .csv files named ‘Z plus down’ and ‘Z plus up’. These values are 2011.62 and 1996.09 respectively. These zero-‘g’ values are required in order to compute X_Offset values. Therefore, the computed X_Offset value is:

$$X_{Offset} = \frac{2005.08 + 2006.60 + 2011.62 + 1996.09}{4} = 2004.85$$

The X_{accel} value is computed considering X_{Raw} , X_{Offset} and X_{Scale} values from Eq. 4.1.

$$Calibrated\ X_{accel} = \frac{X_{Raw} - X_{Offset}}{X_{Scale}} = \frac{1740.03 - 2004.85}{260.51} = -1.02$$

In order to calibrate the Shimmer sensor device with respect to the ‘g’ value, X_{accel} value is multiplied with ‘g’. Thus, the calibrated vibration acceleration of the sensor with respect to the ‘g’ value is ($X_{accel} * 9.80665$) = (-1.02 * 9.80665) = -9.97 m/s². This value corresponds to the sensor position in –‘g’ direction along X axis.

Same procedure is followed for calculating the Y_{accel} and Z_{accel} values. Table 4.4 provides a summary of the calibrated vibration acceleration information with different sensor positions. It can be noticed that the calculated average ‘g’ values (9.82 for X axis, 9.80 for Y axis and 9.81 for Z axis) are very close to the gravitational vibration acceleration 9.81 m/s².

Table 4.4: Summary of sensor calibration data

Sensor position	Avg X_{accel}	Avg Y_{accel}	Avg Z_{accel}	Avg calibrated X_{accel}	Avg X_{accel} (m/s ²)	Avg calibrated Y_{accel}	Avg Y_{accel} (m/s ²)	Avg calibrated Z_{accel}	Avg Z_{accel} (m/s ²)
- ‘g’ direction along X axis	1747.67	2114.178	2008.726	-0.989	-9.697	-0.012	-0.117	0.018	0.172
+ ‘g’ direction along X axis	2268.69	2119.178	2005.836	1.013	9.939	0.008	0.076	0.006	0.059
- ‘g’ direction along Y axis	2005.08	18863.493	1993.301	0.001	0.004	-1.002	-9.830	-0.044	-0.435
+ ‘g’ direction along Y axis	2006.60	2369.151	2009.603	0.006	0.061	0.996	9.761	0.021	0.207
- ‘g’ direction along Z axis	2011.62	2121.616	1750.027	0.026	0.251	0.017	0.171	-1.021	-10.01
+ ‘g’ direction along Z axis	1996.1	2114.041	2248.055	-0.034	-0.334	-0.012	-0.122	0.979	9.598

4.5 SET UP DEVELOPMENT WITH A TOY CAR

4.5.1 Toy Car Set-up Concept

The main concept for developing an experimental set-up is to measure speeds of rotating body with the sensor and transfer the data in the computer. Therefore, the experimental set-up would be a source of speed data, which can be varied randomly. The first concept for developing such facility was started with a remote controlled toy

car. The shimmer device was attached with the toy car to measure the speed of the car by sensing the vibration at different speeds (Figure 4.12).

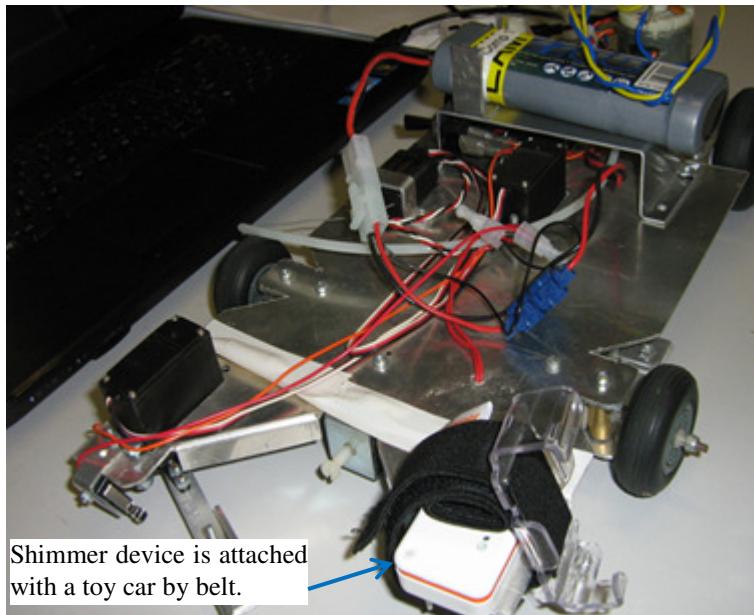


Figure 4.12: Shimmer sensor attached with a remote controlled toy car

4.5.2 Tests with the Toy Car

After attaching the shimmer sensor with the toy car, a wireless connection is established between the sensor and the laptop. The toy car was moved with the remote controller and vibration acceleration data was generated due to vibration from the toy car movement. A wireless connection is established between the sensor and the laptop via Bluetooth. Figure 4.13 demonstrates the transfer of data from the sensor to the laptop presents in a graphical format with respect to the elapsed time. Furthermore, the data is automatically saved in the laptop as tabular format (csv file).

However, this set-up could not be used for random speed data generation from the relationship between vibration and speed of the toy car. Firstly, the toy car movement could not be controlled on a straight path. Furthermore, the toy car speed could not be varied randomly or maintained at known speed. Therefore, no relationship could be established between the car speed and accelerometer output data.

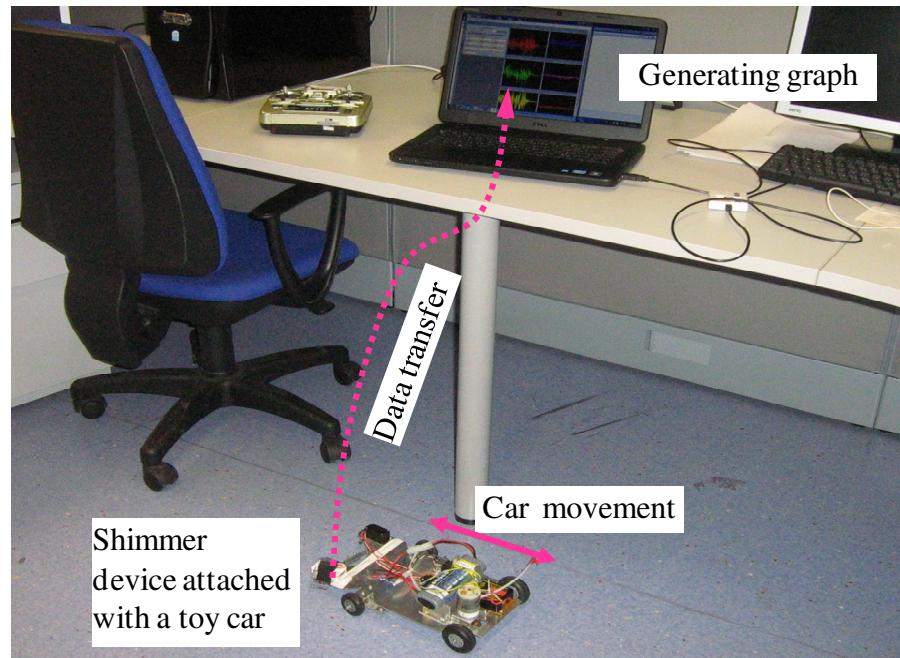


Figure 4.13: Shimmer sensor is generating vibration acceleration data while moving with car

4.6 SET-UP DEVELOPMENT WITH A MECHANICAL WHEEL

4.6.1 Mechanical Wheel Set-up Concept

In order to generate random speed data a new mechanical wheel concept has been developed. A schematic diagram of the concept has been presented in Figure 4.14. A motor drives a wheel mounted on a shaft. The wheel is considered to be a vehicle. A speed controller controls the wheel speed and an integrated tachometer measures the wheel speed in rpm. The sensor is attached with the wheel. By setting the wheel at known speeds, a relationship between wheel speed and vibration of the wheel can be established. This way the wheel speed can be measured indirectly by the sensor and transferred to the computer wirelessly. According to the schematic diagram, a real set-up has been developed as shown in Figure 4.15.

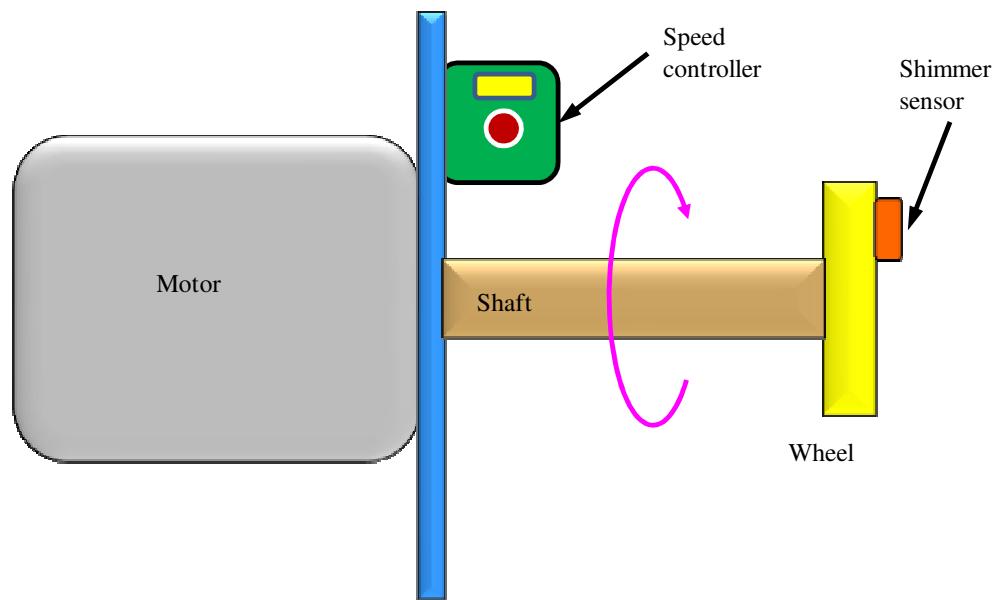


Figure 4.14: Schematic diagram of a mechanical wheel based set-up

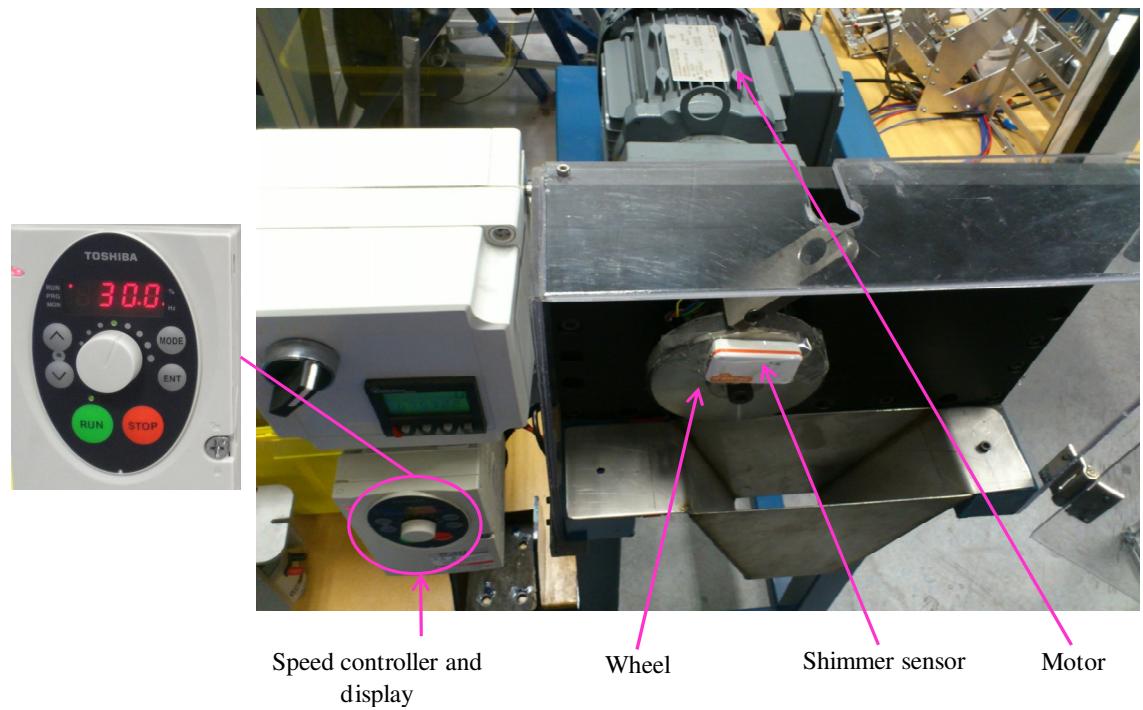


Figure 4.15: Photograph of mechanical wheel based set-up

4.6.2 Connection Establishment

The Shimmer sensor is connected with the laptop through Bluetooth connection to transfer the wheel vibration data wirelessly to the laptop (Figure 4.16). All the vibration data are stored in the computer for further processing.

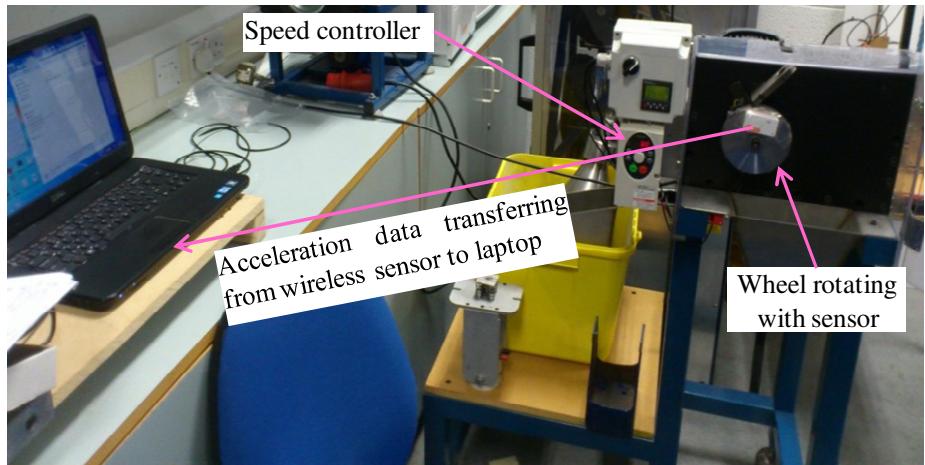


Figure 4.16: Connection establishment between sensor and server computer.

4.6.3 Data Transfer and Storing

When the Shimmer sensor streams the vibration acceleration data, a real-time vibration acceleration plot is obtained through SimmerConnect front panel. This real-time plot for vibration acceleration in three Cartesian coordinates is illustrated in Figure 4.17. Table 4.5 illustrates a sample vibration acceleration data obtained from the Shimmer sensor device set at 100 Hz and saved on the computer when the wheel is running at 20 rpm.

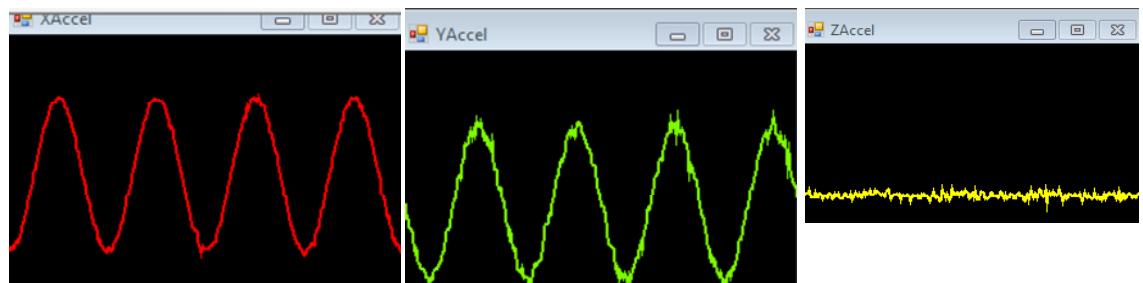


Figure 4.17: Real-time vibration acceleration plot for the wheel rotating at 20 rpm

Table 4.5: Vibration Acceleration data obtained from Shimmer sensor at 20 rpm wheel speed

TimeStamp	XAccel	YAccel	ZAccel		TimeStamp	XAccel	YAccel	ZAccel
58369	3243	2281	1762		833	1601	2746	1818
58689	3259	2334	1775		1153	1516	2566	1719
59009	3219	2472	1804		1473	1420	2613	1750
59329	3154	2492	1774		1793	1363	2562	1769
59649	3186	2561	1771		2113	1363	2549	1804
59969	3124	2693	1785		2433	1286	2452	1762
60289	3066	2761	1755		2753	1283	2335	1789
60609	2946	2767	1783		3073	1273	2281	1791
60929	2931	2848	1796		3393	1265	2172	1781
61249	2913	2924	1746		3713	1279	2052	1765
61569	2799	3073	1754		4033	1300	1931	1794
61889	2697	3056	1793		4353	1338	1874	1823
62209	2674	3082	1848		4673	1354	1839	1805
62529	2582	3159	1832		4993	1369	1772	1786
62849	2426	3091	1799		5313	1397	1571	1748
63169	2417	3080	1766		5633	1473	1595	1743
63489	2281	3089	1897		5953	1563	1506	1869
63809	2246	3000	1790		6273	1612	1469	1852
64129	2111	3015	1714		6593	1625	1346	1824
64449	1962	3102	1745		6913	1711	1314	1844
64769	1933	3084	1829		7233	1799	1294	1815
65089	1851	3011	1794		7553	1901	1220	1788
65409	1759	2935	1771		7873	1978	1217	1771
193	1657	2890	1759		8193	2038	1192	1765
513	1581	2855	1845		8513	2128	1060	1783

4.7 WHEEL SPEED MEASUREMENT WITH THE SENSOR

The wheel speed is measured with the sensor using a calibration technique. The purpose of the calibration is to obtain a feasible mathematical relationship between vibration acceleration of the wheel obtained from the Shimmer device and the rotational speed of the wheel. The general procedure for wheel seed measurement is presented in Figure 4.18.

The wheel can have a wide range of rotational speeds controlled by the speed controller connected to the motor. In order to measure the wheel speed with the sensor, a set of rpm was considered (10, 15, 20, 25, 30, 35, 40, 45, 50 rpm). Vibration Acceleration data in X, Y and Z directions were generated for different known wheel speeds. Initially, the sensor device is set at 100 Hz sampling rate and the wheel is run for 30 seconds at each rpm to collect substantial amount of data. The data were saved in the computer as CSV file after clicking “stop streaming” button on the ShimmerConnect interface page. Average vibration acceleration was calculated for each direction and for all speeds.

From the average vibration accelerations in each direction resultant vibration acceleration was calculated for all speeds using Eq. 4.4 [106]. Further details of the resultant vibration acceleration calculation can be found in Appendix B.

$$R_{Accel} = \sqrt{X_{Accel}^2 + Y_{Accel}^2 + Z_{Accel}^2} \quad \text{---(4.4)}$$

Where, X_{Accel} = vibration Acceleration in X direction, Y_{Accel} = vibration Acceleration in Y direction and Z_{Accel} = vibration Acceleration in Z direction

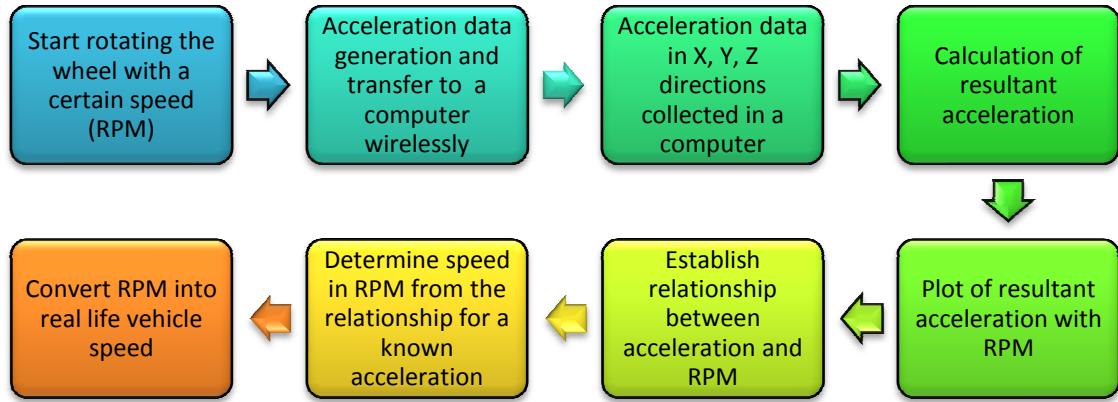


Figure 4.18: Shimmer sensor calibration process

The resultant vibration acceleration for different rotational speeds of the wheel is presented in Table 4.6.

Table 4.6: The average resultant vibration acceleration of the wheel at different rpm

Speed (rpm)	X _{accel} (Avg)	Y _{accel} (Avg)	Z _{accel} (Avg)	Resultant acceleration
10	1845.06	2130.33	1796.48	3342.14
15	2111.56	2119.93	1789.07	3486.20
20	2267.41	2125.68	1782.80	3583.02
25	2477.74	2116.99	1781.09	3713.91
30	2732.15	2110.57	1778.12	3883.40
35	3033.11	2105.47	1774.08	4096.36
40	3323.53	2094.16	1769.19	4308.29
45	3591.63	2089.88	1760.54	4512.97
50	3816.85	2076.37	1754.15	4685.80

The resultant vibration acceleration data were plotted against different wheel speeds. Figure 4.19 displays a plot between the resultant vibration acceleration calculated from

the vibration acceleration components obtained from the Shimmer device and the wheel speed in rpm. The vibration acceleration data obtained from the Shimmer device is represented by an arbitrary unit. It can be seen from the graph that the resultant vibration acceleration increases proportionally with the rotational speed of the wheel.

Therefore, the relationship between the resultant vibration acceleration of the wheel while rotating at a certain rpm can be represented by a linear equation (Eq. 4.5)

$$R_{Accel} = 31.655N + 3024.8 \quad \text{---(4.5)}$$

Where R_{Accel} represents the resultant vibration acceleration and N represents wheel speed in rpm.

For a known value of resultant vibration acceleration, the wheel speed can be calculated by Eq. 4.6

$$N = \frac{R_{Accel} - 3024.8}{31.655} \quad \text{---(4.6)}$$

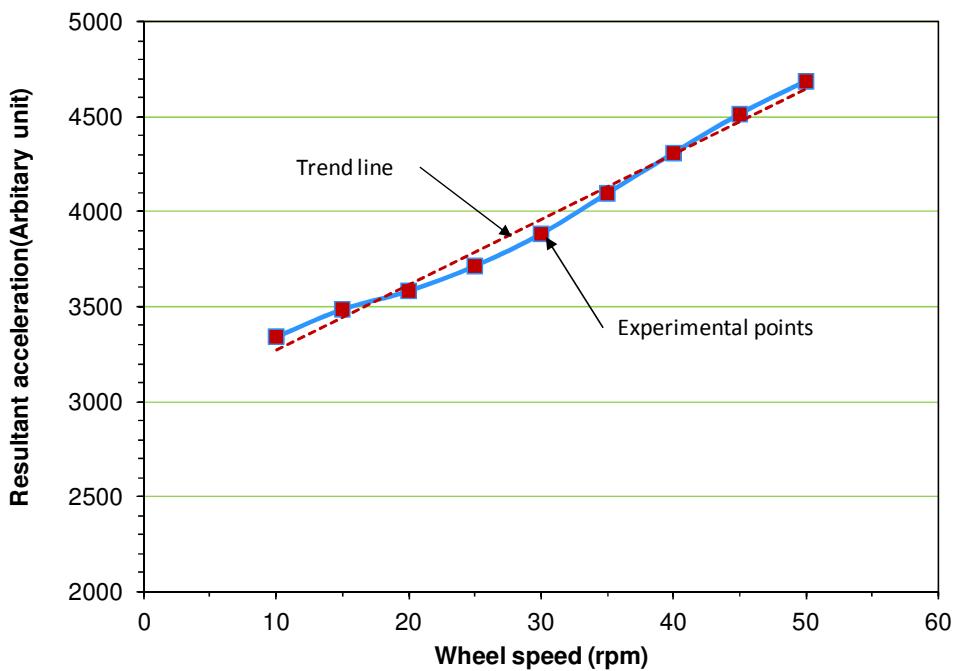


Figure 4.19: The resultant vibration acceleration vs. different rotational speeds (rpm) of the wheel

Similar relationship can also be seen from the snapshots of selected real-time vibration acceleration plots for different wheel speeds (Figure 4.20). The figure shows that with

the increase of wheel speed the wavelengths in the vibration acceleration gradually decreases.

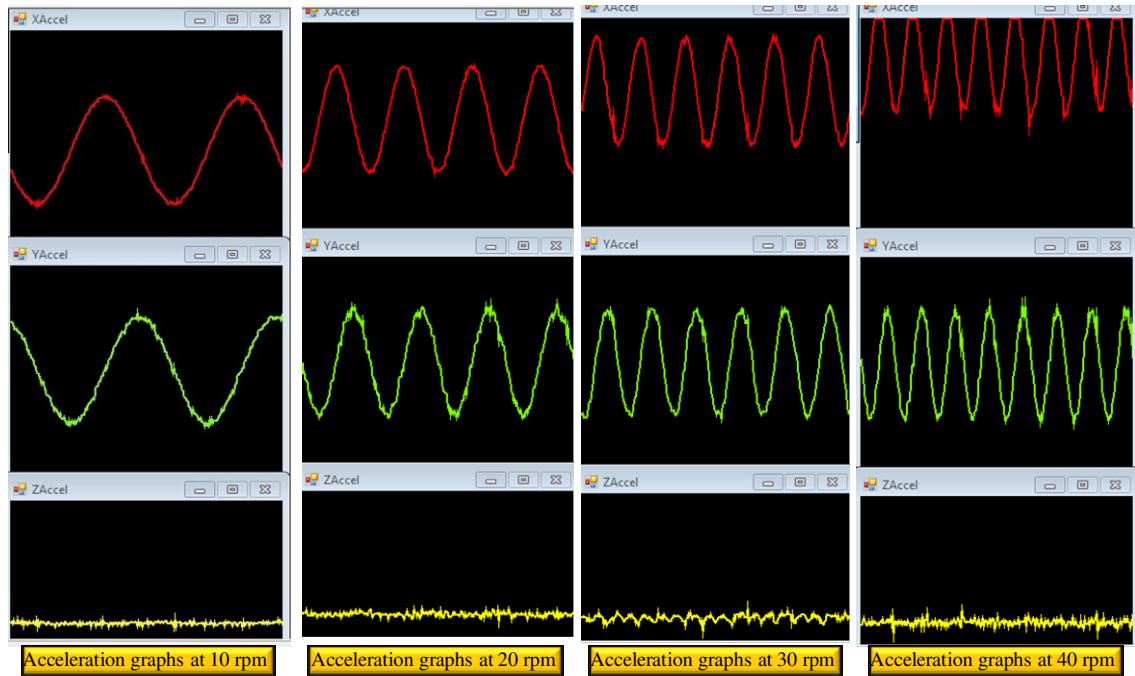


Figure 4.20: Vibration Acceleration graphs in X, Y and Z directions at different rotational speeds of the mechanical wheel

The relationship was validated by calculating the speeds using the equation and comparing that against the actual wheel speeds. Table 4.7 presents a comparison between the actual and calculated speeds. The reasons for differences could be due to some unavoidable experimental error and/or considering the average vibration acceleration values in the calculation.

Table 4.7: Comparison between actual and measured wheel speeds

Test no	Original Speed (rpm)	Calculated Speed (rpm)	Deviation	Percentage of deviation
1	10	10.03	0.03	0.3%
2	15	14.58	-0.42	-2.8%
3	20	17.63	2.37	11.85%
4	25	21.77	3.23	12.92%
5	30	27.12	2.88	9.6%
6	35	33.85	-1.15	-3.29%
7	40	40.55	0.55	1.38%
8	45	47.01	2.01	4.47%
9	50	52.47	2.47	4.94%

The rotational speed of the wheel does not represent a real life vehicle speed. Therefore, first the rotational speed has been converted in to linear speed using the following formula (Eq. 4.7).

$$\text{Simulated vehicle speed (km/hr)} = \frac{2\pi N}{60} \times r \times \frac{1}{1000} \times 3600 \text{---(4.7)}$$

Where, r is the radius of the wheel, which is 0.055 m. For example, when the rotational speed is 10 rpm the simulated vehicle speed is 0.207339 km/h, which is a very small value. Therefore, it has been assumed that 10 rpm (0.207339 km/h) is equivalent to 20 km/h and a multiplying factor can be calculated as $20/0.207339=96.46$. This factor converts all the rotational speeds to a equivalent real life vehicle speed. Therefore, the final siimulated vehicle speed is represented by the following equation.

$$\text{Final simulated vehicle speed (km/hr)} = \frac{2\pi N}{60} \times r \times \frac{1}{1000} \times 3600 \times 96.46 \text{---(4.8)}$$

Where 96.46 is a constant.

4.8. EXPERIMENTATION FOR SPEED DATA COLLECTION

In the previous section, it has been demonstrated that wheel speed can be measured by the Shimmer sensor using a calibration technique. Therefore, the wheel speed now can be varied randomly within a certain range, and the speeds will be measured by the sensor after transferring the vibration data to a computer. The speeds of the wheel can be considered as a vehicle running on a particular road.

4.8.1 Experimental Condition

The Shimmer sensor is attached to the mechanical wheel at the same position during calibration. Figure 4.21 presents the experimental set-up used for simulating the condition of vehicle speed monitoring.

