

CHAPTER 1

INTRODUCTION

An **Event Data Recorder** or **EDR** is a device installed in some automobiles to record information related to vehicle crashes or accidents. In modern diesel trucks, EDRs are triggered by electronically sensed problems in the engine (often called faults), or a sudden change in wheel speed. One or more of these conditions may occur because of an accident. Information from these devices can be collected after a crash and analysed to help determine what the vehicles were doing before, during and after the crash or event. The term generally refers to a simple, tamper-proof, read-write memory device.

1.1 ACCIDENT STATISTICS

A life lost every four minutes! Stats of Indian road accidents in graphs.

Over the years, India has seen a steep rise in road accidents. According to a report on road accidents in India released by the honourable **Ministry of Road Transport and Highways**, **2015** has seen the greatest rise in number of accidents in five years- 12,023 accidents more than the previous years. Such a massive rise has not been seen since 2010 when the number of accidents rose by 13,244. It would be clear from the graph shown below.

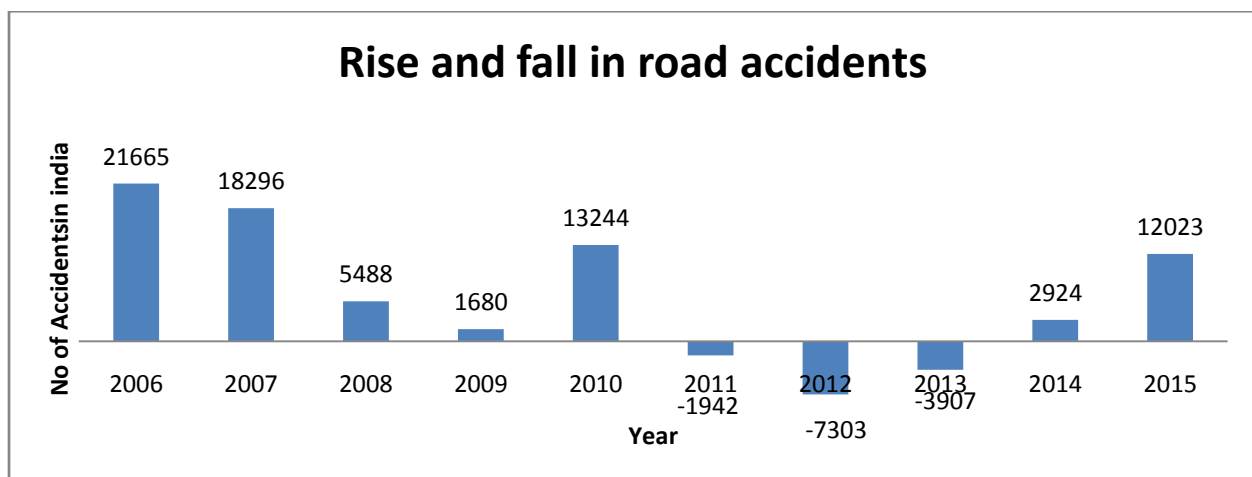


Fig: 1.1 Chart on Rise and fall in road accidents

From around 4.4 lakh annual accidents in 2005, India saw over 5 lakh accidents in 2015. In this period, fatalities rose from 94,968 to 1,46,133 for the same time period. The report says that thirteen top states namely **Tamil Nadu (69,059)**, Maharashtra (63,805), Madhya Pradesh (54,947), Karnataka (44,011), Kerala (39,014), Uttar Pradesh (32,385), Andhra Pradesh (24,258), Rajasthan (24,072), Gujarat (23,183), Telengana (21,252), Chhattisgarh (14,446), West Bengal (13,208) and Haryana (11,174) together account for 86.7 per cent of all road accidents in the country.

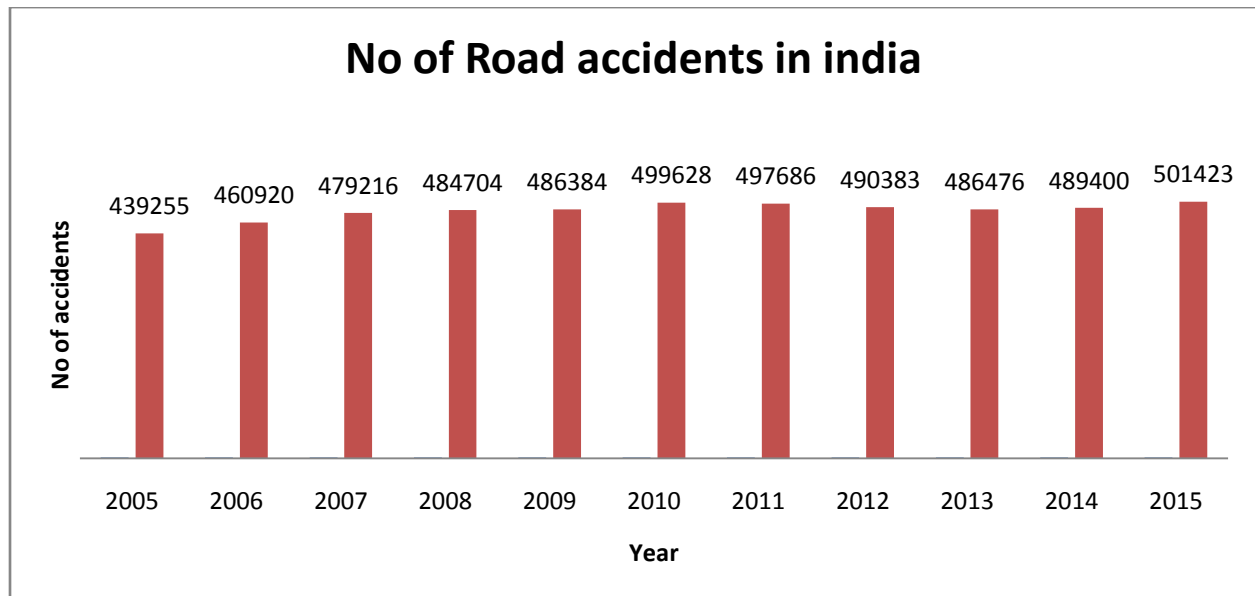


Fig: 1.2 Chart on Number of road accidents in India

The data shows that on an average day 1324 accidents occur on Indian roads leading to the death of 349 people. This means 55 accidents and 15 lives lost per hour. In other words, **a life lost every four minutes. A terrifying number!**

Finally, one serious road accident in the country occurs every minute and 16 die on Indian roads every hour.

