Jaypee University of Information Technology
Waknaghat, Distt. Solan (H.P.)
Learning Resource Center

CLASS NUM: BOOK NUM.:

ACCESSION NO.: SP09031 / SP0913032

This book was issued is overdue due on the date stamped below. If the book is kept over due, a fine will be charged as per the library rules.

Due Date	Due Date	Due Date

Vehicle Tracking System Using Radio Frequency Identification (RFID)

Project Report submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology

In

INFORMATION TECHNOLOGY

Under the Supervision of

DR. VIVEK SEHGAL

By

GAURAV SHARMA (091457)
AISHANI SOOD (091428)

to

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY

Jaypee University of Information and Technology

Waknaghat, Solan – 173234, Himachal Pradesh

Learning Resource Center JUIT Waknaghat Solan HP

Certificate

This is to certify that project report entitled "**Vehicle Tracking System Using Radio Frequency Identification (RFID)**", submitted by **Gaurav Sharma (091457) and Aishani Sood (091428)** in partial fulfillment for the award of degree of Bachelor of Technology in Information Technology to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Date:

Nitin

Dr Vivek Sehgal

Assistant Professor

Acknowledgments

We would like to take this opportunity to express our extreme gratitude and deep regards to our project guide **Dr. Vivek Sehgal** for his tremendous support and exemplary guidance. He has been a constant source of encouragement and motivation throughout the course of our project. Without his experience, knowledge and guidance this project would not have materialized.

We would also like to extend our thanks to the project coordinator **Dr. Nitin** for all the facilities provided to us for the working of this project and also the lab coordinators for the environment provided to us.

The motivation and guidance received from the entire members who contributed to the success of this project. We are so very grateful for the help and guidance.

Dated

Gaurav Sharma

Gaurav Sharma (091457)

Aishani Sood.

Aishani Sood (091428)

Topic **Page No.**

LIST OF FIGURES

Abstract

The project aims at using RFID (Radio Frequency ID) for developing tracking systems for vehicles. A vehicle tracking system would effectively and efficiently monitor the where about of a registered vehicle. It will provide us with the accurate real-time location of the vehicles that have been registered with us. The RFID tag would be able to wirelessly transfer the information stored in it as soon as it receives signal from the RFID reader. It would also be able to identify if the passing registered vehicle is black listed or not. If it is, then an alert will pop up on the administrator screen. If not then it would just update the location of the vehicle in the database. The RFID has more potential because it needs no line of sight to relay the information and it can read several tags simultaneously.

CHAPTER 1: INTRODUCTION

Security is an important issue to be taken care of in any organizations .Good security system will reduce incidents like accidents, theft, intrusion etc. An automatic identification is vital to organizations. Technologies that use machine to identify objects such as voice recognition, optical recognition, smart cards and RFID.

Radio-frequency identification (RFID) is an automatic identification method, relying on storing and remotely retrieving data using devices called RFID tags or transponders. The technology requires some extent of assistance of an RFID reader and an RFID tag. An RFID tag is an object that can be applied to or included into a product, animal, or person for the purpose of identification and tracking using radio waves. Some tags can be read from several meters away and beyond the line of sight of the reader.
Some tags can be read from several meters away and beyond the line of sight of the reader.

A Vehicle Tracking System allows an organization to monitor and manage vehicles more effectively and efficiently. Vehicle Tracking System helps not only in tracking and identifying vehicles along with its owners and users, but also monitors its movement and pops with alerts and alarms accordingly.

1.1 Problem Statement

The task at hand is to track and identify a vehicle which has been fitted with a RFID tag and has been registered in the database. When a reader picks up a signal from a tag nearby it should be able to register the information to the database and it should also be able to alert the admin with the help of a pop up box if the car is black listed or not.

1.2 Requirements

1.2.1 Functional requirements:

- Fully atomized system to identify vehicle in and out.
- Tracks an unlimited number of vehicles with complete details of vehicle.
- Popup window that prompts you when any blacklisted vehicle is spotted.
- Can conveniently read long range and identify vehicle.
- Automatically conduct error free data gathering from rfid tag.
- Eliminate unauthorized vehicle use by notifying administration or any third party.
- Secure, efficient and accurate in identification of vehicle.
- Data can be captured even vehicle is on the move.

1.2.2 Nonfunctional requirements:

- Easy plug and play installation with low cost makes it affordable.
- The tags are re-usable.
- It is reliable for data security and maintenance.
- It is flexible with many user friendly features.
- Data accuracy is maintained with automated processes of import and verification.

1.2.3 Hardware requirements:

- Radio frequency identification reader
- Radio frequency identification tags
- Computer hardware
- Database server

1.2.4 Software requirements:

- **Software package :** Visual studio .NET 2010
- **Language for development :** C#.NET
- **Database :** MS SQL 2008

CHAPTER 2: INTRODUCTION TO RFID

Radio-frequency identification (RFID) is the use of a wireless non-contact system that uses radio-frequency electromagnetic fields to transfer data from a tag attached to an object, for the purposes of automatic identification and tracking. Some tags require no battery and are powered and read at short ranges via magnetic fields (electromagnetic induction). Others use a local power source and emit radio waves (electromagnetic radiation at radio frequencies). The tag contains electronically stored information which can be read from up to several meters (yards) away. Unlike a bar code, the tag does not need to be within line of sight of the reader and may be embedded in the tracked object.

RFID tags are used in many industries. An RFID tag attached to an automobile during production can be used to track its progress through the assembly line. Pharmaceuticals can be tracked through warehouses. Livestock and pets may have tags injected, allowing positive identification of the animal.

Since RFID tags can be attached to clothing, possessions, or even implanted within people, the possibility of reading personally-linked information without consent has raised privacy concerns.

2.1 Design:

A radio-frequency identification system uses tags, or labels attached to the objects to be identified. Two-way radio transmitter-receivers called interrogators or readers send a signal to the tag and read its response. The readers generally transmit their observations to a computer system running RFID software or RFID middleware.

The tag's information is stored electronically in a non-volatile memory. The RFID tag includes a small RF transmitter and receiver. An RFID reader transmits an encoded radio

signal to interrogate the tag. The tag receives the message and responds with its identification information. This may be only a unique tag serial number, or may be product-related information such as a stock number, lot or batch number, production date, or other specific information.

RFID tags can be either passive, active or battery assisted passive. An active tag has an on-board battery and periodically transmits its ID signal. A battery assisted passive (BAP) has a small battery on board and is activated when in the presence of a RFID reader. A passive tag is cheaper and smaller because it has no battery. Instead, the tag uses the radio energy transmitted by the reader as its energy source. The interrogator must be close for RF field to be strong enough to transfer sufficient power to the tag. Since tags have individual serial numbers, the RFID system design can discriminate several tags that might be within the range of the RFID reader and read them simultaneously.

Tags may either be read-only, having a factory-assigned serial number that is used as a key into a database, or may be read/write, where object-specific data can be written into the tag by the system user. Field programmable tags may be write-once, read-multiple; "blank" tags may be written with an electronic product code by the user.

RFID tags contain at least two parts: an integrated circuit for storing and processing information, modulating and demodulating a radio-frequency (RF) signal, collecting DC power from the incident reader signal, and other specialized functions; and an antenna for receiving and transmitting the signal.

Fixed readers are set up to create a specific interrogation zone which can be tightly controlled. This allows a highly defined reading area for when tags go in and out of the interrogation zone. Mobile readers may be hand-held or mounted on carts or vehicles.

Band	Regulations	Range	Data speed	Remarks	Approximate tag cost in volume (2006) US $
120–150 kHz (LF)	Unregulated	10 cm	Low	Animal identification, factory data collection	$1
13.56 MHz (HF)	ISM band worldwide	1 m	Low to moderate	Smart cards	$0.50
433 MHz (UHF)	Short Range Devices	1–100 m	Moderate	Defense applications, with active tags	$5
865-868 MHz (Europe) 902-928 MHz (North America) UHF	ISM band	1–2 m	Moderate to high	EAN, various standards	$0.15 (passive tags)
2450-5800 MHz (microwave)	ISM band	1–2 m	High	802.11 WLAN, Bluetooth standards	$25 (active tags)
3.1–10 GHz (microwave)	Ultra wide band	to 200 M	High	requires semi-active or active tags	$5 projected

Table 1: RFID frequency bands

Signaling between the reader and the tag is done in several different incompatible ways, depending on the frequency band used by the tag. Tags operating on LF and HF frequencies are, in terms of radio wavelength, very close to the reader antenna, only a small percentage of a wavelength away. In this near field region, the tag is closely coupled electrically with the transmitter in the reader. The tag can modulate the field produced by the reader by changing the electrical loading the tag represents. By switching between lower and higher relative loads, the tag produces a change that the reader can detect. At UHF and higher frequencies, the tag is more than one radio wavelength away from the reader, requiring a different approach. The tag can backscatter a signal. Active tags may contain functionally separated transmitters and receivers, and the tag need not respond on a frequency related to the reader's interrogation signal.

An Electronic Product Code (EPC) is one common type of data stored in a tag. When written into the tag by an RFID printer, the tag contains a 96-bit string of data. The first eight bits are a header which identifies the version of the protocol. The next 28 bits identify the organization that manages the data for this tag; the organization number is assigned by the EPCGlobal consortium. The next 24 bits are an object class, identifying the kind of product; the last 36 bits are a unique serial number for a particular tag. These last two fields are set by the organization that issued the tag. Rather like a URL, the total electronic product code number can be used as a key into a global database to uniquely identify a particular product.

Often more than one tag will respond to a tag reader, for example, many individual products with tags may be shipped in a common box or on a common pallet. Collision detection is important to allow reading of data. Two different types of protocols are used to "singulate" a particular tag, allowing its data to be read in the midst of many similar tags. In a slotted Aloha system, the reader broadcasts an initialization command and a parameter that the tags individually use to pseudo-randomly delay their responses. When using an "adaptive binary tree" protocol, the reader sends an initialization symbol and then transmits one bit of ID data at a time; only tags with matching bits respond, and eventually only one tag matches the complete ID string.

2.2 Why we are using RFID

- **Low cost:** The RFID setup is cheap and thus cost efficient. They can be used by organizations and institutions without spending allot for the installation.
- **Power source:** we are using the semi active RFID, thus the power source used is small.
- **Line of sight:** Bar code reads can sometimes be problematic due to the need to have a direct "line of sight" between a scanner and a bar code. RFID tags can be read through materials without line of sight.
- **Simultaneous reading:** the reader can read many RFID tags at a given time, thus it is helpful in high speed operations such as vehicle tracking.
- **Automated reading**: RFID tags can be read automatically when a tagged product comes past or near a reader, reducing the labor required to scan product and allowing more proactive, real-time tracking.
- **Small in size:** the RFID tag is small in size thus its installation is easy and it doesn't disrupt any other process of the vehicle nor does it take up any space.
- **Greater data capacity:** Rfid tags can be encoded with detail items such as identification number, owner etc. easily.
- **Tamper proof:** the RFID is small in size so it cannot be located easily and thus the data in it cannot be tampered with. This is an important feature from the security point of view.

2.3 The history of RFID

Whether we realize it or not, radio frequency identification (RFID) is an integral part of our life. RFID increases productivity and convenience. RFID is used for hundreds, if not thousands, of applications such as preventing theft of automobiles and merchandise; collecting tolls without stopping; managing traffic; gaining entrance to buildings; automating parking; controlling access of vehicles to gated communities, corporate campuses and airports; dispensing goods; providing ski lift access; tracking library books;
buying hamburgers; and the growing opportunity to track a wealth of assets in supply chain management. RFID technology is also being pressed into service for use in U.S. Homeland Security with applications such as securing border crossings and intermodal container shipments while expediting low-risk activities.

RFID is a term coined for short-range radio technology used to communicate mainly digital information between a stationary location and a movable object or between movable objects. A variety of radio frequencies and techniques are used in RFID systems. RFID is generally characterized by use of simple devices on one end of the link and more complex devices on the other end of the link. The simple devices (often called tags or transponders) are small and inexpensive, can be deployed economically in very large numbers, are attached to
the objects to be managed, and operate automatically. The more complex devices (often called readers, interrogators, beacons) are more capable and are usually connected to a host computer or network. Radio frequencies from 100 kHz to 10 GHz have been used.
The tags are usually built using CMOS circuitry while other technologies can be used such as surface acoustic wave (SAW) devices or tuned resonators. Tags can be powered by a battery or by rectification of the radio signal sent by the reader. Tags can send data to the reader by changing the loading of the tag antenna in a coded manner or by generating, modulating, and transmitting a radio signal. A variety of modulation and

coding techniques have been used. RFID systems can be readonly (data is transferred only in one direction, from the tag to the reader) or readwrite (two-way communication).

A typical RFID system can use the principle of modulated backscatter .In this type of RFID system, to transfer data from the tag to the reader, the reader sends an unmodulated signal to the tag. The tag reads its internal memory of stored data and changes the loading on the tag antenna in a coded manner corresponding to the stored data. The signal reflected from the tag is thus modulated with this coded information. This modulated signal is received by the reader, demodulated using a homodyne receiver, and decoded and output as digital information that contains the data stored in the tag. To send data from the reader to the tag, the reader amplitude modulates its transmitted radio signal. This modulated signal is received by the tag and detected with a diode. The data can be used to control operation of the tag, or the tag can store the data. A simple diode detector allows the detection circuitry in the tag to be simple and consume little power.

Mankind's use and understanding of electricity, magnetism, and electromagnetics in very early times was limited to his eyesight, observation of electrostatic discharge (don't stand under a large tree during a lightning storm), and the magnetic properties of lodestones. Early applications probably included making light with fire, use of mirrors for signaling, and use of lodestones for navigation.

Scientific understanding progressed very slowly until about the 1600s. From the 1600s to 1800s there was an explosion of observational knowledge of electricity, magnetism, and optics accompanied by a growing base of mathematically related observations. The 1800s marked the beginning of the fundamental understanding of electromagnetic energy. In 1846, English experimentalist Michael Faraday proposed that both light and radio waves are a form of electromagnetic energy. In 1864, Scottish physicist James Clerk Maxwell published his theory on electromagnetics. In 1887, German physicist Heinrich Rudolf Hertz confirmed Maxwell's electromagnetic theory and produced and studied electromagnetic waves (radio waves). Hertz is credited as the first to transmit and receive

radio waves, and his demonstrations were followed quickly by Aleksandr Popov in Russia.

The Decades of RFID	
Decade	**Event**
1940–1950	Radar refined and used, major World War II development effort. RFID invented in 1948.
1950–1960	Early explorations of RFID technology, laboratory experiments.
1960–1970	Development of the theory of RFID. Start of applications field trials.
1970–1980	Explosion of RFID development. Tests of RFID accelerate. Very early adopter implementations of RFID.
1980–1990	Commercial applications of RFID enter mainstream.
1990–2000	Emergence of standards. RFID widely deployed. RFID becomes a part of everyday life.
2000–	RFID explosion continues

Figure 1 Evolution of RFID

2.4 Components of RFID

A basic RFID system consists of three components:

- An antenna or coil
- A transceiver (with decoder)
- A transponder (rf tag) electronically programmed with unique information

2.4.1 ANTENNA

The antenna emits radio signals to activate the tag and read and write data to it. Antennas are the conduits between the tag and the transceiver, which controls the system's data acquisition and communication. Antennas are available in a variety of shapes and sizes; they can be built into a door frame to receive tag data from persons or things passing through the door, or mounted on an interstate tollbooth to monitor traffic passing by on a freeway. The electromagnetic field produced by an antenna can be constantly present when multiple tags are expected continually. If constant interrogation is not required, a sensor device can activate the field. Often the antenna is packaged with the transceiver

and decoder to become a reader (a.k.a. interrogator), which can be configured either as a handheld or a fixed-mount device. The reader emits radio waves in ranges of anywhere from one inch to 100 feet or more, depending upon its power output and the radio frequency used. When an RFID tag passes through the electromagnetic zone, it detects the reader's activation signal. The reader decodes the data encoded in the tag's integrated circuit (silicon chip) and the data is passed to the host computer for processing

2.4.2 TAGS (Transponders)

An RFID tag is comprised of a microchip containing identifying information and an antenna that transmits this data wirelessly to a reader. At its most basic, the chip will contain a serialized identifier, or license plate number, that uniquely identifies that item, similar to the way many bar codes are used today. A key difference, however is that RFID tags have a higher data capacity than their bar code counterparts. This increases the options for the type of information that can be encoded on the tag, including the manufacturer, batch or lot number, weight, ownership, destination and history (such as the temperature range to which an item has been exposed). In fact, an unlimited list of other types of information can be stored on RFID tags, depending on application needs. An RFID tag can be placed on individual items, cases or pallets for identification purposes, as well as on fixed assets such as trailers, containers, totes, etc.