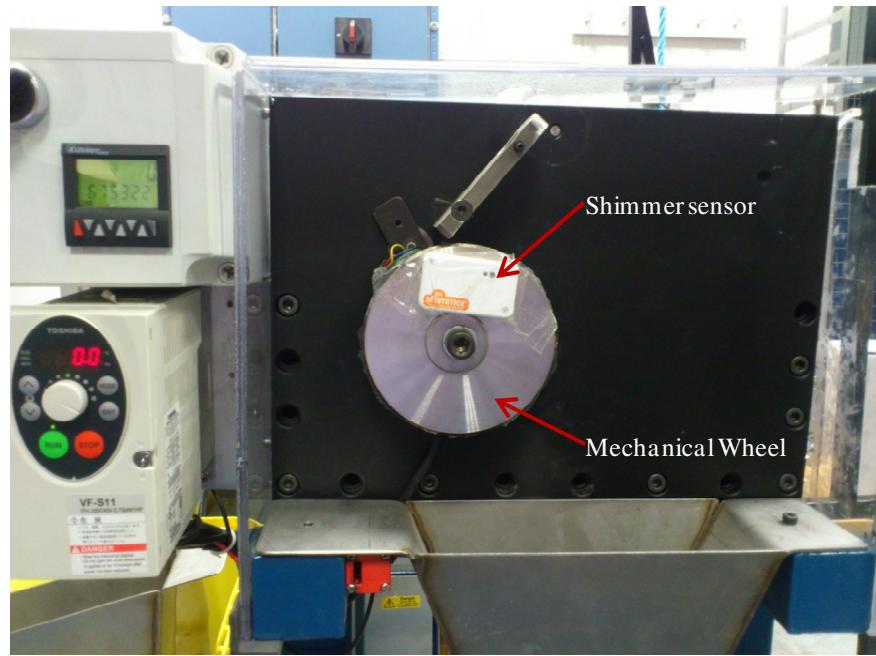


Figure 4.21: Experimental set-up for speed monitoring using wireless sensor

A connection is established between the sensor and a computer through BlueTooth. The ShimmerConnect interface page was configured for collecting data in accelerometer mode. Table 4.8 shows the details of the conditions employed for the experiments.

Table 4.8: Experimental conditions for data collection

Parameter	Description
Wireless sensor	Shimmer
Wheel speed range	0-55 rpm
Sampling rate	100 Hz
Sensor setting	Accelerometer
Acceleration range	± 1.5 g
Axis	X, Y and Z acceleration
File format	.csv

4.8.2 Experimental Procedure

The following procedure was followed to generate wheel speed data from the experimental set-up and transferring the data to the computer for converting into vehicle speed data.

1. At the stationary condition of the wheel, the vibration values in X, Y and Z directions are continuously transferred to the ShimmerConnect interface page in the computer

2. The wheel is rotated at a random value for few minutes
3. The values in the ShimmerConnect will increase due to increased vibration of the wheel while rotating
4. The data were saved in the computer as CSV file after clicking “stop streaming” button on the ShimmerConnect interface page.
5. Average vibration acceleration was calculated for each direction.
6. From the average vibration accelerations in each direction, resultant vibration acceleration was calculated using Equation 4.4.
7. The wheel speed was calculated using equation 4.6 and then converted to realistic vehicle speed using Equation 4.8.
8. This procedure was repeated by setting the wheel speed to another random value.

Experiments have been carried out to collect data in order to simulate four different scenarios: single vehicle running inside a city/town and multiple vehicles running on highway. Further details about the application of experimental data are presented in Chapters 5 and 6.

Chapter 5: Software Development for Intelligent Traffic Speed Monitoring System

5.1. INTRODUCTION

This chapter presents software development steps using Java Socket programming for the proposed system of automatic data management in traffic monitoring and controlling. First, the concept of data management is briefly outlined with an algorithm. Socket programming and establishment of connection between a client and a server are also discussed. Finally, the socket and client algorithms for single and multiple vehicle scenarios are presented with detailed description of each step in the algorithms.

5.2 SYSTEM DEVELOPMENT CONCEPT

The general concept of the proposed system of traffic monitoring has been presented in Figure 5.1. In this system, it has been assumed that the speed of a vehicle can be measured using an accelerometer based wireless sensor when placed within a vehicle. The vibration data of the vehicle are transferred wirelessly to a computer known as a server. These data are then processed in the server to convert to relevant vehicle speed data and transferred wirelessly to a desktop computer, known as client. The system is designed to continuously monitor the speed data displayed in a graphical User Interface (GUI). If the data value is greater than a pre-set speed limit, a report is generated with the details of the car and its speed at that instantaneous time. This report is to be stored in the server and it to be sent to a wireless printer for printing. Therefore, the software system is to manage the data related to road traffic monitoring automatically and wirelessly.

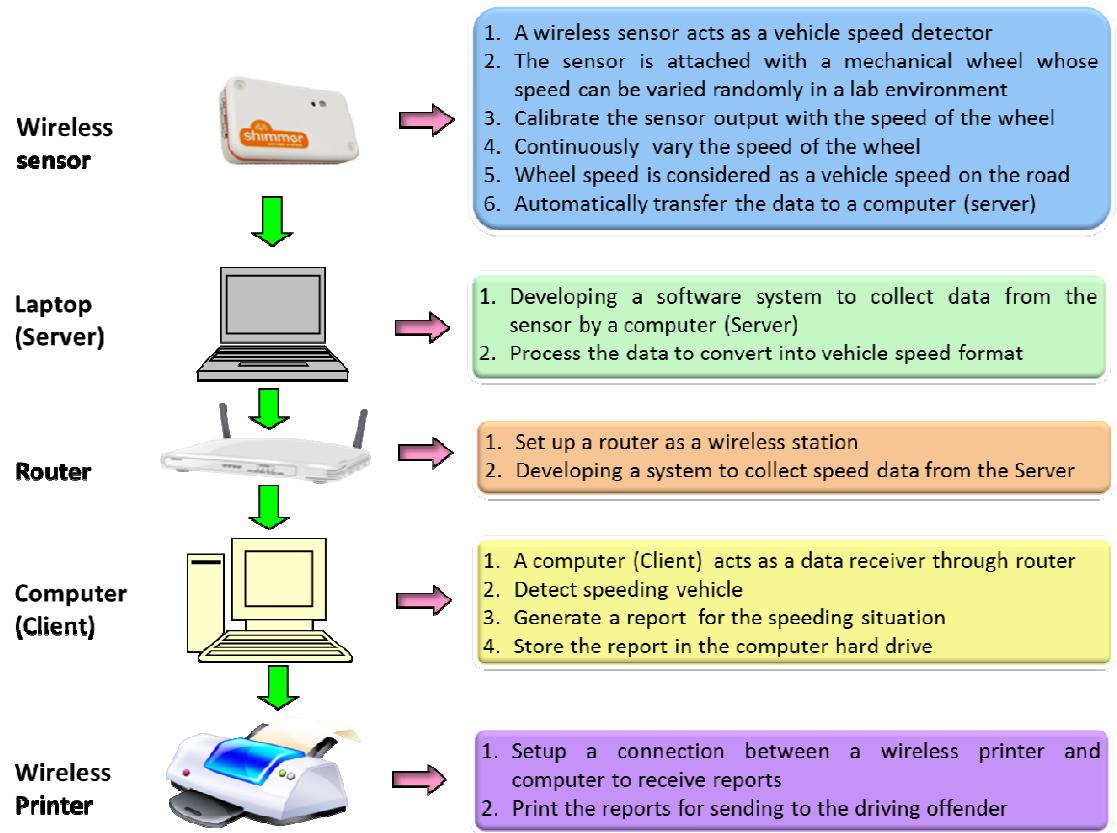


Figure 5.1: System concept for developing intelligent traffic monitoring software

In order to develop the software for intelligent traffic monitoring system a general algorithm has been developed and presented in Figure 5.2.

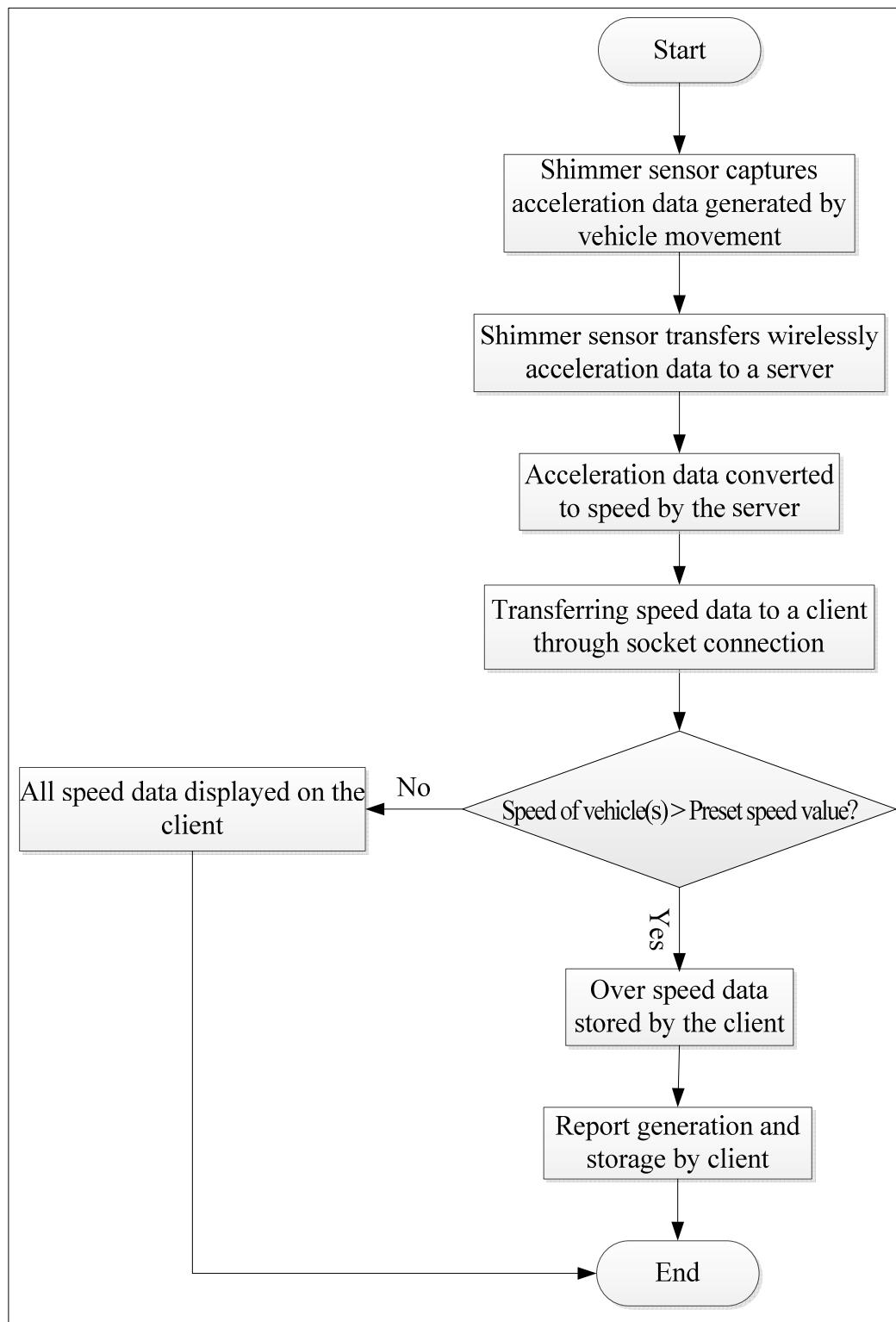


Figure 5.2: A general algorithm for developing traffic monitoring system

The wireless traffic monitoring system considers a wheel as a representation of a car/vehicle within a laboratory environment. A Shimmer device is mounted on the wheel. The Shimmer device acts as an accelerometer to measure the wheel vibration when the wheel rotates and transfers the vibration acceleration data to the server

wirelessly. These data are communicated to the client wirelessly using Java socket programming. Figure 5.3 presents the server and client used in this work.

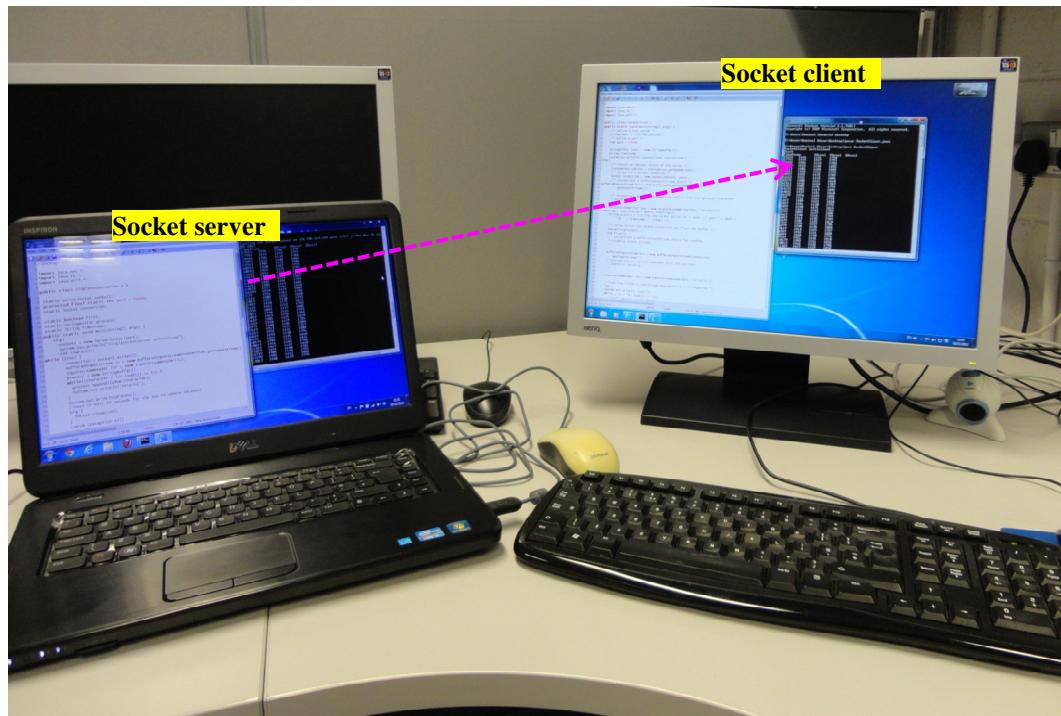


Figure 5.3: Communication between client and server

The vibration acceleration data are processed by the server through a set of Java codes (server program) to convert them as vehicle speed data. A maximum limit of vehicle speed is set in the socket client program. When a vehicle crosses the maximum set speed limit, the system compares the vehicle speed with the pre-set speed limit. When the vehicle speed is less than the pre-set limit, the speed data are stored in the server and displayed in the client. If the vehicle speed is greater than the pre-set speed limit, the socket client program stores the speed data in the client. A report about the speeding vehicle is also generated and subsequently stored in the client computer.

5.3 COMMUNICATION WITH SOCKET PROGRAMMING

The server and the client are two different systems accomplishing different functions. The server provides services to other systems within a network while the client uses remote services provided by the server. Servers may be of different types. The Socket helps to communicate between the server and the client as illustrated in Figure 5.4.

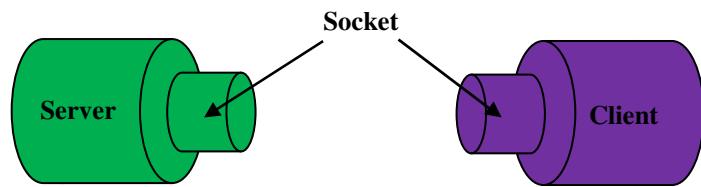


Figure 5.4: Socket in server and client [107]

The socket is known as one end-point of a two-way communication link between two programs running on a network using server and client [108]. The server creates a socket object on its end of the communication once the connection is made. The client and server can now communicate by writing to and reading from the socket. In socket programming, the socket classes are used for building the connection between a client program and a server program. Two classes are provided by the java.net package, e.g., `Socket` and `ServerSocket`. `Socket` class implements the client side of the connection and while the `ServerSocket` class provides a mechanism for the server program in order to listen for the client and establish connections with it. The saved vibration data generated by sensor device can be sent from the server to the client by the following steps (Figure 5.5).

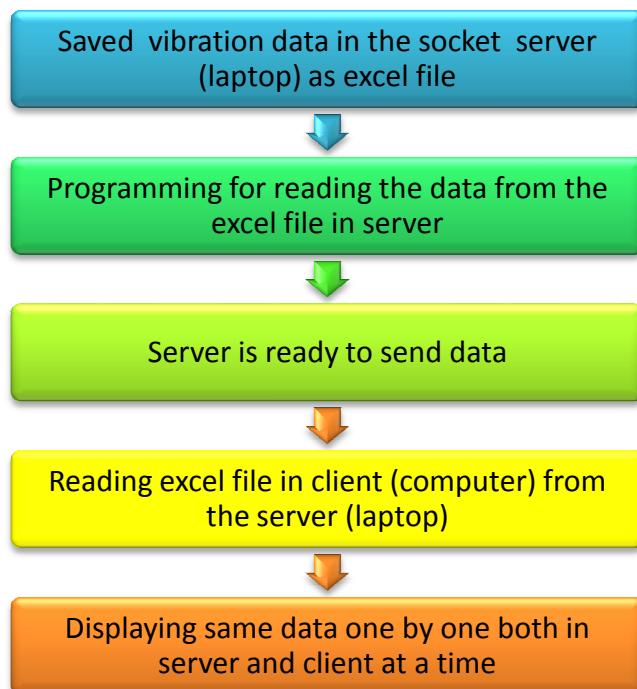


Figure 5.5: Steps of transferring data from server to client using socket programming

5.4 SOCKET SERVER ALGORITHM FOR SINGLE MOVING VEHICLE

In this section, an algorithm has been developed for a database server as shown in Figure 5.6. The socket server algorithm includes different classes in order to obtain a port and communicate subsequently with the client requests for a single moving vehicle on the road. Several steps are included in the socket server algorithm and the operation of each step is coded with Java language. The steps of the algorithm with corresponding java codes have been presented in the following sections.

5.4.1 Setting up Data Channel

The first step is to declare Seversocket and Socket for the server and the client respectively. This is achieved by the following codes

```
static Socket connection=null;  
static ServerSocket socket1;  
int character;
```

Declaration of the socket and server socket follows declaration of a port on the server using the following code:

```
protected final static int port = 19999;
```

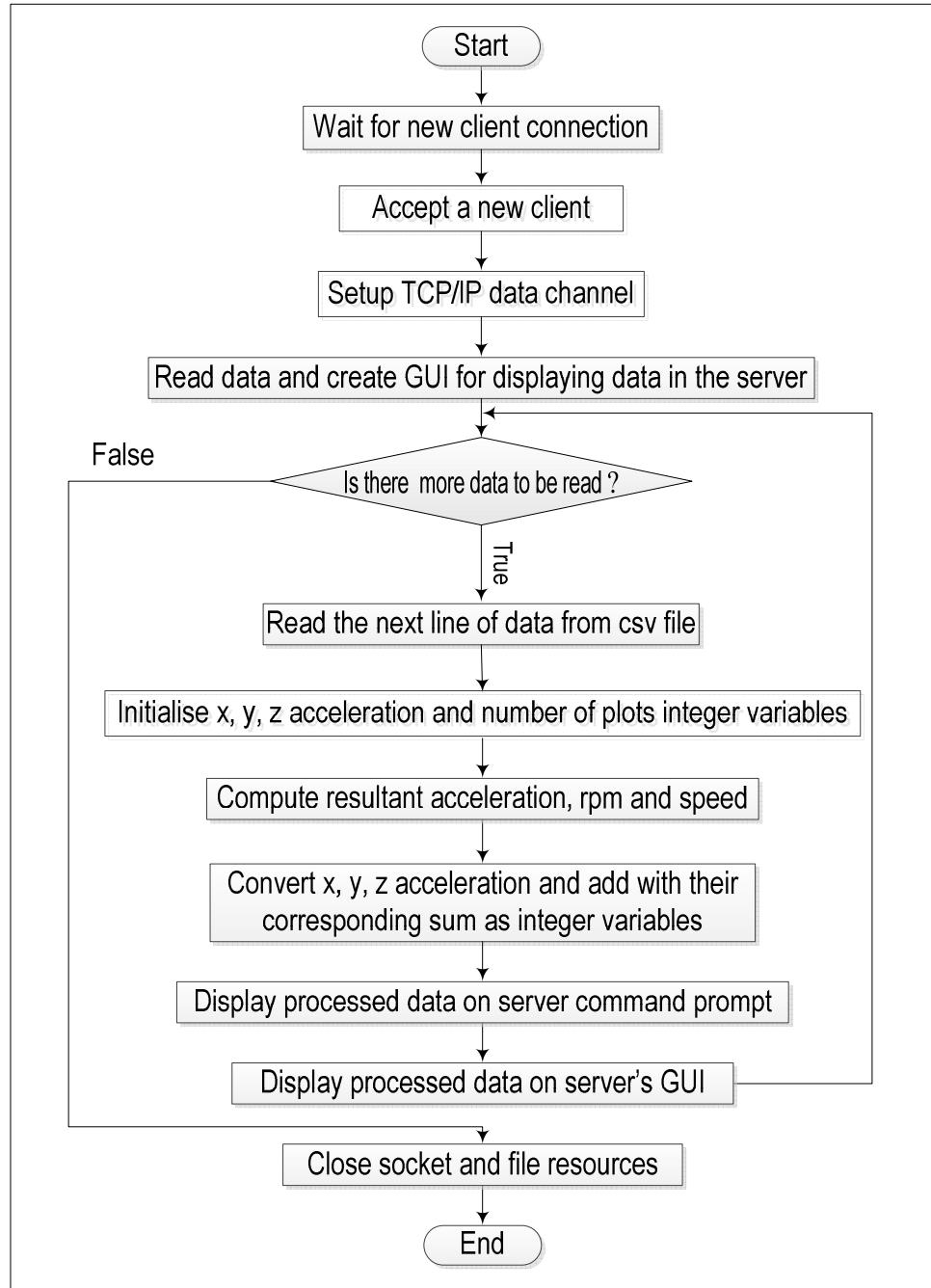


Figure 5.6: Socket Server algorithm for a single moving vehicle

5.4.2 Creating String and Object, and Connecting Client

A string of car file name is created. Try catch IO exception is set up. The car file name considered here is “Car_1000.csv”:

```
String filename = "Car_1000.csv";
```

A buffer reader object is created to read the above-mentioned .csv file.

```
BufferedReader CSVfile = new BufferedReader(new FileReader(filename));
```

Output stream writer is created in order to write the output stream for writing the output data from the .csv file.

```
BufferedOutputStream os = new BufferedOutputStream(connection.getOutputStream());
OutputStreamWriter osw = new OutputStreamWriter(os, "US-ASCII");
```

Once the .csv file is read, a server socket is created. This initiates activation of the server port to communicate with the client:

```
socket1 = new ServerSocket(port);
```

The following code accepts the connection between the server and client:

```
connection = socket1.accept();
```

Connection initialisation message is displayed on the server:

```
System.out.println("SingleSocketServer Initialized");
```

5.4.3 Creating GUI for Displaying Server Data

A Graphical User Interface (GUI) enables the users to access the server's front-end screen. Therefore, a GUI set-up is essential to be designed in the server algorithm for the moving vehicle. The current characteristics of the single moving vehicle are displayed through GUI. The following codes are written on server in order to create GUI for displaying server's raw data:

```
GridLayout experimentLayout = new GridLayout(7,1,0,0);
JFrame myFrame = new JFrame("Server");
myFrame.setLayout(experimentLayout);
myFrame.setSize(300,400);
myFrame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
myFrame.setVisible(true);
JLabel text1 = new JLabel("Sending the following data to client");
JLabel text2 = new JLabel("          ");
JLabel text3 = new JLabel("          ");
JLabel text4 = new JLabel("          ");
JLabel text5 = new JLabel("          ");
JLabel text6 = new JLabel("          ");
JLabel text7 = new JLabel("          ");
myFrame.add(text1);
myFrame.add(text2);
myFrame.add(text3);
myFrame.add(text4);
myFrame.add(text5);
myFrame.add(text6);
myFrame.add(text7);
```

5.4.4 Reading Data from the .csv File

The first line of data of the .csv file is read. A check is incorporated to determine if the data row is null. The null data row guarantees that the end of the file is reached, and the socket and file resources are closed.

```
CSVFile.readLine();
String dataRow = CSVFile.readLine();
```

If the data rows are not null, they are progressively read and vehicle data are computed. In order to compute the next data row, the next line of data is read in the following manner:

```
dataRow = CSVFile.readLine();
```

Output stream is re-initialised:

```
osw.flush();
osw = new OutputStreamWriter(os, "US-ASCII");
```

5.4.5 Processing of Vibration Acceleration Data

Variables related to vibration acceleration data are declared and initialised. These include vibration accelerations in x, y and z directions and number of plots of the single moving vehicle. The following codes initialise vehicle vibration acceleration data:

```
int XAccel    = 0;
int YAccel    = 0;
int ZAccel    = 0;
int num_of_plots = 1;
```

A string array is considered in order to compute the speed data. Each column of the speed data is converted into an element in the string array.

```
String[] dataArray = dataRow.split(",");
```

These elements in the string array, i.e., x, y and z vibration accelerations string values, are converted into integers as follows:

```
XAccel  = Integer.parseInt(dataArray[1]);
YAccel  = Integer.parseInt(dataArray[2]);
ZAccel  = Integer.parseInt(dataArray[3]);
```

The server computes resultant vibration acceleration from the raw vibration acceleration data. The resultant vibration acceleration data are converted to speed data using the relationship established between vibration acceleration and wheel speed in chapter 4.

```

double Resultant_Acceleration = Math.sqrt ((XAccel*XAccel) + (YAccel*YAccel) +
(ZAccel*ZAccel));
double RPM = Math.abs((Resultant_Acceleration - 3024.8)) / 31.655;
double Speed = (2*3.1415*RPM*0.055*3600/(60*1000))*96.457;

```

5.4.6 Displaying Data on Server's DOS Window

The vibration acceleration data of the vehicle is displayed on the command prompt of the server when the server communicates with the client. The vibration acceleration of the car in x, y and z axes, the computed resultant vibration acceleration of the car, rpm of the car wheel and speed of the car are displayed on the command prompt.

```

String message = "-----\n"+
"XAccel:      "+XAccel+"\n"+
"YAccel:      "+YAccel+"\n"+
"ZAccel:      "+ZAccel+"\n"+
"Resultant_Acceleration: "+Resultant_Acceleration+"\n"+
"RPM:         "+RPM+"\n"+
"Speed:        "+Speed+"\n";
System.out.println(message);

```

5.4.7 Displaying Data on Server's GUI

Like the server command prompt, the processed data of the car is displayed on the GUI of the server. In addition to the vibration acceleration in x, y and z axes, the resultant vibration acceleration, rpm and speed of the car, an additional element is added to the GUI of the server. This additional information is the car number.

```

DecimalFormat df = new DecimalFormat("#.##");
text1.setText(" Car_number:           Car_1000");
text2.setText(" XAccel:             "+XAccel);
text3.setText(" YAccel:             "+YAccel);
text4.setText(" ZAccel:             "+ZAccel);
text5.setText(" Resultant_Acceleration: "+df.format(Resultant_Acceleration));
text6.setText(" RPM:                "+df.format(RPM));
text7.setText(" Speed:               "+df.format(Speed));

```

All the processed data on the server's GUI are written to client along with the end of line character. The following Java codes serve the purpose:

```

osw.write(XAccel + ",");
osw.write(YAccel + ",");
osw.write(ZAccel + ",");
osw.write(Resultant_Acceleration + ",");
osw.write(RPM + ",");
osw.write(Speed+ "\n");
osw.write((char) 13);

```

The server processes plots for different speeds of the moving car in client. This plot is visualised in client only. If the number of plot in the client is greater than one a delay is

created for 30 seconds. This follows an increase in the number of plot variables. However, if the number of plot in client is not greater than one the number of plot variables increase without having a delay for 30 seconds.

```
if(num_of_plots > 1);  
Thread.sleep(30000);  
num_of_plots++;
```

This follows sending of the end of transmission character to the client:

```
osw.write((char) 23);
```

5.4.8 Closing Socket and File Resources

Once all the processed data of the car file are displayed on command prompt and server's GUI, the socket is closed.

```
osw.close();  
connection.close();
```

5.5 SOCKET CLIENT ALGORITHM FOR SINGLE MOVING VEHICLE

The server computer communicates data to client computer using the codes written on the server computer. The client computer should have a set of coded instructions in order to listen to the server's communication. Therefore, an algorithm has been developed and named as socket client as illustrated in Figure 5.7.

Initially a port on the client computer is defined that accepts the set of coded communications from the server computer. In order to define a port on the client computer an integer, called port, is created. The client computer will listen to the communication of the server computer on port 19999. This is the reason for which the port is initialised to 19999. Therefore, the port is defined as int port = 19999.

A GUI enables the users to access the system's front end screens developed by the Java programming. Therefore, a GUI set up is essential to be designed in the socket client algorithm designed for the single moving vehicle. In this algorithm, the speed of the car is displayed on a 2-D plot using GUI. This GUI window is known as "SpeedManage".