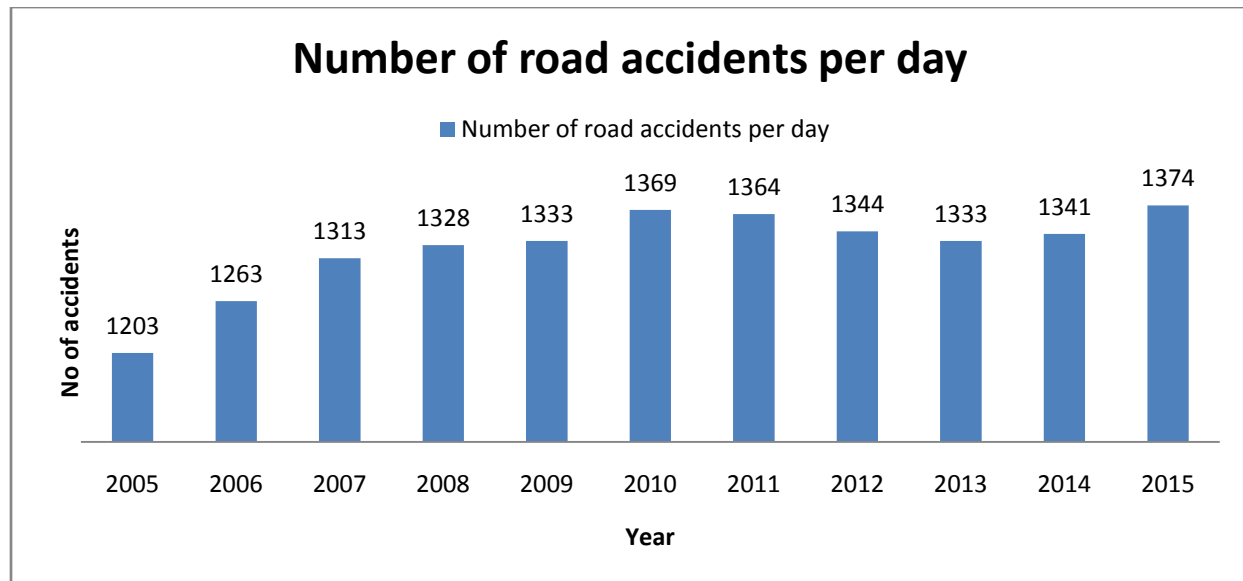


Fig: 1.3 Chart on Number of road accidents per day

1214 road crashes occur every day in India. **Four wheelers** account for **45%** of total road crash deaths. **20 children under the age of 14 die every day due to road crashes** in the country. **377 people die every day**, equivalent to a **jumbo jet crashing every day**. Two people die every hour in Uttar Pradesh – State with maximum number of road crash deaths.

Tamil Nadu is the state with the maximum number of road crash injuries

Top 10 Cities with the highest number of Road Crash Deaths (Rank –Wise):

1. Delhi
2. **Chennai**
3. Jaipur
4. Bengaluru
5. Mumbai
6. Kanpur
7. Lucknow

8. Agra
9. Hyderabad
10. Pune

So as Responsible Youths and Engineering Graduates we used our knowledge and skill to take this awful problem and came with a solution to aid investigators to study the cause of an accident, thereby minimizing its occurrence in future and developing highly reliable automobiles with utmost safety systems.

1.2 FEATURES OF EDR'S

There are many different patents related to various types of EDR features. Some EDRs continuously record data, overwriting the previous few minutes until a crash stops them, and others are activated by crash-like events (such as sudden changes in velocity) and may continue to record until the accident is over, or until the recording time is expired.

EDRs may record a wide range of data elements, potentially including whether the brakes were applied, the speed at the time of impact, the steering angle, and whether seat belt circuits were shown as "Buckled" or "Unbuckled" at the time of the crash. Current EDRs store the information internally on an EEPROM until recovered from the module. Some vehicles have communications systems (such as GM's On Star system) that may transmit some data, such as an alert that the airbags have been deployed, to a remote location.

Most EDRs in automobiles and light trucks are part of the restraint system control module, which senses impact accelerations and determines what restraints (airbags and/or seatbelt tensioners) to deploy. After the deployment (or non-deployment) decisions are made, and if there is still power available, the data are written to memory. The data downloaded from older EDRs usually contain 6 to 8 pages of information, though many newer systems include many more data elements and require more pages, depending on the make/model/year of the

vehicle being evaluated. Depending on the type of EDR, it may contain either a deployment file or a non-deployment file or sometimes both, depending on the circumstances of the collisions and the time interval between them, among other things.

It is also possible that no data can be recovered from a data recorder. One situation where this might occur is a catastrophic loss of electrical power early in a collision event. In this situation, the power reserve in the restraint system control module capacitors may be completely spent by the deployment of the air bags, leaving insufficient power to write data to the EEPROM. There are other circumstances where a module may fail to record a data file as well.

Most EDRs in heavy trucks are part of the engine electronic control module (ECM), which controls fuel injection timing and other functions in modern heavy-duty diesel engines.

1.3 TYPES OF EDR'S

The EDR functions are different for different engine manufacturers, but most recognize engine events such as sudden stops, low oil pressure, or coolant loss. Detroit Diesel, Caterpillar Inc., Mercedes-Benz, Mack Trucks and Cummins engines are among those that may contain this function. When a fault-related event occurs, the data is written to memory. When an event triggered by a reduction in wheel speed is sensed, the data that is written to memory can include almost two minutes of data about vehicle speed, brake application, clutch application, and cruise control status. The data can be downloaded later using the computer software and cables for the specific engine involved. These software tools often allow monitoring of the driver hours of service, fuel economy, idle time, average travel speeds, and other information related to the maintenance and operation of the vehicle.