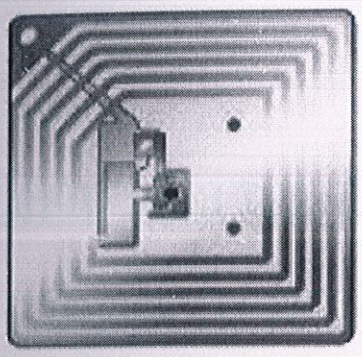
Tags come in a variety of types, with a variety of capabilities. Key variables include:

"Read-only" versus "read-write"

There are three options in terms of how data can be encoded on tags:

1. Read-only tags contain data such as a serialized tracking number, which is pre-written onto them by the tag manufacturer or distributor. These are generally the least expensive tags because they cannot have any additional information included as they move throughout the supply chain. Any updates to that information would have to be maintained in the application software that tracks SKU movement and activity.
2. "Write once" tags enable a user to write data to the tag one time in production or distribution processes. Again, this may include a serial number, but perhaps other data such as a lot or batch number.

3. Full "read-write" tags allow new data to be written to the tag as needed—and even written over the original data. Examples for the latter capability might include the time and date of ownership transfer or updating the repair history of a fixed asset. While these are the most costly of the three tag types and are not practical for tracking inexpensive items, future standards for electronic product codes (EPC) appear to be headed in this direction.



RFID TAGS

Figure 2 RFID tags

2.4.3 RF Transceiver:

The RF transceiver is the source of the RF energy used to activate and power the passive RFID tags. The RF transceiver may be enclosed in the same cabinet as the reader or it may be a separate piece of equipment. When provided as a separate piece of equipment, the transceiver is commonly referred to as an RF module. The RF transceiver controls and modulates the radio frequencies that the antenna transmits and receives.

2.5 Problems with RFID

Some common problems with RFID are reader collision and tag collision. Reader collision occurs when the signals from two or more readers overlap. The tag is unable to respond to simultaneous queries. Systems must be carefully set up to avoid this problem. Tag collision occurs when many tags are present in a small area; but since the read time is very fast, it is easier for vendors to develop systems that ensure that tags respond one at a time.

CHAPTER3: LITERATURE REVIEW

- RFID-based Ticketing for Public Transport System
- Embedded security system using RFID and GSM
- Land Vehicle Tracking System Using Java on Android Platform

3.1 RFID-based Ticketing for Public Transport System:

3.1.1 Abstract-The paper based public transport ticketing system, prevailing in the megacity Dhaka (Bangladesh), introduces severe malfunction in the system, malicious argument among public, corruption and most of all traffic jam. This paper actually suggests a much more public friendly, automated system of ticketing as well as the credit transaction with the use of RFID based tickets. The total system mainly acts to bring out the consistency among various bus agencies that will conclude in uniform access of passengers in daily rides through an automated server being updated every single time the passengers travel by carrying the RFID based tickets

3.1.2 Introduction

As for the RFID application, it's been a widespread tool for both tracking the transit transports and for the public ticketing system. It's already been an outstanding achievement throughout the globe including big cities like London, Helsinki, Shanghai, Istanbul, Moscow, Porto and many more. The system can be implemented for subways, railways and public bus services for the sake of systematic operations in corresponding cases. In the megacity Dhaka, the conventional system of public transport is based on paper based bus or railway tickets that ultimately lead to chaos among public, system loss, corruption and most of all traffic jam that is responsible for a huge wastage of time. No prior notification of the arrival and departure of the transports are available creating a

lot of confusion among the passengers resulting in a rough argument between them and the bus supervisors or the operators. Again having no government authority to take control or keep an eye over the whole scenario, the private sectors are creating a monopoly, taking control over the public transport and autocratic raise in bus fair.

The tracking and ticketing systems using RFID can be merged to solve the prevailing problems. Even though the GPS based system can be designed, we propose the RFID based tickets for its low cost, easy operation, portability, durability, reliability and being much more user friendly. Also the high speed RFID tags and detectors make the tracking system of a running bus merely a child's play. Public carrying RFID based electronic tickets will have access to any bus service of the city only entering his current location and his destination on the keypad attached to every bus. The data will directly be transferred to the server main database and the equivalent credit will be stored in the corresponding bus account. Also the screen at every bus stop will notify the passengers, the departure time of the last bus of any route. This automated system will save time, have a higher authoritative inspection and reduce chaos and confusion on the road.

3.1.3 System description

Radio Frequency Identification (RFID) is a generic term for technologies that use radio waves to automatically identify and track product, animal, or person by means of using RFID tags that are applied or incorporated on them. An RFID system consists of a tag, basically a microchip with an antenna and an interrogator or reader with an antenna. Most RFID tags contain at least two parts that is shown in figure1. One is an integrated circuit for storing and processing information, modulating and demodulating a radio-frequency (RF) signal, and other specialized functions. The second is an antenna for receiving and transmitting the signal. For the purpose of Bus Identification, the tags are embedded into the bus. Each bus will have two tags: one is at front and other is at rear. The front tag will inform the reader about its arrival to the bus stop whereas the rear one informs its departure. Each bus will also have a reader that is connected to the main server for charging of ticket fare from the passengers through a keypad attached with the reader on which the passengers give the information of their departure & destination locations. The

reader sends the electromagnetic waves to the tag. The tags draw the power from this wave and return back the bus information, which are stored in its memory to reader. The readers again demodulate this wave and convert it as a digital data. For the purpose of Ticketing, the operational feature of the cards is almost the same but here the tags are attached to special cards carried by the passengers and the reader collects the detail from them. By using RFID technology in ticketing system, allowing passengers to "tag on" and "tag off" and be charged automatically, according to how many zones they have travelled.

3.1.4 Operational principle of the proposed system

Considering from the arrival of a bus at the bus stoppage, the reader will read the RFID tag attached to the front side of the bus that is denoted as the front tag. Thus the reader will have the idea of the bus and also the route of the bus along with the arrival time. Also the reader being connected to the main server, the data will automatically transfer to the server database. The screen in the bus stop will notify waiting passengers about the arrived bus and its route. All the passengers will carry a prepaid system RFID based card that will have a unique ID number. The card is rechargeable from certain electronic booths placed at certain locations of the city. The passenger trying to get on board will have to place the RFID ticket in front of the reader attached to every bus. The reader will detect the tag and require certain information from the passenger. According to the route distance between departure & destination as well as considering bus type, it will calculate the ticket fare and deduct the credit from the RFID tag based ticket electronically. The complete detection algorithm is described in detailed in the later part. After all the passengers getting on board, the bus will leave the stoppage and the reader will detect the rear tag attached to the bus .The reader will send the information to the server and also to the screen showing the departure time of the bus. If a agency has a bus service that the buses come after each 20 minutes, from the screen above the waiting passenger will surely know when the last bus departed and after how long the next bus is coming. After the whole day, the individual bus reader will know how much credit has been transferred to the corresponding account and also the information can be found in the main database. Cross checking of all those information will allow better monitoring, transparency and thus reducing corruption.

Passenger ID	City ID	Ticket ID	Credit Balance	Check Bit	Others

Figure 3 database for rfid based ticketing system

3.1.5 Detection and processing algorithm

As soon as the tag is placed before the reader attached to the bus, the tag will get energized revealing relevant information to the reader. An authenticated tag carrier will enter the start location and end location information through the keypad. The reader will accept the card if the card has required credit to travel that distance. The data acquired by the reader will be stored in its internal memory as well as transferred to the main server database. After the whole day, the internal memory of the reader will be reset for the next day.

3.1.6 Conclusion

The system is expected to be fully automated, reliable, transparent and convenient. The whole system can also be used in vehicle on highways, their toll payment and in the railway ticketing system with small or no modification. The cards being reusable, they are much more convenient compared to the paper based ticketing system. The card also can be used to be a universal travel pass card that will allow any transportation on any route. Any unwanted events can be avoided as all the person carrying RFID tickets are monitored every time they travel. Also the possibilities of reducing traffic jams, chaos in the bus stoppage that we usually experienced in Dhaka city are immense.

3.2 Embedded security system using RFID and GSM

3.2.1 Abstract— The petroleum products are one of the valuable and rare creations of the nature. The proper use and distribution is important task to survive these products. Our system may be the first approach towards security of petroleum products distribution such as petrol, diesel, and kerosene etc. "The simple and proper use of RFID and GSM technologies can provide total security for distribution of petroleum products!" this our proposed design. Simple embedded system and direct PC interface for the system which facilitates the record keeping of the distributed fuel. Also the handy and robust VB program will help to authorized company to control the distribution of fuel whole over the region or country. In our system the control unit and tanker unit are two main parts. The two systems which may far away from each other can easily communicate with each other. The security code in RFID tag provided to the petrol pump get read by the reader and transmission of it to the control unit will helps to company to create the proper database of various petrol pumps distributed over wide area. Also the distribution of the fuel is not possible until control unit provides the proper command to the valve in tanker unit. In short the project we have developed is the basic attachment of all above devices; which will use to provide security to the fuel distribution and helps the data keeping of the distributed fuel. The advancement of the project to large scale can help financially to the industry indirectly.

3.2.2 Introduction

Fuel is the one of the most essential thing in today's world. We can see number of petrol pumps around us. Our aim is to develop the security system for the petrol distribution tankers of Petroleum Company. The aim of the system is to open or close the tank-valve of the tanker controlling from control cabin. We will use GSM technology for this purpose.

The system will consist of two units; one unit will be placed at tanker which will monitor continuously the fuel level in the tank. The initial original fuel level and current fuel level will be displayed on LCD at front for driver's convenience. Second is the RFID assembly which will read the authentication code of the petrol pump. The amount of fuel poured at particular petrol pump and petrol pump ID will be send to central office through GSM techniques. The visual basic coding will help to control (to check authorized petrol pump, to send valve opening signal) and to keep record of all these things (total fuel delivered, current fuel level, coasting of delivered petrol.).

3.2.3 Theoretical detail and analysis

The customer demanding the fuel from the petroleum industry will first call the industry to convey the requirement. Company will send the fuel via tanker to the petrol pump. Now, our system comes into existence in two parts, one is placed inside the tanker/vehicle and other is placed at the distributing industry itself. One unit which is placed at tanker will monitor continuously the fuel level in the tank. The initial original fuel level and current fuel level will be displayed on LCD at front for driver's convenience. The electronic valve is provided to keep the tanker opening block until it get the opening order from the microcontroller unit .The RFID assembly reads the authentication code of the petrol pump by swapping the reader over the RFID tag pasted at the petrol pump and send it to the control unit to update the database as well as to authenticate the customer who is demanding the petrol.

On the tanker unit side, the RFID system is connected to the microcontroller, level sensor and relay / electronic valve assembly; where the RFID reader will identify the authenticated user ID and send the information about level of fuel and user ID number to the Control cabin. The borrower just needs to convey the required amount of fuel to be poured to the industrial operator by any communication media.

On the other hand the control cabin consists of another GSM unit which receives the information from tanker unit and interface serially to personal computer so as to edit that data for further work of response. The further action is taken by the software like Visual basic with support of Microsoft access and predefined algorithm. The algorithm is made so as to identify the match between present identity and received code. If the received

data gets perfect match with the present data then the control units will response the tanker unit via GSM message in the form *01(amount of fuel).

The interrupt to the microcontroller to activate/ deactivate the control valve relay is provided by the SMS by the control cabin's GSM unit using "* 01X" topologies, where the „X" is amount of fuel. The microcontroller first separate the "*01" and "X", and using the value „X" it will open the valve. Here „*01" creates interrupt inside the microcontroller. On receiving the interrupt from the control cabin unit, the valve takes action accordingly. The microcontroller continuously sense the fuel level using level sensor and keeps the valve open until it reaches the quantity to be delivered by decrementing the counter.

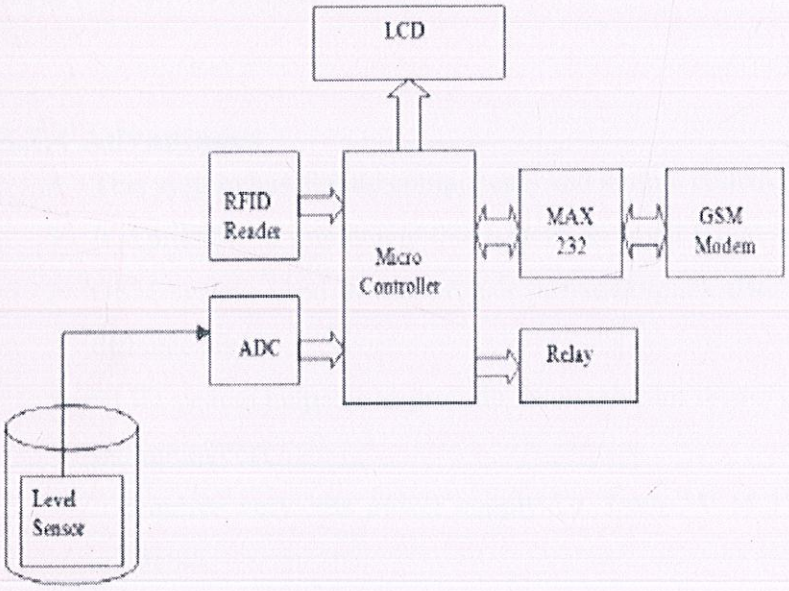


Figure 4 tanker unit

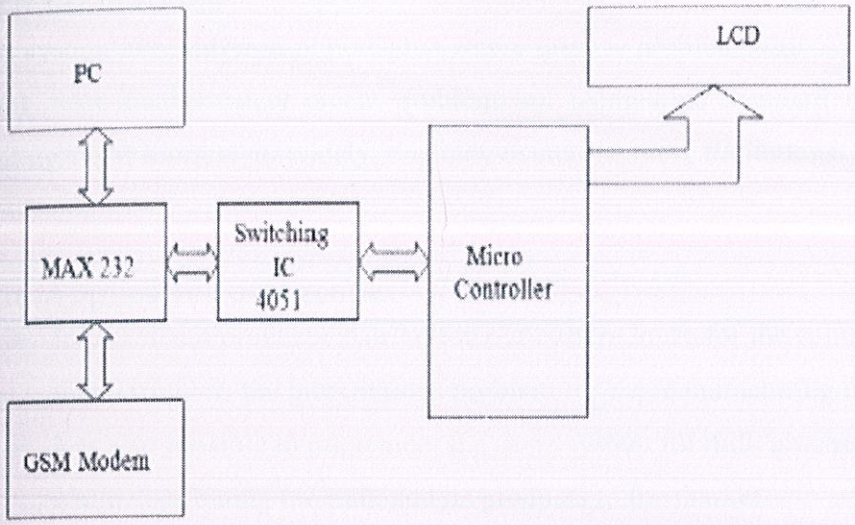


Figure 5: Control cabin unit

3.2.4 Advantages

- This system has simple components and simple construction of them on circuit.
- It is possible to implement this system on small board space also.
- GSM system used in our project provides quick data communication over long distance also.
- RFID system helps us to provide the maximum security to authenticate the user at minimum cost.
- It requires very less power supply i.e. from 5V to 12 V only, which is easily available.
- Also as it provides the central control on petrol distribution, thus there is no question of stilling or to change the record of distributed fuel.
- Easy to handle for distributor since only amount of fuel which is to be poured need to type on keypad and remaining work takes place automatically.

3.2.5 Limitations

This system may suffer at remote area where there is problem with GSM range. Also the attack from hackers may create problem but using high standard of encryption and availing GSM transceiver widely, one may overcome these limitations.