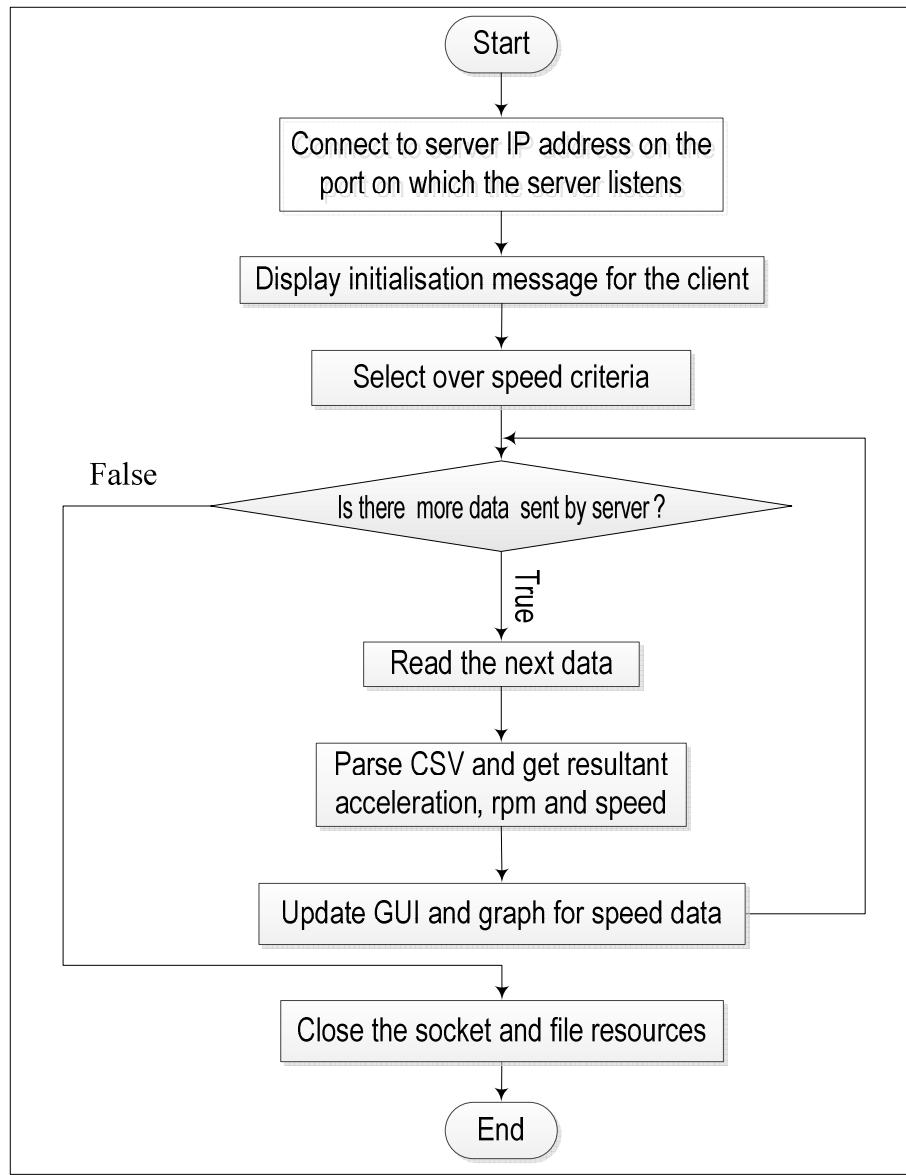


Figure 5.7: Socket client algorithm for a single moving vehicle

5.5.1 Connection to Server

In order to establish communication between the server computer and the client computer, the server IP address is assigned to the client computer:

```
String host = "136.206.95.82";
```

The processed data of the server are written to the client. List of options are created to save the processed data of the server on the client computer using ‘ArrayList’ constructor. The main advantage of an ArrayList is that the dynamic array automatically expands as data is added:

```
ArrayList<String> temp_line_array = new ArrayList<String>();
```

The processed data is stored as strings in the client. Thus, string buffer object is created:

```
StringBuffer instr = new StringBuffer();
```

5.5.2 Client Initialisation

Once the socket client is run, server computer gets connected with the client computer. This means that the socket client algorithm is ready to display the processed data. At this stage an initialisation message for client computer is displayed in DOS prompt:

```
System.out.println("SocketClient initialized\nStart");
```

In order to display the processed data, a try catch is set up under which a number of coded instructions are provided. The output of the processed data in the client computer is written with some headers to a .csv file. An address object of the server is obtained from the server:

```
PrintWriter out = new PrintWriter(new FileWriter("output.csv"));
out.println("XAccel,YAccel,ZAccel,ResultantAcceleration, RPM, Speed");
InetAddress address = InetAddress.getByName(host);
```

5.5.3 Speed Limit Settings

A GUI is generated in the client computer. This GUI provides several options, in drop down menu, in order to set the speed limits of the moving cars depending on the type of the roads. This is illustrated in the following codes:

```
final int LOWEST = 30;
final int INCREMENT = 5;
final int HIGHEST = 80;
Object[] speedlimits = new Integer[(HIGHEST - LOWEST)/INCREMENT + 1];
int i, j;
for (i = LOWEST, j = 0; j < speedlimits.length; i+=INCREMENT, j++) {
    speedlimits[j] = new Integer(i);
}
Integer s = (Integer) JOptionPane.showInputDialog(null,
    "Please choose an appropriate speed limit:\n",
    "Speed Limit Settings",
    JOptionPane.PLAIN_MESSAGE, null, speedlimits, new Integer(LOWEST));
if (s == null) {
    out.close();
    return;
}
speed_limit = (double)s.intValue();
```

5.5.4 Establishing Socket Connection

In order to set up a socket connection with the server computer, the client computer establishes communication with the port as defined in the server algorithm.

Furthermore, the socket connection is established by connecting to the IP address of the server computer. The following code helps in establishing the socket connection:

```
Socket connection = new Socket(address, port);
```

A socket connection has been established between the client and server computers. An object, “*InputStreamReader*”, reads the processed data of the server in the client computer. The ASCII values of the processed data of the server computer are read. The following codes initiate an *InputStreamReader* with the optional character encoding:

```
BufferedInputStream bis = new BufferedInputStream(connection.getInputStream());
InputStreamReader isr = new InputStreamReader(bis, "US-ASCII");
```

5.5.5 Creating ‘SpeedManage’ GUI

‘SpeedManage’ is a GUI that shows speed in the form of plots of the car while it is moving on the road. Different maximum speed limits are set for different road conditions. If the car crosses a particular speed limit set by the user on the client computer, the said speed of the car is considered as an over speed on that road. The over speed is indicated on ‘SpeedManage’ using a red-coloured box. The speeds of the car at different time stamps are indicated using blue-coloured boxes, which indicate that the speeds of the car are below the speed limit. Yellow-coloured boxes in ‘SpeedManage’ indicate that the speed of the car, at that particular point of time, is at the maximum speed limit on the road. A horizontal red line is drawn on the ‘SpeedManage’ window reflecting the maximum speed limit of the car. This horizontal line is placed in different positions of the ‘SpeedManage’ window for different maximum speed limits. The processed data are displayed in ‘SpeedManage’ using the following codes:

```
final PlotWithArrowLabelsAndLines demo = new
    PlotWithArrowLabelsAndLines("SpeedManage", speed_limit);
demo.pack();
RefineryUtilities.centerFrameOnScreen(demo);
demo.setVisible(true);
```

The colours in the ‘SpeedManage’ GUI, as stated above, are set by calling the following header file:

```
import java.awt.Color;
```

The speed of the moving vehicle is defined with a parameter “y”. The parameters “x” and “y” are time in second and speed of the car at a particular time respectively. These

parameters aid the ‘SpeedManage’ GUI to obtain different colours for the car’s speeds at different time stamps:

```
double y = dataset.getYValue(row, col);
double xVal = this.lastValue + seconds;
```

The parameters “x” and “y” are initialised and relationships with the corresponding conditional threshold values are defined:

```
if( (int) y > (int) Threshold )
{
    return Color.red;
}
else if( (int) y == (int) Threshold )
{
    return Color.yellow;
}
else
{
    return Color.blue;
}
```

‘SpeedManage’ reads first data as socket’s InputStream, appends to a StringBuffer and displays in the following manner:

```
int c;
c = isr.read();
int counter=0;
```

5.5.6 Displaying Data in DOS Window and ‘SpeedManage’ GUI

Once ‘SpeedManage’ starts on the client computer, the DOS window starts displaying the processed data on the client computer. The displayed data in the DOS window includes car wheel vibration acceleration data in x, y and z axes, resultant vibration acceleration of the moving car, rpm of the car wheel and speed of the moving car.

```
System.out.println("XAccel : " +temp_line_array.get(0));
System.out.println("YAccel: " +temp_line_array.get(1));
System.out.println("ZAccel: " +temp_line_array.get(2));
System.out.println("Resultant_Acceleration: " +temp_line_array.get(3));
System.out.println("RPM: " +temp_line_array.get(4));
System.out.println("Speed: " +temp_line_array.get(5));
```

The displayed data in the DOS window e.g., vibration acceleration in x, y and z directions, resultant vibration acceleration, rpm and speed of the moving vehicle, are also displayed in the ‘SpeedManage’ GUI. Here, the vehicle name, speed and time stamp are specified as strings.

```

double speed_as_an_double = Double.parseDouble(temp_line_array.get(5));

String fn = "Car_1000";
String sp = temp_line_array.get(5);
double yVal = Double.valueOf(sp);
double tmpY = yVal*100;
tmpY = Math.round(tmpY);
double tmpFinal = tmpY/100;
String speed = String.valueOf(tmpFinal);
String timeAndDateStr = getTimeAndDateString(counter);
demo.plotData( temp_line_array.get(5) );
demo.addMessage(timeAndDateStr);
demo.addMessage("Car Speed: " + speed + " Km/Hr");
demo.addMessage("Speed limit: " + speed_limit + " Km/Hr");
demo.addMessage("Car Number: Car_1000");
demo.addMessage("-----");

```

Based on the conditional relationships, ‘SpeedManage’ displays different colours for the speeds below, equal and above the maximum speed limits:

```

if( (int) yVal > Threshold )
{
    xypointerannotation.setBackgroundPaint(Color.red);
}
else if( (int) yVal == Threshold )
{
    xypointerannotation.setBackgroundPaint(Color.yellow);
}
else
{
    xypointerannotation.setBackgroundPaint(Color.cyan);
}

subplot.addAnnotation(xypointerannotation);
System.out.println("Plotting X = " + xVal + " Y=" + yVal);

```

The maximum speed limit is indicated in ‘SpeedManage’ using a red line horizontal to x-axis of the plot:

```
ValueMarker mark = new ValueMarker(Threshold,Color.RED, new BasicStroke(2));
```

5.5.7 Updating the Processed Data on the Client

The latest processed car data stored on the server computer are read and subsequently displayed on the client computer’s DOS window and ‘SpeedManage’ GUI as explained before. At the end of reading and displaying the set of processed data, a temporary array is defined to facilitate the re-initialisation of the processed data. This helps to locate if any remaining processed data is left for reading and displaying on the client computer thereby updating the string arrays of the processed data on the client computer. The characters embedded within the string arrays are transmitted from the server computer

to the client computer. Therefore, the client computer checks, using the following code, if the end of character transmission is reached:

```
while( c != 23 )  
{
```

Furthermore, the client computer checks if the transmission of the end of a line character is reached:

```
while( c != 13 )  
{
```

If the transmission of the end of a line character is not reached, the value of the object and data written in the string arrays in the client computer is updated by the following codes:

```
instr.append( (char) c);  
c = isr.read();  
}
```

The client computer converts the string buffer to a string and adds each comma separated column into the temporary array list:

```
String buffer_as_string = instr.toString();  
StringTokenizer st = new StringTokenizer(buffer_as_string, ",");  
while (st.hasMoreTokens())  
{  
    String column = st.nextToken();  
    temp_line_array.add(column);  
}
```

5.5.8 Generating Reports on Client Computer

The speed of the vehicle is displayed in the DOS and ‘SpeedManage’ windows on the client computer. The client computer stores the information of the car running at a speed over the maximum road speed limit corresponding to the time. Car number, road speed limit, speed of the car, time and date are the stored information of the speeding car. The information are stored in two different file formats, e.g., .csv and .pdf. The following codes result in a report in .csv file format:

```
PrintWriter out = new PrintWriter(new FileWriter("output.csv"));  
out.println("XAccel,YAccel,ZAccel,ResultantAcceleration, RPM, Speed");
```

A report is generated in .pdf file format with the over speed car data. Initially a file is saved in .pdf file format when the car starts running with a speed over the maximum

road speed limit. Additional over speed car data is stored in the same file. These functions are performed using the following codes:

```
public static void CreatePdf(String fName, String Speed, int counter, String timeStamp)
{
    String reportFileName = "report_for_" + fName + ".pdf";
    File f = new File(reportFileName);
    Document document = new Document();
    if(f.exists()) {
        try {
            System.out.println("file already exists. Adding to report..\n");
            String prevContent = "";
            if(hMap.containsKey(fName))
            {
                prevContent = hMap.get(fName);
            }
        }
    }
}
```

5.5.9 Closing the Socket Connection and File Resources

Once the report is generated for the over speed car, socket connection is closed. Simultaneously all the file resources are also closed.

```
connection.close();
out.close();
```

5.6 SOCKET SERVER ALGORITHM FOR MULTIPLE MOVING VEHICLES

In the previous section, the communication between the server and client computers has been demonstrated. In order to establish the communication, a server algorithm and a client algorithm are generated. The algorithms are coded in Java language. Figure 5.8 presents a server algorithm for multiple vehicles. The following sections present the coding steps of the server algorithm for multiple vehicles.

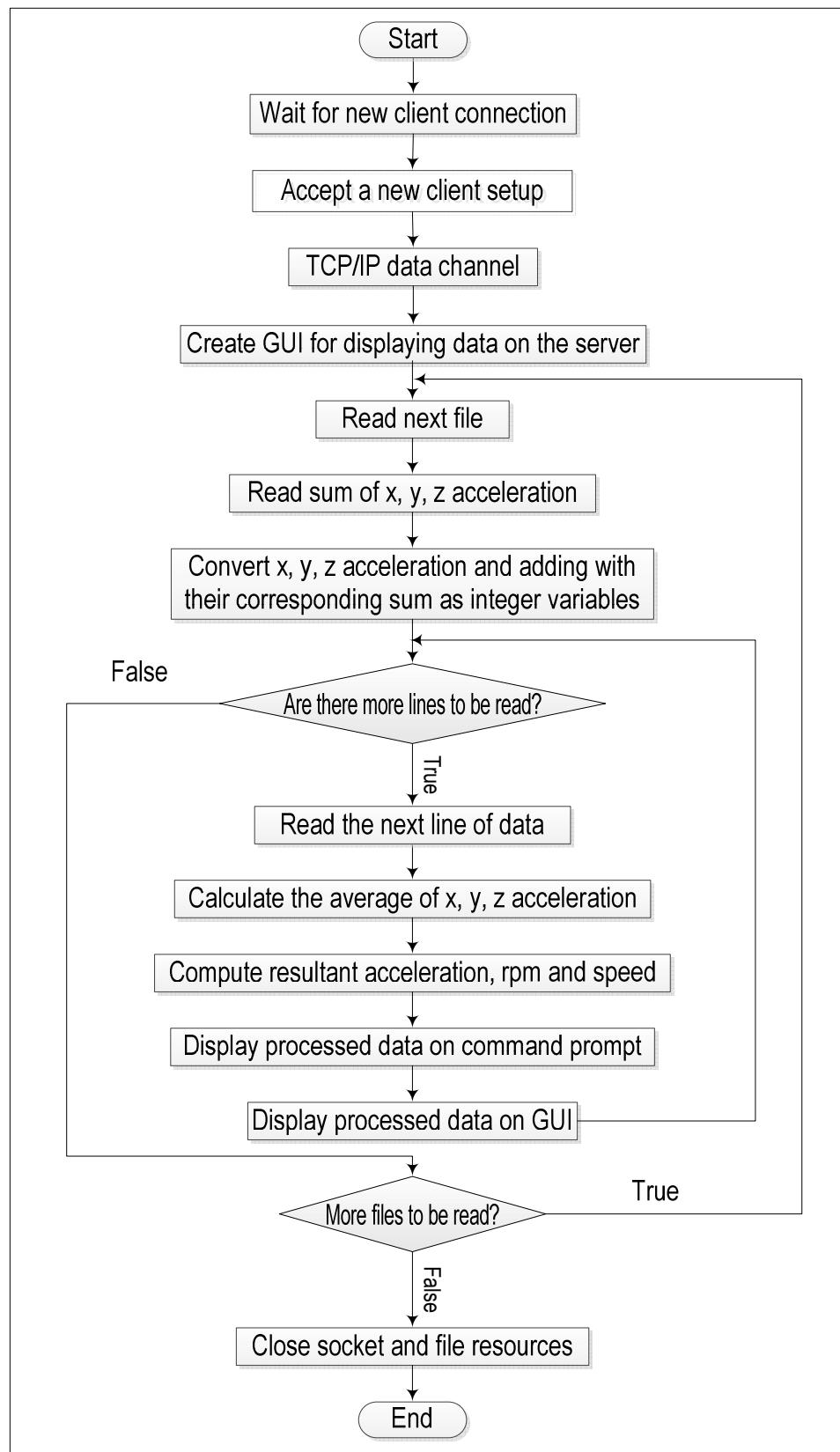


Figure 5.8: Socket server algorithm for multiple moving vehicles

5.6.1 Setting up Data Channel

TCP/IP data channel is a networking protocol that provides end-to-end connectivity. This protocol specifies data in relation to its formatting, address, transmission, routing and reception at the destination. As the first step of this algorithm, a connection is declared as a socket object. A ServerSocket object is declared as socket1:

```
static Socket connection;  
static ServerSocket socket1;
```

A data channel is established for its task-specific data exchange for the end-to-end data transfer. Port number is used for receiving service requests from client computer. Through the ‘listening’ port client computer gets connected to the server computer. Therefore, the server computer establishes a one-to-one server-client connection:

```
final static int port = 19999;
```

5.6.2 Creation of String and Object, and Connecting Client

Multiple strings of vehicle file names are created. ‘Car_array’ contains strings that are the suffix of the filename used to read in data about each vehicle. Try catch IO exception is set up. A new server socket is created by using the specified port. Server socket waits for a client connection to be made and then accepts it. The following codes define all the functionalities:

```
{  
String[] Car_array = { "1052", "1250", "2453", "2201", "0151", "3402", "0302", "1103", "4502",  
"1350" };  
  
socket1 = new ServerSocket(port);  
  
connection = socket1.accept();
```

A new buffered output stream is created to write data to the specified underlying output stream. An OutputStreamWriter is created that uses the given character set. The declared variable index is of integer type and assigns it the value 0.

```
BufferedOutputStream os = new BufferedOutputStream(connection.getOutputStream());  
  
OutputStreamWriter osw = new OutputStreamWriter(os, "US-ASCII");  
  
int index = 0;
```

Connection initialisation message has been printed to the DOS command prompt using the following code:

```
System.out.println("SingleSocketServerAverage Initialized");
```

5.6.3 Creating GUI for Displaying Server Data

A Graphical User Interface (GUI) helps to visualise the current status of the vehicles. The information, e.g., vehicle number, vibration accelerations in x, y and z directions, resultant vibration acceleration, rpm and speed of the vehicle are displayed in this GUI. The GUI is updated with respect to the new vehicle status once it passes through the road. The following codes create a GUI on the server computer:

```
GridLayout experimentLayout = new GridLayout(7,1,0,0);
JFrame myFrame = new JFrame("Server");
myFrame.setLayout(experimentLayout);
myFrame.setSize(300,400);
myFrame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
myFrame.setVisible(true);
JLabel text1 = new JLabel("      Sending the following data to client");
JLabel text2 = new JLabel("      ");
JLabel text3 = new JLabel("      ");
JLabel text4 = new JLabel("      ");
JLabel text5 = new JLabel("      ");
JLabel text6 = new JLabel("      ");
JLabel text7 = new JLabel("      ");
myFrame.add(text1);
myFrame.add(text2);
myFrame.add(text3);
myFrame.add(text4);
myFrame.add(text5);
myFrame.add(text6);
myFrame.add(text7);
```

5.6.4 Reading Data From the .csv File

A string file name is created. The prefix of the file name is joined together with the string “_Car.csv” in order to obtain a file name as shown below:

```
String Car_number = "Car_"+Car_array[index]+ ".csv";
```

In order to read the vehicle file a buffered reader object is created. The created buffered reader object for the .csv file is as follows:

```
BufferedReader CSVFile = new BufferedReader(new FileReader(Car_number));
```

The header and first data row of the .csv file is read. The first data row is read, but it does not assign it to a variable as the first line is going to be. TimeStamp, XAccel, YAccel, ZAccel values are not sent to the client computer. Subsequently, the second data row is read.

```

CSVFile.readLine();
String dataRow = CSVFile.readLine();

```

5.6.5 Processing of Vibration Acceleration Data

Vehicle data related to integer variables are declared and subsequently initialised. The variables are vibration accelerations in x, y and z directions. These variables are used to calculate the average acceleration.

```

int XAccel_Sum = 0;
int YAccel_Sum = 0;
int ZAccel_Sum = 0;
int num_of_lines = 0;

```

The comma separated lines are converted into a string array. Each of the columns is converted into an element as string array:

```
String[] dataArray = dataRow.split(",");
```

Each of the strings is converted into integers:

```

XAccel_Sum = XAccel_Sum + Integer.parseInt(dataArray[1]);
YAccel_Sum = YAccel_Sum + Integer.parseInt(dataArray[2]);
ZAccel_Sum = ZAccel_Sum + Integer.parseInt(dataArray[3]);

```

Number of line increment variable is declared and the next data line is read:

```

num_of_lines = num_of_lines + 1;
dataRow = CSVFile.readLine();

```

Once all the columns of the data are read, the averages of the vibration accelerations in x, y and z directions are computed:

```

int XAccel_Average = XAccel_Sum / num_of_lines;
int YAccel_Average = YAccel_Sum / num_of_lines;
int ZAccel_Average = ZAccel_Sum / num_of_lines;

```

The resultant vibration acceleration is calculated using the following formula.

$$\text{Resultant Acceleration} = \sqrt{((X\text{Accel Average})^2 + (Y\text{Accel Average})^2 + (Z\text{Accel Average})^2)}$$

The wheel rpm and vehicle speed are also computed:

```

double RPM = Math.abs(Resultant_Acceleration - 3024.8) / 31.655;
double Speed = (2 * 3.1415 * RPM * 0.055 * 3600 / (60 * 1000)) * 96.457;
DecimalFormat df = new DecimalFormat("#.##");

```

A while loop checks to see if the data is null in the .csv file. If the data is null, the end of the file is reached; otherwise, the data is processed further.

```
while (dataRow != null)
```

5.6.6 Displaying Data on Server's DOS and GUI windows

The processed data of all the vehicles are displayed in a DOS window. The vibration acceleration in x, y and z directions, computed resultant vibration accelerations, wheel rpm and the vehicle speeds are displayed. The following codes are employed in order to achieve these functionalities:

```
String message = "Car_number:      Car_"+Car_array[index]+"\n"+  
"XAccel_Average:    "+XAccel_Average+"\n"+  
"YAccel_Average:    "+YAccel_Average+"\n"+  
"ZAccel_Average:    "+ZAccel_Average+"\n"+  
"Resultant_Acceleration: "+Resultant_Acceleration+"\n"+  
"RPM:              "+RPM+"\n"+  
"Speed:            "+Speed+"\n";  
System.out.println("-----");
```

Like the server DOS window, the processed data are printed in a GUI window of the server. Vehicle number is an additional component included in the GUI window. This component is dependent on the number of cars considered.

```
text1.setText(" Car_number:      Car_"+Car_array[index]);  
text2.setText(" XAccel_Average:    "+XAccel_Average);  
text3.setText(" YAccel_Average:    "+YAccel_Average);  
text4.setText(" ZAccel_Average:    "+ZAccel_Average);  
text5.setText(" Resultant_Acceleration: "+df.format(Resultant_Acceleration) );  
text6.setText(" RPM:              "+df.format(RPM));  
text7.setText(" Speed:            "+df.format(Speed));
```

All the processed data on the server computer's GUI are written to the client computer along with the end of the character. The heading of each row is written in the .csv file format.

```
osw.write(Car_array[index] + ",");  
osw.write(XAccel_Average + ",");  
osw.write(YAccel_Average + ",");  
osw.write(ZAccel_Average + ",");  
osw.write(Resultant_Acceleration + ",");  
osw.write(RPM + ",");  
osw.write(Speed+ "\n");
```

The char 13 is sent to let the client know that it has reached the end of the file. The current data file is closed. The data from the next data file is read afterwards.

```
osw.write((char) 13);  
CSVfile.close();
```

5.6.7 Checking the Total Number of Data Files

The processing of the first data file is completed. A while loop is used to read and process more data files. The while loop will be continued until the end of the data files are reached. The processed data is displayed on the client computer with a delay for 30 seconds.

```
while (index < Car_array.length )
index = index + 1;
try
{
    Thread.sleep(30000);
}
```

5.6.8 Closing Socket and File Resources

Once all the data files are read and displayed on the server computer, the ‘char 23’ is sent to the client computer in order to confirm that all the data has been communicated. This ends the communication between the server and client computers.

```
osw.write((char) 23);
osw.close();
connection.close();
```

5.7 SOCKET CLIENT ALGORITHM FOR MULTIPLE MOVING VEHICLES

The client computer runs a set of coded instructions in order to listen to the server’s communication. Therefore, an algorithm has been developed on the client computer as illustrated in Figure 5.9. A port on the client computer is defined that accepts the set of coded communications from the server computer. The client computer listens to the communication of the server computer set on port 19999. A GUI set up is essential in the socket client algorithm designed for the moving car that enables the users to access the system’s front-end screens developed by Java codes. In this algorithm, the speeds of multiple vehicles are displayed on a 2-D plot of the “SpeedManage” GUI.

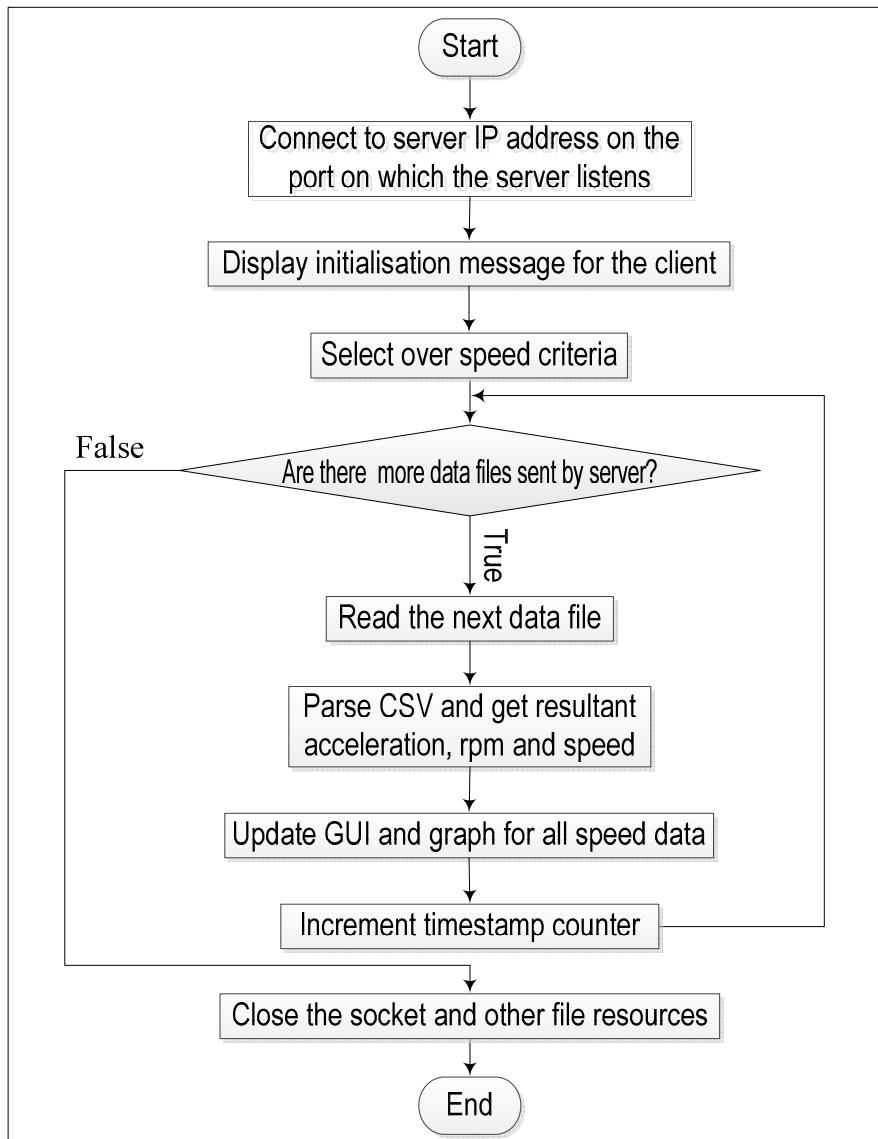


Figure 5.9: Socket client algorithm for multiple moving vehicles

5.7.1 Connection to Server

The server IP address is assigned to the client computer for establishing communication between the server computer and the client computer:

```
final String host = "136.206.95.82";
```

The processed data stored in the server computer are written to the client. ‘ArrayList’ constructor has been declared to save these processed data of the server on the client computer. ‘ArrayList’ constructor automatically expands the string array as the data is added. String buffer object is created in order to store the processed data as strings in the client. The following codes perform these operations:

```
ArrayList<String> temp_line_array = new ArrayList<String>();
```

```
StringBuffer instr = new StringBuffer();
```

5.7.2 Client Initialisation

Once the socket client is run, server computer gets connected with the client computer. This means that the socket client algorithm is ready to display the processed data. At this stage, an initialisation message for the client computer is displayed in a DOS prompt:

```
System.out.println("SocketClient1 initialized \nStart");
```

A try catch is set up in order to display the processed data. A number of coded instructions are provided. Some headers to a .csv file are written in the client computer for the output of the processed data. An address object of the server computer is obtained from the server computer.

```
PrintWriter out = new PrintWriter(new FileWriter("output.csv"));
out.println("Car_number,XAccel(Avg),YAccel(Avg),ZAccel(Avg),ResultantAcceleration,
RPM,Speed");
InetAddress address = InetAddress.getByName(host);
```