Some EDRs only keep track of the car's speed along its length and not the speed going sideways. Analysts generally look at the momentum, energy, and crush damage, and then compare their speed estimates to the number coming out of the EDR to create a complete view of the accident.

1.4 AN EXAMPLE IN TRUCKS

The Eaton Vehicle On-board Radar (VORAD) Collision Warning System is used by many commercial trucking firms to aid drivers and improve safety. The system includes forward and side radar sensors to detect the presence, proximity and movements of vehicles around the truck and then alert the truck driver. When sensors determine that the truck is closing on a vehicle ahead too quickly or that a nearby vehicle is potentially hazardous, the VORAD system gives the driver both a visual and audible warning. The VORAD system also monitors various parameters of the truck including vehicle speed and turn rate plus the status of vehicle systems and controls. The monitored data is captured and recorded by the VORAD system. This monitored data can be extracted and analyzed in the event of an accident. The recorded data can be used by accident investigators and forensic engineers to show the movement and speed of the host vehicle plus the position and speeds of other vehicles prior to the incident. In accident reconstruction, the VORAD system is a step above the EDR systems in that VORAD monitors other vehicles relative to the host vehicle, while EDR's only record data about the host vehicle.

1.5 EVOLUTION OF EDR'S

Event data recorders were introduced to American open-wheel championship CART in the 1993 season, and the Formula One World Championship in 1997. This allowed studying crashes that allow developing new car rules and tracking safety measures that reduce damages.

Usage of the device in road vehicles varies widely from manufacturer to manufacturer. General Motors and Ford implement the technology on most of their recent models, while Mercedes-Benz and Audi do not use EDRs at all. As of 2003, there were at least 40 million vehicles equipped with the devices. In the UK many police and emergency service vehicles are fitted with a more accurate and detailed version that is produced by one of several independent companies. Both the Metropolitan police and the City of London police are long-term users of EDRs and have used the data recovered after an incident to convict both police officers and members of the public.



Fig: 1.4 Downloading a module via the DLC

Downloading an airbag module in most vehicles is best accomplished by connecting the appropriate scanning tool to the Diagnostic Link Connector (DLC) usually found under the vehicle's dashboard near the driver's knees. The photo given above shows a DLC download in progress. Alternately, some modules can be downloaded "on the bench" after removal from the vehicle, as shown below.



Fig: 1.5 Conducting a bench download

The only system capable of downloading commercially available crash data in North America is Bosch Diagnostic's Crash Data Retrieval System.

1.6 STANDARDIZATION OF EDR'S BY NHTSA

From 1998 to 2001, the National Highway Traffic Safety Administration (NHTSA) sponsored a working group specifically tasked with the study of EDRs. After years of evaluation, NHTSA released a formal Notice of Proposed Rulemaking in 2004. This notice declared NHTSA's intent to standardize EDRs.

It was not until August 2006 that NHTSA released its final ruling (49 CFR Part 563). The ruling was lengthy (207 pages), consisting of not only definitions and mandatory EDR standards, but also acted as a formal reply to the dozens of petitions received by NHTSA after the 2004 notice.

Since there was already an overwhelming trend for voluntary EDR installation, the ruling did not require manufacturers to install EDRs in vehicles produced for North America. Based on its analysis, NHTSA estimated that by 2010, over 85% of vehicles would already have EDRs installed in them, but warned that if the trend did not continue, the agency would revisit their decision and possibly make installation a requirement.

The mandate did, however, provide a minimum standard for the type of data that EDRs would be required to record: at least 5 types of crash data. Some of the required crash data include pre-crash speed, engine throttle, brake use, measured changes in forward velocity (Delta-V), driver safety belt use, airbag warning lamp status and airbag deployment times.

In addition to the required data, NHTSA also set standards for 30 other types of data if EDRs were voluntarily configured to record them. For example, if a manufacturer configured an EDR to record engine RPMs or ABS activity, then the

EDR would have to record 5 seconds of those pre-crash data in half-second increments.

Besides the requirement that all data be able to survive a 30 MPH barrier crash and be measured with defined precision, NHTSA also required that all manufacturers make their EDR data publicly available. As of October 2009, only General Motors, Ford and Daimler Chrysler had released their EDR data to be publicly read. In the August 2006 ruling, NHTSA set a time table for all vehicle manufacturers to be in compliance with the new EDR standards. The compliance date was originally set for all vehicles manufactured after September 1, 2010. NHTSA has since updated its ruling (49 CFR Part 563 Update) to give vehicle manufacturers until September 1, 2014 to be in compliance with the original ruling.

1.7 PRIVACY CONCERNS

Despite alerts and warnings in their vehicle owner's manual, many drivers are not aware of their vehicle's recording capability. Civil liberty and privacy groups have raised concerns about the implications of data recorders 'spying' on car users, particularly as the issue of 'who owns the data' has not yet been fully resolved, and there has been some controversy over the use of recorded data as evidence in court cases and for insurance claims against the driver of a crashed vehicle. But the use of EDR data in civil and criminal court cases is on the rise as they become more accepted as source reliable empirical evidence in accident reconstruction.

Fourteen states have statutes specific to EDRs. Generally, these state statutes restrict access to the EDR or limit the use of recovered EDR information.

1.8 AS EVIDENCE IN COURTS

There have been a number of trial cases in the US and Canada involving EDRs. Drivers have been convicted and exonerated as a result of EDR evidence.