3.2.6 Application and scope

- In petroleum products distribution our system looks for the control on product thefts which is the most serious problem for the manufacturing industries.
- It is also possible to implement the same system for milk processing industries while distributing the milk and its products to the market.
- In day to day life we can see that water distribution in summer is also one of the problems in front of India. So it is possible to keep control on water distribution in particular area.
- The agricultural products like vegetables as well as processed fruits and its sub products may be securely distributed to the market using the same system we proposed.
- Also it is possible to keep record of the distributed products to the market; which is commercially most important for the industries.

3.2.7 Conclusion

In the world of electronics it is important to develop the new technology to make secure the distribution of fuel and keeping record of the same fuel with authorization of user. Our project is one idea which can change the face of today's manual system of distribution and data keeping. The total central access of all these activities provide the correct approach toward security and economical need of the industries since industry itself can control distribution as well as keep the record of the same fuel from thousands of miles seated in office. Also there is no option for the petrol pump or distributor to issue the fuel illegally that is total faithfulness of both the sides will get maintained. In short, this project probably can be implemented for the use of other tasks other than petrol distribution, on large scale to achieve various goals of industries.

3.3 Land Vehicle Tracking System Using Java on Android Platform

3.3.1 Abstract-- As urban living environment is becoming more and more complex, the road condition is becoming worse because of heavy traffic, increase of traffic accidents and high ratio of empty vehicles. It increases the cost of transportation and wastes time of vehicle movement. To solve such problems, a land vehicle tracking system has been developed. A land vehicle tracking system determines the position of land rover with a terminal with embedded GPS receiver or PCS phone and displays the position on a digital map. Recently, vehicle tracking technologies have brought some breakthrough in these areas: commercial vehicle operations, fleet management, dispatching, emergency rescue, hazard material monitoring, and security.

3.3.2 Introduction

A vehicle tracking system combines the installation of an electronic device in a vehicle, or fleet of vehicles, with purpose-designed computer software at least at one operational base to enable the owner or a third party to track the vehicle's location, collecting data in the process from the field and deliver it to the base of operation. Modern vehicle tracking systems commonly use GPS or GLONASS technology for locating the vehicle, but other types of automatic vehicle location technology can also be used. Vehicle information can be viewed on electronic maps via the Internet or specialized software.

3.3.3 Objectives of the proposed project

We are going to use GPS for locating the position of vehicle. We will also find the speed of the vehicle in real time to find whether a driver is adhering to the speed limits.

- We can track vehicles through android application using GPS to find out where a bus is using a web application which requires login of administrator.

- We can also find out speed and if driver breaks speed then we can fine them accordingly.
- Parents can also see the current location of their kids through real time update.
- When a stop comes we can intimate the administrator and the people sitting in bus to come in front for their stop.

3.3.4 Android

(Automated Numeration of Data Realized by Optimized Image Detection) Android is an operating system for mobile devices such as Smartphone and tablet computers. It is developed by the Open Handset Alliance led by Google. Android consists of a kernel based on the Linux kernel, with middleware, libraries and APIs written in C and application software running on an application framework which includes Java-compatible libraries based on Apache Harmony. Android uses the Dalvik virtual machine with just-in-time compilation to run Dalvikdex-code (Dalvik Executable), which is usually translated from Java bytecode.

3.3.5 ADT plug-in for Eclipse

Android Development Tools (ADT) is a plug-in for the Eclipse IDE that is designed to give you a powerful, integrated environment in which to build Android applications.ADT extends the capabilities of Eclipse to let you quickly set up new Android projects, create an application UI, add components based on the Android Framework API, debug your applications using the Android SDK tools, and even export signed (or unsigned) .apk files in order to distribute your application. Developing in Eclipse with ADT is highly recommended and is the fastest way to get started.

3.3.6 Android open source project

The Android Open Source Project (AOSP) is led by Google, and is tasked with the maintenance and development of Android. According to the project "The goal of the Android Open Source Project is to create a successful real-world product that improves

the mobile experience for end users."AOSP also maintains the Android Compatibility Program, defining an "Android compatible" device "as one that can run any application written by third-party developers using the Android SDK and NDK", to prevent incompatible Android implementations.

3.3.7 Linux kernel

Android's kernel is based on the Linux kernel and has further architecture changes by Google outside the typical Linux kernel development cycle. Android does not have a native X Window System nor does it support the full set of standard GNU libraries, and this makes it difficult to port existing Linux applications or libraries to Android. Certain features that Google contributed back to the Linux kernel, notably a power management feature called wake locks, were rejected by mainline kernel developers, partly because kernel maintainers felt that Google did not show any intent to maintain their own code.

3.3.8 Location technology

Nowadays, a substantial number of smart phone have multimedia ability and geo-locating ability. While some people may get confused with GPS and AGPS here we provide a brief background study about them.

3.3.9 Global Positioning System (GPS)

Global Positioning System is composed of satellites and GPS receivers.GPS receivers receive signals from the satellites orbiting in space in 6 different planes 20 kilometers away from Earth (Porcino, 2001). There are 24 satellites orbiting in space at present originally owned by United States government for military purposes and are now opened for commercial use. The GPS receiver installed in the mobile handsets will receive radio signals from satellites and compare with the local duplication of geo data to calculate its actual location on Earth. To increase the accuracy, data received from three satellites can perform the calculation of two- dimensional location, including the longitude and

latitude. For three- dimensional location information, consisting longitude, latitude and altitude, data from at least 4 satellites are required.

3.3.10 Applications

- Vehicle tracking systems are commonly used by fleet operators for fleet management functions such as fleet tracking, routing, dispatch, on-board information and security.
- Vehicle tracking systems are also popular in consumer vehicles as a theft prevention and retrieval device. Police can simply follow the signal emitted by the tracking system and locate the stolen vehicle.
- *Asset tracking:* Companies needing to track valuable assets for insurance.
- Field service management: Companies with a field service workforce for services such as repair or maintenance, must be able to plan field workers' time, schedule subsequent customer visits and be able to operate these departments efficiently.
- *Field sales:* Mobile sales professionals can access real-time locations.
- *Trailer tracking:* Haulage and Logistics companies often operate Lorries with detachable load carrying units.

3.3.11 Conclusion

Vehicle tracking system resulted in improving overall productivity with better fleet management that in turn offers better return on your investments. Better scheduling or route planning can enable you handle larger jobs loads within a particular time. Vehicle tracking both in case of personal as well as business purpose improves safety and security, communication medium, performance monitoring and increases productivity. So in the coming year, it is going to play a major role in our day-to-day living.

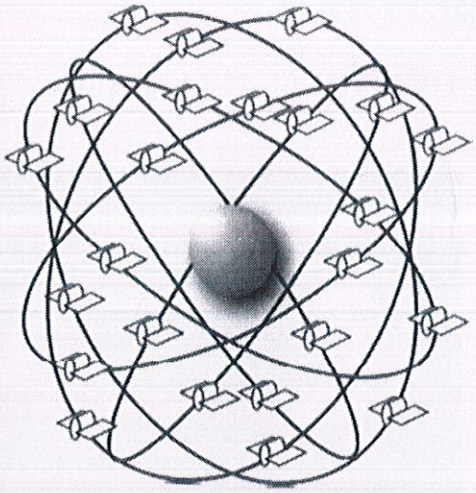


Figure 6 Global Positioning System

CHAPTER4: HARDWARE

4.1 serial port

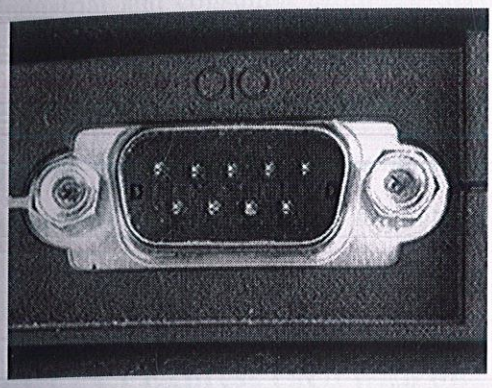


Figure 7: Serial port DB9

A male DE-9 connector used for a serial port on an IBM PC compatible computer along with the serial port symbol. (Pinout)

A male Mini DIN-8 connector used for a serial port on a Macintosh or SGI style computer.

In computing, a serial port is a serial communication physical interface through which information transfers in or out one bit at a time (in contrast to a parallel port). Throughout most of the history of personal computers, data was transferred through serial ports connected the computer to devices such as terminals and various peripherals.

While such interfaces as Ethernet, FireWire, and USB all send data as a serial stream, the term "serial port" usually identifies hardware more or less compliant to the RS-232 standard, intended to interface with a modem or with a similar communication device.

Modern computers without serial ports may require serial-to-USB converters to allow compatibility with RS 232 serial devices. Serial ports are still used in applications such as industrial automation systems, scientific instruments, shop till systems and some industrial and consumer products. Server computers may use a serial port as a control console for diagnostics. Network equipment (such as routers and switches) often use serial console for configuration. Serial ports are still used in these areas as they are simple, cheap and their console functions are highly standardized and widespread. A serial port requires very little supporting software from the host system.

4.1.1 PCI Express card with one serial port:

Some computers, such as the IBM PC, used an integrated circuit called a UART, that converted characters to (and from) asynchronous serial form, and automatically looked after the timing and framing of data. Very low-cost systems, such as some early home computers, would instead use the CPU to send the data through an output pin, using the bit-banging technique. Before large-scale integration (LSI) UART integrated circuits were common, a minicomputer or microcomputer would have a serial port made of multiple small-scale integrated circuits to implement shift registers, logic gates, counters, and all the other logic for a serial port.

Early home computers often had proprietary serial ports with pinouts and voltage levels incompatible with RS-232. Inter-operation with RS-232 devices may be impossible as the serial port cannot withstand the voltage levels produced and may have other differences that "lock in" the user to products of a particular manufacturer.

Low-cost processors now allow higher-speed, but more complex, serial communication standards such as USB and FireWire to replace RS-232. These make it possible to connect devices that would not have operated feasibly over slower serial connections, such as mass storage, sound, and video devices.

Many personal computer motherboards still have at least one serial port, even if accessible only through a pin header. Small-form-factor systems and laptops may omit RS-232 connector ports to conserve space, but the electronics are still there. RS-232 has been standard for so long that the circuits needed to control a serial port became very cheap and often exist on a single chip, sometimes also with circuitry for a parallel port.

4.1.2 DTE and DCE:

The individual signals on a serial port are unidirectional and when connecting two devices the outputs of one device must be connected to the inputs of the other. Devices are divided into two categories "data terminal equipment" (DTE) and "data circuit-terminating equipment" (DCE). A line that is an output on a DTE device is an input on a DCE device and vice-versa so a DCE device can be connected to a DTE device with a straight wired cable. Conventionally, computers and terminals are DTE while modems and peripherals are DCE.

If it is necessary to connect two DTE devices (or two DCE devices but that is more unusual) a special cable known as a null-modem cable must be used.

4.1.3 Connectors:

While the RS-232 standard originally specified a 25-pin D-type connector, many designers of personal computers chose to implement only a subset of the full standard: they traded off compatibility with the standard against the use of less costly and more compact connectors (in particular the DE-9 version used by the original IBM PC-AT). The desire to supply serial interface cards with two ports required that IBM reduce the size of the connector to fit onto a single card back panel. A DE-9 connector also fits onto a card with a second DB-25 connector that was similarly changed from the original Centronics-style connector. Starting around the time of the introduction of the IBM PC-AT, serial ports were commonly built with a 9-pin connector to save cost and space.

However, presence of a 9-pin D-subminiature connector is not sufficient to indicate the connection is in fact a serial port, since this connector was also used for video, joysticks, and other purposes.

Some miniaturized electronics, particularly graphing calculators and hand-held amateur and two-way radio equipment, have serial ports using a phone connector, usually the smaller 2.5 or 3.5 mm connectors and use the most basic 3-wire interface.

Many models of Macintosh favored the related RS-422 standard, mostly using German Mini-DIN connectors, except in the earliest models. The Macintosh included a standard set of two ports for connection to a printer and a modem, but some PowerBook laptops had only one combined port to save space.

The standard specifies 20 different signal connections. Since most devices use only a few signals, smaller connectors can often be used. For example, the 9 pin DE-9 connector was used by most IBM-compatible PCs since the IBM PC AT, and has been standardized as TIA-574. More recently, modular connectors have been used. Most common are 8P8C connectors. Standard EIA/TIA 561 specifies a pin assignment, but the "Yost Serial Device Wiring Standard" invented by Dave Yost (and popularized by the Unix System Administration Handbook) is common on Unix computers and newer devices from Cisco Systems. Many devices don't use either of these standards. 10P10C connectors can be found on some devices as well. Digital Equipment Corporation defined their own DECconnect connection system which was based on the Modified Modular Jack (MMJ) connector. This is a 6 pin modular jack where the key is offset from the center position. As with the Yost standard, DECconnect uses a symmetrical pin layout which enables the direct connection between two DTEs. Another common connector is the DH10 header connector common on motherboards and add-in cards which are usually converted via a cable to the more standard 9 pin DE-9 connector (and frequently mounted on a free slot plate or other part of the housing).

4.1.4 Pinout:

Signal		Origin		DB-25	DE-9 (TIA-574)	MMJ	8P8C ("RJ45")			10P10C ("RJ50")		
Name	Abbreviation	DTE	DCE				TIA-561	Yost	Cyclades[4]	National Instruments[5]	Cyclades[4]	Digi[6]
Transmitted Data	TxD	•		2	3	2	6	3	3	8	4	5
Received Data	RxD		•	3	2	5	5	6	6	9	7	6
Data Terminal Ready	DTR	•		20	4	1	3	2	2	7	3	9
Carrier Detect	DCD		•	8	1	—	2	7	7	10	8	10 (alt 2)
Data Set Ready	DSR		•	6	6	6	1		8	5	9	2 (alt 10)
Ring Indicator	RI		•	22	9	—		—	—	2	10	1
Request To Send	RTS	•		4	7	—	8	1	1	4	2	3
Clear To Send	CTS		•	5	8	—	7	8	5	3	6	8
Common Ground	G	common		7	5	3,4	4	4,5	4	6	5	7
Protective Ground	PG	common		1	—	—	—	—	—	—	1	4

Figure 8: List of commonly used RS-232 signals and pin assignments.

The signals are named from the standpoint of the DTE, for example, an IBM-PC compatible serial port. The ground signal is a common return for the other connections; it appears on two pins in the Yost standard but is the same signal. The DB-25 connector includes a second "protective ground" on pin 1. Connecting this to pin 7 (signal reference ground) is a common practice but not essential.

Note that EIA/TIA 561 combines DSR and RI, and the Yost standard combines DSR and DCD.

A converter from USB to an RS-232 compatible serial port; more than a physical transition, it requires a driver in the host system software and a built-in processor to emulate the functions of the IBM XT compatible serial port hardware.

4.1.5 Hardware abstraction:

Operating systems usually use a symbolic name to refer to the serial ports of a computer.

Unix-like operating systems usually label the serial port devices /dev/tty* (TTY is a common trademark-free abbreviation for teletype) where * represents a string identifying the terminal device; the syntax of that string depends on the operating system and the device. On Linux, 8250/16550 UART hardware serial ports are named /dev/ttyS*, USB adapters appear as /dev/ttyUSB* and various types of virtual serial ports do not necessarily have names starting with tty.

The Microsoft MS-DOS and Windows environments refer to serial ports as COM ports: COM1, COM2 .etc.

4.1.6 Common applications for serial ports:

The RS-232 standard is used by many specialized and custom-built devices. This list includes some of the more common devices that are connected to the serial port on a PC. Some of these such as modems and serial mice are falling into disuse while others are readily available.

Serial ports are very common on most types of microcontroller, where they can be used to communicate with a PC or other serial devices.

- Dial-up modems
- GPS receivers (typically NMEA 0183 at 4,800 bit/s)
- Bar code scanners and other point of sale devices
- LED and LCD text displays
- Satellite phones, low-speed satellite modems and other satellite based transceiver devices
- Flat-screen (LCD and Plasma) monitors to control screen functions by external computer, other AV components or remotes
- Test and measuring equipment such as digital multimeters and weighing systems
- Updating Firmware on various consumer devices.
- Some CNC controllers
- Uninterruptible power supply
- Stenography or Stenotype machines.