5.7.3 Speed Limit Settings

The “SpeedManage” GUI is generated in the client computer. The GUI has several options in a drop down menu. The drop down menu helps to set the speed limits of the moving vehicles depending on the type of the roads. Maximum speed limits for different roads have been set between 30 km/hr to 80 km/hr with an increment of 5 km/hr. Drop down menu helps in selecting the appropriate maximum speed limit for a particular road. The following codes illustrate how the maximum speed limits for different roads are determined for the vehicles.

```
final int LOWEST = 30;
final int INCREMENT = 5;
final int HIGHEST = 80;
Object[] speedlimits = new Integer[(HIGHEST - LOWEST)/INCREMENT + 1];
int i, j;
for (i = LOWEST, j = 0; j < speedlimits.length; i+=INCREMENT,j++) {
speedlimits[j] = new Integer(i);
}
Integer s = (Integer) JOptionPane.showInputDialog(null,
"Please choose an appropriate speed limit:\n",
"Speed Limit Settings",
JOptionPane.PLAIN_MESSAGE, null, speedlimits, new Integer(LOWEST));

if (s == null) {
out.close();
return;
}
```

```
speed_limit = (double)s.intValue();
```

5.7.4 Establishing Socket Connection

The client computer establishes communication with the port as defined in the server algorithm in order to set up a socket connection with the server computer. The socket connection is established by connecting to the IP address of the server computer. The following code builds the socket connection.

```
Socket connection = new Socket(address, port);
```

A socket connection is built between the client and server computers. An ‘InputStreamReader’ object is created that reads the processed data of the server computer in the client computer. The client computer reads the ASCII values of the server computer’s processed data. The ‘InputStreamReader’ object initiates optional character encoding:

```
BufferedInputStream bis = new BufferedInputStream(connection.getInputStream());
InputStreamReader isr = new InputStreamReader(bis, "US-ASCII");
```

5.7.5 Creating ‘SpeedManage’ GUI

The functionality of the ‘SpeedManage’ GUI has been described before. The processed data of the server computer are displayed in ‘SpeedManage’ GUI using the following codes.

```
PlotWithArrowLabelsAndNoLines demo = new
    PlotWithArrowLabelsAndNoLines("SpeedManage", (int) speed_limit);
demo.pack();
RefineryUtilities.centerFrameOnScreen(demo);
demo.setVisible(true);
```

Parameter ‘y’ defines the speeds of the vehicles at a particular time. The parameter ‘x’ refers to time in seconds. The ‘SpeedManage’ GUI obtains different colours for the vehicles’ speeds at different time stamps using these two parameters. The parameters ‘x’ and ‘y’ are initialised and subsequently the relationships with the corresponding conditional threshold values are defined:

```
double y = dataset.getYValue(row, col);
double xVal = this.lastValue + 30.0;

if( (int) y > (int) Threshold )
{
    return Color.red;
}
else if( (int) y == (int) Threshold )
```

```

{
    return Color.yellow;
}
else
{
    return Color.blue;
}

```

The ‘SpeedManage’ GUI reads first data as socket’s InputStream. The GUI appends to a StringBuffer and simultaneously displays the speeds of the vehicles in the following way.

```

int c;
c = isr.read();
int counter=0;

```

5.7.6 Displaying Data in DOS Window and ‘SpeedManage’ GUI

A DOS window displays the processed data on the client computer. At the same time, the ‘SpeedManage’ GUI window starts on the client computer. The DOS window displays vibration acceleration data in x, y and z directions, resultant vibration acceleration, rpm and speeds of the moving vehicles. The following codes help in displaying the data related to the vehicle speeds in the DOS window:

```

System.out.println("XAccel:           "+temp_line_array.get(0));
System.out.println("YAccel:           "+temp_line_array.get(1));
System.out.println("ZAccel:           "+temp_line_array.get(2));
System.out.println("Resultant_Acceleration: "+temp_line_array.get(3));
System.out.println("RPM:               "+temp_line_array.get(4));
System.out.println("Speed:             "+temp_line_array.get(5));

```

The ‘SpeedManage’ GUI window displays the vehicle speed data e.g., vibration acceleration in x, y and z directions, resultant vibration acceleration, rpm and speeds of the vehicles. The vehicle names, vehicle speeds and time stamp are specified as strings.

```

double speed_as_an_double = Double.parseDouble(temp_line_array.get(5));

String fn = "Car_1000";
String sp = temp_line_array.get(5);
double yVal = Double.valueOf(sp);
double tmpY = yVal*100;
tmpY = Math.round(tmpY);
double tmpFinal = tmpY/100;
String speed = String.valueOf(tmpFinal);
String timeAndDateStr = getTimeAndDateString(counter);
demo.plotData( temp_line_array.get(5) );
demo.addMessage(timeAndDateStr);
demo.addMessage("Car Speed:   " + speed + " Km/Hr");
demo.addMessage("Speed limit: " + speed_limit + " Km/Hr");
demo.addMessage("Car Number:  Car_1000");
demo.addMessage("-----");

```

The ‘SpeedManage’ GUI window displays different colours for the speeds below, equal to and above the maximum speed limit based on some conditional relationships:

```
if( (int) yVal > Threshold )
{
    xypointerannotation.setBackgroundPaint(Color.red);
}
else if( (int) yVal == Threshold )
{
    xypointerannotation.setBackgroundPaint(Color.yellow);
}
else
{
    xypointerannotation.setBackgroundPaint(Color.cyan);
}

subplot.addAnnotation(xypointerannotation);
System.out.println("Plotting X = " + xVal + " Y=" + yVal);
```

The maximum speed limits for the vehicles on different roads are indicated in ‘SpeedManage’ using a red line horizontal to x-axis of the plot:

```
ValueMarker mark = new ValueMarker(Threshold,Color.RED, new BasicStroke(2));
```

5.7.7 Updating the Processed Data on the Client

The latest processed data for the vehicles are read and subsequently displayed on the client computer’s DOS window and ‘SpeedManage’ GUI. A temporary array is defined to facilitate the re-initialisation of the processed data at the end of reading and displaying the set of processed data. This aids in locating if any remaining processed data is left for reading and subsequently displaying on the client computer. This process updates the string arrays of the processed data on the client computer. The characters embedded within the string arrays are transmitted from the server computer to the client computer. The client computer checks, using the following code, if the end of character transmission is reached.

```
while( c != 23 )
{
```

Furthermore, the client computer checks if the transmission of the end of a line character is reached:

```
while( c != 13 )
{
```

The value of the object and data are updated in the string arrays until the transmission of end line character. The client computer converts the string buffer to strings.

Furthermore, the client computer adds each comma separated column into the temporary array list:

```
instr.append( (char) c);
c = isr.read();
}

String buffer_as_string = instr.toString();
StringTokenizer st = new StringTokenizer(buffer_as_string, ",");
while (st.hasMoreTokens())
{
    String column = st.nextToken();
    temp_line_array.add(column);
}
```

The GUI is updated. Incremental values for the time stamp counter are provided to check if there is any vehicle file sent by the server computer to be read. The following code performs this process:

```
counter++;
```

5.7.8 Generating Reports on Client Computer

Both the DOS and ‘SpeedManage’ windows appeared on the client computer display the speeds of the vehicles. The client computer stores the information of the vehicles running at the speeds over the maximum speed limits on different roads. Vehicle numbers, road speed limits, speeds of the vehicles, time and date are stored for the speeding vehicles. The information are stored in two different file formats, e.g., .csv and .pdf. The following codes generate a report in .csv file format:

```
PrintWriter out = new PrintWriter(new FileWriter("output.csv"));
out.println("Car_number,XAccel(Avg),YAccel(Avg),ZAccel(Avg),ResultantAcceleration,
RPM,Speed");
```

Reports are generated in .pdf file format with the speeding vehicles’ data. Several files are saved in .pdf file format for the discrete vehicles moving on the roads above the speed limit. Reports for individual vehicle are generated using the following codes:

```
public static void CreatePdf(String fName, String Speed, int counter, String timeStamp)
{
    String reportFileName = "report_for_" + fName + ".pdf";
    File f = new File(reportFileName);
    Document document = new Document();

    if(f.exists()) {
        try {
            System.out.println("file already exists. Adding to report..\n");
            String prevContent = "";
            if(hMap.containsKey(fName)) {
```

```
    prevContent = hMap.get(fName);
}
PdfWriter.getInstance(document, new FileOutputStream(reportFileName));
document.setPageSize(PageSize.A4);
```

5.7.9 Closing the Socket Connection and File Resources

The report generation, in both the .pdf and .csv formats, for the speeding vehicles, follows closure of the socket connection between the server and client computers. The closure of the socket connection leads to the closure of all the file resources.

```
connection.close();
out.close();
```

Chapter 6: Results and Discussions

6.1 INTRODUCTION

The concept for intelligent traffic monitoring system using wireless sensor has been modelled in a laboratory environment. The software has been developed to monitor and control the related data automatically as mentioned in Chapter 5. This chapter demonstrates the functioning of the system with example scenarios. The graphical user interface of the software can display the vehicle speed information and generates report in the event of speeding by comparing the speed with the speed limit. The system is also capable of monitoring and controlling single vehicle and multiple vehicles.

6.2 SCENARIO DESIGN

In order to demonstrate the functioning of the system two scenarios have been taken into consideration. The first scenario can simulate the road traffic situation inside a city/town (Slower speed limit), where the speed limit varies between 30-50 km/hr whereas the second scenario considers the situation in highway (Faster speed limit) where speed limit varies between 50 to 80 km/hr. In both scenarios, single vehicle and multiple vehicles have been considered (Figure 6. 1).

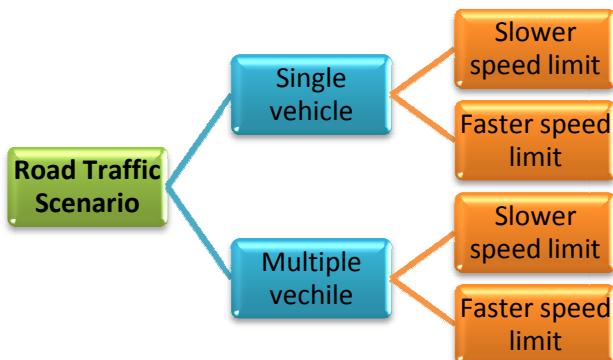


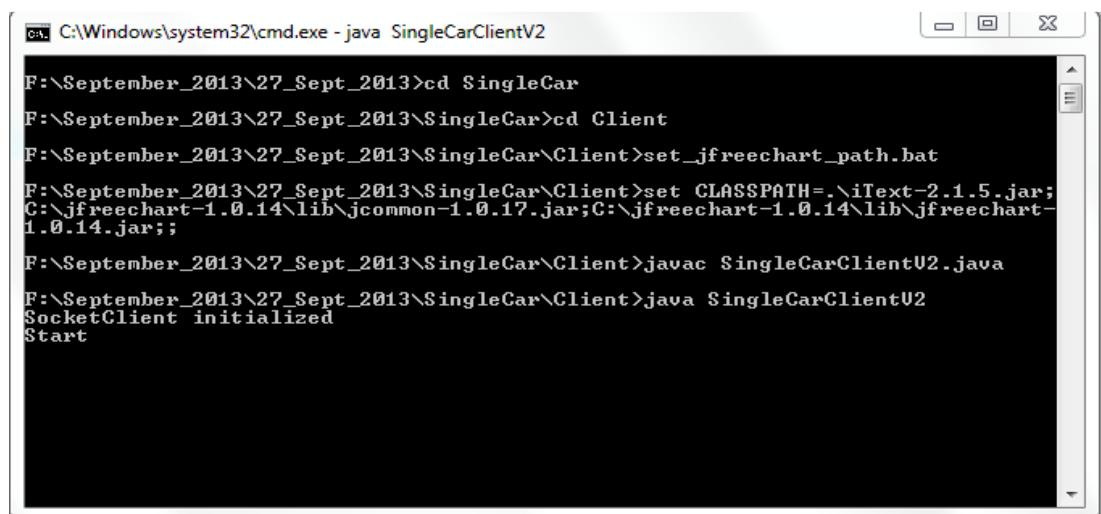
Figure 6.1: Road traffic scenarios considered for demonstration

6.3 SIMULATING A SINGLE VEHICLE SCENARIO

6.3.1 Single Vehicle Running at Lower Speed Limit

6.3.1.1. Connection Establishment

First, a connection needs to be established between a server computer and a client computer in order to display the vehicle speed information in a graphical user interface. Figure 6.2 displays confirmation of the connection between the server and client in a DOS window.



```
C:\Windows\system32\cmd.exe - java SingleCarClientV2
F:\September_2013\27_Sept_2013>cd SingleCar
F:\September_2013\27_Sept_2013\SingleCar>cd Client
F:\September_2013\27_Sept_2013\SingleCar\Client>set_jfreechart_path.bat
F:\September_2013\27_Sept_2013\SingleCar\Client>set CLASSPATH=.\\iText-2.1.5.jar;
C:\\jfreechart-1.0.14\\lib\\jcommon-1.0.17.jar;C:\\jfreechart-1.0.14\\lib\\jfreechart-
1.0.14.jar;;
F:\September_2013\27_Sept_2013\SingleCar\Client>javac SingleCarClientV2.java
F:\September_2013\27_Sept_2013\SingleCar\Client>java SingleCarClientV2
SocketClient initialized
Start
```

Figure 6.2: Connection established between a server computer and a client computer

6.3.1.2. Setting of Speed Limit for a Single Vehicle

In the case of vehicle running inside a city/town, it is assumed that the maximum speed limit is 40 km/hr. The speed limit can be set using a drop down menu from Speed Limit Settings dialog box in the client computer (Figure 6.3).

The speed limit for the scenario of single vehicle running inside a city/town is set to 40 km/h by clicking OK in the dialog box (Figure 6.4).

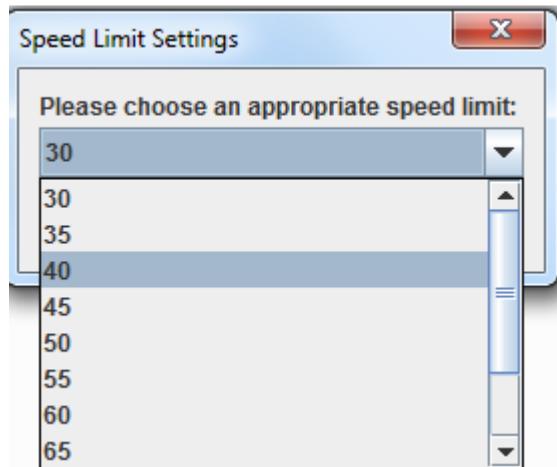


Figure 6.3: Selection of speed limit in the client computer for the scenario of single vehicle running at slower speed limit

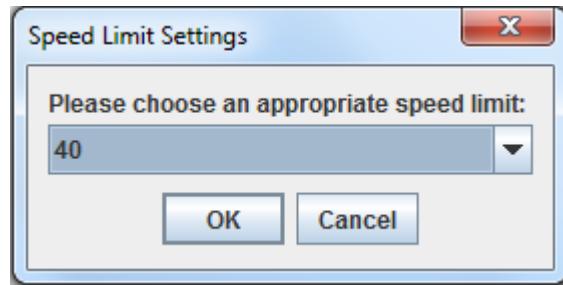


Figure 6.4: Set speed limit for a single vehicle running at slower speed limit

6.3.1.3 Displaying Single Vehicle Data in the Server Computer

After setting the speed limit, the server computer displays the vibration acceleration data related to vehicle speed, converts the data into rotational speed (rpm) and then vehicle speed format. All the calculated data are displayed in a DOS window as shown in Figure 6.5.

```

cmd C:\Windows\system32\cmd.exe - java SingleCarServer

XAccel:          2569
YAccel:          1536
ZAccel:          1768
Resultant_Acceleration: 3476.3315434520914
RPM:             14.264146057560929
Speed:           28.52729866223469

-----
XAccel:          2768
YAccel:          1744
ZAccel:          1757
Resultant_Acceleration: 3713.543994622926
RPM:             21.25782639781791
Speed:           43.51412522868743

-----
XAccel:          2747
YAccel:          2036
ZAccel:          1818
Resultant_Acceleration: 3872.5223046484834
RPM:             26.280044373668716
Speed:           53.558208582016064

-----
XAccel:          1215
YAccel:          2900
ZAccel:          1811
Resultant_Acceleration: 3628.490870871801
RPM:             19.070948376932574
Speed:           38.14055782644279

-----
XAccel:          1084
YAccel:          2768
ZAccel:          1831
Resultant_Acceleration: 3491.337995668709
RPM:             14.738208676945469
Speed:           29.475382618157607

-----
XAccel:          1418
YAccel:          2984
ZAccel:          1799
Resultant_Acceleration: 3761.832133415844
RPM:             23.283276999394847
Speed:           46.564919333463095

-----
XAccel:          1642
YAccel:          3112
ZAccel:          1855
Resultant_Acceleration: 3977.654208198596
RPM:             30.10122281467685
Speed:           60.2003322917327

-----
XAccel:          2656
YAccel:          1657
ZAccel:          1764
Resultant_Acceleration: 3593.282760930456
RPM:             17.95870355174398
Speed:           35.916146264216614

-----
XAccel:          2598
YAccel:          1518
ZAccel:          1800
Resultant_Acceleration: 3506.2698127782464
RPM:             15.209913529560772
Speed:           30.41875920607543

```

Figure 6.5: Display of vehicle speed data in a DOS window of the server computer for single vehicle running at slower speed limit

At the same time, the vehicle speed data with car number and other details are displayed in a server dialog box (Figure 6.6). In contrast to DOS window, the dialog box only displays the latest vehicle speed information.

6.3.1.4 Displaying Single Vehicle Data in the Client Computer

When the vehicle speed details are displayed in server computer, at the same time the client computer can read all data and display the same information. A DOS window is created in the client computer where all the speed data will be updated and displayed as

a chart for monitoring purpose. Figure 6.7 displays the vehicle speed data in a DOS window of the client computer.

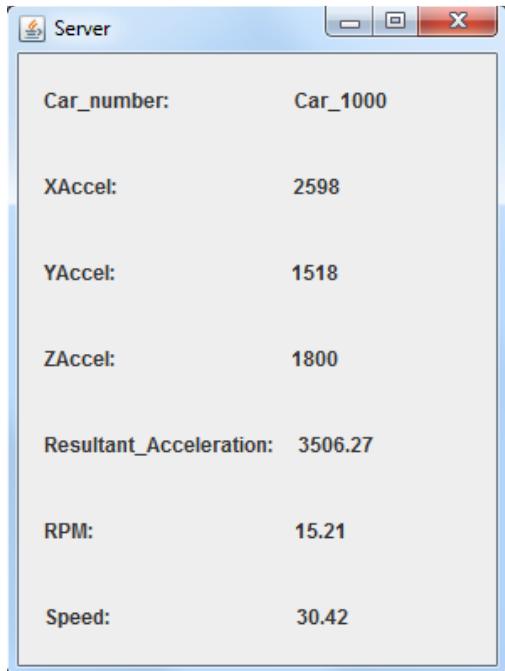


Figure 6.6: Display of latest speed data with vehicle details in a server dialog box for a single vehicle running at slower speed limit

A graphical user interface (GUI) is generated in the client computer for the case of a vehicle running inside a city/town. Figure 6.8 displays a snapshot of vehicle speed plot at any particular moment. The GUI shows a red line across the plot, which indicates the speed limit set for the particular road traffic condition. The speed plot generally displays the speeds in every 30 seconds. Each point in the plot also displays the actual value of the speed. The plot will automatically highlight the speed situations with different colours. Red colour indicates that the vehicle is running at a higher speed than the set limit on the road. On the other hand, a cyan colour indicates that the vehicle is running below the speed limit. Vehicle speed equal to the speed limit is indicated by the amber colour. The speed details are also continuously updated on the left hand side of the GUI.

```
C:\Windows\system32\cmd.exe - java SingleCarClientV2
ZAccel:          1818
Resultant_Acceleration: 3872.5223046484934
RPM:            26.780044373668716
Speed:           53.558208582016064
timeAndDateStr is Time:      19:53:46
Date:             Tues 3 Dec 2013
file already exists. Adding to report..
Plotting X = 600.0 Y=53.558208582016064
-----
XAccel :          1215
YAccel:           2900
ZAccel:           1811
Resultant_Acceleration: 3628.490870871801
RPM:             19.070948376932574
Speed:            38.14055782644279
timeAndDateStr is Time:      19:54:16
Date:             Tues 3 Dec 2013
Plotting X = 630.0 Y=38.14055782644279
-----
XAccel :          1084
YAccel:           2768
ZAccel:           1831
Resultant_Acceleration: 3491.337995668709
RPM:             14.738208626945469
Speed:            29.475382618157607
timeAndDateStr is Time:      19:54:46
Date:             Tues 3 Dec 2013
Plotting X = 660.0 Y=29.475382618157607
-----
XAccel :          1418
YAccel:           2984
ZAccel:           1799
Resultant_Acceleration: 3761.832133415844
RPM:             23.283276999394847
Speed:            46.564919333463095
timeAndDateStr is Time:      19:55:16
Date:             Tues 3 Dec 2013
file already exists. Adding to report..
Plotting X = 690.0 Y=46.564919333463095
-----
XAccel :          1642
YAccel:           3112
ZAccel:           1855
Resultant_Acceleration: 3977.654208198596
RPM:             30.10122281467685
Speed:            60.2003322917327
timeAndDateStr is Time:      19:55:46
Date:             Tues 3 Dec 2013
file already exists. Adding to report..
Plotting X = 720.0 Y=60.2003322917327
-----
XAccel :          2656
YAccel:           1657
ZAccel:           1764
Resultant_Acceleration: 3593.282760930456
RPM:             17.95870355174398
Speed:            35.916146264216614
timeAndDateStr is Time:      19:56:16
Date:             Tues 3 Dec 2013
Plotting X = 750.0 Y=35.916146264216614
-----
XAccel :          2598
YAccel:           1518
ZAccel:           1800
Resultant_Acceleration: 3506.2698127782464
RPM:             15.209913529560772
Speed:            30.41875920607543
timeAndDateStr is Time:      19:56:46
Date:             Tues 3 Dec 2013
Plotting X = 780.0 Y=30.41875920607543
```

Figure 6.7: Display of vehicle speed data in a DOS window of the client computer for a single vehicle running at slower speed limit

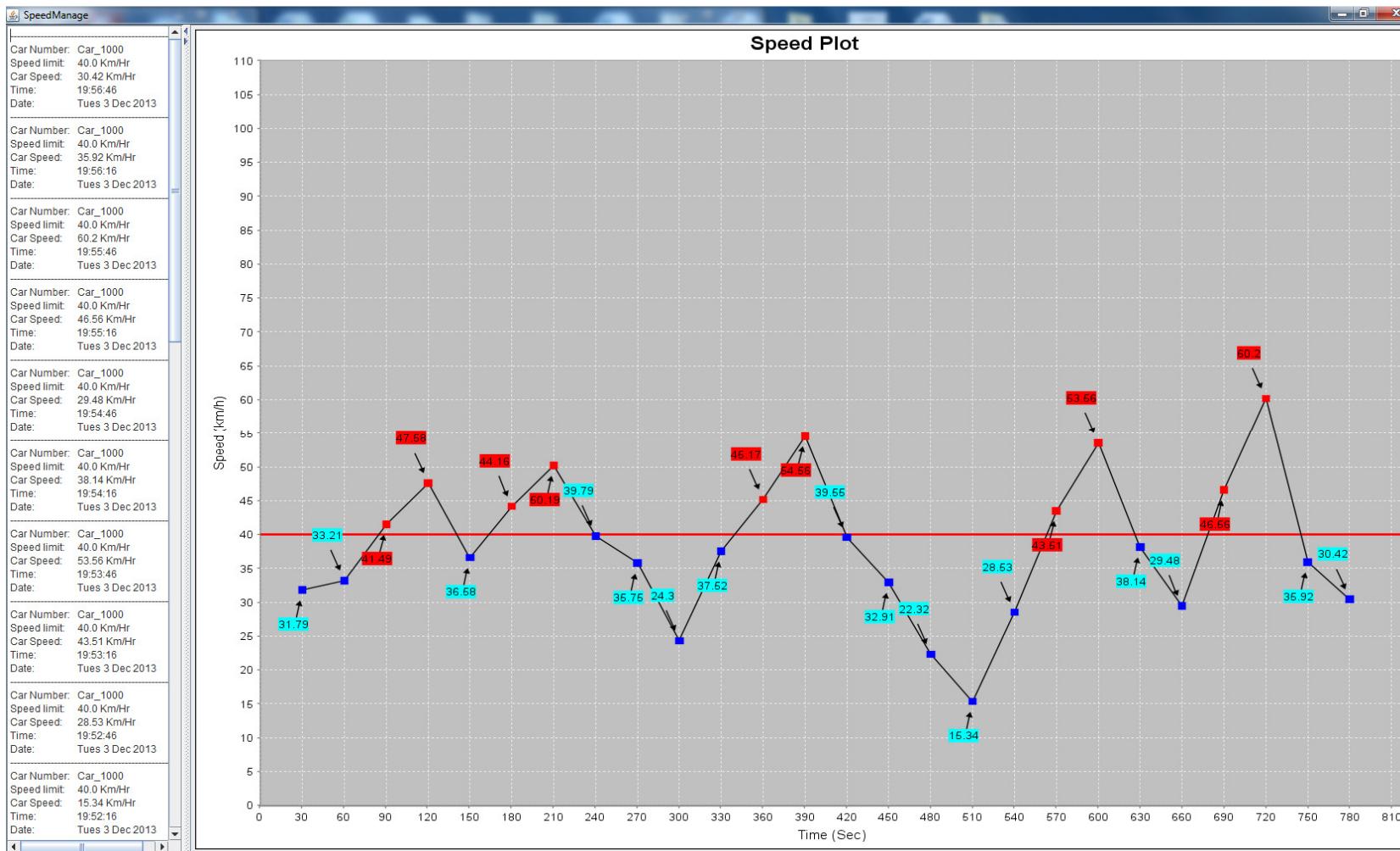


Figure 6.8: Display of vehicle speed plot on a GUI in the client computer for a single vehicle running at slower speed limit

6.3.1.5 Report Generation for Single Vehicle

The system can automatically detect the speeding vehicle and generate a report in .csv and .pdf file formats and save in the client computer. Therefore, the report can be sent to the offender for imposing penalty. The tabular (csv) can only display speed as shown in Table 6.1. However, the pdf report will save the details of the vehicle, speed, date and time (Figure 6.9).

Table 6.1: Speeding report in tabular format for a single vehicle running at slower speed limit

XAccel	YAccel	ZAccel	Resultant acceleration(Arbitrary unit)	Rotational Speed (RPM)	Linear Speed (km/h)
970	3008	1888	3681.51	20.75	41.49
1699	2822	1850	3777.93	23.79	47.58
1021	3054	1870	3723.74	22.08	44.15
1603	2908	1887	3819.27	25.10	50.19
1323	2953	1875	3739.81	22.59	45.17
1513	3075	1837	3888.36	27.28	54.56
2768	1744	1757	3713.54	21.76	43.51
2747	2036	1818	3872.52	26.78	53.56
1418	2984	1799	3761.83	23.28	46.56
1642	3112	1855	3977.65	30.10	60.20
970	3008	1888	3681.51	20.75	41.49

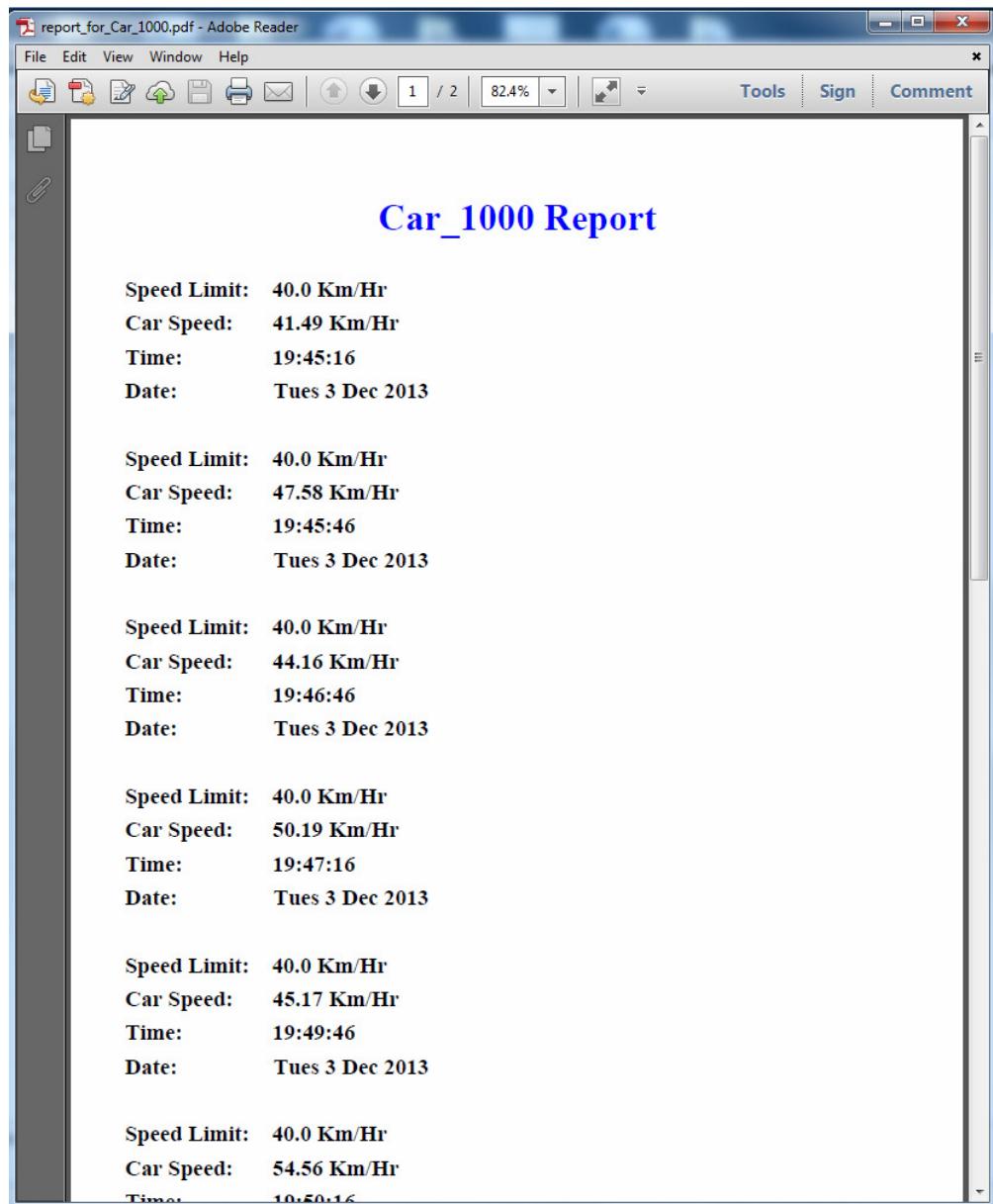


Figure 6.9: Example of speeding report in pdf format for a single vehicle running at slower speed limit

6.3.2 Vehicle Running at Higher Speed Limit

In order to monitor a vehicle running on highway, a new speed limit can be set in the software. For demonstration purpose, a speed limit of 60 km/hr has been considered (Figure 6.10). Similar to a single vehicle running inside a city, vibration acceleration data from the sensor are converted into vehicle speed format displayed in a DOS window of the server. In addition, a dialog box in the server displays the speed details (Figure 6.11). At the same time, the client computer can read all the data and display in a client DOS window.