Some examples include:

- In New South Wales, Australia, a teen-aged female (a probationary driver) was convicted of dangerous driving "causing death/occasioning grievous bodily harm" in 2005. Evidence from the Peugeot's EDR showed that the car was being driven in excess of the posted speed limit. An injunction against the use of EDR evidence, obtained by the owner of the car (the parents of the defendant), was overturned in the NSW Supreme Court.
- In Quebec, Canada, the driver of a car who sped through a red light, crashing into another car at the intersection and killing the other driver, was convicted of "dangerous driving" in 2001 after EDR information revealed that it was he, not the deceased driver of the other car (as the defendant asserted), who was speeding. There were no other witnesses to the crash.
- The first such use of EDR evidence in the United Kingdom was at Birmingham Crown Court during the trial of Antonio Boparan-Singh who crashed the Range Rover Sport he was driving into a Jeep in 2006. The accident left a baby girl paralyzed and the driver, who was aged 19 at the time of the incident, was sentenced to 21 months in prison. The EDR evidence allowed investigators to determine the driver was speeding at 72 mph in a 30 mph zone.

CHAPTER 2

LITERATURE SURVEY

2.1 AUTOMOBILE'S FIRST INFORMATION RECORD

Murugesh Gorajanal, Priyanka Mannikeri, Venkatesh Nayak, VikasDeshpande, MahalaxmiBhille (2015) proposed a project regarding First Information Record (FIR) used in Automobiles. Various FIR's used in high end automobiles like flights, cars and some two wheeler like Kawasaki's Ninja. This project works on the questions like:

1. What are the root causes for so many accidents to happen?
2. Where are the details stored after the accident?
3. Was there any fault with the motorcycle?
4. How about the fault insurance claims?

This project work addresses the above questions and aims to collect the information which aids investigations of standards. Information from this device can be collected to determine the condition of motorcycle before the time of accident. The results of analysis show that the recorders can report real world crash data and therefore be a powerful tool by providing useful information to crash reconstruction experts.

Actually, this project documents the process of using event data recorders in motorcycles. The report details the use of Auto-FIR on Discover-125cc motorcycle. Auto-FIR is designed to record data at regular time intervals. The data recorded in normal conditions is overwritten for every 5 seconds. Even if the system is restarted the values previously recorded and stored is displayed. Auto-FIR does not store any personal data (e.g., name, gender, age etc.).

Auto-FIR records motorcycle data such as:

1. Speed
2. Braking status
3. Stand status
4. Gear status at the time of crash

Typically, three entities conduct accident investigation including government agencies, insurance companies and law enforcement. Each of the entity searches for the reasons for the accident to happen and the actual committer of the accident all based on the pre, post-crash data which is given to them. Data from Auto-FIR helps in crash reconstruction process.

2.2 DRIVING BEHAVIOUR-BASED EVENT DATA RECORDER

Bing-Fei Wu, Ying-Han Chen, Chung-HsuanYeh (2014) proposed a project on general event data recorder which is used in order to record information related to vehicle crashes or accidents. This project made a study regarding a prototype of a driving behaviour-based event data recorder, which provides the information of driving behaviours and a danger level.

In this, the authors approach is to recognize the seven behaviours:

1. Normal driving
2. Acceleration
3. Deceleration
4. Changing to the left lane
5. Changing to the right lane
6. Zigzag driving
7. Approaching the car in front by the hidden Markov models.

Here, all data were collected from a real vehicle and evaluated in a real road environment. The experimental results show that the proposed method achieved an average detection ratio of 95% for behaviour recognition.

In this project,

- Driving behavior is firstly recognized using hidden Markov models (HMMs)
- The danger level is inferred by the fuzzy logic.

To examine the driving behaviour clearly and to study further information, driving behaviours are divided into three categories:

1. Longitudinal behaviors,
2. Lateral behaviors and
3. Car-following behaviors.

With these behaviour categories, it is easier to describe one driving section, and further analyse the danger level. Therefore the HMM is applied to recognize the driving behaviours.

The major task in the second stage is to combine the effect of each driving event in order to infer the danger level using an FIS. The degree of each driving event is estimated as a quantifiable indicator that represents a more explicit description. Decision strategy is also presented to improve the efficiency of the fuzzy rule selection by using the recognized results obtained from the first stage as conditions.

DBEDR recorded the recognized driving behaviours and the danger level, and the places were stored with the assistance of a global positioning system receiver. By integrating Google Maps, the locations, the driving behaviour occurrences, the danger level on the travel routes and the recorded

images, the proposed DBEDR could be more useful compared with the traditional EDRs.

2.3 MONITORING DRIVING HABITS THROUGH ANAUTOMOTIVE CAN NETWORK

J. E. Pacheco, E. López (2013) proposed a project in order to design and build an EDR (event data recorder) device that stores data about the car. To obtain such information, the device must be able to communicate with the car through the CAN protocol (Controller Area Network). The device should listen to network messages and decrypt their content.

Measurements like:

1. Engine temperature,
2. Enginerpm,
3. Speed reached,
4. Messages from the brake,
5. Accelerators are obtained.

Messages will be stored on the EDR and then transmitted to the PC. This information is used to obtain driving habits of the driver.

In this project the messages to be analysed are obtained from the electrical system of a car. The car has two low-speed CAN networks and one high speed CAN network. The EDR is based on a microcontroller with two CAN nodes, one set at low speed and the other at high speed. The system is captures messages from the car, rejects unwanted messages and then transmits to the PC desired messages.

The information extracted from the messages is:

- Throttle position
- Vehicle speed in km/h

- Engine RPMs.

In this project Test and Results of the measured parameters are carried out in two ways which are as follows:

- Simulation Test.
- On-Board Test.

Simulation Test consists on emulate message from other CAN nodes. In On-Board Test, the EDR is connected to Bora Variant and some tests are carried out. By using this way, the Test and Results of the measured parameters are carried out.

The hardware used in this project implements a CAN node with full functionality. It can be programmed to allow only desired messages. At this point, complete paths of the throttle position, engine rpm and vehicle speed are required to analyse driver habits.

CHAPTER 3

SYSTEM REQUIREMENTS

Before seeing in detail about the requirements of our system let us see in brief about some of the basic concepts of an embedded system. An embedded system is a computer system with a dedicated function within a larger mechanical or electrical system, often with real-time computing constraints. It is embedded as part of a complete device often including hardware and mechanical parts. Embedded systems control many devices in common use today. Ninety-eight percent of all microprocessors are manufactured as components of embedded systems. Modern embedded systems are often based on microcontrollers (i.e. CPU's with integrated memory or peripheral interfaces), but ordinary microprocessors (using external chips for memory and peripheral interface circuits) are also common, especially in more-complex systems.