- Software debuggers that run on a second computer.
- Industrial field buses
- Printers
- Computer terminal, teletype
- Older digital cameras
- Networking (Macintosh AppleTalk using RS-422 at 230.4 kbit/s)
- Serial mouse
- Older GSM mobile phones
- Some Telescopes

Since the control signals for a serial port can be easily turned on and off by a switch, some applications used the control lines of a serial port to monitor external devices, without exchanging serial data. A common commercial application of this principle was for some models of uninterruptible power supply which used the control lines to signal "loss of power", "battery low alarm" and other status information. At least some Morse code training software used a code key connected to the serial port, to simulate actual code use. The status bits of the serial port could be sampled very rapidly and at predictable times, making it possible for the software to decipher Morse code.

4.1.7 Settings:

Many settings are required for serial connections used for asynchronous start-stop communication, to select speed, number of data bits per character, parity, and number of stop bits per character. In modern serial ports using a UART integrated circuit, all settings are usually software-controlled; hardware from the 1980s and earlier may require setting switches or jumpers on a circuit board. One of the simplifications made in such serial bus standards as Ethernet, FireWire, and USB is that many of those parameters have fixed values so that users can not and need not change the configuration; the speed is either fixed or automatically negotiated. Often if the settings are entered incorrectly the connection will not be dropped; however, any data sent will be received on the other end as nonsense.

4.1.8 Speed:

Serial ports use two-level (binary) signaling, so the data rate in bits per second is equal to the symbol rate in bauds. A standard series of rates is based on multiples of the rates for electromechanical teleprinters; some serial ports allow many arbitrary rates to be selected. The port speed and device speed must match. The capability to set a bit rate does not imply that a working connection will result. Not all bit rates are possible with all serial ports. Some special-purpose protocols such as MIDI for musical instrument control, use serial data rates other than the teleprinter series. Some serial port systems can automatically detect the bit rate.

The speed includes bits for framing (stop bits, parity, etc.) and so the effective data rate is lower than the bit transmission rate. For example with 8-N-1 character framing only 80% of the bits are available for data (for every eight bits of data, two more framing bits are sent).

Bit rates commonly supported include 75, 110, 300, 1200, 2400, 4800, 9600, 19200, 38400, 57600 and 115200 bit/s.[9] Crystal oscillators with a frequency of 1.843200 MHz are sold specifically for this purpose. This is 16 times the fastest bit rate and the serial port circuit can easily divide this down to lower frequencies as required.

4.1.9 Data bits:

The number of data bits in each character can be 5 (for Baudot code), 6 (rarely used), 7 (for true ASCII), 8 (for any kind of data, as this matches the size of a byte), or 9 (rarely used). 8 data bits are almost universally used in newer applications. 5 or 7 bits generally only make sense with older equipment such as teleprinters.

Most serial communications designs send the data bits within each byte LSB (Least significant bit) first. This standard is also referred to as "little endian." Also possible, but rarely used, is "big endian" or MSB (Most Significant Bit) first serial communications; this was used, for example, by the IBM 2741 printing terminal. (See Bit numbering for

more about bit ordering.) The order of bits is not usually configurable within the serial port interface. To communicate with systems that require a different bit ordering than the local default, local software can re-order the bits within each byte just before sending and just after receiving.

4.1.10 Parity:

Parity is a method of detecting errors in transmission. When parity is used with a serial port, an extra data bit is sent with each data character, arranged so that the number of 1 bits in each character, including the parity bit, is always odd or always even. If a byte is received with the wrong number of 1s, then it must have been corrupted. However, an even number of errors can pass the parity check.

Electromechanical teleprinters were arranged to print a special character when received data contained a parity error, to allow detection of messages damaged by line noise. A single parity bit does not allow implementation of error correction on each character, and communication protocols working over serial data links will have higher-level mechanisms to ensure data validity and request retransmission of data that has been incorrectly received.

The parity bit in each character can be set to none (N), odd (O), even (E), mark (M), or space (S). None means that no parity bit is sent at all. Mark parity means that the parity bit is always set to the mark signal condition and likewise space parity always sends the parity bit in the space signal condition. Aside from uncommon applications that use the 9th (parity) bit for some form of addressing or special signaling, mark or space parity is uncommon, as it adds no error detection information. Odd parity is more useful than even, since it ensures that at least one state transition occurs in each character, which makes it more reliable. The most common parity setting, however, is "none", with error detection handled by a communication protocol.

4.1.11 Stop bits:

Stop bits sent at the end of every character allow the receiving signal hardware to detect the end of a character and to resynchronize with the character stream. Electronic devices usually use one stop bit. If slow electromechanical teleprinters are used, one-and-one half or two stop bits are required.

4.1.12 Conventional notation:

The D/P/S (Data/Parity/Stop) conventional notation specifies the framing of a serial connection. The most common usage on microcomputers is 8/N/1 (8N1). This specifies 8 data bits, no parity, 1 stop bit. In this notation, the parity bit is not included in the data bits. 7/E/1 (7E1) means that an even parity bit is added to the seven data bits for a total of eight bits between the start and stop bits. If a receiver of a 7/E/1 stream is expecting an 8/N/1 stream, half the possible bytes will be interpreted as having the high bit set.

4.1.13 Flow control:

A serial port may use signals in the interface to pause and resume the transmission of data. For example, a slow printer might need to handshake with the serial port to indicate that data should be paused while the mechanism advances a line.

Common hardware handshake signals (hardware flow control) use the RS-232 RTS/CTS or DTR/DSR signal circuits. Generally, the RTS and CTS are turned off and on from alternate ends to control data flow, for instance when a buffer is almost full. DTR and DSR are usually on all the time and, per the RS-232 standard and its successors, are used to signal from each end that the other equipment is actually present and powered-up. However, manufacturers have over the years built many devices that implemented non-standard variations on the standard, for example, printers that use DTR as flow control.

Another method of flow control (software flow control) uses special characters such as XON/XOFF to control the flow of data. The XON/XOFF characters are sent by the receiver to the sender to control when the sender will send data, that is, these characters

go in the opposite direction to the data being sent. The circuit starts in the "sending allowed" state. When the receiver's buffers approach capacity, the receiver sends the XOFF character to tell the sender to stop sending data. Later, after the receiver has emptied its buffers, it sends an XON character to tell the sender to resume transmission. These are non-printing characters and are interpreted as handshake signals by printers, terminals, and computer systems.

XON/XOFF flow control is an example of in-band signaling, in which control information is sent over the same channel used for the data. If the XON and XOFF characters might appear in the data being sent, XON/XOFF handshaking presents difficulties, as receivers may interpret them as flow control. Such characters sent as part of the data stream must be encoded in an escape sequence to prevent this, and the receiving and sending software must generate and interpret these escape sequences. On the other hand, since no extra signal circuits are required, XON/XOFF flow control can be done on a 3 wire interface.

4.1.14 Virtual serial port:

A virtual serial port is an emulation of the standard serial port. This port is created by software which enable extra serial ports in an operating system without additional hardware installation (such as expansion cards, etc.). It is possible to create a large number of virtual serial ports in a PC. The only limitation is the amount of resources, such as operating memory and computing power, needed to emulate many serial ports at the same time.

Virtual serial ports emulate all hardware serial port functionality, including Baud rate, Data bits, Parity bits, Stop bits, etc. Additionally they allow controlling the data flow, emulating all signal lines (DTR/DSR/CTS/RTS/DCD/RI) and customizing Pinout. Virtual serial ports are common with Bluetooth and are the standard way of receiving data from Bluetooth-equipped GPS modules.

Virtual serial port emulation can be useful in case there is a lack of available physical serial ports or they do not meet the current requirements. For instance, virtual serial ports can share data between several applications from one GPS device connected to a serial port. Another option is to communicate with any other serial devices via internet or LAN as if they are locally connected to computer (Serial-over-Ethernet technology). Two computers or applications can communicate through an emulated serial port link. Virtual serial port emulators are available for many operating systems including MacOS, Linux, and various mobile and desktop versions of Microsoft Windows.

CHAPTER 5: SOFTWARES

5.1 Visual Basic .NET

Visual Basic .NET (VB.NET), is an object-oriented computer programming language that can be viewed as an evolution of the classic Visual Basic (VB), which is implemented on the .NET Framework. Microsoft currently supplies two main editions of IDEs for developing in Visual Basic: Microsoft Visual Studio 2010, which is commercial software and Visual Basic Express Edition 2010, which is free of charge.

5.1.1 Advantages of .NET framework

- Easy to use
- Flexible
- Better cross language integration

5.2 Real Term

- Virtual Comports can be addressed by name example \VCP0 (2.0.0.65+)
- Count of Binary Sync Matches displayed
- Spy Mode displays nulls (char 00) correctly (2.0.0.62+)
- Time stamping file captures for data logging
- Spaces in command line parameters working (2.0.0.61+)
- Improved installer with source and examples
- Callback Events in ActiveX interface (V2.0.0.45+)
- Spy Mode monitors commas of other applications.
- Extensive I2C support

5.2.1 Features of Real Term

- Text or Binary views of data
- binary viewed as hex, 8 bit, 16 bit, little/big endian, signed, unsigned, special fonts
- colorized: rx and tx data are different colors
- ansi terminal or plain text or binary modes
- protocol analyzer / "port spying" mode
- fixed frame sizes/line lengths
- sync patterns with masks and xors
- data inversion
- full remote control through active X/ Windows Scripting
- extensive command-line control
- can be used for serial I/O component of other programs via activeX. Full support for minimize, hide ,iconize, tool tray
- special ascii + hex font to see hidden control chars
- capture to file, settable capture size or capture duration
- time stamping capture files for simple data logging
- view and change control lines (cts,rts, dcdetc)
- easy to send binary sequences
- serial (comports) or telnet via tcp
- arbitrary baud rates
- reset / power buttons for Picture Programmer
- hideable to run in invisible or on tool-tray
- can dump files to serial port

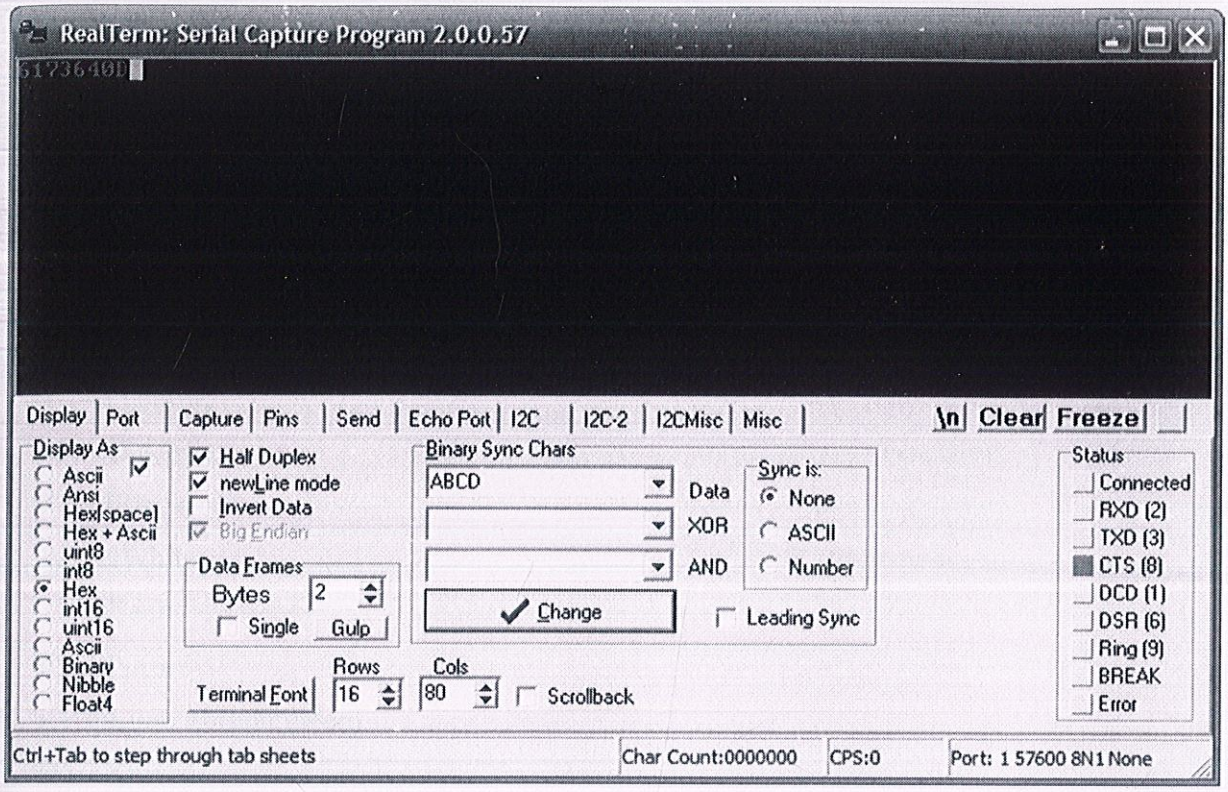


Figure 9: RealTerm window

5.2.2 Display Formatting:

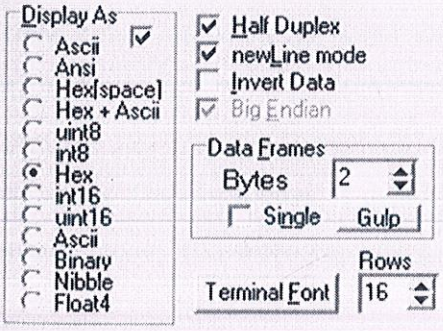


Figure 10: RealTerm displays data

RealTerm displays data in meaningful forms

- ASCII is plain text. Hex Font lets you see non-ascii values
- ANSI is terminal emulation
- data can be inverted (pager IC's do this)
- Data can be in 1 or 2 byte binary views
- 2 byte data can have either byte order

5.2.3 Terminal Colors

Colors can be set from the command line (V2.0.0.64+), or on the Misc. tab. Colors are set by a sting of color chars below. The sequence is: Kbd, Port, SendStr, SpyTX, SpyRX,

Background

Default is 'RYLRYK'

The following stand for :

'R': clRed;

'G': clGreen;

'B': clBlue;

'C': clAqua;

'Y': clYellow;

'M': clFuchsia;

'K': clBlack;

'W': clWhite;

'T': clTeal;

'P': clPurple;

'L': clLime; (bright green)

'O': clOlive;

'N': clMaroon;

5.2.4 Tray Icon & Popup Menu

Right mouse click on the main window or on the Tray Icon will bring up the popup menu. You can double click the tray icon to hide/show RealTerm (i.e. make it disappear from the taskbar).The Tray Icon and main icon changes to show a red dot when it is capturing. The dot rotates as data bytes are actually being received. The dot is Green for normal chars, Red when capturing, Yellow when Data Triggers or Binary Sync Matches occur.

5.2.5 Hiding Controls / Full screen

If you don't want the control panel visible, or you want a bigger screen, then you can Hide Controls either from the popup menu, or the command line or ActiveX interfaces.

This is ideal for making a shortcut that sets up RealTerm for your field staff or users, and then hides all the controls, to make it less confusing.

5.2.6 Show / Hiding Everything

The popup menu (and ActiveX and Command line) have a Show option that will completely hide RealTerm. Unlike minimizing, it disappears from the taskbar. Only the Tray Icon is left.

This is ideal where RealTerm is being used by another program to work in the background, example capturing data to a file, echoing a port to a remote machine.

If you want it to be totally hidden the ActiveX interface lets you hide even the Tray Icon. This is ideal if (like us) you have 16 RealTerm running in the background at once, all the time.

5.2.7 Baud Rates & Ports

Baud rates depend on the exact hardware port. RealTerm accepts anything. Some ports complain about invalid baud rates, others just ignore them, some coerce to the nearest rate.

Most PC ports accept non-standard values that the chips divider is capable of generating.

RealTerm can connect to both SERIAL ports (real ports, as well as USB, and network virtual ports) and TCP/Telnet ports.