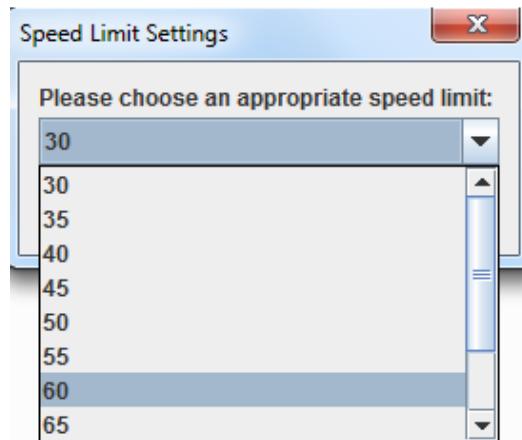


Figure 6.10: Selection of speed limit in the client computer for the scenario of single vehicle running at faster speed limit

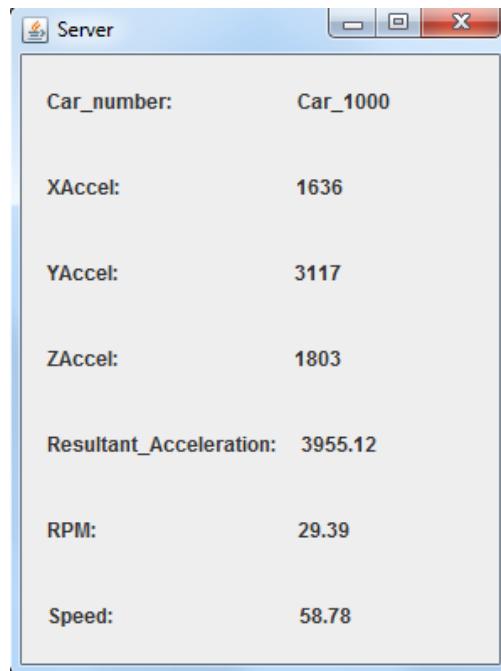


Figure 6.11: Display of latest vehicle speed data with vehicle details in a server dialog box for a single vehicle running at faster speed limit

A graphical user interface is also generated in the client computer where all the speeds of the vehicle will be updated one by one with a chart to monitor and detect the speeding data (Figure 6.12). The speed plot shows that at any particular moment, the vehicle is running below the speed limit. However, over the time the vehicle crosses the speed limit as indicated by the red points and returns below the limit. The speeding details of the vehicle are saved one by one as csv file format and also in pdf format, where vehicle details, speed, time and date are recorded.

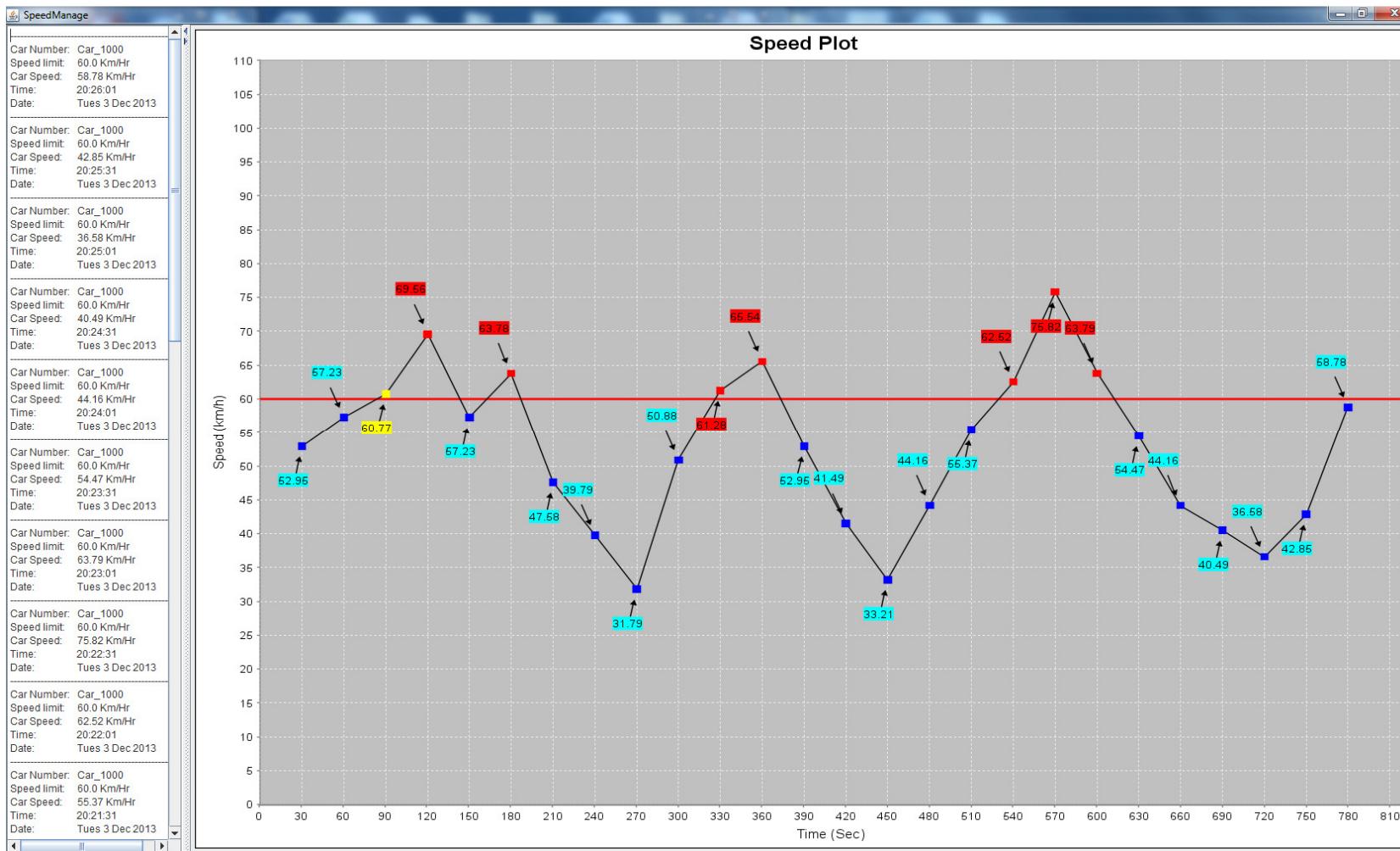


Figure 6.12: Display of a vehicle speed plot on a GUI in the client computer for a single vehicle running at faster speed limit

6.4 SIMULATING MULTIPLE VEHICLES SCENARIO

The system is also capable of monitoring and controlling multiple vehicle scenarios inside a city and on highway. The following sections demonstrate the functioning of the software in case multiple vehicle scenarios.

6.4.1 Multiple Vehicles Running at Slower Speed Limit

In this case, it is assumed that the maximum speed limit inside a city/town is 40 km/hr and ten different vehicles are running inside a city/town.

6.4.1.1. *Setting of Speed Limit for Multiple Vehicles*

After getting the connection between the server and client computers, a speed limit of 40 km/hr has been set from the Speed Limit Settings dialog box (Figure 6.13).

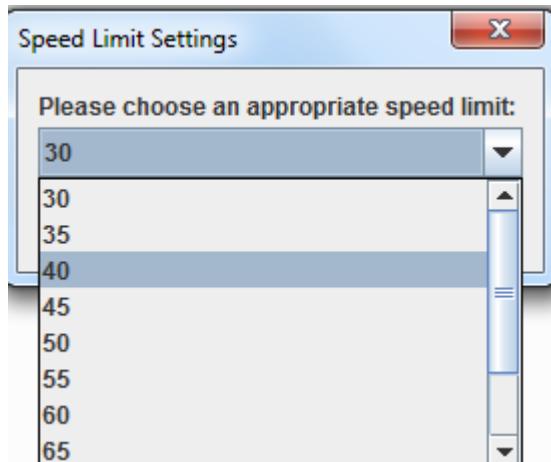


Figure 6.13: Selection of speed limit in the client computer for the scenario of multiple vehicles running at slower speed limit

6.4.1.2 *Displaying Multiple Vehicles Data in the Server Computer*

Similar to the single vehicle case, the server computer displays the vibration acceleration, rpm and speed data as shown in Figure 6.14. In this case, the data are shown for individual vehicles.

The server dialog box displays the vehicle details with other speed details in the server computers. The dialog box always displays the latest detected vehicle details (Figure 6.15).

```
C:\Windows\system32\cmd.exe - java MultipleCarServer
Car_number: Car_1250
XAccel_Average: 2477
YAccel_Average: 2116
ZAccel_Average: 1791
Resultant_Acceleration: 3712.808371031287
RPM: 21.73458761747866
Speed: 43.467649299550246

-----
Car_number: Car_2453
XAccel_Average: 3591
YAccel_Average: 2089
ZAccel_Average: 1760
Resultant_Acceleration: 4511.851283010113
RPM: 46.97682145032736
Speed: 93.95034476606737

-----
Car_number: Car_2201
XAccel_Average: 2267
YAccel_Average: 2125
ZAccel_Average: 1782
Resultant_Acceleration: 3581.9600779461516
RPM: 17.60101336111677
Speed: 35.200790995567786

-----
Car_number: Car_0151
XAccel_Average: 2111
YAccel_Average: 2119
ZAccel_Average: 1789
Resultant_Acceleration: 3485.255083921405
RPM: 14.546045930229184
Speed: 29.09107061602951

-----
Car_number: Car_3402
XAccel_Average: 3323
YAccel_Average: 2094
ZAccel_Average: 1769
Resultant_Acceleration: 4307.728635835828
RPM: 40.52846740912423
Speed: 81.05408940776715

-----
Car_number: Car_0302
XAccel_Average: 2732
YAccel_Average: 2110
ZAccel_Average: 1778
Resultant_Acceleration: 3882.9380628591025
RPM: 27.10908427923242
Speed: 54.216265292000494

-----
Car_number: Car_1103
XAccel_Average: 1845
YAccel_Average: 2130
ZAccel_Average: 1796
Resultant_Acceleration: 3341.637422856683
RPM: 10.009081436003246
Speed: 20.017460157419755

-----
Car_number: Car_4502
XAccel_Average: 3816
YAccel_Average: 2076
ZAccel_Average: 1754
Resultant_Acceleration: 4684.885057287958
RPM: 52.443059778485484
Speed: 104.88243765036296

-----
Car_number: Car_1350
XAccel_Average: 3033
YAccel_Average: 2105
ZAccel_Average: 1774
Resultant_Acceleration: 4095.9968261706454
RPM: 33.83973546582357
Speed: 67.67709512165148
```

Figure 6.14: Display of vehicle speed data in a DOS window of the server computer for multiple vehicles running at slower speed limit

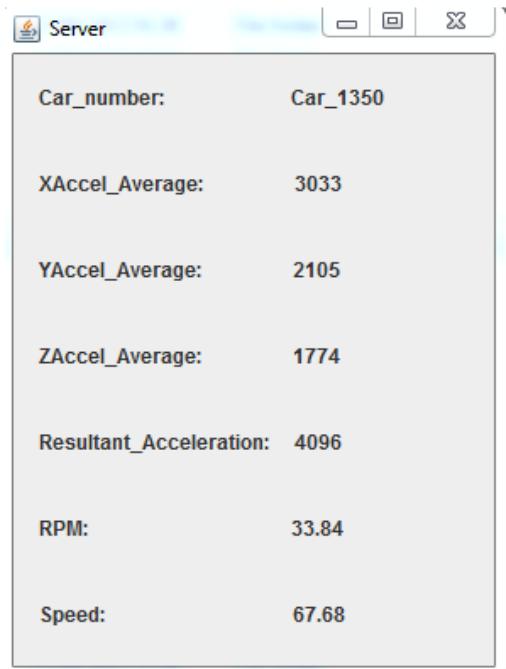


Figure 6.15: Display of latest vehicle details in a server dialog box for multiple vehicles running at slower speed limit

6.4.1.3 Displaying Multiple Vehicles Data in the Client Computer

When multiple vehicle speed details are displayed in the server computer, at the same time client computer can read all the data and can display the same information in a DOS window of the client computer. Figure 6.16 displays the multiple vehicles data in a DOS window of the client computer.

Similar to the previous cases, the system generates a graphical user interface in the client computer where speeds of different vehicles are updated and a speed plot is created as well to monitor and detect the speeding vehicles (Figure 6.17). The speed plot shows that at any particular moment six vehicles are running above the speed limit and four below the limit. Each vehicle can be identified with a car number and speed at which it is running. For example, a vehicle '1052_Car' is running at the lowest speed of 19.94 km/h and another vehicle '4502_Car' is running at the highest speed of 104.88 km/h. The speed plot follows the same colour convention as in the previous cases. The vehicle speed details are also updated in the left hand side of the chart.

```
C:\Windows\system32\cmd.exe - java MultipleCarClientV2
Car_number: 2453
XAccel_Average: 3591
YAccel_Average: 2089
ZAccel_Average: 1760
Resultant_Acceleration: 4511.851283010113
RPM: 46.97682145032736
Speed: 93.95034476606737
Plotting X = 90.0 Y=93.95034476606737
-----
Car_number: 2201
XAccel_Average: 2267
YAccel_Average: 2125
ZAccel_Average: 1782
Resultant_Acceleration: 3581.9600779461516
RPM: 17.60101336111677
Speed: 35.200790995567786
Plotting X = 120.0 Y=35.200790995567786
-----
Car_number: 0151
XAccel_Average: 2111
YAccel_Average: 2119
ZAccel_Average: 1789
Resultant_Acceleration: 3485.255083921405
RPM: 14.546045930229184
Speed: 29.09107061602951
Plotting X = 150.0 Y=29.09107061602951
-----
Car_number: 3402
XAccel_Average: 3323
YAccel_Average: 2094
ZAccel_Average: 1769
Resultant_Acceleration: 4307.728635835828
RPM: 40.52846740912423
Speed: 81.05408940776715
Plotting X = 180.0 Y=81.05408940776715
-----
Car_number: 0302
XAccel_Average: 2732
YAccel_Average: 2110
ZAccel_Average: 1778
Resultant_Acceleration: 3882.9380628591025
RPM: 27.18908427923242
Speed: 54.216265292008494
Plotting X = 210.0 Y=54.216265292008494
-----
Car_number: 1103
XAccel_Average: 1845
YAccel_Average: 2130
ZAccel_Average: 1796
Resultant_Acceleration: 3341.637472856683
RPM: 10.009081436003246
Speed: 20.017460157419755
Plotting X = 240.0 Y=20.017460157419755
-----
Car_number: 4502
XAccel_Average: 3816
YAccel_Average: 2076
ZAccel_Average: 1754
Resultant_Acceleration: 4684.885057287958
RPM: 52.443059778485484
Speed: 104.88243765036296
Plotting X = 270.0 Y=-104.88243765036296
-----
Car_number: 1350
XAccel_Average: 3033
YAccel_Average: 2105
ZAccel_Average: 1774
Resultant_Acceleration: 4095.9968261706454
RPM: 33.83973546582357
Speed: 67.67709512165148
Plotting X = 300.0 Y=67.67709512165148
```

Figure 6.16: Display of vehicle speed data in a DOS window of the client computer for multiple vehicles running at slower speed limit

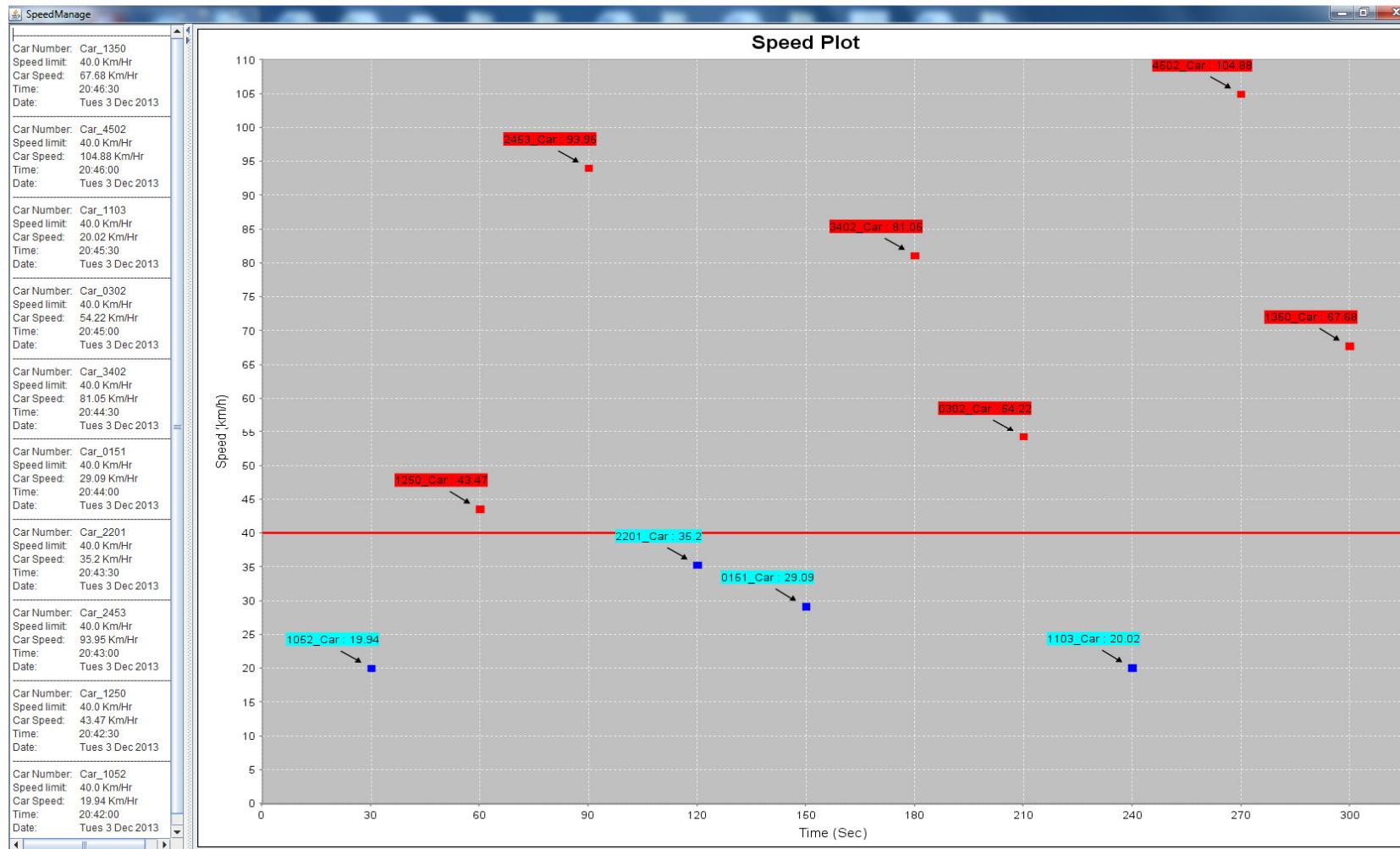


Figure 6.17: Display of a vehicle speed plot on a GUI in the client computer for multiple vehicles running at slower speed limit

6.3.1.4 Report Generation for Multiple Vehicles

The speeding vehicles and speed details are saved one by one in a csv file while the status of the vehicles is displayed on the speed plot in the client computer. Table 6.2 presents the report in a tabular format for multiple vehicles running inside a city/town. It should be noted that the table lists only the vehicles, which crossed the speed limit.

Table 6.2: Speeding report in tabular format for multiple vehicles running at slower speed limit

Car number	XAccel (Avg)	YAccel (Avg)	ZAccel (Avg)	Resultant Acceleration (Arbitrary Unit)	Rotational Speed (RPM)	Linear Speed (km/h)
1250	2477	2116	1781	3712.81	21.73	43.47
2453	3591	2089	1760	4511.85	46.98	93.95
3402	3323	2094	1769	4307.73	40.53	81.05
0302	2732	2110	1778	3882.94	27.11	54.22
4502	3816	2076	1754	4684.88	52.44	104.88
1350	3033	2105	1774	4095.99	33.84	67.68

At the same time, speeding vehicle details are also saved in individual pdf file for different vehicles, which cross the speed limit. The speed limit, vehicle speed, time and date for different vehicles are saved in individual pdf report as shown in Figure 6.18.

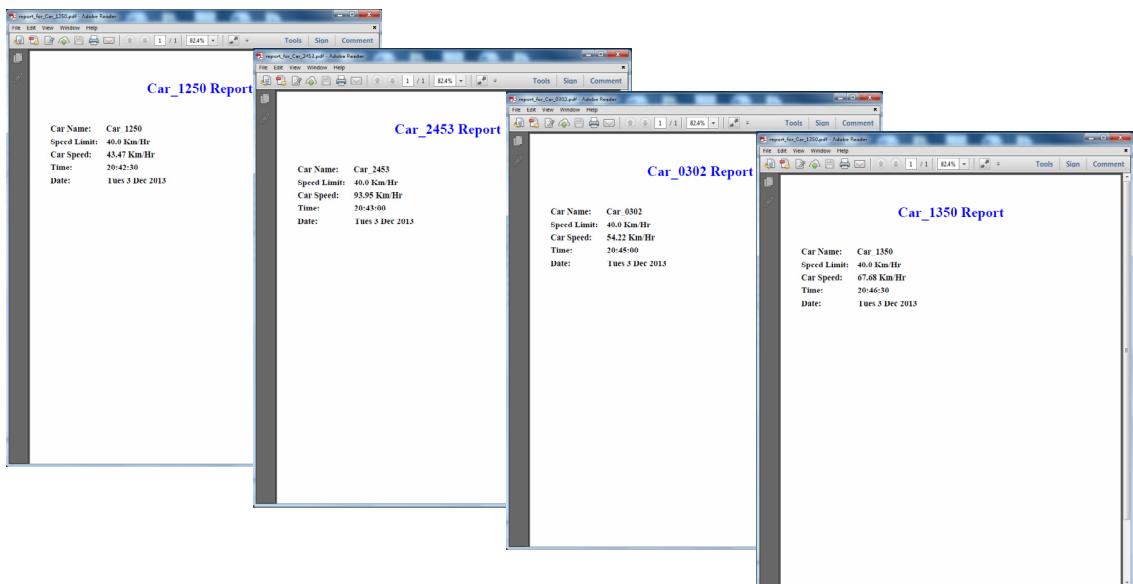


Figure 6.18: Example of speeding report in pdf format for multiple vehicles running at slower speed limit

6.4.2 Multiple Vehicles Running at Faster Speed Limit

In the case of multiple vehicles running on the highway, 60 km/hr has been considered the maximum speed limit. After getting connection between the server and client computers, a new speed limit of 60 km/hr has been set in the Speed Limit Settings dialog box (Figure 6.19).

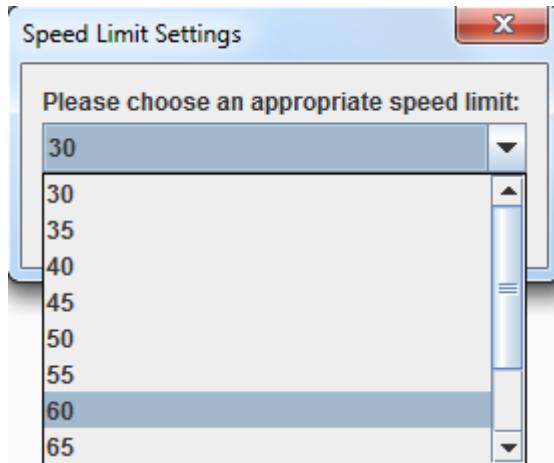


Figure 6.19: Selection of speed limit in the client computer for the scenario of multiple vehicles running at faster speed limit

The server processes the raw vibration acceleration data into speed format and displays in both a DOS window and a server dialog box (Figure 6.20). The dialog box always displays the latest detected vehicle details.

The client computer also reads all the data and displays the same information in a DOS window or in a graphical user interface of the client computer. The speed plot in Figure 6.21 indicates that at any particular moment four vehicles are running over the speed limit and other six vehicles running below the speed limit. At the same time, the details of the speeding vehicles are saved in csv file and in separate pdf files. Table 6.3 shows the report of the four speeding vehicles.

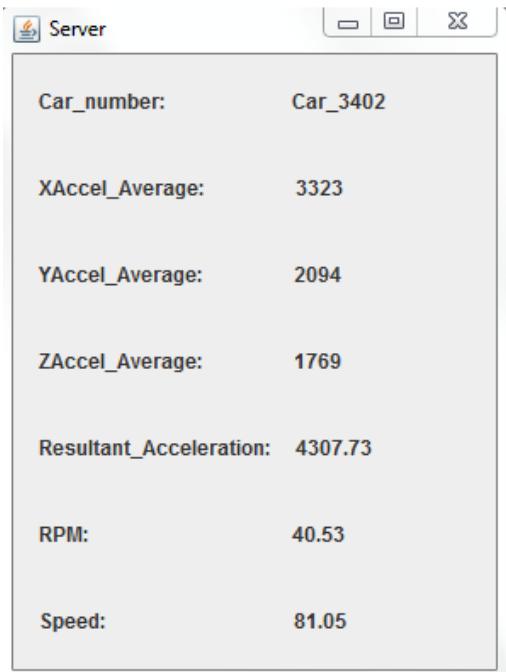


Figure 6.20: Display of latest vehicle speed data with vehicle details in a server dialog box for multiple vehicles running at faster speed limit

Table 6.3: Report for over speed vehicles details in .csv format running at faster speed limit

Vehicle number	XAccel (Avg)	YAccel (Avg)	ZAccel (Avg)	Resultant Acceleration (Arbitrary unit)	Rotational Speed (RPM)	Linear Speed (km/h)
2453	3591	2089	1760	4511.85	46.98	93.95
3402	3323	2094	1769	4307.73	40.53	81.05
4502	3816	2076	1754	4684.89	52.44	104.88
1350	3033	2105	1774	4095.99	33.84	67.68

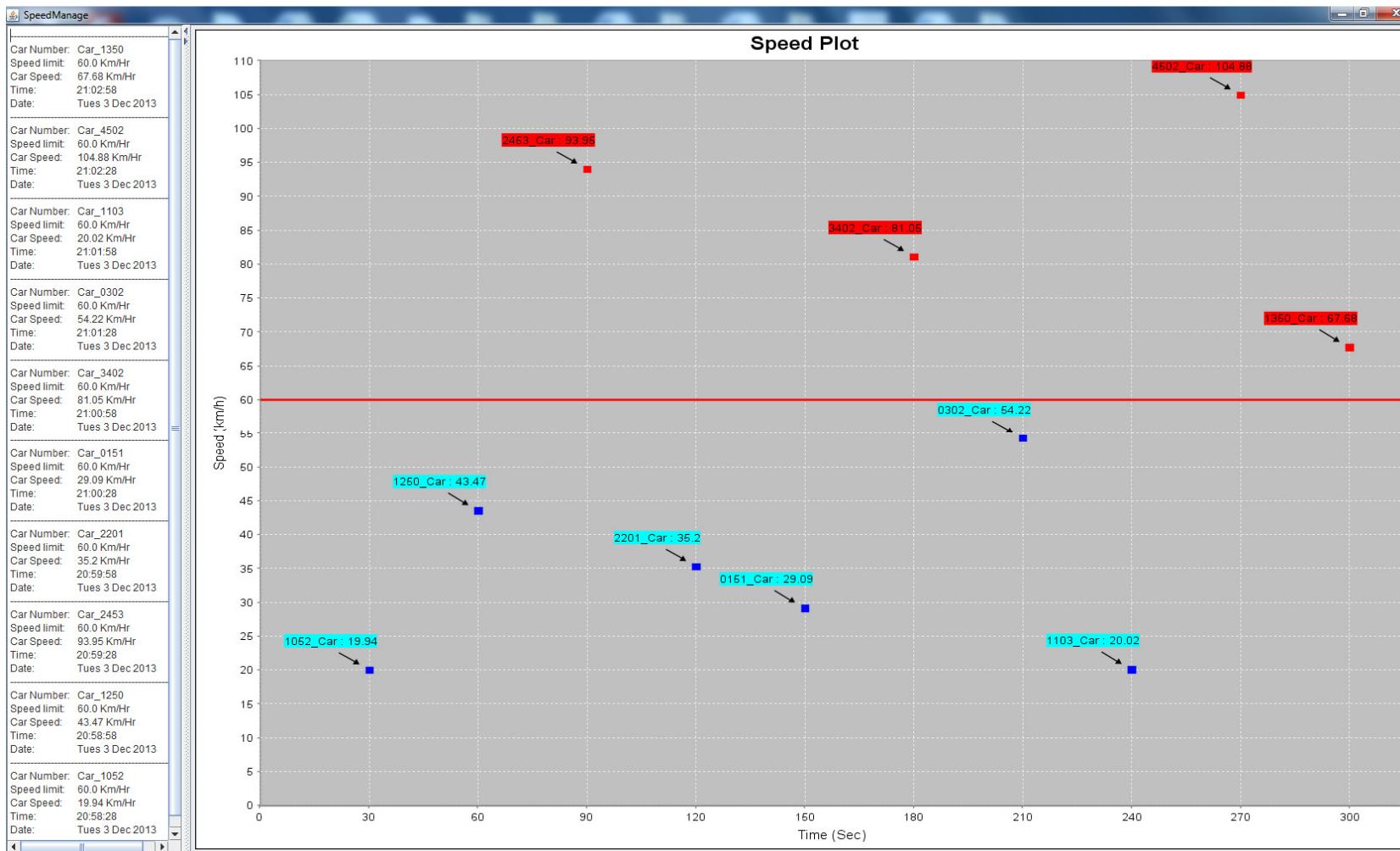


Figure 6.21: Display of a vehicle speed plot on a GUI in the client computer for multiple vehicles running at faster speed limit

6.5 SUMMARY

The SpeedManage software developed in this work can detect speeding vehicles and display the information both in a DOS window and in a graphical user interface. The software also automatically generates report for the speeding vehicle in csv or pdf file formats. It would be usable for different road traffic conditions with the capability of monitoring single and multiple vehicles.