Embedded systems are commonly found in consumer, cooking, industrial, automotive, medical, commercial and military applications.

Transportation systems from flight to automobiles increasingly use embedded systems. New airplanes contain advanced avionics such as inertial guidance systems and GPS receivers that also have considerable safety requirements. Various electric motors — brushless DC motors, induction motors and DC motors — use electric/electronic motor controllers. Automobiles, electric vehicles, and hybrid vehicles increasingly use embedded systems to maximize efficiency and reduce pollution. Other automotive safety systems include anti-lock braking system (ABS), Electronic Stability Control (ESC/ESP), traction control (TCS) and automatic four-wheel drive.

Since the embedded system is dedicated to specific tasks, design engineers can optimize it to reduce the size and cost of the product and increase the reliability and performance, which is an important requirement.

Thus here we are going to see in detail about the software and hardware requirements of our system.

3.1 SOFTWARE REQUIREMENTS

Arduino is a computer hardware and software company, project, and user community that designs and manufactures microcontroller kits for building digital devices and interactive objects that can sense and control objects in the physical world. The project's products are distributed as open-source hardware and software, which are licensed under the GNU Lesser General Public License (LGPL) or the GNU General Public License (GPL),^[1] permitting the manufacture of Arduino boards and software distribution by anyone. Arduino boards are available commercially in preassembled form, or as do-it-yourself kits.

A program for **ARDUINO 1.8.1** may be written in any programming language (Java, C and C++) for a compiler that produces binary machine code for the target processor. Atmel provides a development environment for their microcontrollers, AVR Studio and the newer Atmel Studio. The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux.

The Arduino project provides the Arduino Integrated development environment (IDE), which is a cross-platform application written in the programming language Java. It originated from the IDE for the languages Processing and Wiring. It includes a code editor with features such as text cutting and pasting, searching and replacing text, automatic indenting, brace matching, and syntax highlighting, and provides simple one-click mechanisms to compile and upload programs to an Arduino board. It also contains a message area, a text console, a toolbar with buttons for common functions and a hierarchy of operation menus.

A program written with the IDE for Arduino is called a sketch. Sketches are saved on the development computer as text files with the file extension .ino. Arduino Software (IDE) pre-1.0 saved sketches with the extension .pde.

The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub `main()` into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program `avrdude` to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.

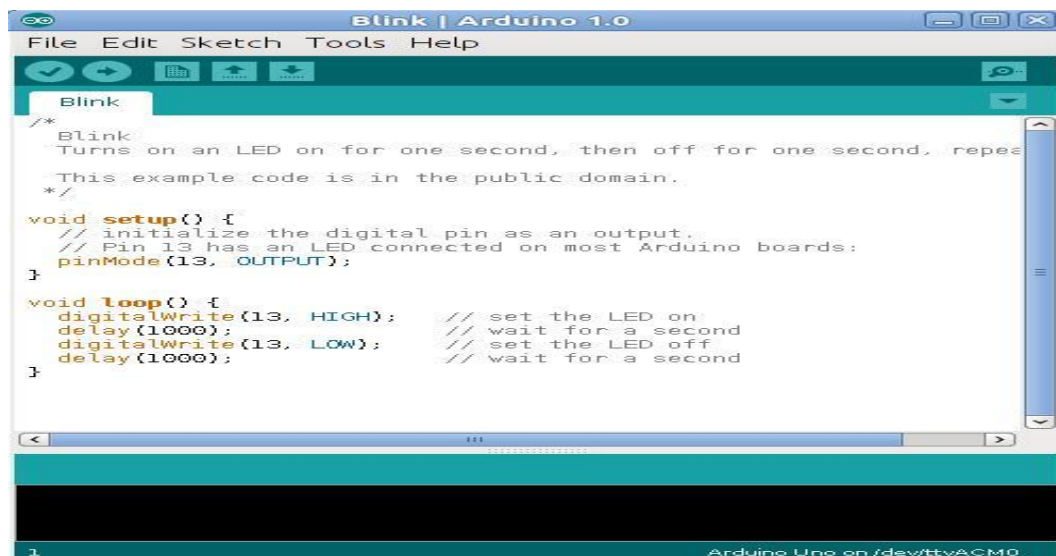


Fig 3.1 Screenshot of the Arduino IDE showing the Blink simple beginner program

A minimal Arduino C/C++ sketch, as seen by the Arduino IDE programmer, consists of only two functions:

- *setup*: This function is called once when a sketch starts after power-up or reset. It is used to initialize variables, input and output pin modes, and other libraries needed in the sketch.

- *loop*: After *setup* has been called, function *loop* is executed repeatedly in the main program. It controls the board until the board is powered off or is reset.^[43]

Most Arduino boards contain a [light-emitting diode](#) (LED) and a load resistor connected between pin 13 and ground, which is a convenient feature for many tests and program functions. A typical program for a beginning Arduino programmer blinks an LED repeatedly.

```
#define LED_PIN 13                // Pin number attached to LED.

void setup() {
  pinMode(LED_PIN, OUTPUT);      // Configure pin 13 to be a digital output.
}

void loop() {
  digitalWrite(LED_PIN, HIGH);   // Turn on the LED.
  delay(1000);                  // Wait 1 second (1000 milliseconds).
  digitalWrite(LED_PIN, LOW);    // Turn off the LED.
  delay(1000);                  // Wait 1 second.
}
```

Fig 3.2 Screenshot of the Arduino Programming for blinking an LED repeatedly.

This program uses the functions `pinMode`, `digitalWrite`, and `delay`, which are provided by the internal libraries included in the IDE environment. The program is usually loaded in the Arduino by the manufacturer.

3.2 HARDWARE COMPONENTS

Here we are going to see in depth about the various Hardware components that are required for making the entire system. At first let us see in detail about the processor board in detail.