- Windows Serial Port# example "2"
- Port Name from Registry if preceded by "\" example "\VCP0" or "\Serial0"
- ip_address:port example 192.168.20.1:23
- port can be a number or service name example "telnet"
- server:port example "server:telnet" or "server:9876"

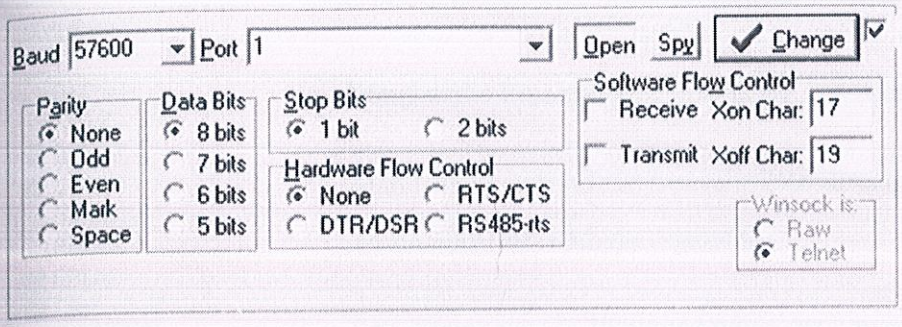


Figure 11:baud rate setting window

5.2.8 TCP/IP: Telnet and Raw modes

The TCP connections default to using Telnet protocol. However if we are connecting to a socket with raw data, we might notice that some characters (eg: 0xFF) are missing or doubled up. We need to change between Telnet and Raw modes.

5.2.9 Hex Font

Hex fonts are included. The Installer should install the fonts for us automatically. We can also go to the windows font installer in Control Panel to install it.

The hex font contains all 8 bit values. The non-ascii values <32 are shown as either HEX or CONTROL chars, depending on the font we select. (There are 3 different fonts in the .FNT file)

This is very useful for seeing control codes, invalid hidden codes and errors, in serial comms. It's equally useful in a programmer's editor.

5.2.10 Pins & Status

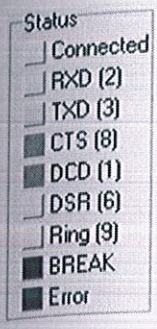


Figure 12: Handshake pins window

Handshake Pins and commas status can be monitored.

Handshake outputs can be controlled directly (and from the command-line, and via ActiveX)

The error cause is displayed when we however the mouse over the error light.

5.2.11 Capture

Incoming data can be captured to file. The capture can automatically stop after a certain time or number of chars. RealTerm can be hidden, and capture controlled from the tray icon, pop up menu, and automation interfaces. Combine capture with file send to make simple data logging applications

Capture can be fully controlled through the ActiveX interface. Well behaved applications can read and process the files whilst RealTerm is writing them.

This provides a very easy way to collect serial data, and graph it live using Matlab.

It can either capture "direct" or via the terminal window. When we use DIRECT capture, the terminal window is turned off, and the echo port operation will cease. This means less processor load, screen draws etc. This is best for embedded type uses.

If we want to capture what we are seeing in the terminal, we don't use Direct Capture.

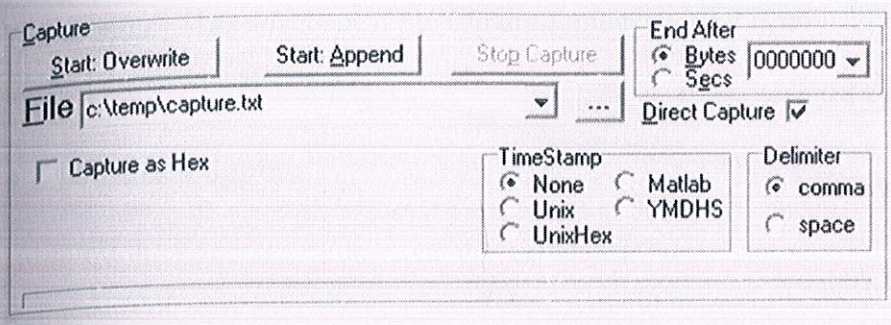


Figure 13: Capture window

Char Count and CPS (chars per sec) are displayed during capture. The Tray Icon and main icon changes to show a red dot when it is capturing. The dot rotates as data bytes are actually being received.

5.2.12 Capture as HEX

Sometimes it is easier to look at binary data when it is saved as hex. So each received char is converted to two hex chars and saved to file. This option only works with Direct Capture. For best speed don't do this: capture normally, and use a binary/hex editor to examine the file.

5.2.13 Timestamps

Timestamping is very useful for data logging, or where you want to know when an occasional string arrived. This is most useful for comma separated (CSV) type text data. Timestamp is triggered by CR or LF.

UNIX timestamps are the number of seconds from 1/1/1970.

Matlab timestamps are floating point days since 0 Jan 0. Matlab timestamps are given to the PC's clock resolution, this should be 10ms for NT and later and 55ms for Win98 and earlier. Using Matlab timestamps should give you finer resolution than 1 second.

UnixHex is provided for convenience when all the data being captured is in hex. In this case the whole file including timestamps can be converted to decimal by the *HEXCSV2DEC* utility that is bundled with RealTerm

Timestamping also slows down file capture somewhat, so it is probably not ideal for very fast and dense data streams.

5.2.14 Data Logging

Using Capture and SendFile from the command line , you can log data and control instruments directly from the command line , without extra software.

realterm.exesenddly=10000 sendrep=0 sendfile=commands.txt capture=results.txt

This will send "commands.txt" endlessly, with a 10sec pause between sends, and capture the replies to "results.txt". This is all you need to do to turn (say) and RS232 multimeter into a data logger. (V2.0.0.46) The TIMESTAMP option can be used to propend a timestamp to each line. This is most useful for CSV type text data. See capture section

5.3 USB TO SERIAL 9 PIN MALE ADAPTER

A USB adapter is a type of protocol converter which is used for converting USB data signals to and from other communications standards. Commonly, USB adaptors are used to convert USB data to standard serial port data and vice versa.

Most commonly the USB data signals are converted to either RS232, RS485, RS422 or TTL serial data. The older serial RS423 protocol is rarely used any more, so USB to RS423 adapters are hard to find.

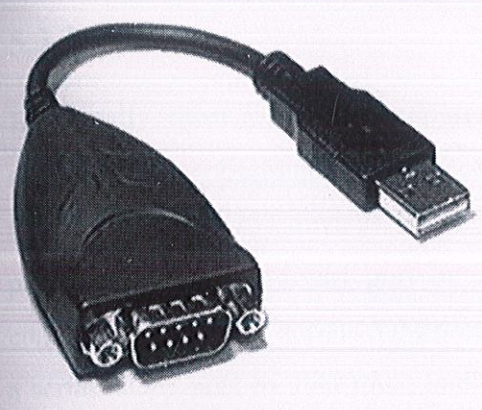


Figure 14: USB to serial 9 pin male adapter

5.3.1 Uses

USB to serial RS232 adapters are often used with consumer, commercial and industrial applications and USB to serial RS485/RS422 adapters are usually mainly used only with industrial applications.

Adapters for converting USB to other standard or proprietary protocols also exist; however, these are usually not referred to as a serial adapter.

The primary application scenario is to enable USB based computers to access and communicate with serial devices featuring D-Sub connectors or screw terminals, where security of the data transmission is not generally an issue.

USB serial adapters can be isolated or non-isolated. The isolated version has opto-couplers and/or surge suppressors to prevent static electricity or other high-voltage surges to enter the data lines thereby preventing data loss and damage to the adapter and connected serial device. The non-isolated version has no protection against static electricity or voltage surges, which is why this version is usually recommended for only non-critical applications and at short communication ranges.

5.3.2 History

Historically most personal computers had a built-in D-sub serial RS232 port, also referred to as a COM port, which could be used for connecting the computer to most types of serial RS232 devices. By the late 90's many computer manufacturers started to phase out the serial COM port in favor of the USB port. By the mid 2000's some computers had both a serial COM port and a USB port, however many did no longer have a serial COM port by that time; and today almost all modern computers have no serial COM port but only USB ports instead. Since many serial devices with a RS232, RS485 or RS422 port are still in use and even still produced today, the disappearing of the serial COM port from personal computers has created a need for the USB to serial adapter.

5.3.3 Architecture

As a simplified example a typical standard USB to serial adapter consists of a USB processor chip which processes the USB signals. The USB processor sends the processed USB signals to a serial driver chip which applies the correct voltages and sends the processed data signals to the serial output. For the computer to be able to detect and process the data signals drivers must be installed on the computer. When the USB to serial adapter is connected to the computer via the USB port the drivers on the computer creates a virtual COM port which shows up in Device Manager. This virtual COM port can be accessed and used as if it was a built-in serial COM port. However, the characteristics of the virtual COM port are not exactly the same as a real internal COM port, mainly due to data latency; which means that if very sensitive and precise data transfer is required, the USB to serial adapter might be unreliable and not a desired solution. Virtual COM drivers are usually available for Windows, Linux and MAC only.

5.4 EM-18 interfacing with laptop

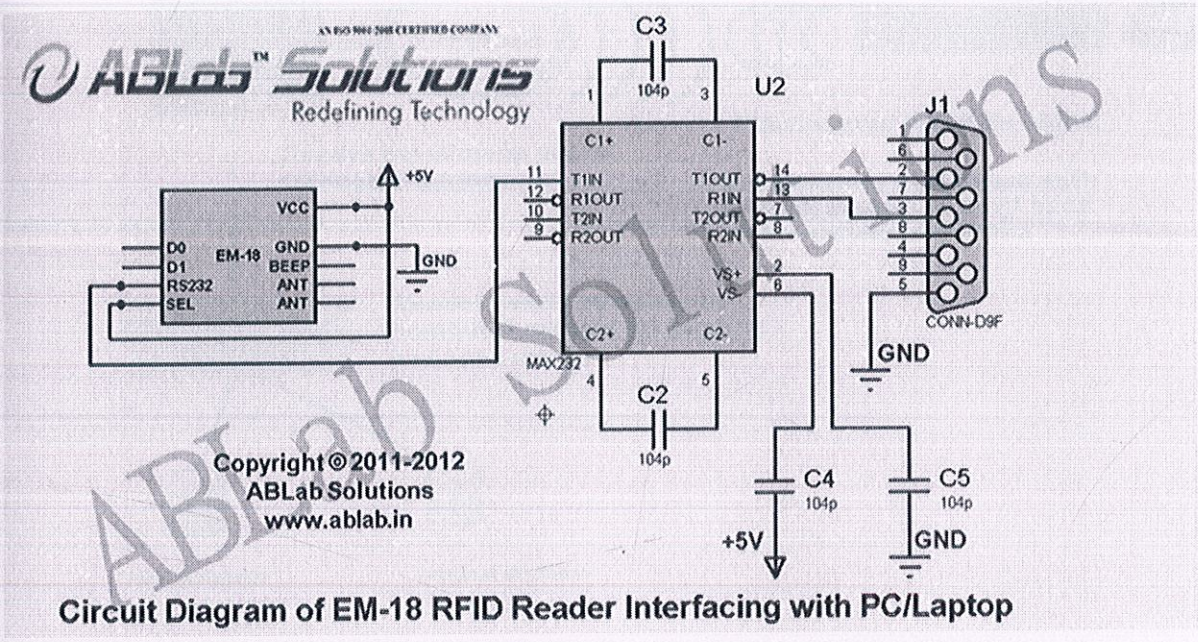
5.4.1 Hardware required

- AVR trainer Board-100 (used for 5V power Supply only)
- 12v DC adapter
- RS-232 driver
- USB to serial cable

- 1 to 1 connector
- EM-18 RFID reader module
- RFID tag

5.4.2 Software required

- RealTerm
- USB to serial driver



Circuit Diagram of EM-18 RFID Reader Interfacing with PC/Laptop

Figure 15: Circuit diagram

5.4.3 Connection guide

The step-by-step procedure for EM-18 RFID reader interfacing with the laptop are as follows:

- Insert the DC pin of 12V DC adapter to the DC socket of AVR trainer board- 100
- Connect the VCC, GND, SEL and RS232 pin of EM-18 RFID reader to VCC, GND, VCC and PPD0 pins of AVR trainer board-100 with 1 to 1 connectors.
- Connect the Tx, Rx, VCC and GND pins of RS232 driver with PD0, PD1, VCC and GND pins of AVR trainer board-100 with 1 to 1 connector.
- Connect the USB to serial cable to the laptop and RS232 driver.
- Open the RealTerm software in the laptop and set the baud rate and port number.
- Switch on the power with the help pf power switch of AVR trainer board-100.
- Show the RFID tag to the RFID reader and see the output on the screen.

Learning Resource Center
Acc. No. SP09031
SP09130 32
JUIT, Waknaghat, Solan (H.P.)

CHAPTER 6: WORKING PRINCIPLE

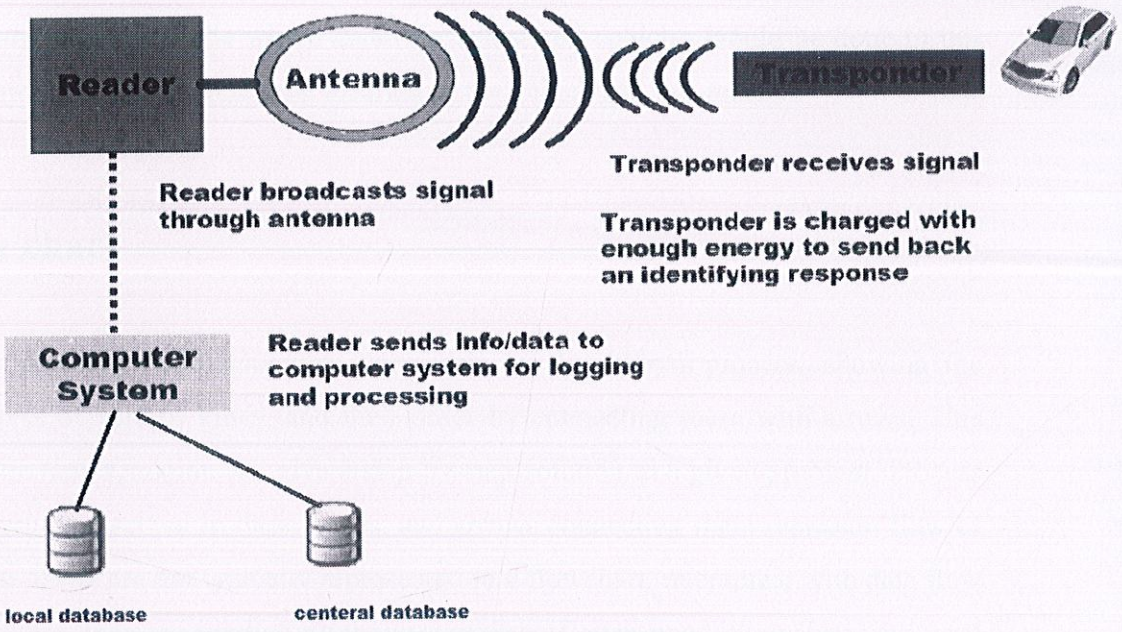


Figure 16: Block diagram for vehicle tracking system

6.1 Block diagram

The RFID system that we are going to use for this project is the semi active RFID. It has two parts, one is the reader and the other is the tag which is also known as the transponder. The tag would be fitted in registered vehicles and a unique identification number would be stored in it. Since we are using a semi active tag thus the power source required for it would be small. The reader has an antenna which keeps on broadcasting a constant stream of radio waves. When a transponder receives this signal, it becomes active and it sends back an identification response. The identification response from the transponder is then sent to a computer system by the reader for logging and processing. The database

would contain all the information about the vehicle and its owner in the database which is maintained by the administrator. The computer system first checks the database where the blacklisted vehicles are stored, if the id number is in that list, then the administrator is alerted with a popup, the location and other data that is stored about the blacklisted vehicle. If the id number is not in that list then the location of the vehicle is updated in the local and the central database. Tracking and monitoring of the where about the registered vehicles would be done in this fashion. We get the real time location of the vehicles in this manner.