Chapter 7. Conclusions and Suggestion for Future Work

This chapter presents general conclusions of the whole project work and thesis contribution. Furthermore, the chapter provides future directions on the use of wireless sensor network in the intelligent transportation system for monitoring vehicle speed and managing the relevant data in an intelligent way.

7.1. CONCLUSIONS

The following conclusions can be drawn based on the work carried out in the project.

1. An experimental setup with a wireless sensor has been developed to simulate the detection and measurement of vehicle speed in a laboratory environment. A wireless sensor attached to a mechanical wheel has been used to measure the wheel speed by sensing the wheel vibration acceleration. The wireless sensor also transfers the wheel vibration acceleration data to a server computer wirelessly.
2. A software (SpeedManage) has been developed with Java Socket programming language to convert the wheel vibration acceleration data into speed format automatically and to transfer from a server computer to a client computer wirelessly. The software is capable of detecting simulated speeding vehicle by comparing with a set speed limit and generating a report in the event of speeding.
3. The software has also been successfully tested with different speed limit scenarios for single and multiple vehicles. The graphical user interface displays the data in red colour to identify the speeding vehicle inside and outside the city with speed limit set at 40 km/h and 60 km/h respectively.

7.2. THESIS CONTRIBUTION

This section presents the main contributions in this project.

1. The current state-of-the-art research activities related to the application of wireless sensor network in road traffic monitoring and controlling has been documented.
2. A novel concept of road traffic monitoring and data management using wireless sensor network has been proposed.

3. An experimental set-up has been developed to simulate the real life traffic situation for monitoring and controlling.
4. A software has been developed with Java socket programming language for vehicle detection, speed measurement and monitoring, and collating and managing vehicle speed data.

7.3. SUGGESTIONS FOR FUTURE WORK

The following work can be carried out as a follow up of the current work.

1. For practical applications, the sensor can be attached with the wheel of a vehicle to get the wheel vibration, which can be used for measuring vehicle speed.
2. Multiple sensors can be placed beside the road to measure the vehicle speed. For example, two sensors can be set up on the road with a known distance apart (Figure 7.1). Two highest vibration peaks can be seen while the vehicle will be at the nearest position of the sensors. Then the time between the appearances of the two peaks can be calculated. Thus speed can be determined by the dividing the distance between two sensors with the calculated time.

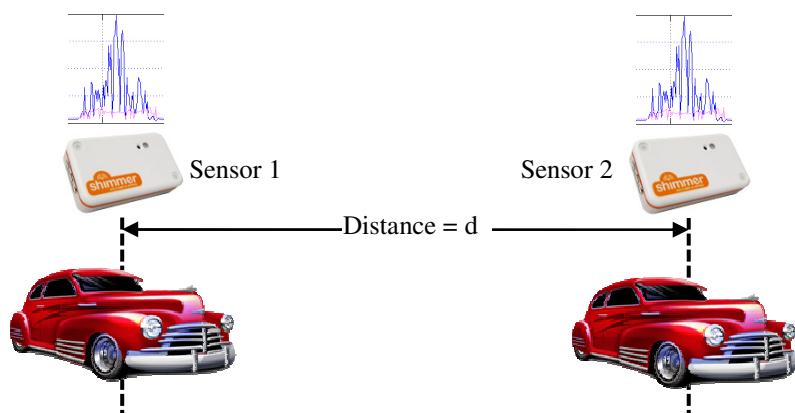


Figure 7.1: Illustration of vechicle speed measurement by vibration based wireless sensors

3. A GPS system can be added to determine the particular speed violation location of vehicles as a feature of identical proofs of the offenders.
4. The data can be processed in real time though the current system is unable to do that. The sensor has limited access to send data from the server computer to the client

computer in real time. In future, the sensor accessibility can be developed to manage data completely in real time mode.

5. A wireless printer can be integrated with this system to generate hard copy of report automatically. That may reduce the manual work for printing the reports of speeding vehicles manually.
6. The graphical user interface can be further developed to make it more user-friendly by adding new features.

(i) LIST OF REFERENCES

- [1] <http://www.roadsafetymayo.ie/CausesofAccidents/> (accessed December 2013)
- [2] <http://www.transportscotland.gov.uk/strategy-and-research/publications-and-consultations/j10724c-04.htm> (accessed December 2013)
- [3] Cheung, S.Y. and Varaiya, P., (2007) Traffic surveillance by wireless sensor network: final report, California PATH Research Report, UCB-ITS-PRR-2007-4, California Path Program Institute of Transportation Studies University of California, Berkeley
- [4] <http://justicescale.blogspot.co.uk/2012/03/california-traffic-accidents-in.html> (accessed December 2013)
- [5] Georgoulas, D. and Blow, K. (2006) In-Motes: An Intelligent Agent Based Middleware for Wireless Sensor Networks. In: Proceedings of the 5th WSEAS International Conference on Applications of Electrical Engineering, Prague, Czech Republic, pp. 225-231
- [6] Trichias, K. (2011) Modelling and evaluation of LTE in Intelligent Transportation Systems, University of Twente, Enschede, The Netherlands.
- [7] Intelligent Transportation System. <http://www.oring-networking.com/doc/view/sn/276/Intelligent%20Transportation%20System> (accessed December 2013)
- [8] Tate, F. and Carsten, O. (2008) Intelligent speed adaptation, Implementation scenarios, Transport Technology and Standards Division, Department for Transport, Project Partners: The University of Leeds and MIRA Ltd.
- [9] Bamashak, O. Al-Mehmadi, G., Al-Ghamdi, A., Al-Mutawa, A. and Al-Ghamdi, M. (2011) Monitored Intelligent Car Speed Adaptation, International Journal of Electrical and Computer Sciences, Vol. 11, No. 3, pp. 1-13
- [10] Long, C. and Shuai, M. (2010) Wireless Sensor Networks: Traffic Information Providers for Intelligent Transportation System, In: 18th International Conference on Geoinformatics, Beijing, China, pp. 1-5
- [11] Duvvuru, S.K. (2008) Design and Deployment of Vehicular Traffic Monitoring Sensor Networks, Dissertation, M. Tech, Indian institute of technology, Bombay
- [12] Fraden, J. (2003) Handbook modern sensors, 3rd edition, Advanced Monitors Corporation, San Diego, USA

- [13] Dhar, A. (2008) Traffic and road condition monitoring system. M. Tech. Dept. of Computer Science and Engineering Indian Institute of Technology, Bombay, pp. 1-22
- [14] Dargie, W. and Poellabauer, C. (2010) Fundamental of wireless sensor network, Wiley Series on Wireless Communications and Mobile Computing, John Wiley and Sons Ltd.
- [15] Molisch, A. F. (2011) Wireless Communications, Second Edition, John Wiley and Sons Ltd.
- [16] Mukhopadhyay, S. C., Gaddam, A. and Gupta, G. S. (2008) Wireless Sensors for Home Monitoring - A Review, Recent Patents on Electrical Engineering, Vol. 1, pp. 32-39
- [17] Howitt, I. and Gutierrez, J. A. (2003) IEEE802.15.4 Low rate-wireless personal area network coexistence issues, Wireless Communications and Networking, Vol. 3, pp. 1481–1486
- [18] ISA100.11,<http://www.isa.org/MSTemplate.cfm?MicrositeID=1134&CommitteeID=6891> (accessed December, 2013)
- [19] Stallings, W. (2005) Wireless Communication and Networks, Pearson Education Ltd.
- [20] Rackley, S. (2007) Wireless Networking Technology, Elsevier
- [21] Ng, E.H., Tan, S.L. and Guzman, J. G. (2008) Road traffic monitoring using a wireless vehicle sensor network, In: International Symposium on Intelligent Signal Processing and Communications Systems, ISPACS, Banhkok, Thailand, pp. 1-4
- [22] Megalingam, R.K., Mohan, V., Leons, P., Shooja, R. and Ajay, M. (2011) Smart Traffic Controller using Wireless Sensor Network for Dynamic Traffic Routing and Over Speed Detection, Global Humanitarian Technology Conference (GHTC), Amritapuri, India, pp. 528-533
- [23] Kameswari, U. J., Satwik, M. and Lokesh, A. (2011) A Design Model for Automatic Vehicle Speed Controller, International Journal of Computer Applications, Vol. 35, No. 9, pp. 19-24
- [24] Stankovic, J. A. (2008) Wireless Sensor Networks, Chapter in Handbook of Real-Time and Embedded Systems, CRC Press
- [25] Culler, D., Estrin D. and Srivastava M. (2004) Overview of Sensor Networks, Computer, In: IEEE Computer Society, Washington, DC, pp. 40-49

- [26] Verdone, R., Dardari, D., Mazzini G. and Conti A. (2008) Wireless Sensor and Actuator Networks, Academic Press/Elsevier, London
- [27] <http://www.omega.com/prodinfo/wirelessensors.html> (accessed December 2013)
- [28] Yu, H. and Guo, M. (2010) An Efficient Freeway Traffic Information Monitoring Systems Based on Wireless Sensor Networks and Floating Vehicles, In: First International Conference on Pervasive Computing Signal Processing and Applications (PCSPA), Jingzhou, China, pp. 1065-1068
- [29] Pathan, A. S. K., Hong, C. S. and Lee, H. W. (2006) Smartening the Environment using Wireless Sensor Networks in a Developing Country, The 8th International Conference on Advanced Communication Technology, 20-22 Feb. pp. 705- 709
- [30] Sun Z., Bebis, G. and Miller R., (2004) On-road vehicle detection using optical sensors: a review, In: 7th International IEEE Conference on Intelligent Transportation Systems, 3-6 October, NV, USA, pp. 585-590
- [31] Zengqiang, M., Guosheng G., Wanmin, S. and Yan, Y. (2008) Wireless Monitoring System of Vehicle Overspeed on Freeway Based on GPRS, In: Chinese 27th Control Conference, 16-18 July 2008, Beijing, China 2008, pp. 550-553
- [32] Arnold, D.V., Dougall, Jr., J.B., Giles, B.C., Jarrett, B.R., Karlinsey, T.W., Waite, J.L. (2008) Systems and methods for monitoring speed. US Pat. No. 7426450B2
- [33] Wang, Y., Lynch, J. P. and Law, K. H. (2005) Wireless Structural Sensors using Reliable Communication Protocols for Data Acquisition and Interrogation, In: Proceedings of the 23rd International Modal Analysis Conference, 31st January-3rd February 2005
- [34] Dondi, D., Di Pompeo, A., Tenti, C., and Rosing, T.S., (2010) Shimmer: a wireless harvesting embedded system for active ultrasonic structural health monitoring. In: Proceedings of the IEEE Sensors 2010 Conference, Kona, HI, USA, 1-4 Nov., pp. 2325–2328.
- [35] Weissman, I. (1997) Method and system for regional traffic monitoring. US Pat. No. 5663720
- [36] Matsui, T., Yuzawa, H. and Kobayashi, A. (2004) Road traffic monitoring system. US Pat. No. 6781523B2

- [37] Coleri, S., S.Y. Cheung, P. varaiya, (2004) Sensor Networks for Monitoring Traffic, In: 42nd Annual ALLERTON Conference on Communication, Control, and Computing, Invited paper, September 2004.
- [38] Mandhyan, I.B. and Trovato, K.I. (1996) Traffic monitoring system with reduced communications requirements. US Patent No. 5,539,645,
- [39] Hassett, J.J., Kowal, K.E., Rourke, J. A. (1994) Traffic monitoring and management method and apparatus. US Pat. No. 5,289,183
- [40] Kilger, M., (1992a) A shadow handler in a video-based real-time traffic monitoring system. In: Proceedings of the IEEE Workshop on Applications of Computer Vision, 30 Nov-2 Dec, Palm Springs: CA, pp. 11–18
- [41] Kilger, M., (1992b) Video-based traffic monitoring. In: Proceedings of the 4th International Conference on Image Processing and its Applications, Maastricht: Netherlands, 7-9 April, pp 89-92
- [42] Robert, K. (2009) Video-based traffic monitoring at day and night vehicle features detection tracking. In: Proceedings of the 12th International IEEE Conference on Intelligent Transportation Systems, St. Louis: MO, 4-7 Oct., pp. 1–6
- [43] Laparmonpinyo, P. and Chitsobhuk, O. (2010) A video-based traffic monitoring system based on the novel gradient-edge and detection window techniques. In: Proceedings of the 2nd International Conference on Computer and Automation Engineering, Singapore, 26-28 Feb., Vol. 3, pp. 30–34
- [44] Dell'Acqua, G., Luca, M. d., Mauro, R., and Lamberti, R. (2011) Motorway Speed Management in Southern Italy. Procedia - Social and Behavioral Sciences, Vol. 20, pp. 49–58.
- [45] Dickinson, K.W. and Wan, C.L. (1989) Road traffic monitoring using the TRIP II system. In: Second International Conference on Road Traffic Monitoring, London, 07-09 Feb, pp. 56–60
- [46] Derry, J.D., Afukaar, F.K., Donkor, P., and Mock, C. (2007). Study of vehicle speeds on a major highway in Ghana: Implication for monitoring and control. Traffic Injury Prevention, Vol. 8, No. 2, pp. 142-146
- [47] Andrews, J.W. (1994) The IVHS potential of surface detection radar. In: Proceedings of the IVHS American Annual Meeting, V1, Atlanta GA, Publisher – ITS America: Washington, DC, pp. 894-901

- [48] Hongyu, Z., Chunyan, X and Yimin, L. (2011) Application of Virtual Instrument in Vehicle Detection System, International Conference on Mechatronic Science, Electric Engineering and Computer, August 19-22, Jilin, China, pp. 442-445
- [49] Yu, M., Zhang, D., Cheng, Y. and Wang, M. (2011) An RFID electronic tag based automatic vehicle identification system for traffic iot applications, Control and Decision Conference (CCDC), Mianyang, China, 23-25 May, pp. 4192-4197
- [50] Runge, H., Eineder, M., Palubinskas, G., Suchandt, S. and Meyer, F. (2005) Traffic monitoring with TerraSAR-X, In: Proceedings of the International Radar Symposium IRS2005, Berlin, pp. 629–634
- [51] Suchandt, S., Eineder, M., Breit, H., Runge, H., (2006) Analysis of ground moving objects using SRTM/X-SAR data. ISPRS Journal of Photogrammetry and Remote Sensing, Vol. 61, No. 3-4, pp. 209–224
- [52] Breit, H., Eineder, M., Holzner, J., Runge, H., and Bamler, R. (2003) Traffic monitoring using SRTM along-track interferometry. In: Proceedings of the IEEE International Conference on Geoscience and Remote Sensing Symposium, 21–25 July, Vol. 2, pp. 1187–1189
- [53] Luciani, S. (2003) Traffic monitoring system and method. US Pat. No. 6,505,114B2
- [54] Wen Y., Pan J. L. and Le J. F., (2007) Survey on Application of Wireless Sensor Networks for Traffic Monitoring, International Conference on Transportation Engineering (ICTE 2007), pp. 442-445
- [55] Tubaishat, M., Zhuang, P., Qi, Q. and Shang Y., (2008) Wireless sensor networks in intelligent transportation systems, Wireless Communications and Mobile Computing, Vol. 9, pp. 287–302
- [56] Mohan, P., Padmanabhan, V.N. and Ramjee, R. (2008) Nericell: Rich Monitoring of Road and Traffic Conditions using Mobile Smartphones, In: Proceedings of the 6th ACM conference on Embedded network sensor systems, pp. 323-336
- [57] Ma, W., Xing, D., McKee, A., Bajwa, R., Flores, C., Fuller, B., and Varaiya, P. (2013) A wireless accelerometer-based automatic vehicle classification prototype system, IEEE Transactions on Intelligent Transportation Systems, Vol. PP, No. 99, pp. 1–8

- [58] Mohan, P., Padmanabhan, V.N., and Ramjee, R., (2008) TrafficSense: Rich monitoring of road and traffic conditions using mobile smartphones. Technical Report MSR-TR-2008-59, April, pp. 1-29
- [59] Sensys Networks, Inc. (2010) Sensys™ Wireless Vehicle Detection System. Reference Guide P/N 152-240-001-001 Rev D.
- [60] Hisao, O. and Takashi, A. (1998) Vehicle detecting device and traffic quantity measuring equipment. Japanese Pat. No 2729977(B2)
- [61] Matsuo, T., Kaneko, Y. and Matano, M. (1998) Introduction of intelligent vehicle detection sensors, Japan, Vol. 98, No. 23-29, pp. 19-23
- [62] Chen, S., Sun, Z.P. and Bridge, B. (1997) Automatic traffic monitoring by intelligent sound detection. In: IEEE Conference on Intelligent Transportation System, Boston: MA, 9-12 Nov, pp. 171 - 176
- [63] Coifman, B., and Kim, S. (2009) Speed estimation and length based vehicle classification from freeway single-loop detectors. *Transportation Research Part C: Emerging Technologies*, Vol. 17, No. 4, pp. 349–364
- [64] Bachmann, C., Abdulhai, B., Roorda, M.J. and Moshiri, B. (2013). A comparative assessment of multi-sensor data fusion techniques for freeway traffic speed estimation using microsimulation modeling. *Transportation Research Part C: Emerging Technologies*, Vol. 26, pp. 33–48
- [65] Zhang, L., Wang, R., and Cui, L. (2011) Real-time traffic monitoring with magnetic sensor networks. *Journal of Information Science & Engineering*, Vol. 27, No. 4, pp. 1473-1486
- [66] Zhang, P. C., Shi, Z. K. and Xu, M. (2010) Design and implementation of vehicle monitoring system based on GPRS system, Second International Conference on Information Technology and Computer Science (ITCS), 24-25 July, Nanjing, China, pp. 413-416
- [67] Karthikeyan, B., and Tamileniyam, M. (2010) Dynamic Data update for Intelligent Speed Adaptation (ISA) System, Vol. 11, No.1, pp. 8-13
- [68] Lapidot, D. and Silverberg, J. (2002) Traffic monitoring system and methods for traffic monitoring and route guidance useful therewith. US Pat. No. 6490519B1
- [69] Endo, S., Ukawa, H., Sanda, K. and Kitagawa, A. (1999) Simulation of speed control in acceleration mode of a heavy-duty vehicle, Vol. 20, pp. 81-86
- [70] Smith, B.L., Zhang, H., Fontaine, M., and Green, M. (2004) Wireless location technology-based traffic monitoring: Critical assessment and evaluation of an

- early-generation system. *Journal of Transportation Engineering*, Vol. 130, No. 5, pp. 576–584
- [71] Fontaine, M.D. and Smith, B.L. (2007) Investigation of the performance of wireless location technology-based traffic monitoring systems. *Journal of Transportation Engineering*, Vol. 133, No. 3, pp. 157–165
- [72] Herrera, J. C., Work, D. B., Herring, R., Ban, X., Jacobson, Q. and Bayen, A. M. (2010) Evaluation of traffic data obtained via GPS-enabled mobile phones: The Mobile Century field experiment, Volume 18, 4th August, pp. 568-583
- [73] Sridharamurthy, K., Govinda, A. P., Gopal, J.D. and Varaprasad, G. (2012) Violation detection method for vehicular ad hoc networking, *Security and Communication Networks*, DOI: 10.1002/sec.427
- [74] Ghadiri, S.M.R., Prasetijo, J., Sadullah, A.F. and Hoseinpour M. (2011) Intelligent Speed Adaptation, Results of Pre-Test in Penang, Malaysia, 3rd International Conference on Road Safety and Simulation, September 14-16, Indianapolis, USA, pp. 1-13
- [75] Agerholm, N., Waagepetersen, R., Tradisauskas, N. and Lahrmann, H. (2008) Intelligent Speed Adaptation in Company Vehicles, *Intelligent Vehicles Symposium*, 2008 IEEE, 4-6 June, Aalborg, pp. 936-943
- [76] Megalingam, R.K., Mohan, V., Mohanan, A., Leons, P., Shooja, R. and Vidyapeetham, A.V. (2010) Wireless Sensor Network for Vehicle Speed Monitoring and Traffic Routing System, International Conference on Mechanical and Electrical Technology (ICMET 2010), 10-12 Sept, Kerala, India, pp. 631-635
- [77] Murty, R.N., Mainland, G., Rose, I., Chowdhury, A.R., Gosain, A., Bers, J., Welsh, M., (2008) CitySense: An urban-scale wireless sensor network and testbed. In: Proceedings of the IEEE Conference on Technologies for Homeland Security, Waltham: MA 12-13 May, pp. 583–588
- [78] Atluri, M., Chowdhury, M., Kanhere, N., Fries, R., Sarasua, W., and Ogle, J. (2009) Development of a sensor system for traffic data collection, *Journal of Advanced Transportation*, Vol. 43, No. 1, pp. 1–20
- [79] Pornpanomchai, C. and Kongkittisan, K., (2009) Vehicle Speed Detection System, *IEEE International Conference on Signal and Image Processing Applications*, pp. 135-139

- [80] Roetenberg, D., Slycke, P.J., and Veltink, P.H. (2007) Ambulatory position and orientation tracking fusing magnetic and inertial sensing. *IEEE Transactions on Biomedical Engineering*, Vol. 54, No. 5, pp. 883–890
- [81] Bamberg, S., Benbasat, A.Y., Scarborough, D.M., Krebs, D.E., and Paradiso, J.A. (2008) Gait analysis using a shoe-integrated wireless sensor system. In: *IEEE Transactions on Information Technology in Biomedicine*, Vol. 12, No. 4, pp. 413–423
- [82] Otto, C., Milenkovic, A., Sanders, C., and Jovanov, E. (2006) System architecture of a wireless body area sensor network for ubiquitous health monitoring. *Journal of Mobile Multimedia*, Vol. 1, No. 4, 307–326
- [83] Westeyn, T., Presti, P., and Starner, T. (2006) ActionGSR: A Combination Galvanic Skin Response-Accelerometer for Physiological Measurements in Active Environments. In:: Proceedings of the 10th IEEE International Symposium on Wearable Computers, Montreux, Switzerland, 11-14 Oct., pp. 129–130
- [84] Bellis, S.J., Delaney, K., O'Flynn, B., Barton, J., Razeeb, K.M., and O'Mathuna, C. (2005) Development of field programmable modular wireless sensor network nodes for ambient systems. *Computer Communications*, Vol. 28, No. 13, pp. 1531–1544
- [85] Benini, L., Farella, E. and Guiducci, C., (2006) Wireless sensor networks: Enabling technology for ambient intelligence. *Microelectronics Journal*, Vol. 37, No. 12, pp. 1639–1649
- [86] Burns, A., Greene, B.R., McGrath, M.J., O'Shea, T.J., Kuris, B., Ayer, S.M., Stroiescu, F., and Cionca, V. (2010) SHIMMER™ – A wireless sensor platform for noninvasive biomedical research *IEEE Sensors Journal*, Vol. 10, No. 9, pp. 1527-1534
- [87] Patel, S., Lorincz, K., Hughes, R., Huggins, N., Growdon, J., Standaert, D., Akay, M., Dy, J., Welsh, M., Bonato, P. (2009) Monitoring motor fluctuations in patients with Parkinson's disease using wearable sensors. *IEEE Transactions on Information Technology in Biomedicine*, Vol. 13, No. 6, pp. 864 - 873.
- [88] Chen, B., Varkey, J.P., Pompili, D., Li, J.K.-J. and Marsic, I. (2010) Patient vital signs monitoring using wireless body area networks, In: Proceedings of the 36th IEEE Annual Northeast Bioengineering Conference, New York, USA, 26-28 March, pp. 1–2

- [89] O'Connell, E., O'Keeffe, S., Newe, T., Healy, M., Lewis, E., Lyons, W.B. (2009) Development of a prototyping platform for the integration of multiple fiber optic sensing devices to a SHIMMER™ system for in-situ maritime monitoring. In: Proceedings of the Eighth IEEE Conference on Sensors, Christchurch, New Zealand, 25–28 Oct., pp. 1800–1803
- [90] Nadeem, T., Dashtinezhad, S., Liao, C. and Iftode, L. (2004) TrafficView: a scalable traffic monitoring system, In: Proceedings of the IEEE International Conference on Mobile Data Management, pp. 13–26.
- [91] Huang, Q. and Miller, R. (2003) The design of reliable protocols for wireless traffic signal system. Technical Report WUCS-02-45, Washington University, Department of Computer Science and Engineering, St. Louis, USA
- [92] Chen, X. Zhang, J. Qian, S. and Xu, P (2012) Applied Research on Traffic Information Collection Based on Wireless Sensor Networks. Energy Procedia, Vol. 17, Part A, pp. 602-606
- [93] Douxchamps, D., Macq B. and Chihara, K. (2006) High accuracy traffic monitoring using road-side line-scan cameras, In: Proceedings of the IEEE ITSC Intelligent Transportation Systems Conference, Toronto, Canada, pp. 875 - 878
- [94] Roberts, Jr., E.V. (2002) Traffic monitoring system and method. US Pat. No. 6384739B1
- [95] Mizunuma, I. and Masaki, I. (2002) Integrated traffic monitoring assistance, and communications system. US Pat. No. 6,411,889B1
- [96] Jorgenson, D.L., Karlaftis, M.G., Sinha, K.C. (2000) Vehicle speed considerations in traffic management: Development of a new speed monitoring program, Journal of Transportation and Statistics, Vol. 3, No. 3, pp. 69–82
- [97] Zhou, B., Cao, J., Zeng, X. and Wu, H. (2010) Adaptive Traffic Light Control in Wireless Sensor Network-based Intelligent Transportation System, Vehicular Technology Conference Fall (VTC 2010-Fall), 2010, 6-9 Sept. Hong Kong, China, pp. 1-5
- [98] Wen, W. (2010) An intelligent traffic management expert system with RFID technology, Expert Systems with Applications, Vol. 37, No. 4, pp. 3024–3035
- [99] Balaji, P.G. and Srinivasan, D. (2011) Type-2: fuzzy logic based urban traffic management, Engineering Applications of Artificial Intelligence, Vol. 24, No.1, pp. 12–22

- [100] Zhou, J., Gao, D. and Zhang, D. (2007) Moving vehicle detection for automatic traffic monitoring, IEEE Transactions on Vehicular Technology, Vol. 56, No. 1, pp. 51-59
- [101] Hull, B., Bychkovsky, V., Zhang, Y., Chen, K., Goraczko, M., Miu, A., Shih, E., Balakrishnan, H., and Madden, S. (2006) CarTel: A distributed mobile sensor computing system, In: Proceedings of the 4th international conference on Embedded networked sensor systems, 1–3 November, Boulder, Colorado: USA, pp. 125-138
- [102] Messelodi, S., Modena, C.M., Zanin, M., De Natale, F.G.B., Granelli, F., Betterle, E., and Guarise, A. (2009) Intelligent extended floating car data collection, Expert Systems with Applications, Vol. 36, No. 3, Part 1, pp. 4213–4227
- [103] Li, X., Shu, W. Li, M., Huang, H.-Y., Luo, P.-E. and Wu, M.-Y. (2009) Performance Evaluation of Vehicle-Based Mobile Sensor Networks for Traffic Monitoring, IEEE Transactions on Vehicular Technology, Vol. 58, No. 4, pp. 1647 - 1653
- [104] Hong, J.W., Park, S.-U., Kang, Y.-M., and Park, J.-T. (2001) Enterprise Network Traffic Monitoring, Analysis, and Reporting Using Web Technology, Journal of Network and Systems Management, Vol. 9, No.1, pp. 89-111
- [105] Shimmer Research, www.shimmerresearch.com (accessed December 2013)
- [106] Török, J. (2000) Analytical Mechanics: With Introduction to Dynamical Systems, John Wiley & Sons, Inc.: USA
- [107] Buyya, R., Selvi, S. T. and Chu, X. (2009) Object Oriented Programming with Java: Essentials and Applications, Tata McGraw Hill
- [108] Harold, E. R. (2005) Java Network Programming, 3rd Edition, O'Reilly Media, Inc.