3.2.1 ARDUINO UNO BOARD

Arduino is open-source hardware. The hardware reference designs are distributed under a Creative Commons Attribution Share-Alike 2.5 license and are

available on the Arduino website. Layout and production files for some versions of the hardware are also available. The source code for the IDE is released under the GNU General Public License, version 2. Nevertheless, an official Bill of Materials of Arduino boards has never been released by Arduino staff.

Although the hardware and software designs are freely available under copy left licenses, the developers have requested that the name Arduino be exclusive to the official product and not be used for derived works without permission. The official policy document on use of the Arduino name emphasizes that the project is open to incorporating work by others into the official product.

An Arduino board consists of an Atmel8, 16 or 32-bit AVR microcontroller (ATmega8, ATmega168, ATmega328, ATmega1280, ATmega2560), but other makers' microcontrollers have been used since 2015. The boards use single-row pins or female headers that facilitate connections for programming and incorporation into other circuits. These may connect with add-on modules termed **shields**. Multiple, and possibly stacked shields may be individually addressable via an **I²C** serial bus. Most boards include a 5 V linear regulator and a 16 MHz crystal oscillator or ceramic resonator. Some designs, such as the Lily Pad, run at 8 MHz and dispense with the on-board voltage regulator due to specific form-factor restrictions.

Arduino microcontrollers are pre-programmed with a boot loader that simplifies uploading of programs to the on-chip flash memory. The default boot loader of the Aduino UNO is the optibootloader. Boards are loaded with program code via a serial connection to another computer. Some serial Arduino boards contain a level shifter circuit to convert between RS-232 logic levels and transistor–transistor logic (TTL) level signals. Current Arduino boards are programmed via Universal Serial Bus (USB), implemented using USB-to-serial adapter chips such as the FTDI FT232. Some boards, such as later-model Uno

boards, substitute the FTDI chip with a separate AVR chip containing USB-to-serial firmware, which is reprogrammable via its own ICSP header. Other variants, such as the Arduino Mini and the unofficial Boarduino, use a detachable USB-to-serial adapter board or cable, Bluetooth or other methods, when used with traditional microcontroller tools instead of the ArduinoIDE, standard AVR in-system programming (ISP) programming is used.

The Arduino board exposes most of the microcontroller's I/O pins for use by other circuits. The Uno provide 14 digital I/O pins, six of which can produce pulse-width modulated signals, and six analog inputs, which can also be used as six digital I/O pins. These pins are on the top of the board, via female 0.1-inch (2.54 mm) headers. Several plug-in application shields are also commercially available. The Arduino Nano, and Arduino-compatible Bare Bones Board and Boarduino boards may provide male header pins on the underside of the board that can plug into solder less breadboards.

Many Arduino-compatible and Arduino-derived boards exist. Some are functionally equivalent to an Arduino and can be used interchangeably. Many enhance the basic Arduino by adding output drivers, often for use in school-level education, to simplify making buggies and small robots. Others are electrically equivalent but change the form factor, sometimes retaining compatibility with shields, sometimes not. Some variants use different processors, of varying compatibility.

The photo of the Uno processor used in this project is described below :

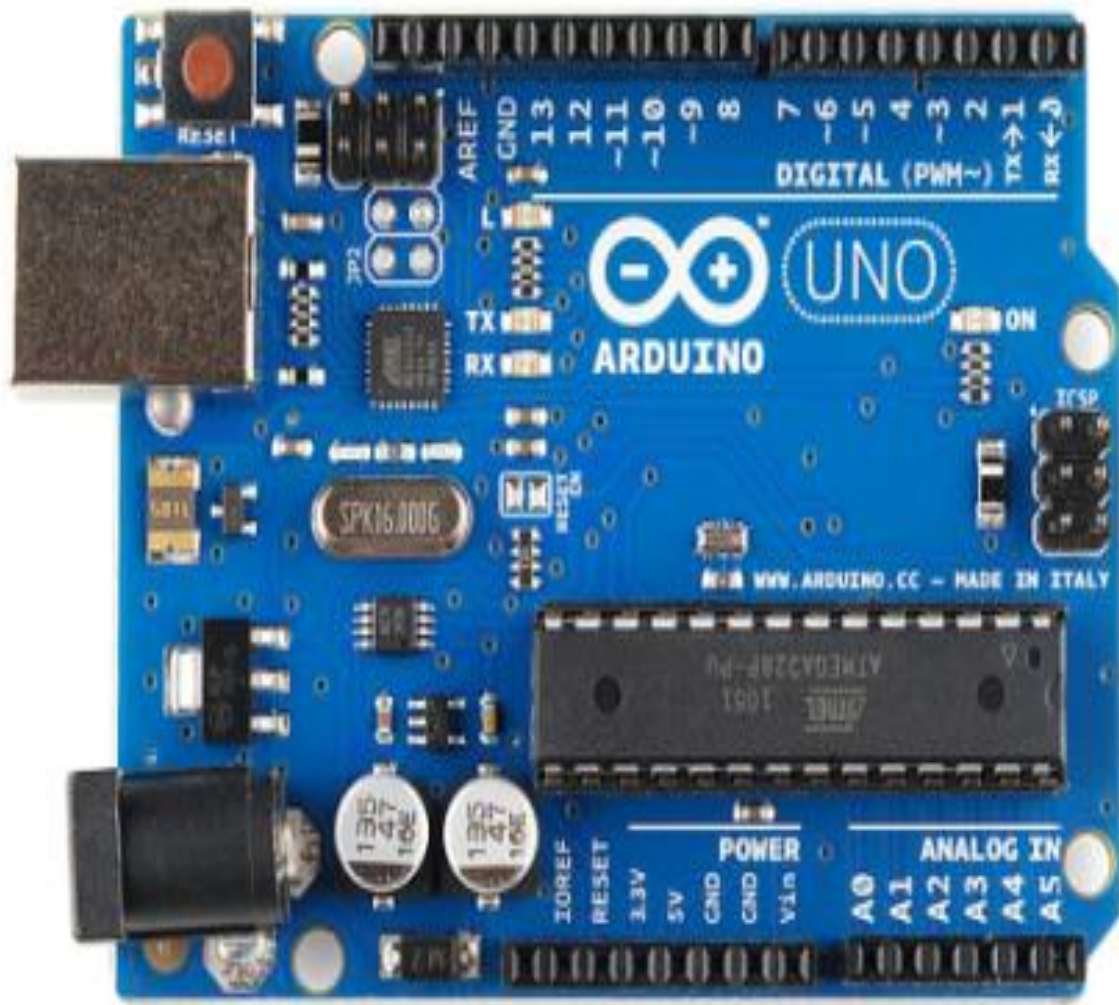


Fig 3.3 Uno board

3.2.1.1 SPECIFICATION OF ARDUINO UNO BOARD

The specification of Arduino Uno Board is clearly shown in the next page as a table.