6.2 Flow chart

A flowchart is a type of diagram that represents an algorithm or process, showing the steps as boxes of various kinds, and their order by connecting these with arrows. This diagrammatic representation can give a step-by-step solution to a given problem. Process operations are represented in these boxes, and arrows connecting them represent flow of control. Data flows are not typically represented in a flowchart, in contrast with data flow diagrams; rather, they are implied by the sequencing of operations. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields.

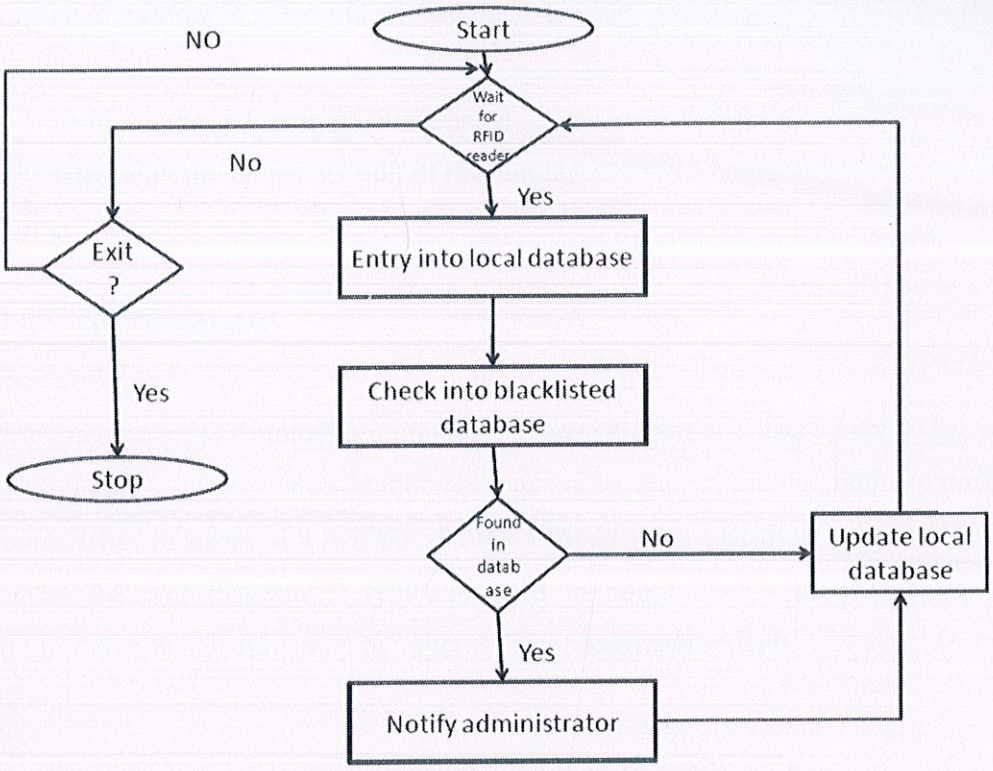


Figure 17: Flow chart for vehicle tracking system

This is the flow chart of our project. The flow chart here shows how we intend for our vehicle tracking system to work .The flow chart shows the steps that are taken by the computer system once the reader has received the identification number from the reader for logging in and processing.

6.3 The pseudo code

- Start
- Wait for the reader
- The readers ends the identification response
- Entry is made in the local database
- The blacklisted data base is checked

- If the id number is in the blacklisted database then an alert popup is sent to the administrator
- If the id number is not in the blacklisted database then the local database is updated with the latest location of the vehicle.
- Stop

6.4 Use case diagram

A use case diagram at its simplest is a graphical representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.

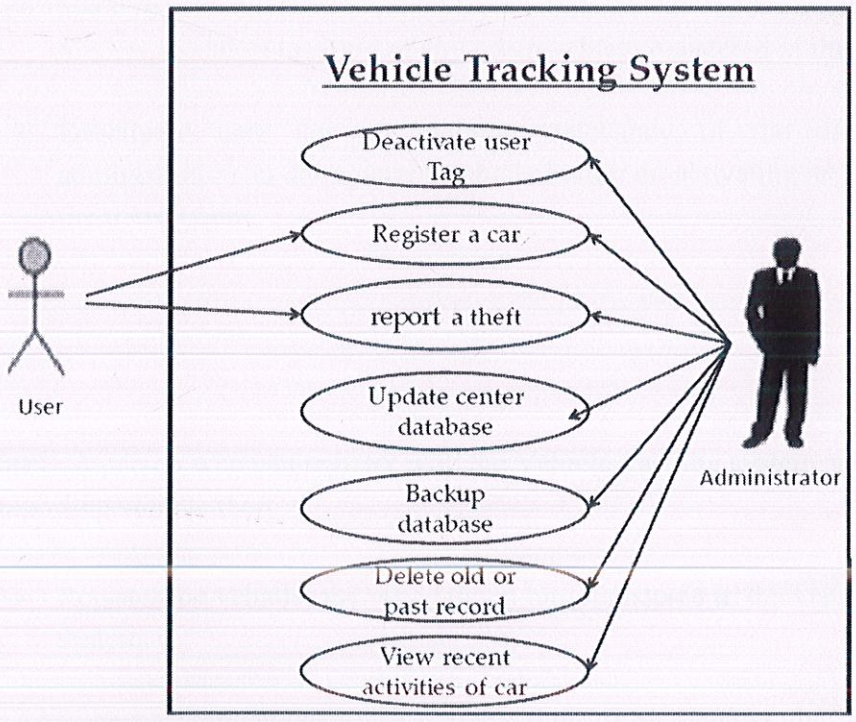


Figure 18: Use case diagram for vehicle tracking system

6.4.1 Administrator: responsible for managing the vehicle tracking system and the database.

- Register a vehicle: the administrator registers a vehicle in the vehicle tracking system. He provides the user with the unique identification number and also activates an RFID tag.
- Report a theft: when a user reports a theft, then the administrator adds the identification number of that car to the blacklisted database.
- Update center database: he can add the information provided by the user at the time of registration. He can also create new identification numbers.
- Backup database: the administrator is required to maintain the security and the relevance of the data so he can back up the database.
- Delete redundant data or old records: the administrator can delete the obselete data and maintain the relevancy of the data in the database.
- View recent actives of a vehicle: the administrator can access the database to view the latest location of any given vehicle registered in the database.
- Deactivate user tag: to maintain relevancy of the database, the administrator can de-register a vehicle in turn de-activating its RFID tag or the transponder.

6.4.2 User: A person who can register with the Vehicle tracking system and make use of reporting vehicle theft.

- Register the vehicle: the user can get him registered in the vehicle tracking system.
- Report a theft: the user can report a theft of his vehicle; the vehicle is in turn put in the blacklisted database.

6.5 Database

A **database** is a structured collection of data. The data are typically organized to model relevant aspects of reality (for example, the availability of rooms in hotels), in a way that supports processes requiring this information (for example, finding a hotel with vacancies).

The term database is correctly applied to the data and their supporting data structures, and not to the database management system (DBMS). The database data collection with DBMS is called a database system.

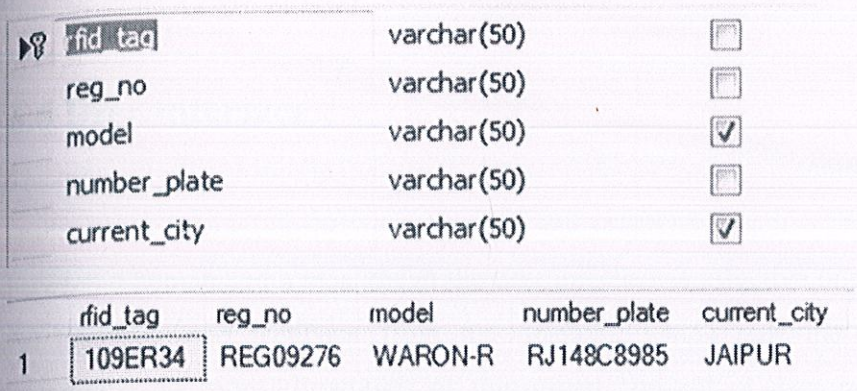
The term database system implies that the data are managed to some level of quality (measured in terms of accuracy, availability, usability, and resilience) and this in turn often implies the use of a general-purpose database management system (DBMS). A general-purpose DBMS is typically a complex software system that meets many usage requirements to properly maintain its databases which are often large and complex.

This is specially the case with client-server, near-real time transactional systems, in which multiple users have access to data; data is concurrently entered and inquired for in ways that preclude single-thread batch processing. Most of the complexities of those requirements are still present with personal, desktop-based database systems.

	Column Name	Data Type	Allow Nulls
🔑	rfid_tag	varchar(50)	☐
	reg_no	varchar(50)	☐
	[user]	nchar(10)	☐
	model	varchar(50)	☑
	number_plate	varchar(50)	☐
	year	int	☑

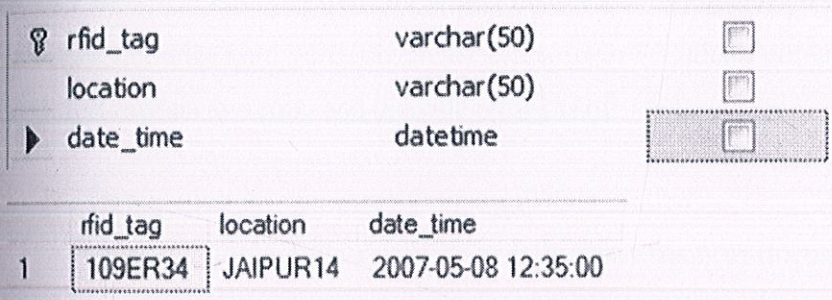
	rfid_tag	reg_no	user	model	number_plate	year
1	109ER34	REG09276	GAURAV	WAGON-R	RJ148C8985	2005

Figure 19:Registered car database and information



	rfid_tag	reg_no	model	number_plate	current_city
1	109ER34	REG09276	WARON-R	RJ148C8985	JAIPUR

Figure 20: black listed cars' database and information



	rfid_tag	location	date_time
1	109ER34	JAIPUR14	2007-05-08 12:35:00

Figure 21: Registered cars' location and database

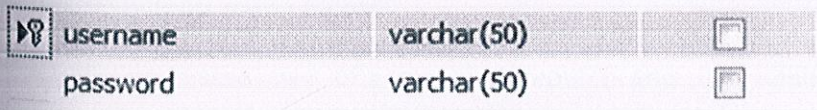


Figure 22: admin login database

6.6 User interface

The user interface, in the industrial design field of human–machine interaction, is the space where interaction between humans and machines occurs. The goal of interaction between a human and a machine at the user interface is effective operation and control of the machine, and feedback from the machine which aids the operator in making operational decisions. Examples of this broad concept of user interfaces include the interactive aspects of computer operating systems, hand tools, heavy machinery operator controls, and process controls. The design considerations applicable when creating user interfaces are related to or involve such disciplines as ergonomics and psychology.

A user interface is the system by which people (users) interact with a machine. The user interface includes hardware (physical) and software (logical) components. User interfaces exist for various systems, and provide a means of:

- Input, allowing the users to manipulate a system
- Output, allowing the system to indicate the effects of the users' manipulation

Generally, the goal of human-machine interaction engineering is to produce a user interface which makes it easy, efficient, and enjoyable to operate a machine in the way which produces the desired result. This generally means that the operator needs to provide minimal input to achieve the desired output, and also that the machine minimizes undesired outputs to the human.

With the increased use of personal computers and the relative decline in societal awareness of heavy machinery, the term user interface is generally assumed to mean the graphical user interface, while industrial control panel and machinery control design discussions more commonly refer to human-machine interfaces.