(ii) APPENDIX

Appendix - A: Calibration Data of the Shimmer Sensor

Table A.1: X-axis Shimmer sensor data along the axis of the positive gravitational vibration acceleration ('+g') ['X plus up.csv' file]

Time Stamp	X_accel	Y_accel	Z_accel	Calibrated X_accel	X_accel (m/s ²)	Calibrated Y_accel	Y_accel (m/s ²)	Calibrated Z_accel	Z_accel (m/s ²)
8961	2263	2120	2006	0.992	9.724	0.011	0.108	0.007	0.065
12161	2271	2119	1998	1.022	10.026	0.007	0.070	-0.025	-0.250
15361	2270	2119	2006	1.019	9.988	0.007	0.070	0.007	0.065
18561	2267	2122	2006	1.007	9.875	0.019	0.186	0.007	0.065
21761	2272	2113	2006	1.026	10.064	-0.017	-0.163	0.007	0.065
24961	2267	2118	2003	1.007	9.875	0.003	0.031	-0.005	-0.053
28161	2269	2123	2004	1.015	9.951	0.023	0.225	-0.001	-0.014
31361	2273	2122	2013	1.030	10.101	0.019	0.186	0.035	0.341
34561	2267	2115	2005	1.007	9.875	-0.009	-0.085	0.003	0.026
37761	2272	2117	2005	1.026	10.064	-0.001	-0.008	0.003	0.026
40961	2266	2121	2011	1.003	9.837	0.015	0.147	0.027	0.262
44161	2265	2119	2003	0.999	9.800	0.007	0.070	-0.005	-0.053
47361	2269	2119	1999	1.015	9.951	0.007	0.070	-0.021	-0.211
50561	2268	2119	2009	1.011	9.913	0.007	0.070	0.019	0.183
53761	2269	2120	2007	1.015	9.951	0.011	0.108	0.011	0.104
56961	2274	2115	2004	1.034	10.139	-0.009	-0.085	-0.001	-0.014
60161	2270	2118	2009	1.019	9.988	0.003	0.031	0.019	0.183
63361	2273	2121	2007	1.030	10.101	0.015	0.147	0.011	0.104
1025	2267	2118	2004	1.007	9.875	0.003	0.031	-0.001	-0.014
4225	2264	2115	2006	0.995	9.762	-0.009	-0.085	0.007	0.065
7425	2267	2119	2003	1.007	9.875	0.007	0.070	-0.005	-0.053
10625	2270	2117	2008	1.019	9.988	-0.001	-0.008	0.015	0.144
13825	2277	2119	2011	1.045	10.252	0.007	0.070	0.027	0.262
17025	2272	2120	2010	1.026	10.064	0.011	0.108	0.023	0.223
20225	2274	2106	2003	1.034	10.139	-0.044	-0.434	-0.005	-0.053
23425	2272	2117	2014	1.026	10.064	-0.001	-0.008	0.039	0.380
26625	2269	2122	2004	1.015	9.951	0.019	0.186	-0.001	-0.014
29825	2267	2113	2006	1.007	9.875	-0.017	-0.163	0.007	0.065
33025	2270	2116	2007	1.019	9.988	-0.005	-0.047	0.011	0.104
36225	2261	2117	2011	0.984	9.649	-0.001	-0.008	0.027	0.262
39425	2267	2125	2010	1.007	9.875	0.031	0.302	0.023	0.223
42625	2271	2117	2010	1.022	10.026	-0.001	-0.008	0.023	0.223
45825	2268	2125	2002	1.011	9.913	0.031	0.302	-0.009	-0.092
49025	2277	2120	1998	1.045	10.252	0.011	0.108	-0.025	-0.250
52225	2268	2115	2005	1.011	9.913	-0.009	-0.085	0.003	0.026
55425	2266	2121	2010	1.003	9.837	0.015	0.147	0.023	0.223
58625	2261	2117	2001	0.984	9.649	-0.001	-0.008	-0.013	-0.132
61825	2262	2119	2008	0.988	9.687	0.007	0.070	0.015	0.144
65025	2269	2120	2005	1.015	9.951	0.011	0.108	0.003	0.026
2689	2269	2119	2006	1.015	9.951	0.007	0.070	0.007	0.065
5889	2270	2115	2009	1.019	9.988	-0.009	-0.085	0.019	0.183
9089	2267	2121	1998	1.007	9.875	0.015	0.147	-0.025	-0.250
12289	2266	2118	2009	1.003	9.837	0.003	0.031	0.019	0.183
15489	2267	2123	2009	1.007	9.875	0.023	0.225	0.019	0.183
18689	2268	2119	2002	1.011	9.913	0.007	0.070	-0.009	-0.092
21889	2265	2113	2005	0.999	9.800	-0.017	-0.163	0.003	0.026

25089	2266	2129	2003	1.003	9.837	0.047	0.457	-0.005	-0.053
28289	2268	2121	2008	1.011	9.913	0.015	0.147	0.015	0.144
31489	2269	2124	2005	1.015	9.951	0.027	0.263	0.003	0.026
34689	2272	2111	2007	1.026	10.064	-0.025	-0.240	0.011	0.104
37889	2261	2120	2009	0.984	9.649	0.011	0.108	0.019	0.183
41089	2268	2123	2008	1.011	9.913	0.023	0.225	0.015	0.144
44289	2272	2122	2005	1.026	10.064	0.019	0.186	0.003	0.026
47489	2270	2124	2002	1.019	9.988	0.027	0.263	-0.009	-0.092
50689	2265	2120	2004	0.999	9.800	0.011	0.108	-0.001	-0.014
53889	2268	2119	2011	1.011	9.913	0.007	0.070	0.027	0.262
57089	2269	2123	2002	1.015	9.951	0.023	0.225	-0.009	-0.092
60289	2268	2115	2003	1.011	9.913	-0.009	-0.085	-0.005	-0.053
63489	2272	2124	2009	1.026	10.064	0.027	0.263	0.019	0.183
1153	2269	2123	2007	1.015	9.951	0.023	0.225	0.011	0.104
4353	2267	2118	2005	1.007	9.875	0.003	0.031	0.003	0.026
7553	2271	2116	2008	1.022	10.026	-0.005	-0.047	0.015	0.144
10753	2276	2122	1997	1.042	10.214	0.019	0.186	-0.030	-0.289
13953	2277	2122	2001	1.045	10.252	0.019	0.186	-0.013	-0.132
17153	2273	2117	2007	1.030	10.101	-0.001	-0.008	0.011	0.104
20353	2269	2115	2012	1.015	9.951	-0.009	-0.085	0.031	0.301
23553	2258	2126	2006	0.972	9.536	0.035	0.341	0.007	0.065
26753	2271	2124	2004	1.022	10.026	0.027	0.263	-0.001	-0.014
29953	2269	2119	2007	1.015	9.951	0.007	0.070	0.011	0.104
33153	2269	2124	2005	1.015	9.951	0.027	0.263	0.003	0.026
36353	2261	2117	2003	0.984	9.649	-0.001	-0.008	-0.005	-0.053
39553	2272	2117	2007	1.026	10.064	-0.001	-0.008	0.011	0.104
42753	2268	2119	2006	1.011	9.913	0.007	0.070	0.007	0.065

Table A.2: X-axis Shimmer sensor data along the axis of the negative gravitational vibration acceleration ('-g') [‘X plus down.csv’ file]

Time Stamp	X_accel	Y_accel	Z_accel	Calibrated X_accel	X_accel (m/s ²)	Calibrated Y_accel	Y_accel (m/s ²)	Calibrated Z_accel	Z_accel (m/s ²)
28513	1740	2121	2007	-1.018	-9.986	0.015	0.147	0.011	0.104
31713	1755	2118	2008	-0.961	-9.421	0.003	0.031	0.015	0.144
34913	1740	2118	2003	-1.018	-9.986	0.003	0.031	-0.005	-0.053
38113	1742	2121	2006	-1.011	-9.910	0.015	0.147	0.007	0.065
41313	1748	2115	2004	-0.988	-9.684	-0.009	-0.085	-0.001	-0.014
44513	1745	2111	2008	-0.999	-9.797	-0.025	-0.240	0.015	0.144
47713	1740	2115	2015	-1.018	-9.986	-0.009	-0.085	0.043	0.420
50913	1747	2113	2008	-0.991	-9.722	-0.017	-0.163	0.015	0.144
54113	1748	2111	2015	-0.988	-9.684	-0.025	-0.240	0.043	0.420
57313	1753	2112	2013	-0.968	-9.496	-0.021	-0.202	0.035	0.341
60513	1748	2121	1999	-0.988	-9.684	0.015	0.147	-0.021	-0.211
63713	1750	2114	2013	-0.980	-9.609	-0.013	-0.124	0.035	0.341
1377	1744	2113	2012	-1.003	-9.835	-0.017	-0.163	0.031	0.301
4577	1752	2115	2010	-0.972	-9.534	-0.009	-0.085	0.023	0.223
7777	1750	2120	2010	-0.980	-9.609	0.011	0.108	0.023	0.223
10977	1747	2119	2015	-0.991	-9.722	0.007	0.070	0.043	0.420
14177	1746	2119	2002	-0.995	-9.760	0.007	0.070	-0.009	-0.092
17377	1748	2116	2012	-0.988	-9.684	-0.005	-0.047	0.031	0.301
20577	1747	2117	2009	-0.991	-9.722	-0.001	-0.008	0.019	0.183
23777	1739	2116	2013	-1.022	-10.024	-0.005	-0.047	0.035	0.341
26977	1751	2110	2015	-0.976	-9.571	-0.028	-0.279	0.043	0.420
30177	1745	2117	2010	-0.999	-9.797	-0.001	-0.008	0.023	0.223

33377	1745	2111	2007	-0.999	-9.797	-0.025	-0.240	0.011	0.104
36577	1748	2115	2001	-0.988	-9.684	-0.009	-0.085	-0.013	-0.132
39777	1747	2117	2007	-0.991	-9.722	-0.001	-0.008	0.011	0.104
42977	1744	2116	2017	-1.003	-9.835	-0.005	-0.047	0.051	0.498
46177	1755	2107	2011	-0.961	-9.421	-0.040	-0.395	0.027	0.262
49377	1743	2110	2013	-1.007	-9.873	-0.028	-0.279	0.035	0.341
52577	1744	2113	2014	-1.003	-9.835	-0.017	-0.163	0.039	0.380
55777	1754	2112	2010	-0.964	-9.458	-0.021	-0.202	0.023	0.223
58977	1741	2114	2008	-1.014	-9.948	-0.013	-0.124	0.015	0.144
62177	1751	2111	2003	-0.976	-9.571	-0.025	-0.240	-0.005	-0.053
65377	1751	2112	2000	-0.976	-9.571	-0.021	-0.202	-0.017	-0.171
3041	1754	2110	2014	-0.964	-9.458	-0.028	-0.279	0.039	0.380
6241	1748	2120	2008	-0.988	-9.684	0.011	0.108	0.015	0.144
9441	1743	2113	2013	-1.007	-9.873	-0.017	-0.163	0.035	0.341
12641	1753	2110	1998	-0.968	-9.496	-0.028	-0.279	-0.025	-0.250
15841	1757	2107	2010	-0.953	-9.345	-0.040	-0.395	0.023	0.223
19041	1746	2118	2009	-0.995	-9.760	0.003	0.031	0.019	0.183
22241	1749	2118	2008	-0.984	-9.647	0.003	0.031	0.015	0.144
25441	1748	2111	2011	-0.988	-9.684	-0.025	-0.240	0.027	0.262
28641	1746	2107	2010	-0.995	-9.760	-0.040	-0.395	0.023	0.223
31841	1753	2111	2010	-0.968	-9.496	-0.025	-0.240	0.023	0.223
35041	1748	2110	2003	-0.988	-9.684	-0.028	-0.279	-0.005	-0.053
38241	1751	2115	2006	-0.976	-9.571	-0.009	-0.085	0.007	0.065
41441	1744	2110	2005	-1.003	-9.835	-0.028	-0.279	0.003	0.026
44641	1739	2115	2026	-1.022	-10.024	-0.009	-0.085	0.087	0.853
47841	1750	2117	2003	-0.980	-9.609	-0.001	-0.008	-0.005	-0.053
51041	1748	2109	2013	-0.988	-9.684	-0.032	-0.318	0.035	0.341
54241	1748	2120	2013	-0.988	-9.684	0.011	0.108	0.035	0.341
57441	1747	2117	2009	-0.991	-9.722	-0.001	-0.008	0.019	0.183
60641	1751	2111	2010	-0.976	-9.571	-0.025	-0.240	0.023	0.223
63841	1743	2116	2003	-1.007	-9.873	-0.005	-0.047	-0.005	-0.053
1505	1750	2118	2002	-0.980	-9.609	0.003	0.031	-0.009	-0.092
4705	1742	2109	2009	-1.011	-9.910	-0.032	-0.318	0.019	0.183
7905	1747	2112	2015	-0.991	-9.722	-0.021	-0.202	0.043	0.420
11105	1753	2115	1998	-0.968	-9.496	-0.009	-0.085	-0.025	-0.250
14305	1742	2121	2011	-1.011	-9.910	0.015	0.147	0.027	0.262
17505	1753	2116	2008	-0.968	-9.496	-0.005	-0.047	0.015	0.144
20705	1753	2115	2004	-0.968	-9.496	-0.009	-0.085	-0.001	-0.014
23905	1750	2119	2004	-0.980	-9.609	0.007	0.070	-0.001	-0.014
27105	1746	2106	2013	-0.995	-9.760	-0.044	-0.434	0.035	0.341
30305	1749	2110	2010	-0.984	-9.647	-0.028	-0.279	0.023	0.223
33505	1745	2115	2005	-0.999	-9.797	-0.009	-0.085	0.003	0.026
36705	1751	2115	2004	-0.976	-9.571	-0.009	-0.085	-0.001	-0.014
39905	1753	2114	2015	-0.968	-9.496	-0.013	-0.124	0.043	0.420
43105	1746	2114	2008	-0.995	-9.760	-0.013	-0.124	0.015	0.144
46305	1739	2109	2016	-1.022	-10.024	-0.032	-0.318	0.047	0.459
49505	1742	2110	2014	-1.011	-9.910	-0.028	-0.279	0.039	0.380
52705	1754	2108	2006	-0.964	-9.458	-0.036	-0.357	0.007	0.065
55905	1754	2123	1995	-0.964	-9.458	0.023	0.225	-0.038	-0.368
59105	1749	2117	2009	-0.984	-9.647	-0.001	-0.008	0.019	0.183
62305	1748	2114	2011	-0.988	-9.684	-0.013	-0.124	0.027	0.262

Appendix - B: Determination of Resultant Acceleration

The mechanical wheel in the experimental set-up is considered to be a rigid body during speed measurement with the sensor device (Figure B.1). The Shimmer sensor is attached to the surface of the rotating wheel thereby forming an integral part of the rigid body. The rectangular Cartesian co-ordinate system on the Shimmer sensor moves together with the rotating wheel. The mass of the Shimmer device is considered to be negligible. The rigid body can rotate with respect to its centre of mass. Therefore, it is essential to examine the kinematics of the rotating rigid body. In this regard, a relationship between the rotating frame of reference and the inertial frame of reference is examined. A rigid body rotation is viewed as a coordinate transformation in space [106].

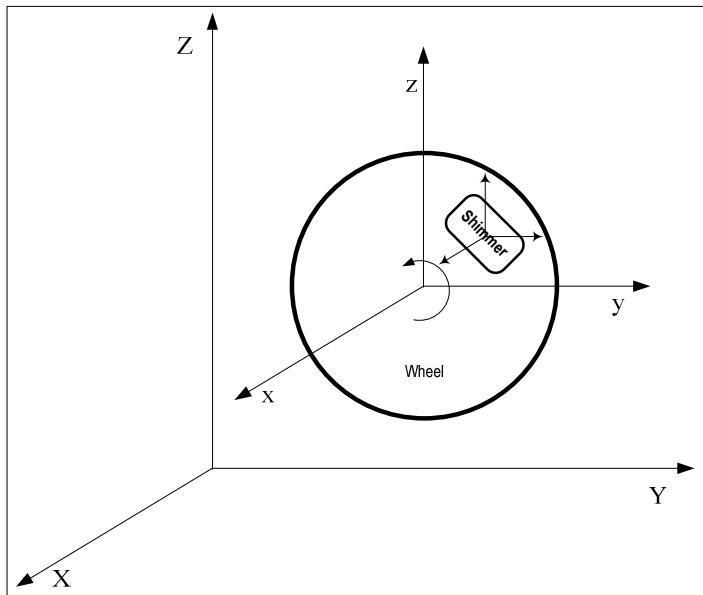


Figure B.1: The rotating rigid body considering the Shimmer sensor as an integral part

In angular motion, the velocity of a body is known as angular velocity. Angular velocity is measured in terms of the angle (θ) covered by the rigid body per unit time. Angular velocity is denoted by ω . The direction of the angular velocity may be clockwise or counter clockwise. The angular acceleration of the body may be of centripetal or centrifugal. The resultant acceleration is broken down into different components depending on the nature of the motion. Figure B.2 demonstrates the components of the angular acceleration of the body 'A' rotating in counter clockwise direction with an angular velocity.

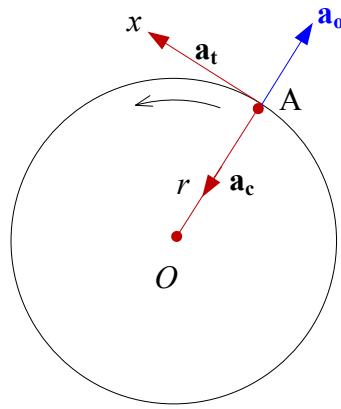


Figure B.2: The angular velocity of a rigid body ‘A’

The definition of the average angular velocity is $\omega = \frac{\theta}{t}$, where θ is the angle covered in time t . From the definition of the angular velocity a relationship between the linear velocity (v) and the radius of the circle (r) can be drawn: $\omega = \frac{v}{r}$.

Angular acceleration of a rigid body in motion is referred as the rate of change of the angular velocity of the rigid body with respect to time. It is denoted by α . Considering an infinitesimal time period $\omega = \frac{d\theta}{dt}$. Therefore, $\alpha = \frac{d\omega}{dt}$, or $\alpha = \frac{d^2\theta}{dt^2}$ leading to $\alpha = \frac{v^2}{r}$.

The physics of the rigid body rotation is more complex as it involves a rectangular 3-D Cartesian coordinate system during the circular motion of a rigid body. At this juncture, a vector representation of the angular motion is strongly recommended. Let us consider a right-handed rectangular Cartesian co-ordinate system as defined in Figure B.3 and Figure B.4.

The velocity (V_{rel}) and acceleration (a_{rel}) as observed from within the moving coordinate system are represented as:

$$\mathbf{v}_{rel} = \frac{dx}{dt} \mathbf{i} + \frac{dy}{dt} \mathbf{j} + \frac{dz}{dt} \mathbf{k} \quad \text{---(B1)}$$

$$\mathbf{a}_{rel} = \frac{d^2x}{dt^2} \mathbf{i} + \frac{d^2y}{dt^2} \mathbf{j} + \frac{d^2z}{dt^2} \mathbf{k} = a_x \mathbf{i} + a_y \mathbf{j} + a_z \mathbf{k} \quad \text{---(B2)}$$

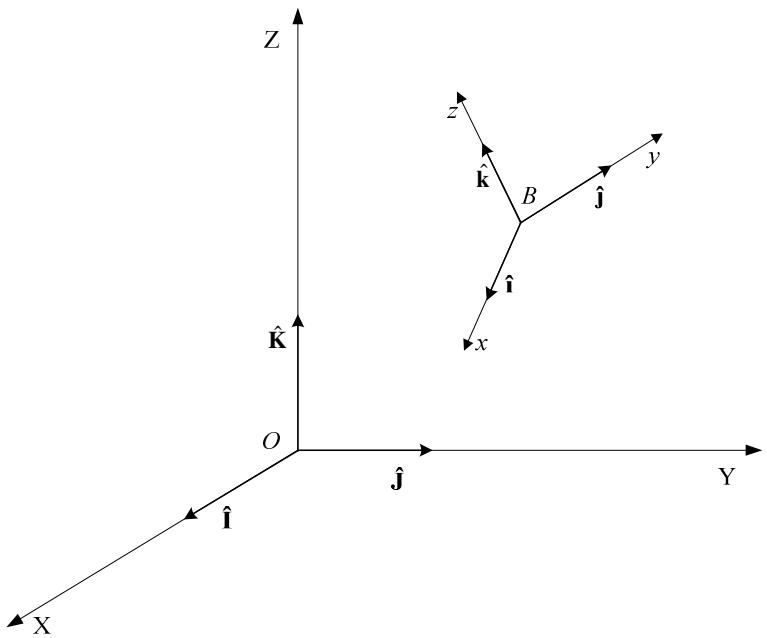


Figure B.3: The motion of the reference frame xyz with respect to the inertial reference frame XYZ (adapted from [106])

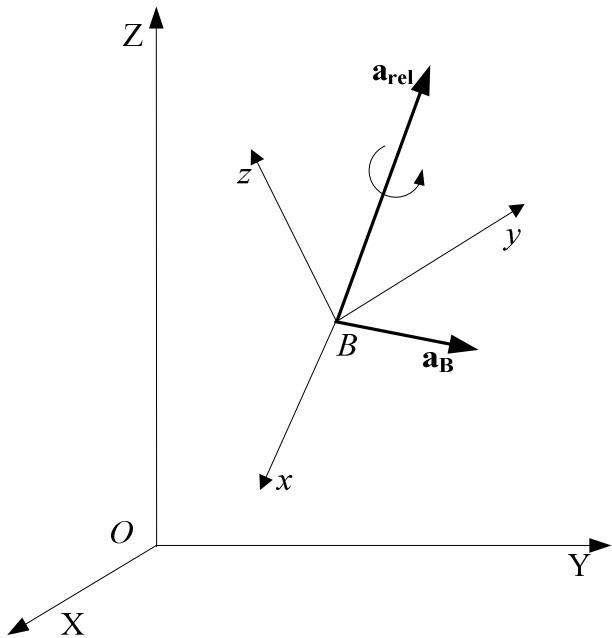


Figure B.4: The resultant acceleration with respect to the reference frame (adapted from [106])

The acceleration vector \mathbf{a}_{rel} determines both the magnitude and direction of the relative acceleration (Figure B.5). The acceleration data of the car wheel obtained from the Shimmer sensor is calculated considering only the magnitude of the relative

acceleration. The magnitude of the resultant acceleration vector \mathbf{a}_{rel} is given by the following equation.

$$|\mathbf{a}_{rel}| = \sqrt{a_x^2 + a_y^2 + a_z^2} \quad \text{-----(B3)}$$

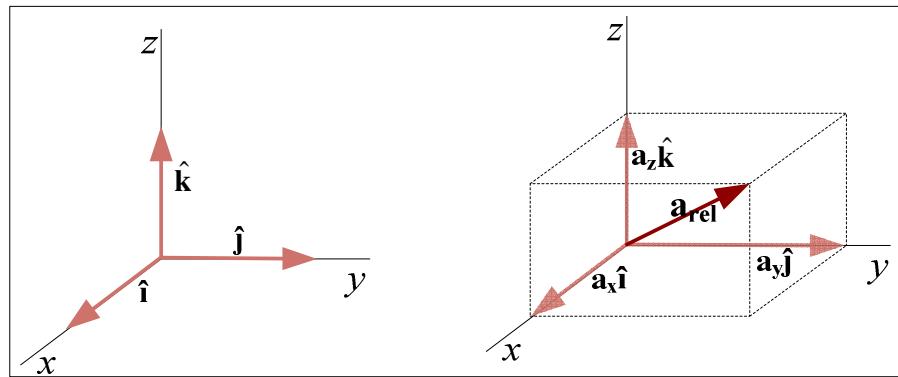


Figure B.5: Resultant acceleration \mathbf{a}_{rel}

Appendix - C: Java™ Socket Programming Codes Developed for Single Vehicle

C1. Codes for Server Computer

```
import java.net.*;
import java.io.*;
import java.util.*;
import java.awt.Color;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.text.*;

public class SingleCarServer
{
    static Socket connection=null;
    static ServerSocket socket1;
    int character;
    protected final static int port = 19999;

    public static void main(String[] args)
    {
        try
        {
            String filename = "Car_1000.csv";

            BufferedReader CSVFile = new BufferedReader(new FileReader(filename));

            socket1 = new ServerSocket(port);
            connection = socket1.accept();

            BufferedOutputStream os = new BufferedOutputStream(connection.getOutputStream());
            OutputStreamWriter osw = new OutputStreamWriter(os, "US-ASCII");

            System.out.println("SingleSocketServer Initialized");
            GridLayout experimentLayout = new GridLayout(7,1,0,0);
            JFrame myFrame = new JFrame("Server");
            myFrame.setLayout(experimentLayout);
            myFrame.setSize(300,400);
            myFrame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
            myFrame.setVisible(true);

            JLabel text1 = new JLabel("      Sending the following data to client");
            JLabel text2 = new JLabel("      ");
            JLabel text3 = new JLabel("      ");
            JLabel text4 = new JLabel("      ");
            JLabel text5 = new JLabel("      ");
            JLabel text6 = new JLabel("      ");
            JLabel text7 = new JLabel("      ");

            myFrame.add(text1);
            myFrame.add(text2);
            myFrame.add(text3);
            myFrame.add(text4);
            myFrame.add(text5);
            myFrame.add(text6);
            myFrame.add(text7);

            String headerRow = CSVFile.readLine();
            String dataRow = CSVFile.readLine();
```

```

        int XAccel = 0;
        int YAccel = 0;
        int ZAccel = 0;

        int num_of_plots = 1;
        while (dataRow != null)
        {
            String[] dataArray = dataRow.split(",");
            XAccel = Integer.parseInt(dataArray[1]);
            YAccel = Integer.parseInt(dataArray[2]);
            ZAccel = Integer.parseInt(dataArray[3]);

            double Resultant_Acceleration = Math.sqrt((XAccel*XAccel)+(YAccel*YAccel)+(ZAccel*ZAccel));
            double RPM = Math.abs((Resultant_Acceleration - 3024.8)) / 31.655;
            double Speed = (2*3.1415*RPM*0.055*3600/(60*1000))*96.457;

            String message = "-----\n+
"XAccel:      "+XAccel+"\n"+
"YAccel:      "+YAccel+"\n"+
"ZAccel:      "+ZAccel+"\n"+
"Resultant_Acceleration: "+Resultant_Acceleration+"\n"+
"RPM:         "+RPM+"\n"+
"Speed:        "+Speed+"\n";
System.out.println(message);

DecimalFormat df = new DecimalFormat("#.##");
text1.setText(" Car_number:           Car_1000");
text2.setText(" XAccel:             "+XAccel);
text3.setText(" YAccel:             "+YAccel);
text4.setText(" ZAccel:             "+ZAccel);
text5.setText("                   Resultant_Acceleration:
"+df.format(Resultant_Acceleration));
text6.setText(" RPM:                "+df.format(RPM));
text7.setText(" Speed:               "+df.format(Speed));

osw.write(XAccel + ",");
osw.write(YAccel + ",");
osw.write(ZAccel + ",");
osw.write(Resultant_Acceleration + ",");
osw.write(RPM + ",");
osw.write(Speed+ "\n");
osw.write((char) 13);

if (num_of_plots > 1)
{
    try
    {
        Thread.sleep(30000);
    }
    catch (Exception e)
    {
    }
}
num_of_plots++;
dataRow = CSVFile.readLine();
osw.flush();
osw = new OutputStreamWriter(os, "US-ASCII");
}

osw.write((char) 23);
osw.close();

```

```
connection.close();
    }
    catch (IOException e)
    {
        System.out.println("ERROR on Server code "+e.toString());
    }
}
```

C2. Codes for Client Computer

```
import java.net.*;
import java.io.*;
import java.util.*;
import org.jfree.ui.RefineryUtilities;
import java.awt.Color;
import javax.swing.JOptionPane;

//http://itextpdf.com/
import com.lowagie.text.Chunk;
import com.lowagie.text.Document;
import com.lowagie.text.DocumentException;
import com.lowagie.text.Element;
import com.lowagie.text.Font;
import com.lowagie.text.PageSize;
import com.lowagie.text.Paragraph;
import com.lowagie.text.Phrase;
import com.lowagie.text.pdf.BaseFont;
import com.lowagie.text.pdf.ColumnText;
import com.lowagie.text.pdf.PdfContentByte;
import com.lowagie.text.pdf.PdfReader;
import com.lowagie.text.pdf.PdfStamper;
import com.lowagie.text.pdf.PdfWriter;

public class SingleCarClientV2
{
    public static HashMap<String, String> hMap = new HashMap<String, String>();
    public static double speed_limit;
    public static void main(String[] args)
    {
        String host = "136.206.95.82";
        int port = 19999;

        ArrayList<String> temp_line_array = new ArrayList<String>();
        StringBuffer instr = new StringBuffer();
        System.out.println("SocketClient initialized \nStart");

        try
        {
            PrintWriter out = new PrintWriter(new FileWriter("output.csv"));
            out.println("XAccel,YAccel,ZAccel,ResultantAcceleration, RPM, Speed");
            InetAddress address = InetAddress.getByName(host);

            final int LOWEST = 30;
            final int INCREMENT = 5;
            final int HIGHEST = 80;
            Object[] speedlimits = new Integer[(HIGHEST - LOWEST)/INCREMENT + 1];
            int i, j;
            for (i = LOWEST, j = 0; j < speedlimits.length; i+=INCREMENT,j++)
            {
                speedlimits[j] = new Integer(i);
            }
            Integer s = (Integer)JOptionPane.showInputDialog(null, "Please choose an appropriate speed limit:\n", "Speed Limit Settings", JOptionPane.PLAIN_MESSAGE, null, speedlimits, new Integer(LOWEST));

            if (s == null)
            {
                out.close();
            }
        }
    }
}
```

```

return;
    }
    speed_limit = (double)s.intValue();
    Socket connection = new Socket(address, port);

BufferedInputStream bis = new BufferedInputStream(connection.getInputStream());
InputStreamReader isr = new InputStreamReader(bis, "US-ASCII");

final PlotWithArrowLabelsAndLines demo = new PlotWithArrowLabelsAndLines("SpeedManage",
speed_limit);
    demo.pack();
    RefineryUtilities.centerFrameOnScreen(demo);
    demo.setVisible(true);

int c;
c = isr.read();
int counter=0;

        while( c != 23 )
    {
while( c != 13 )
{
instr.append( (char) c);
c = isr.read();
}
String buffer_as_string = instr.toString();
 StringTokenizer st = new StringTokenizer(buffer_as_string, ",");
while (st.hasMoreTokens())
{
String column = st.nextToken();
temp_line_array.add(column);
}

System.out.println("-----");
System.out.println("XAccel : "+temp_line_array.get(0));
System.out.println("YAccel: "+temp_line_array.get(1));
System.out.println("ZAccel: "+temp_line_array.get(2));
System.out.println("Resultant_Acceleration: "+temp_line_array.get(3));
System.out.println("RPM: "+temp_line_array.get(4));
System.out.println("Speed: "+temp_line_array.get(5));

double speed_as_an_double = Double.parseDouble(temp_line_array.get(5));

String fn = "Car_1000";
String sp = temp_line_array.get(5);
double yVal = Double.valueOf(sp);
double tmpY = yVal*100;
tmpY = Math.round(tmpY);
double tmpFinal = tmpY/100;
String speed = String.valueOf(tmpFinal);

String timeAndDateStr = getTimeAndDateString(counter);
if ( (int) speed_as_an_double > speed_limit )
{
    out.println(buffer_as_string);
    CreatePdf(fn,speed,counter, timeAndDateStr);
}

demo.plotData( temp_line_array.get(5 ) );
demo.addMessage(timeAndDateStr);
demo.addMessage("Car Speed: " + speed + " Km/Hr");
demo.addMessage("Speed limit: " + speed_limit + " Km/Hr");