Table 3.1 Specification of Arduino Uno Board

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by boot loader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

3.2.2 DS3231 RTCMODULE

This module is a low-cost, extremely accurate real-time clock (RTC) to provide date and time data in required formats. The device incorporates a battery input so that it also maintains accurate timekeeping after being disconnected from the main power supply. Address and data are transferred serially through an I²C bi directional bus using the SCL and SDA pins.

The photo of Real Time Clock DS3231 can be given as follows:



Fig 3.4 Real Time Clock DS3231

Using the standard methods in the DS3231.h library, the date and time can be read from the board.

3.2.3 LM393 RPM SENSOR

It is an opto coupler based voltage comparator in which one side holds a light emitting diode and a photo transistor on the other side whose line of conduction will be periodically interrupted by an encoder disc. The encoder disc is attached to the shaft of the wheel and it rotates with the speed of the shaft.



Fig 3.5 RPM Sensor

This sensor gives the number of interruptions in its signal conduction as a digital input to the arduino board, which can be read using the method `digitalRead()`. This value can be used for further calculations.

3.2.4 HC-SR04 Ultrasonic range sensor

The HC-SR04 ultrasonic sensor uses sonar to determine distance to an object. Its Trig pin emits a 40,000 Hz, 8 cycle sonic burst at the speed of sound and if there is an object or obstacle on its path, it will bounce back to the module and it will be received in the Echo pin after a certain time interval. Considering the time

and the speed of the sound you can calculate the distance. It offers non-contact range detection with high accuracy and stable readings from 2cm to 400cm.

By reading the digital outputs from the Trig and Echo pins using `digitalRead()`, the controller calculates the distance of the obstacle.

The photo of this sensor is shown in the next page as follows:



Fig 3.6 Ultrasonic Sensor

3.2.5 POTENTIOMETER

A potentiometer is a three-terminal resistor with a sliding or rotating contact that forms an adjustable voltage divider. If only two terminals are used, it acts as a variable resistor or as an Rheostat.



Fig 3.7 Potentiometer

These are the various hardware components required to make the system.

3.2.6 SD CARD

SD cards are a form of flash memory used to store data in a digital format. The solid-state chip contains many tiny electrical circuits, which are a series of NAND chips.



Fig 3.8 SD Card

When the card is not in use, the circuits retain their charges without any additional power. When a card is placed in an activated device, a small electrical current from the device moves electrons to the flash memory chip. The digital patterns stored on the chip correlate to the data stored there. Data is erased when a slightly higher voltage is applied to the circuit. This allows for rewriting. The NAND chips in an SD card do not wear out easily, so data can be written to them thousands of times, making SD cards resilient and long-lasting in use. SD cards write and erase memory in blocks or sections, which makes them faster than some other varieties of data storage. In contrast to traditional hard drives, SD cards also contain no moving parts, so they are significantly more resilient to accidental bumps. To prevent the accidental loss of data, many SD cards come equipped with a lock for the data contained on the card. This is accomplished via a small switch, which, if enabled, does not allow new data to be written or old data to be overwritten on the card.

CHAPTER 4

SYSTEM ANALYSIS

In this Chapter we are going to see in detail about the various important aspects of the Existing System and Proposed System. Let us see about them in Detail.

4.1 EXISTING SYSTEM

At present, Event Data Recorder (EDR) is used in various automobiles such as airplane, truck, cars etc. These EDR'S are used in various forms depending on the automobiles at which it is installed. The specification of EDR varies with respect to vehicles such as EDR which is used in trucks is different from the EDR which is installed in cars. Even these two EDR is different from the one which is used in airplane (black box). Though these EDR's differ based on the parameters which is measured yet at some parameter measurement these EDR's become closely related to each other. Some of the parameters used by EDR in the existing system for recording purposes are as follows:

- Pre-crash speed
- Engine throttle
- Brake status etc.

The above parameters used in the existing system were not sufficient enough to reconstruct exact situation of the crash. Also these types of EDR's were mainly used in the cars manufactured by American companies. These EDR's were mainly suitable only to the country like America. Country like India is suffering from various deaths due to road accidents which are nearly equal to the death caused when an atom bomb imposed over a country. In this critical situation, recording only 3 to 4 parameters is consider to be unlikely for our country situation. Every data recorded using this type of EDR's would be in a time limit of 5 seconds. Till

now this time limit is considered to be somewhat efficient. But in countries like India even these 5 seconds time slot measurement is considered to be as a crossing a huge mountain which is one of the main drawbacks of the existing EDR's. Even the present EDR system does not contain the steering angle measurement which is also considered to be an important parameter measurement for cars running in India but does not has one. These are the various drawbacks in the existing EDR's which led us to the modification of the existing EDR's to our proposed one.

4.2 PROPOSED SYSTEM:

The drawbacks of the existing EDR's were analysed properly and from that our system is proposed mainly for the benefits of Indian roads. Our proposed EDR's also record various parameters such as:

- Date and Time.
- Speed of the vehicle.
- Brake pedal status.
- Steering angle.

Like existing EDR's our proposed EDR's also used for the reconstruction of pre-crash and crash information. But our proposed work differs mainly in additional efficient parameter measurement. First in our proposed system the time slot for each parameter updates is kept as 1 seconds unlike the existing EDR's which has 5 seconds for parameter updates. In countries like India which is suffering from tragic loss due to road accidents, every second is considered to be as a valuable one for reconstruction of accidents during forensic analysis etc. when compared to 5 seconds update used in the existing system.