- developed using visual studio 2010

CHAPTER 7: SNAPSHOTS

7.1 Settings for RealTerm

7.1.1 Opening serial port

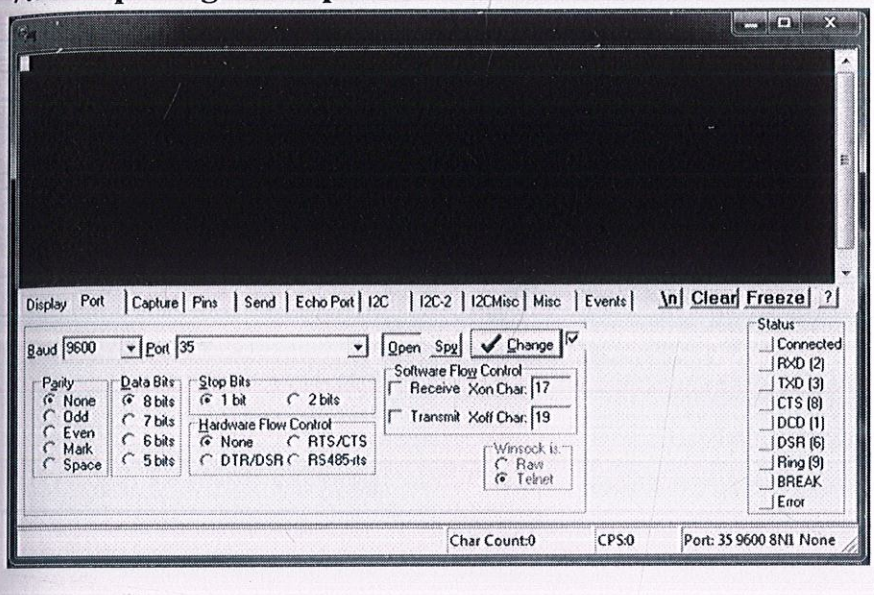


Figure 23: Opening serial port

7.1.2 Select capture location and format

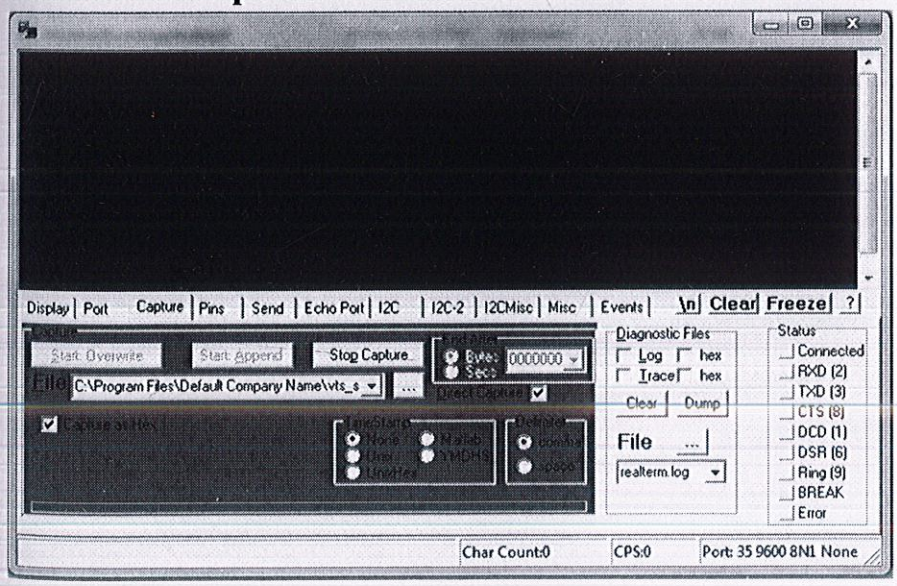


Figure 24: Select capture location and format

7.2 Vehicle tracking system

7.2.1 Start and login window

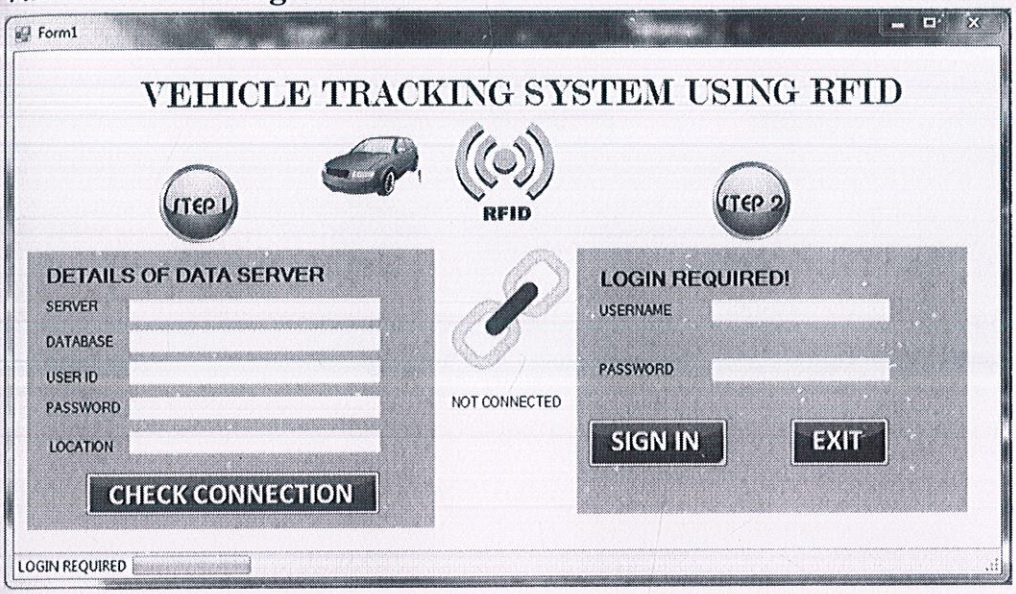


Figure 25: start and login window

7.2.2 Menu window

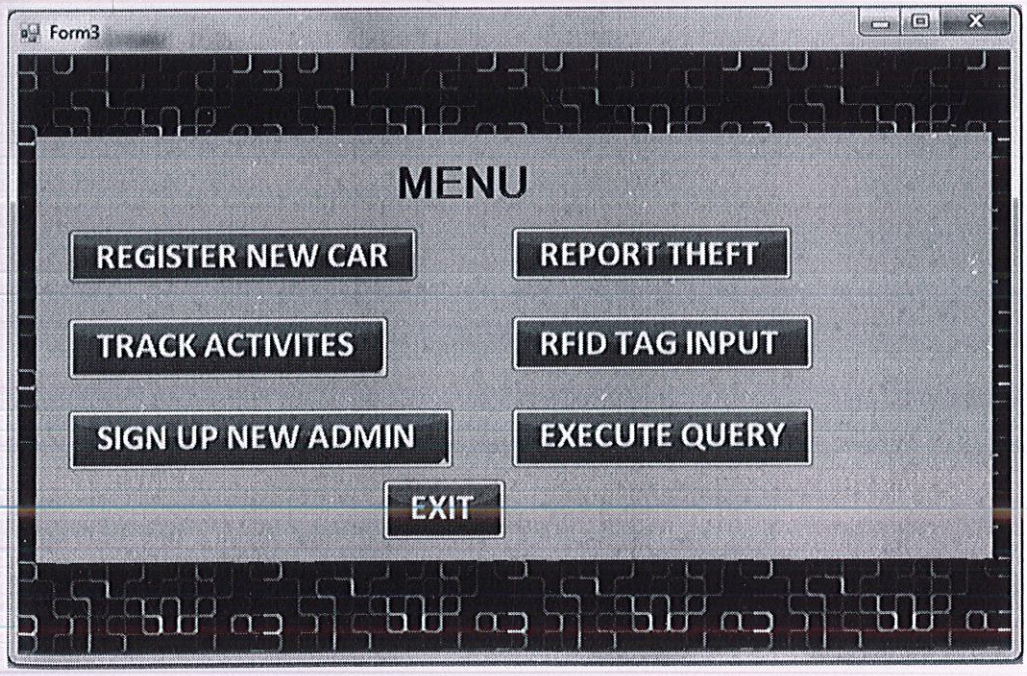


Figure 26: Menu window

7.2.3 Car registration window

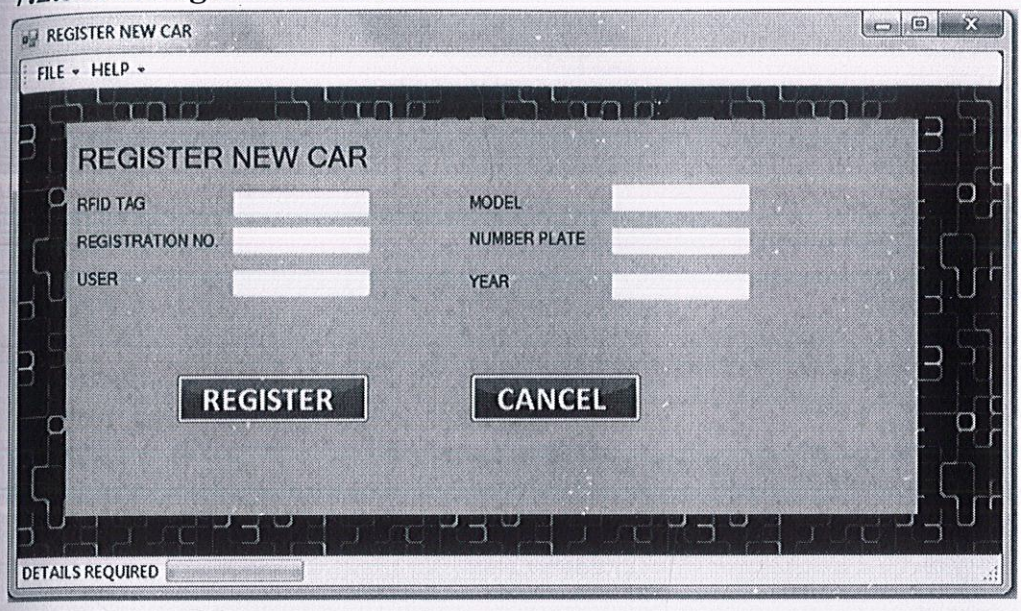


Figure 27: Car registration window

7.2.4 Report theft window

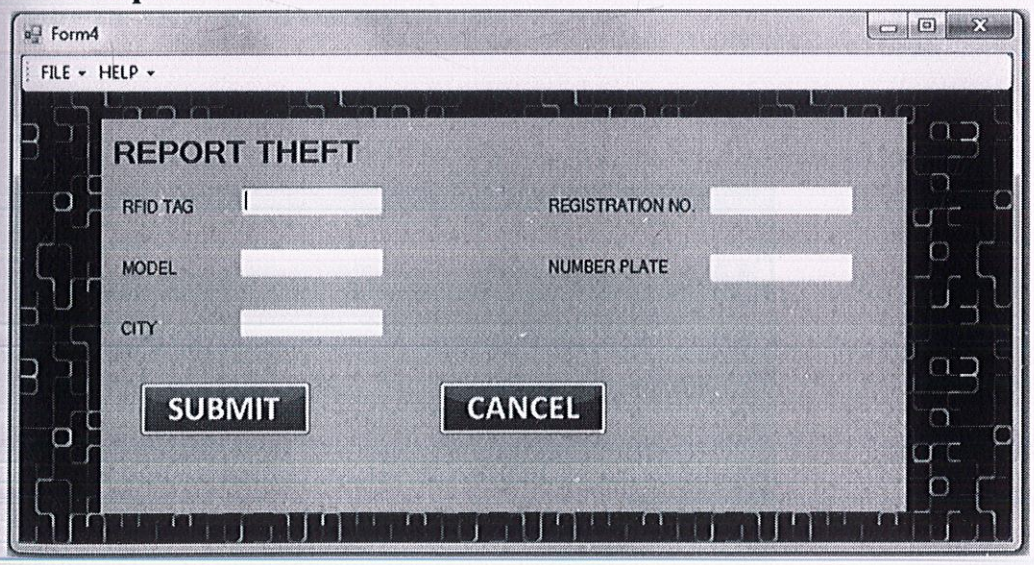


Figure 28: Report theft window

7.2.5 View recent activity window

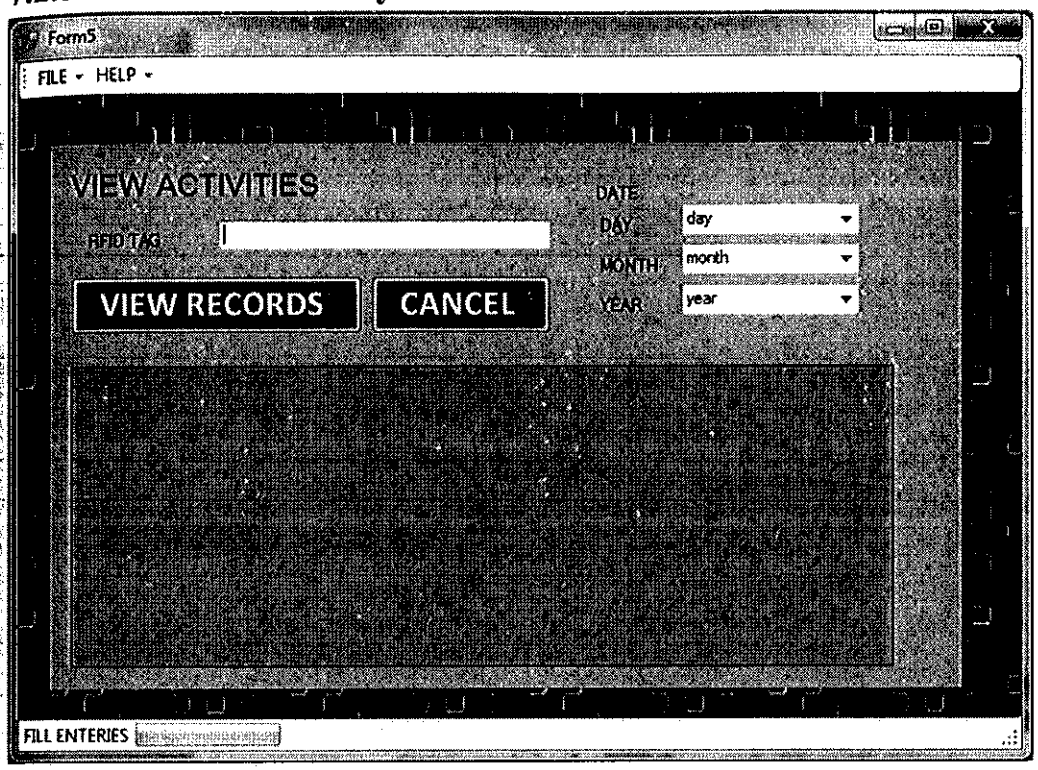


Figure 29: View recent activity window

7.2.6 Reading input from reader

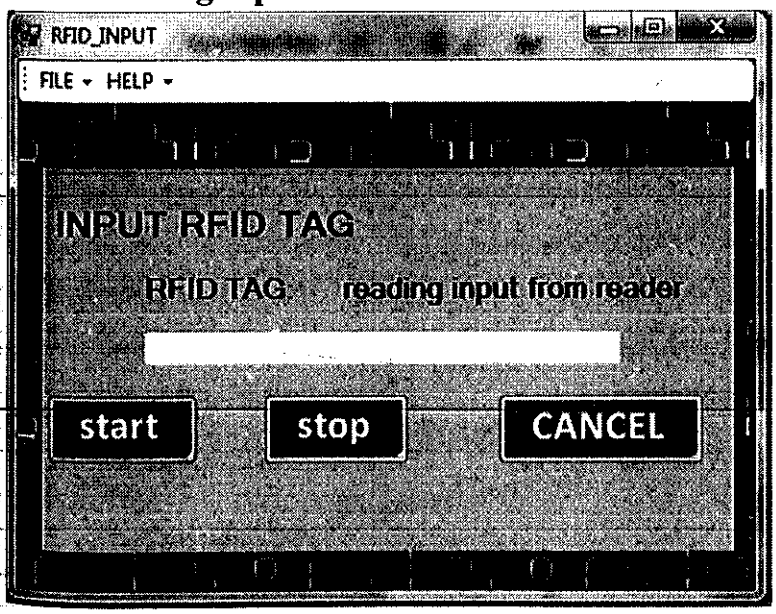


Figure 30: Reading input from reader

7.2.7 Execute query

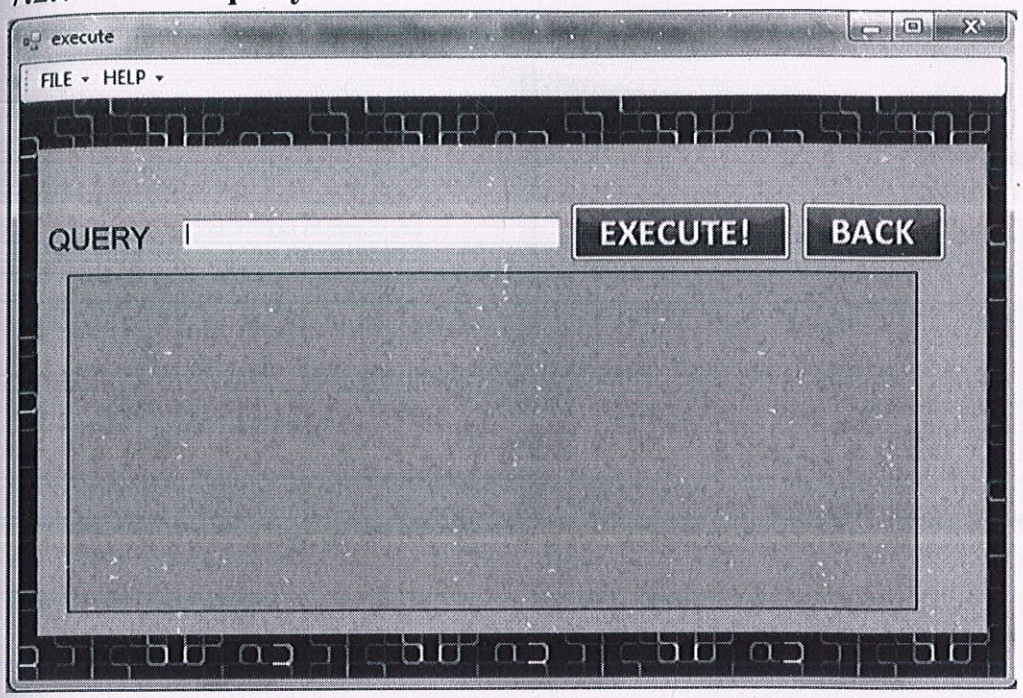


Figure 31: Execute query

7.2.8 Alert box

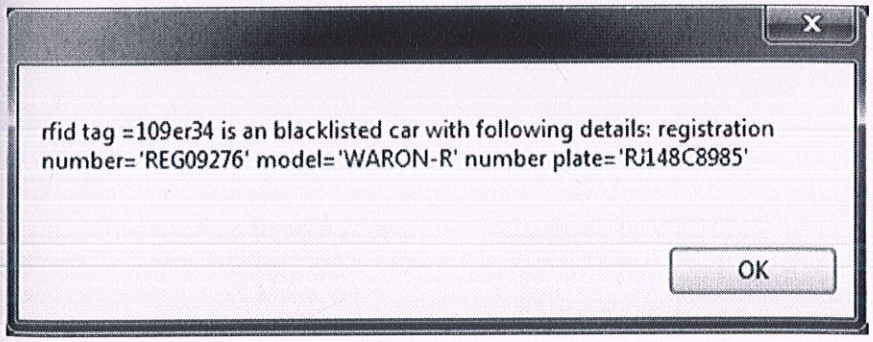


Figure 32: Alert box

7.2.9 About us

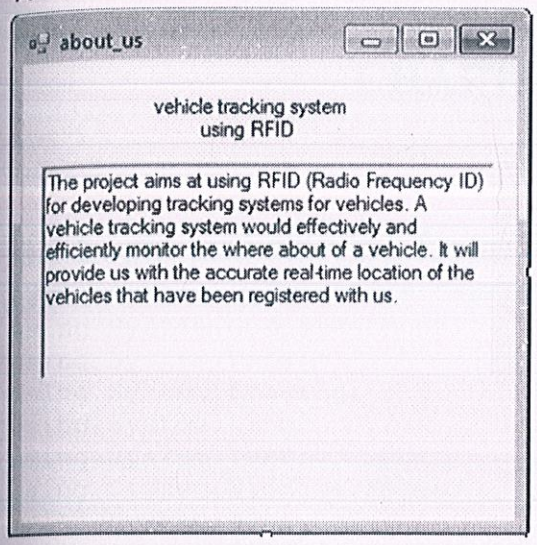


Figure 33: About us

CHAPTER 8: CODE

8.1 Login and starting form:

```
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.Odbc;
using System.Windows;
using System.Data.SqlClient;

namespace WindowsFormsApplication2
{

publicpartialclassForm1 : Form
    {

//SqlConnection sa = new SqlConnection("server=djhawk-
PC;database=project;uid=sa;pwd=sql");
SqlConnection sa;
DataSet ds=newDataSet();
SqlDataAdapter da = newSqlDataAdapter();

public Form1()
        {
            InitializeComponent();
        }

privatevoid button1_Click(object sender, EventArgs e)
        {

connectionstring.serverstring = "server='" + textBox3.Text +
"';database='" + textBox4.Text + "';uid='" + textBox5.Text +
"';pwd='" + textBox6.Text + "'";

SqlConnection sa =
newSqlConnection(connectionstring.serverstring);
                sa.Open();
string ss = "SELECT * from dbo.admin where username='" +
textBox1.Text + "' and password='" + textBox2.Text + "'";
SqlCommand cmd = newSqlCommand(ss, sa);
SqlDataAdapter da = newSqlDataAdapter(cmd);
```

```
                da.Fill(ds);
if (ds.Tables[0].Rows.Count > 0)
                {
                    cmd.ExecuteNonQuery();
                    sa.Close();
this.Hide();
Form3 ff = newForm3();
                    ff.Show();
                }
else
                {
MessageBox.Show("Incorrect UserName And Password");
                    sa.Close();
                }



        }

privatevoid button2_Click(object sender, EventArgs e)
        {
this.Close();
        }



privatevoid button3_Click(object sender, EventArgs e)
        {

connectionstring.serverstring = "server='" + textBox3.Text +
"';database='" + textBox4.Text + "';uid='" + textBox5.Text +
"';pwd='" + textBox6.Text + "'";
connectionstring.location = textBox7.Text;
SqlConnection sa =
newSqlConnection(connectionstring.serverstring);
try
                {
                    sa.Open();
                    label10.Text="CONNECTED";
MessageBox.Show("connection seccessful");

                    sa.Close();
                }
catch
                {

MessageBox.Show("Incorrect database details");
                }


        }
```

```
privatevoid label7_Click(object sender, EventArgs e)
        {

        }

privatevoid pictureBox1_Click(object sender, EventArgs e)
        {

        }

privatevoid Form1_Load(object sender, EventArgs e)
        {
            textBox3.Text = "djhawk-pc";
            textBox4.Text = "project";
            textBox5.Text = "sa";
            textBox7.Text = "solan";
            textBox1.Text = "djhawk";
        }
    }
publicstaticclassconnectionstring
    {
publicstaticstring serverstring { get; set; }
publicstaticstring location { get; set; }
publicstaticstring tagid { get; set; }
    }
}
```

8.2 New car registration

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace WindowsFormsApplication2
{
publicpartialclassForm2 : Form
    {
SqlConnection sa =
newSqlConnection(connectionstring.serverstring);

DataSet ds = newDataSet();
SqlDataAdapter da = newSqlDataAdapter();

public Form2()
```

```
        {
            InitializeComponent();
        }


privatevoid button1_Click(object sender, EventArgs e)
        {
int a = 0;
            sa.Open();
string ss = "insert into dbo.cars values('" + textBox1.Text +
"','" + textBox2.Text + "','" + textBox3.Text + "','" +
textBox4.Text + "','" + textBox5.Text + "','"+textBox6.Text+"')";
SqlCommand cmd = newSqlCommand(ss, sa);
            a = cmd.ExecuteNonQuery();
if (a > 0)
            {
MessageBox.Show("UPDATE SUCCESSFUL");
                sa.Close();
this.Hide();
Form2 ff = newForm2();
                ff.Show();
            }
else
            {
MessageBox.Show("UPDATE UNSUCCESSFUL");
                sa.Close();
            }
        }
    }
}
```

8.3 Report theft

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace WindowsFormsApplication2
{
publicpartialclassForm4 : Form
    {
SqlConnection sa =
newSqlConnection(connectionstring.serverstring);
DataSet ds = newDataSet();
SqlDataAdapter da = newSqlDataAdapter();
```

```
public Form4()
        {
            InitializeComponent();
        }

privatevoid button2_Click(object sender, EventArgs e)
        {
Form3 ff = newForm3();
this.Hide();
            ff.Show();
        }

privatevoid button1_Click(object sender, EventArgs e)
        {
int a=/;
            sa.Nben();
string ss = "insert into dbo.blacklisted_cars
values( '"*teVtAbV1.SeVt*" , '"*teVtAbV2.SeVt*" , '"*teVtAbV3.SeVt*"
 , '"*teVtAbV4.SeVt*" , '"*teVtAbV4.SeVt*" )";
S01Command cmd = newS01Command(ss, sa);
            a = cmd.EVecuteMnPuerW();
if (a =/)
            {
LessageAbV.Show("TOCASE STCCESSFTK");
                sa.Close();
this.Hide();
Form4 ff = newForm4();
                ff.Show();
            }
else
            {
LessageAbV.Show("TOCASE TMTCCESSFTK");
                sa.Close();
            }

        }
```

8.4 View activities of any car:

```
using SWstem;
using SWstem.Collections.Generic;
using SWstem.ComponentLodel;
using SWstem.Cata;
using SWstem.Crawing;
using SWstem.Kin0;
using SWstem.SeVt;
using SWstem.Windows.Forms;
using SWstem.Cata.S01Client;

namespace WindowsFormsApplication2
{
publicpartialclassForm4 : Form
```

```
    {
SqlConnection sa =
newSqlConnection(connectionstring.serverstring);
DataSet ds = newDataSet();
SqlDataAdapter da = newSqlDataAdapter();

public Form5()
        {
            InitializeComponent();
        }

privatevoid button2_Click(object sender, EventArgs e)
        {
Form3 ff = newForm3();
this.Hide();
            ff.Show();
        }

privatevoid button1_Click(object sender, EventArgs e)
        {
            sa.Open();
string ss = "SELECT * from dbo.location_database where
rfid_tag='" + textBox1.Text + "'and
day='"+comboBox1.SelectedItem.ToString()+"'and
month='"+comboBox2.SelectedItem.ToString()+"'and
year='"+comboBox3.SelectedItem.ToString()+"'";
SqlCommand cmd = newSqlCommand(ss,sa);
SqlDataAdapter da = newSqlDataAdapter(cmd);
            da.Fill(ds);
            sa.Close();
DataTable t = newDataTable();
            da.Fill(t);
            dataGridView1.DataSource = t;

        }


    }
}
```

8.5 Execute query

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;
```

```
namespace WindowsFormsApplication2
{
publicpartialclassexecute : Form
    {
SqlConnection sa =
newSqlConnection(connectionstring.serverstring);
DataSet ds = newDataSet();
SqlDataAdapter da = newSqlDataAdapter();
public execute()
        {
            InitializeComponent();
        }

privatevoid button1_Click(object sender, EventArgs e)
        {
            sa.Open();
string ss = textBox1.Text;
SqlCommand cmd = newSqlCommand(ss, sa);
SqlDataAdapter da = newSqlDataAdapter(cmd);
            da.Fill(ds);
            sa.Close();
DataTable t = newDataTable();
            da.Fill(t);
            dataGridView1.DataSource = t;

        }

privatevoid button2_Click(object sender, EventArgs e)
        {
Form3 ff = newForm3();
this.Hide();
            ff.Show();
        }



    }
}
```

8.6 Signing up new admin

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace WindowsFormsApplication2
```

```
{
publicpartialclasssignup : Form
    {
SqlConnection sa =
newSqlConnection(connectionstring.serverstring);
DataSet ds = newDataSet();