```

```

        demo.addMessage("Car Number: Car_1000");
        demo.addMessage("-----");

instr      = new StringBuffer();
temp_line_array = new ArrayList<String>();
c = isr.read();
counter++;
    }

connection.close();
out.close();
}
catch (IOException f)
{
System.out.println("ERROR on Client code"+f.toString());
}
catch (Exception g)
{
System.out.println("ERROR on Client code"+g.toString());
g.printStackTrace();
}
}

}

public static String getTimeAndDateString(int counter)
{
String [] days_array  = { "Sun", "Mon", "Tues", "Wed", "Thurs", "Fri", "Sat" };
String [] months_array = { "Jan", "Feb", "Mar", "Apr", "May", "June", "July", "Aug", "Sept", "Oct",
"Nov", "Dec" };

Calendar calendar = Calendar.getInstance();
String time = calendar.get(Calendar.HOUR_OF_DAY)+"";

int MINUTE  = calendar.get(Calendar.MINUTE);
int SECOND  = calendar.get(Calendar.SECOND);

if (MINUTE < 10)
{
    time = time + ":0"+MINUTE;
}
else
{
    time = time + ":"+MINUTE;
}

if (SECOND < 10)
{
    time = time + ":0"+SECOND;
}
else
{
    time = time + ":"+SECOND;
}

int DAY      = calendar.get(Calendar.DAY_OF_WEEK);
int DAYminusOne = calendar.get(Calendar.DAY_OF_WEEK) - 1;
int MONTH     = calendar.get(Calendar.MONTH) + 1;
int YEAR      = calendar.get(Calendar.YEAR);

String name_of_month = months_array[calendar.get(Calendar.MONTH)];
String date   = days_array[DAYminusOne]+ " "+DAY+" "+name_of_month+
"+YEAR;
String time_and_date = "Time:           "+time+"\n"+Date:           "+date;
}

```

```

        return time_and_date;
    }
    public static void CreatePdf(String fName, String Speed, int counter, String timeStamp)
    {

        String reportFileName = "report_for_" + fName + ".pdf";
        File f = new File(reportFileName);
        Document document = new Document();

        if(f.exists()) {
            try {
                System.out.println("file already exists. Adding to report..\n");
                String prevContent = "";
                if(hMap.containsKey(fName))
            {
                prevContent = hMap.get(fName);
            }
                PdfWriter.getInstance(document, new FileOutputStream(reportFileName));
                document.setPageSize(PageFormat.A4);

                document.open();

                Font font = new Font(Font.TIMES_ROMAN, 25, Font.BOLD);
                font.setColor(new Color(0, 0, 255));

                Chunk chunk = new Chunk(fName + " Report", font);

                Paragraph para = new Paragraph(chunk);
                para.setAlignment(Element.ALIGN_CENTER);
                document.add(para);

                String timeAndDateStr = getTimeAndDateString(counter);

                font = new Font(Font.TIMES_ROMAN, 15, Font.BOLD);
                font.setColor(new Color(0, 0, 0));

                String currContent = "\n" + "\nSpeed Limit: " + speed_limit + " Km/Hr\nCar Speed: " + Speed + " Km/Hr\n" + timeAndDateStr;

                chunk = new Chunk(prevContent + currContent, font);
                hMap.put(fName, prevContent + currContent );

                para = new Paragraph(chunk);
                para.setAlignment(Element.ALIGN_LEFT);
                document.add(para);
                document.close();
            }
        catch (IOException ioex)
        {
            ioex.printStackTrace();
        }
        catch(DocumentException e)
        {
            e.printStackTrace();
        }
    }
    else
    {
        try {
            PdfWriter.getInstance(document, new FileOutputStream(reportFileName));
            document.setPageSize(PageFormat.A4);
            document.open();

```


C3. GUI Plot Codes

```
import java.awt.*;
import java.io.*;
import javax.swing.*;
import org.jfree.chart.*;
import org.jfree.chart.axis.*;
import org.jfree.chart.plot.*;
import org.jfree.chart.renderer.xy.XYLineAndShapeRenderer;
import org.jfree.data.*;
import org.jfree.data.xy.*;
import org.jfree.ui.*;
import org.jfree.chart.annotations.XYPointerAnnotation;

public class PlotWithArrowLabelsAndLines extends ApplicationFrame
{
    private static final long serialVersionUID = 1L;
    private XYSeriesCollection dataset;
    private double lastValue;

    private static double Threshold;
    private XYPlot subplot;
    private static double seconds = 30.0;

    JTextPane status;
    private JSplitPane splitPane;
    private static double radians = 140.0D;
    XYLineAndShapeRenderer renderer = new XYLineAndShapeRenderer()
    {
        private static final long serialVersionUID = 1L;
        @Override
        public Paint getItemPaint(int row, int col)
        {
            Paint cpaint = getItemColor(row, col);
            if (cpaint == null)
            {
                cpaint = super.getItemPaint(row, col);
            }
            return cpaint;
        }
        public Color getItemColor(int row, int col)
        {
            double y = dataset.getYValue(row, col);
            if( (int) y > (int) Threshold )
            {
                return Color.red;
            }
            else if( (int) y == (int) Threshold )
            {
                return Color.yellow;
            }
            else
            {
                return Color.blue;
            }
        }
    }

    @Override
    protected void drawFirstPassShape(Graphics2D g2, int pass, int series, int item, Shape shape)
    {
        g2.setStroke(getItemStroke(series, item));
```

```

Color c1 = getItemColor(series, item);
Color c2 = getItemColor(series, item - 1);
Color c3 = new Color (0,0,0,0);

Paint linePaint = Color.black;
    g2.setPaint(linePaint);
    g2.draw(shape);
}

};

public PlotWithArrowLabelsAndLines(final String title, double threshold_value)
{
super(title);
Threshold = threshold_value;

final CombinedDomainXYPlot plot = new CombinedDomainXYPlot(new
NumberAxis("Sec"));

this.lastValue = 0.0;
final XYSeries series = new XYSeries("velo");
this.dataset = new XYSeriesCollection(series);
final NumberAxis rangeAxis = new NumberAxis("Speed");
rangeAxis.setAutoRangeIncludesZero(false);

subplot = new XYPlot( this.dataset, null, rangeAxis, renderer);
subplot.setBackgroundPaint(Color.lightGray);
subplot.setDomainGridlinePaint(Color.white);
subplot.setRangeGridlinePaint(Color.white);

ValueMarker mark = new ValueMarker(Threshold,Color.RED, new BasicStroke(2));
subplot.addRangeMarker(mark, Layer.BACKGROUND);

ValueAxis y = subplot.getRangeAxis();
y.setRange(new Range(0, 110.0));
plot.add(subplot);

final JFreeChart chart = new JFreeChart("Speed Plot", plot);

chart.setBorderPaint(Color.black);
chart.setBorderVisible(true);
chart.setBackgroundPaint(Color.white);
chart.removeLegend();

plot.setBackgroundPaint(Color.lightGray);
plot.setDomainGridlinePaint(Color.white);
plot.setRangeGridlinePaint(Color.white);

NumberAxis xAxis = (NumberAxis) plot.getDomainAxis();
xAxis.setAutoRange(true);
xAxis.setTickUnit(new NumberTickUnit(seconds));

JPanel content = new JPanel(new BorderLayout());
status = new JTextPane();
JSScrollPane sp = new JSScrollPane(status);
final ChartPanel chartPanel = new ChartPanel(chart);

chartPanel.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));

splitPane = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT, sp , chartPanel);
splitPane.setOneTouchExpandable(true);

```

```

splitPane.setDividerLocation(210);

sp.setMinimumSize(new Dimension(80, 50));
sp.setMaximumSize(new Dimension(210, 50));

chartPanel.setMinimumSize(new Dimension(715, 50));
chartPanel.setMaximumSize(new Dimension(1000, 50));

splitPane.setPreferredSize(new Dimension(1000, 700));
this.setContentPane(splitPane);
}

public void addMessage(String msg) {
String content = status.getText();
content = msg + "\n" + content;
status.setText(content);
}

public void plotData(String yVal_string)
{
    double xVal = this.lastValue + seconds;
    this.lastValue = xVal;

    double yVal = Double.valueOf( yVal_string );

    double tmpY = yVal*100;
    tmpY = Math.round(tmpY);
    double tmpFinal = tmpY/100;

    String labelString = String.valueOf(tmpFinal);
    this.dataset.getSeries(0).add(xVal, yVal);

XYPointerAnnotation xypointerannotation = new XYPointerAnnotation(labelString, xVal, yVal,
radians);
if (radians == 140.0D)
{
    radians = radians - 10;
}
else
{
    radians = radians + 10;
}
xypointerannotation.setLabelOffset(9D);
xypointerannotation.setTextAnchor(TextAnchor.BOTTOM_CENTER);

if( (int) yVal > Threshold )
{
    xypointerannotation.setBackgroundPaint(Color.red);
}
else if( (int) yVal == Threshold )
{
    xypointerannotation.setBackgroundPaint(Color.yellow);
}
else
{
    xypointerannotation.setBackgroundPaint(Color.cyan);
}
subplot.addAnnotation(xypointerannotation);

System.out.println("Plotting X = " + xVal + " Y=" + yVal);
try
{
    Thread.sleep(100);
}

```

```
        catch (InterruptedException e)
        {
            e.printStackTrace();
        }
    }
```

Appendix - D: Java™ Socket Programming Codes Developed for Multiple Vehicles

D1. Codes for Server Computer

```
import java.net.*;
import java.io.*;
import java.util.*;
import java.awt.Color;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.text.*;

public class MultipleCarServer
{
    static Socket connection;
    static ServerSocket socket1;
    final static int port = 19999;

    public static void main(String[] args)
    {
        String[] Car_array = { "1052", "1250", "2453", "2201", "0151", "3402", "0302", "1103", "4502", "1350" };

        try
        {
            socket1 = new ServerSocket(port);
            int character;
            connection = socket1.accept();
            BufferedOutputStream os = new BufferedOutputStream(connection.getOutputStream());
            OutputStreamWriter osw = new OutputStreamWriter(os, "US-ASCII");

            int index = 0;
            System.out.println("SingleSocketServerAverage Initialized");
            GridLayout experimentLayout = new GridLayout(7,1,0,0);
            JFrame myFrame = new JFrame("Server");
            myFrame.setLayout(experimentLayout);
            myFrame.setSize(300,400);
            myFrame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
            myFrame.setVisible(true);
            JLabel text1 = new JLabel("      Sending the following data to client");
            JLabel text2 = new JLabel("      ");
            JLabel text3 = new JLabel("      ");
            JLabel text4 = new JLabel("      ");
            JLabel text5 = new JLabel("      ");
            JLabel text6 = new JLabel("      ");
            JLabel text7 = new JLabel("      ");
            myFrame.add(text1);
            myFrame.add(text2);
            myFrame.add(text3);
            myFrame.add(text4);
            myFrame.add(text5);
            myFrame.add(text6);
            myFrame.add(text7);
            while (index < Car_array.length )
            {
                String Car_number = "Car_"+Car_array[index]+ ".csv";
                BufferedReader CSVFile = new BufferedReader(new FileReader(Car_number));

```

```

        CSVFile.readLine();
        String dataRow = CSVFile.readLine();

        int XAccel_Sum = 0;
        int YAccel_Sum = 0;
        int ZAccel_Sum = 0;
        int num_of_lines = 0;

        while (dataRow != null)
        {
            String[] dataArray = dataRow.split(",");
            XAccel_Sum = XAccel_Sum + Integer.parseInt(dataArray[1]);
            YAccel_Sum = YAccel_Sum + Integer.parseInt(dataArray[2]);
            ZAccel_Sum = ZAccel_Sum + Integer.parseInt(dataArray[3]);

            num_of_lines = num_of_lines + 1;

            dataRow = CSVFile.readLine();
        }

        int XAccel_Average = XAccel_Sum / num_of_lines;
        int YAccel_Average = YAccel_Sum / num_of_lines;
        int ZAccel_Average = ZAccel_Sum / num_of_lines;

        double Resultant_Acceleration = Math.sqrt(
            ((XAccel_Average*XAccel_Average)+(YAccel_Average*YAccel_Average)+(ZAccel_Average*ZAccel_Average)));

        double RPM = Math.abs((Resultant_Acceleration - 3024.8)) / 31.655;
        double Speed = (2*3.1415*RPM*0.055*3600/(60*1000))*96.457;
        DecimalFormat df = new DecimalFormat("#.##");

        String message = "Car_number:      Car_"+Car_array[index]+"\n"+
            "XAccel_Average:    "+XAccel_Average+"\n"+
            "YAccel_Average:    "+YAccel_Average+"\n"+
            "ZAccel_Average:    "+ZAccel_Average+"\n"+
            "Resultant_Acceleration: "+Resultant_Acceleration+"\n"+
            "RPM:              "+RPM+"\n"+
            "Speed:             "+Speed+"\n";

        System.out.println("-----");
        System.out.println(message);

        text1.setText("  Car_number:      Car_"+Car_array[index]);
        text2.setText("  XAccel_Average:    "+XAccel_Average);
        text3.setText("  YAccel_Average:    "+YAccel_Average);
        text4.setText("  ZAccel_Average:    "+ZAccel_Average);
        text5.setText("  Resultant_Acceleration: "+df.format(Resultant_Acceleration));
        text6.setText("  RPM:              "+df.format(RPM));
        text7.setText("  Speed:             "+df.format(Speed));

        osw.write(Car_array[index] + ",");
        osw.write(XAccel_Average + ",");
        osw.write(YAccel_Average + ",");
        osw.write(ZAccel_Average + ",");
        osw.write(Resultant_Acceleration + ",");
        osw.write(RPM + ",");
        osw.write(Speed+ "\n");

        osw.write((char) 13);
        try

```

```
{  
    Thread.sleep(30000);  
}  
catch (Exception e)  
{  
    System.out.println("ERROR Waiting too long"+e.toString());  
}  
    osw.flush();  
    osw = new OutputStreamWriter(os, "US-ASCII");  
  
    CSVFile.close();  
    index = index + 1;  
}  
osw.write((char) 23);  
osw.close();  
connection.close();  
}  
catch (IOException e)  
{  
    System.out.println("ERROR on Server code "+e.toString());  
}  
}  
}  
}
```

D2. Codes for Client Computer

```
import java.net.*;
import java.io.*;
import java.util.*;
import org.jfree.ui.RefineryUtilities;
import java.awt.Color;
import javax.swing.*;

import com.lowagie.text.Chunk;
import com.lowagie.text.Document;
import com.lowagie.text.DocumentException;
import com.lowagie.text.Element;
import com.lowagie.text.Font;
import com.lowagie.text.PageSize;
import com.lowagie.text.Paragraph;
import com.lowagie.text.Phrase;
import com.lowagie.text.pdf.BaseFont;
import com.lowagie.text.pdf.ColumnText;
import com.lowagie.text.pdf.PdfContentByte;
import com.lowagie.text.pdf.PdfReader;
import com.lowagie.text.pdf.PdfStamper;
import com.lowagie.text.pdf.PdfWriter;

public class MultipleCarClientV2
{
    public static HashMap<String, String> hMap = new HashMap<String, String>();
    public static double speed_limit = 54.0;

    public static void main(String[] args)
    {
        final String host = "136.206.95.82";
        final int port      = 19999;
        ArrayList<String> temp_line_array = new ArrayList<String>();

        StringBuffer instr = new StringBuffer();
        System.out.println("SocketClient1 initialized \nStart");
        try
        {
            PrintWriter out = new PrintWriter(new FileWriter("output.csv"));
            out.println("Car_number,XAccel(Avg),YAccel(Avg),ZAccel(Avg),ResultantAcceleration," +
                       "RPM,Speed");
            InetAddress address = InetAddress.getByName(host);
            final int LOWEST = 30;
            final int INCREMENT = 5;
            final int HIGHEST = 80;
            Object[] speedlimits = new Integer[(HIGHEST - LOWEST)/INCREMENT + 1];
            int i, j;
            for (i = LOWEST, j = 0; j < speedlimits.length; i+=INCREMENT, j++)
            {
                speedlimits[j] = new Integer(i);
            }
            Integer s = (Integer) JOptionPane.showInputDialog(null,
                "Please choose an appropriate speed limit:\n", "Speed Limit Settings",
                JOptionPane.PLAIN_MESSAGE, null, speedlimits, new Integer(LOWEST));
            if (s == null)
            {
                out.close();
                return;
            }
            speed_limit = (double)s.intValue();
        }
    }
}
```

```

Socket connection = new Socket(address, port);

BufferedInputStream bis = new BufferedInputStream(connection.getInputStream());
InputStreamReader isr = new InputStreamReader(bis, "US-ASCII");

PlotWithArrowLabelsAndNoLines demo = new PlotWithArrowLabelsAndNoLines("SpeedManage",
(int) speed_limit;
demo.pack();
RefineryUtilities.centerFrameOnScreen(demo);
demo.setVisible(true);
int c;
c = isr.read();
int counter=0;

while( c != 23 )
{
    while( c != 13 )
    {
        instr.append( (char) c);
        c = isr.read();
    }
}

String buffer_as_string = instr.toString();
 StringTokenizer st = new StringTokenizer(buffer_as_string, ",");
while (st.hasMoreTokens())
{
    String column = st.nextToken();
    temp_line_array.add(column);
}
System.out.println("-----");
System.out.println("Car_number:      "+temp_line_array.get(0));
System.out.println("XAccel_Average:  "+temp_line_array.get(1));
System.out.println("YAccel_Average:  "+temp_line_array.get(2));
System.out.println("ZAccel_Average:  "+temp_line_array.get(3));
System.out.println("Resultant_Acceleration: "+temp_line_array.get(4));
System.out.println("RPM:           "+temp_line_array.get(5));
System.out.println("Speed:          "+temp_line_array.get(6));

double speed_as_an_double = Double.parseDouble(temp_line_array.get(6));
String fn = temp_line_array.get(0);
fn = fn.replace(".csv", "");
fn = "Car_" +fn;
String sp = temp_line_array.get(6);
double yVal = Double.valueOf(sp);
double tmpY = yVal*100;
tmpY = Math.round(tmpY);
double tmpFinal = tmpY/100;
String speed = String.valueOf(tmpFinal);
String timeAndDateStr = getTimeAndDateString(counter);
if ( (int) speed_as_an_double > (int) speed_limit )
{
    out.println(buffer_as_string);
    CreatePdf(fn,speed,counter, timeAndDateStr);
}

demo.plotData( temp_line_array.get(6), temp_line_array.get(0)+"_Car" );
demo.addMessage(timeAndDateStr);
demo.addMessage("Car Speed:   " + speed + " Km/Hr");
demo.addMessage("Speed limit:  " + speed_limit + " Km/Hr");
demo.addMessage("Car Number:  " + fn);
demo.addMessage("-----");

```

```

        instr = new StringBuffer();
        temp_line_array = new ArrayList<String>();
        c = isr.read();
        counter++;
    }

connection.close();
out.close();
}
catch (IOException f)
{
    System.out.println("ERROR on Client code"+f.toString());
}
catch (Exception g)
{
    System.out.println("ERROR on Client code"+g.toString());
    g.printStackTrace();
}
}

public static String getTimeAndDateString(int counter)
{
    String [] days_array = { "Sun", "Mon", "Tues", "Wed", "Thurs", "Fri", "Sat" };
    String [] months_array = { "Jan", "Feb", "Mar", "Apr", "May", "June", "July", "Aug", "Sept",
    "Oct", "Nov", "Dec" };

    Calendar calendar = Calendar.getInstance();
    String time = calendar.get(Calendar.HOUR_OF_DAY)+"";

    int MINUTE = calendar.get(Calendar.MINUTE);
    int SECOND = calendar.get(Calendar.SECOND);

    if (MINUTE < 10)
    {
        time = time + ":0"+MINUTE;
    }
    else
    {
        time = time + ":"+MINUTE;
    }
    if (SECOND < 10)
    {
        time = time + ":0"+SECOND;
    }
    else
    {
        time = time + ":"+SECOND;
    }

    int DAY = calendar.get(Calendar.DAY_OF_WEEK);
    int DAYminusOne = calendar.get(Calendar.DAY_OF_WEEK) - 1;
    int MONTH = calendar.get(Calendar.MONTH) + 1;
    int YEAR = calendar.get(Calendar.YEAR);

    String name_of_month = months_array[calendar.get(Calendar.MONTH)];
    String date = days_array[DAYminusOne]+" "+DAY+" "+name_of_month+" "+YEAR;
    String time_and_date = "Time: "+time+"\n"+Date: "+date;
    return time_and_date;
}

```

```

public static void CreatePdf(String fName, String Speed, int counter, String timeStamp)
{
    String reportFileName = "report_for_" + fName + ".pdf";
    File f = new File(reportFileName);
    Document document = new Document();

    if(f.exists()) {
        try {
            System.out.println("file already exists. Adding to report..\n");
            String prevContent = "";
            if(hMap.containsKey(fName)) {
                prevContent = hMap.get(fName);
            }
            PdfWriter.getInstance(document, new FileOutputStream(reportFileName));

            document.setPageSize(PageFormat.A4);
            document.open();

            Font font = new Font(Font.TIMES_ROMAN, 25, Font.BOLD);
            font.setColor(new Color(0, 0, 255));

            Chunk chunk = new Chunk(fName + " Report", font);
            Paragraph para = new Paragraph(chunk);
            para.setAlignment(Element.ALIGN_CENTER);
            document.add(para);

            String timeAndDateStr = getTimeAndDateString(counter);
            font = new Font(Font.TIMES_ROMAN, 15, Font.BOLD);
            font.setColor(new Color(0, 0, 0));

            String currContent = "\n\nCar Name: " + fName + "\nSpeed Limit: " + speed_limit + " Km/Hr\nCar
Speed: " + Speed + " Km/Hr\n" + timeAndDateStr;

            chunk = new Chunk(prevContent + currContent, font);
            hMap.put(fName, prevContent + currContent );
            para = new Paragraph(chunk);
            para.setAlignment(Element.ALIGN_LEFT);
            document.add(para);
            document.close();
        }
        catch (IOException iex) {
            iex.printStackTrace();
        }
        catch (DocumentException e) {
            e.printStackTrace();
        }
    }
    else {

        try
        {
            PdfWriter.getInstance(document, new FileOutputStream(reportFileName));

            document.setPageSize(PageFormat.A4);
            document.open();

            Font font = new Font(Font.TIMES_ROMAN, 25, Font.BOLD);
            font.setColor(new Color(0, 0, 255));
            Chunk chunk = new Chunk(fName + " Report", font);
            Paragraph para = new Paragraph(chunk);
            para.setAlignment(Element.ALIGN_CENTER);
            document.add(para);

```


D3. GUI Plot Codes

```
import java.awt.BasicStroke;
import java.awt.*;
import java.awt.Color;
import java.awt.GradientPaint;
import java.awt.Graphics2D;
import java.awt.Paint;
import java.awt.Shape;
import java.io.BufferedReader;
import java.io.FileReader;

import javax.swing.*;

import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.axis.NumberAxis;
import org.jfree.chart.axis.NumberTickUnit;
import org.jfree.chart.axis.ValueAxis;
import org.jfree.chart.plot.CombinedDomainXYPlot;
import org.jfree.chart.plot.ValueMarker;
import org.jfree.chart.plot.XYPlot;
import org.jfree.chart.renderer.xy.XYLineAndShapeRenderer;
import org.jfree.data.Range;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;
import org.jfree.ui.ApplicationFrame;
import org.jfree.ui.Layer;
import org.jfree.ui.RefineryUtilities;

import org.jfree.chart.annotations.XYPointerAnnotation;
import org.jfree.ui.TextAnchor;

public class PlotWithArrowLabelsAndNoLines extends ApplicationFrame
{
    private static final long serialVersionUID = 1L;
    private XYSeriesCollection dataset;
    private double lastValue;
    private static double Threshold;
    private XYPlot subplot;
    JTextPane status;
    JSplitPane splitPane;

    private static double seconds = 30.0;
    XYLineAndShapeRenderer renderer = new XYLineAndShapeRenderer()
    {
        private static final long serialVersionUID = 1L;
        @Override
        public Paint getItemPaint(int row, int col)
        {
            Paint cpaint = getItemColor(row, col);
            if (cpaint == null)
            {
                cpaint = super.getItemPaint(row, col);
            }
            return cpaint;
        }
    };

    public Color getItemColor(int row, int col)
    {
        double y = dataset.getYValue(row, col);
```

```

        if( (int) y > (int) Threshold )
        {
            return Color.red;
        }
        else if( (int) y == (int) Threshold )
        {
            return Color.yellow;
        }
        else
        {
            return Color.blue;
        }
    }

    @Override
    protected void drawFirstPassShape(Graphics2D g2, int pass, int series, int item, Shape shape)
    {
        g2.setStroke(getItemStroke(series, item));
        Color c1 = getItemColor(series, item);
        Color c2 = getItemColor(series, item - 1);
        Color c3 = new Color (0,0,0,0);
        Paint linePaint = Color.black;

        g2.setPaint(c3);
        g2.draw(shape);
    }
};

public PlotWithArrowLabelsAndNoLines(final String title, double threshold_value)
{
    super(title);
    Threshold = threshold_value;
    final CombinedDomainXYPlot plot = new CombinedDomainXYPlot(new NumberAxis("Sec"));

    this.lastValue = 0.0;
    final XYSeries series = new XYSeries("velo");

    this.dataset = new XYSeriesCollection(series);

    final NumberAxis rangeAxis = new NumberAxis("Speed");
    rangeAxis.setAutoRangeIncludesZero(false);

    subplot = new XYPlot( this.dataset, null, rangeAxis, renderer);

    subplot.setBackgroundPaint(Color.lightGray);
    subplot.setDomainGridlinePaint(Color.white);
    subplot.setRangeGridlinePaint(Color.white);

    ValueMarker mark = new ValueMarker(Threshold,Color.RED, new BasicStroke(2));
    subplot.addRangeMarker(mark, Layer.BACKGROUND);

    ValueAxis y = subplot.getRangeAxis();
    y.setRange(new Range(0, 110.0));

    plot.add(subplot);

    final JFreeChart chart = new JFreeChart("Speed Plot", plot);

    chart.setBorderPaint(Color.black);
    chart.setBorderVisible(true);
    chart.setBackgroundPaint(Color.white);
    chart.removeLegend();
}

```

```

plot.setBackgroundPaint(Color.lightGray);
plot.setDomainGridlinePaint(Color.white);
plot.setRangeGridlinePaint(Color.white);

NumberAxis xAxis = (NumberAxis) plot.getDomainAxis();
xAxis.setAutoRange(true);

xAxis.setTickUnit(new NumberTickUnit(seconds));
JPanel content = new JPanel(new BorderLayout());

status = new JTextPane();
JScrollPane sp = new JScrollPane(status);

final ChartPanel chartPanel = new ChartPanel(chart);
chartPanel.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));

splitPane = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT, sp , chartPanel);
splitPane.setOneTouchExpandable(true);
splitPane.setDividerLocation(210);
sp.setMinimumSize(new Dimension(80, 50));
sp.setMaximumSize(new Dimension(210, 50));

chartPanel.setMinimumSize(new Dimension(715, 50));
chartPanel.setMaximumSize(new Dimension(1000, 50));

splitPane.setPreferredSize(new Dimension(1000, 700));

this.setContentPane(splitPane);
}

public void addMessage(String msg) {
    String content = status.getText();

    content = msg + "\n" + content;
    status.setText(content);
}

public void plotData(String yVal_string, String filename)
{
    double xVal = this.lastValue + 30.0;
    this.lastValue = xVal;
    double yVal = Double.valueOf( yVal_string );
    double tmpY = yVal*100;
    tmpY = Math.round(tmpY);
    double tmpFinal = tmpY/100;

    String labelString = filename + " : " + String.valueOf(tmpFinal);
    this.dataset.getSeries(0).add(xVal, yVal);

    XYPointerAnnotation xypointerannotation = new
    XYPointerAnnotation(labelString, xVal, yVal, 10.0D);
    xypointerannotation.setLabelOffset(9D);
    xypointerannotation.setTextAnchor(TextAnchor.BOTTOM_CENTER);

    if( (int) yVal > Threshold )
    {
        xypointerannotation.setBackgroundPaint(Color.red);
    }
    else if( (int) yVal == Threshold )
    {
        xypointerannotation.setBackgroundPaint(Color.yellow);
    }
    else
}

```

```
{  
    xypointerannotation.setBackgroundPaint(Color.cyan);  
}  
  
subplot.addAnnotation(xypointerannotation);  
System.out.println("Plotting X = " + xVal + " Y=" + yVal);  
  
try  
{  
    Thread.sleep(100);  
}  
catch (InterruptedException e)  
{  
    e.printStackTrace();  
}  
}  
}  
}
```