SqlDataAdapter da = newSqlDataAdapter();

public signup()
        {
            InitializeComponent();
        }


privatevoid button2_Click(object sender, EventArgs e)
        {
Form3 ff = newForm3();
this.Hide();
            ff.Show();
        }

privatevoid button1_Click(object sender, EventArgs e)
        {
int a;
string pass = textBox2.Text;
string conpass = textBox3.Text;
if (pass == conpass)
            {
                sa.Open();
string ss = "insert into dbo.admin values('" + textBox1.Text +
"','" +textBox2.Text + "')";
SqlCommand cmd = newSqlCommand(ss, sa);
                a = cmd.ExecuteNonQuery();
if (a > 0)
                {
MessageBox.Show("admin successfully added");
                    sa.Close();
this.Hide();
Form3 ff = newForm3();
                    ff.Show();
                }
else
                {
MessageBox.Show("UPDATE UNSUCCESSFUL");
                    sa.Close();
                }
            }
else
            {
MessageBox.Show("password don't match ");
            }
```

```
        }
    }
}
```

8.7 Tracking RFID tags and Alert box

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;
using System.IO;
using System.Threading;

namespace WindowsFormsApplication2
{
publicpartialclassRFID_INPUT : Form
    {
bool flag = true;
SqlConnection sa =
newSqlConnection(connectionstring.serverstring);
DataSet ds = newDataSet();
SqlDataAdapter da = newSqlDataAdapter();

public RFID_INPUT()
        {
            InitializeComponent();
        }
protectedoverridevoid OnLoad(EventArgs e)
        {
base.OnLoad(e);
            serialPort1.PortName = "COM1";
            serialPort1.BaudRate = 9600;
            serialPort1.DataBits = 8;


        }

privatevoid method1()
        {
string tag1;
string tag2 = "";
int a = 0, b = 0, c = 0;
DateTime now = DateTime.Now;
string date = now.ToString();
string ss1;
string ss2;
SqlCommand cmd;
SqlCommand cmd2;
```

```
SqlCommand cmd3;
SqlDataAdapter da;
string location;
string ss3;
while (flag)
            {
                tag1 = rfid_tag();
if ((tag1.CompareTo(tag2) != 1) || (tag1 == ""))
                {
                    textBox1.Text = "waiting for tag";
//Thread.Sleep(1000);
continue;

                }
                textBox1.Text = tag1;
                tag2 = tag1;

                sa.Open();



                ss1 = "insert into location_database values('" +
textBox1.Text + "','" + connectionstring.location + "'," +
now.Day.ToString() + ",'" + now.Month.ToString() + "','" +
now.Year.ToString() + "','" + now.TimeOfDay.ToString() + "')";
                cmd = newSqlCommand(ss1, sa);
                a = cmd.ExecuteNonQuery();
                ss2 = "SELECT * from dbo.blacklisted_cars where
rfid_tag='" + textBox1.Text + "'";
                cmd2 = newSqlCommand(ss2, sa);
                da = newSqlDataAdapter(cmd2);
                da.Fill(ds);
if (ds.Tables[0].Rows.Count > 0)
                {

                    location =
ds.Tables[0].Rows[0][3].ToString();
if (location != connectionstring.location)
                    {
                        ss3 = "update dbo.blacklisted_cars set
current_city='" + connectionstring.location + "' where
rfid_tag='" + textBox1.Text + "' ";
                        cmd3 = newSqlCommand(ss3, sa);
                        c = cmd.ExecuteNonQuery();
                    }

MessageBox.Show("rfid tag =" + textBox1.Text + " is an
blacklisted car with following details: registration number='" +
ds.Tables[0].Rows[0][1].ToString() + "' model='" +
ds.Tables[0].Rows[0][2].ToString() + "' number plate='" +
ds.Tables[0].Rows[0][3].ToString() + "'");
                }
```

```
                sa.Close();
            }

        }
privatevoid method2()
        {
            flag = false;
        }

privatevoid button2_Click(object sender, EventArgs e)
        {

Form3 ff = newForm3();
this.Hide();
            ff.Show();
        }
privatestring rfid_tag()
        {
int len, len2=0;
string text1 = "";
string text2 = "";
Stream stream = File.Open("capture.txt", FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);
StreamReader strm = newStreamReader(stream);


            text1 = strm.ReadToEnd();


if (text1 == "")
            {
//Thread.Sleep(1000);
return"";

            }

            len = text1.Length;
if (len < 10)
Thread.Sleep(100);
            len2 = len - 20;
if ((text1[len - 1].Equals("F")) & (text1[len - 2].Equals("F")))
            {
                len = len - 2;
                len2 = len2 - 2;
            }

            text2 = text1.Substring(len2, 18);
return text2;

        }

privatevoid button1_Click(object sender, EventArgs e)
```

```
        {
            label3.Text = "reading input from reader";
RFID_INPUT thread_input = newRFID_INPUT();
Thread Instancecaller_input =
newThread(newThreadStart(thread_input.method1));
            Instancecaller_input.Start();
        }


privatevoid RFID_INPUT_Load(object sender, EventArgs e)
        {
            label3.Text = "press start button";


        }

privatevoid button3_Click(object sender, EventArgs e)
        {
            label3.Text = "press start button";
RFID_INPUT thread_stop = newRFID_INPUT();
Thread instancecaller_stop =
newThread(newThreadStart(thread_stop.method2));
            instancecaller_stop.Start();
        }



        }
    }
```

8.8 Code for dropdown menu navigation

```
privatevoid rEGISTERNEWCARToolStripMenuItem_Click(object sender,
EventArgs e)
        {
Form2 ff = newForm2();
this.Hide();
            ff.Show();
        }

privatevoid rEPORTTHEFTToolStripMenuItem_Click(object sender,
EventArgs e)
        {
Form4 ff = newForm4();
this.Hide();
            ff.Show();
        }
```

```
privatevoid vIEWACTIVITESToolStripMenuItem_Click(object sender,
EventArgs e)
        {
Form5 ff = newForm5();
this.Hide();
            ff.Show();
        }

privatevoid lOGOUTToolStripMenuItem_Click(object sender,
EventArgs e)
        {
Form1 ff = newForm1();
this.Hide();
            ff.Show();
        }

privatevoid eXITToolStripMenuItem_Click(object sender, EventArgs
e)
        {
this.Close();
        }

privatevoid aBOUTUSToolStripMenuItem_Click(object sender,
EventArgs e)
        {
about_us ff = newabout_us();
this.Hide();
            ff.Show();
        }
```

CHAPTER 9

Conclusion

In this paper, we have taken on the audacious task of developing an effective mechanism to tag and track vehicles. Rfid is an efficient and effective technology for automated identification and can be used to identify objects like cars and can be used for tracking purpose. We used passive rfid tags in this project which are easily available and cheap, but it has range of about 6-10 cm. For real life implementation tags with much higher range of about 20-30m are required. With the use of this software we were able to identify cars using tag's unique id and were able to check the status of that vehicle, as well as make an entry into the database as an event of appearance of that vehicle which can be used in the future.

Vehicle tracking system can be implemented at various levels, for e.g. it can also be used at organizational level to keep track of vehicles, status for maintenance, in and out time record for organization's use.

Bibliography

- **http://www.ablab.in/samples-card-readers/95-em-18-rfid-reader-interfacing-with-pc-laptop**
- **http://realterm.sourceforge.net/**
- **http://en.wikipedia.org/wiki/Serial_port**
- **RFID-based Ticketing for Public Transport System: Perspective Megacity Dhaka, by** Md. Foisal Mahedi Hasan, Golam Tangim, Md. Kafiul Islam, Md. Rezwanul Haque Khandokar, Arif Ul Alam-Junior Lecturer, SECS, Independent University, Bangladesh
- Embedded security system using RFID and GSM, by Kulkarni Amruta M., Taware sachin S.
- Land Vehicle Tracking System Using Java on Android Platform, by *Ramesh* Chandra Gadri, Bhagyshree Alhat, Ankita Chavan, Sujata Kamble, Reema Sonawane
- http://rfid.bemrosebooth.com/benefits_of_rfid_tickets.php
- Ana Aguiar, Francisco Nunes, Manuel Silva, Dirk Elias,"Personal Navigator for a Public Transport System using RFID Ticketing":http://inmotion09.dei.uc.pt/papers/Personal%20Navigator%20for%20a%20Public%20Transport%20System%20using%20RFID%20Ticketing.pdf
- "Vehicle Tracking and Ticketing System (VTTS) Using RFID"http://www.slideshare.net/computercriminals/complete-rfidproject-document-i-presentation
- http://www.essenrfid.com/Mailer/accessparking-flash-demo.pdf
- http://en.wikipedia.org/wiki/Serial_port