Apart from the speed measurement, brake pedal measurement which is carried out in the existing EDR's our proposed EDR uses additional steering angle parameter measurement which lead us to the reconstruction of accidents in an

efficient way. Generally steering angle is used in order to determine the situation or the cause of road accidents by the movement of the steering held out by the driver. In a situation where the cause of the accidents is mainly due to the vehicle coming from the opposite side, the based on rotation of the steering either in clockwise or anticlockwise direction, the angle of steering is determined. From this, the main cause for the accident can be reconstructed.

Other than steering angle and date-time detection, various parameters such as brake pedal status and speed of the vehicle can also be measured in an effective manner. Brake pedal status is determined with the help of ultrasonic sensor which is more efficient when compared to the measurement of brake pedal status carried out in existing system. In the similar manner, speed of the vehicle can also be measured using potentiometer. By this way our proposed work is different from the work which is existing at present

CHAPTER 5

SYSTEM DESIGN

The following section gives a detailed explanation of how the various modules used for the EAIR are interfaced with an Arduino kit, along with a brief overview of how each component works.

5.1 Input Design

This section describes the design of each individual input modules, which includes real time clock DS3231, RPM sensor LM393, potentiometer, and ultrasonic range sensor HC-SR04.

5.1.1 Real Time Clock DS3231

This module is a low-cost, extremely accurate I2C real-time clock (RTC) which has an integrated temperature-compensated crystal oscillator (TCXO), which improves the long term accuracy of the device, and a switched bank of tuning capacitors. The temperature of the crystal is continuously monitored, and the capacitors are adjusted to maintain a stable frequency. The device incorporates a battery input so that it also maintains accurate timekeeping after being disconnected from the main power supply. It maintains seconds, minutes, hours, day, date, month, and year information. Less than 31 days of the month, the end date will be automatically adjusted, including corrections for leap year. The clock operates in the 24 hours or band / AM / PM indication of the 12-hour format. Address and data are transferred serially through an I2C bidirectional bus using the SCL and SDA pins.

The Circuit Connection for Real Time Clock DS3231 with Arduino is given as follows:

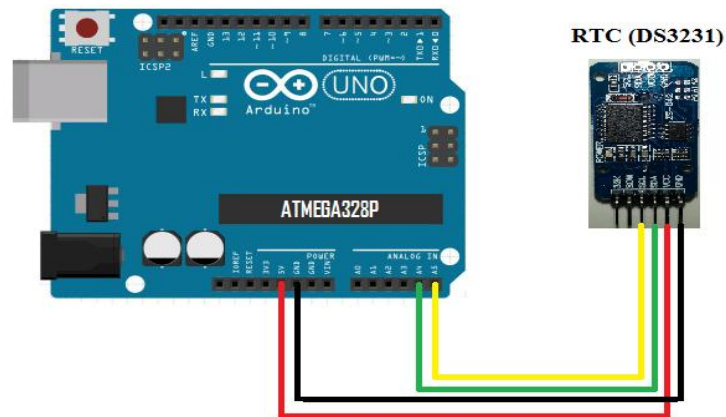


Fig 5.1 RTC DS3231 Connection with Arduino

- VCC ->Arduino +5V pin
- GND ->Arduino GND pin
- SCL ->Arduino Analog Pin 5
- SDA ->Arduino Analog Pin 4

Using the standard methods in the library DS3231.h, the date and time can be read from the board using the methods getDateStr() and getTimeStr() respectively.

5.1.2 RPM Sensor LM393

It is an Optocoupler based voltage comparator in which one side holds a light emitting diode and a photo transistor on the other side. By having the two components matched for a specific frequency of radiation, they are immune to undesired interference. If the visual path is not blocked, phototransistor would conduct, but when something blocks the light falling on the photo transistor it

wouldn't conduct. So an encoder disc (disc with holes) can be placed in the slot of the optical sensor to count the rotation rate of a connected wheel/motor by counting the number of times the sensors goes from low to high.

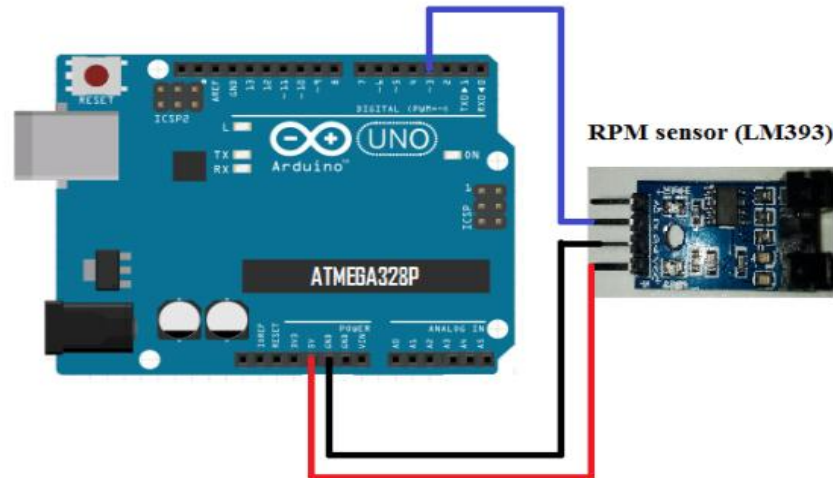


Fig 5.2 RPM Sensor LM393 Connection with Arduino

- VCC ->Arduino +5V pin
- GND ->Arduino GND pin
- DO ->Arduino Digital Pin 3

This sensor gives the number of interruptions in its signal conduction as a digital input to the board, which can be read using the method `digitalRead()`. This value can be used for further calculations.

5.1.3 Potentiometer

A potentiometer (pot) is a three-terminal resistor with a sliding or rotating contact that forms an adjustable voltage divider.

If only two terminals are used, one end and the wiper, it acts as a variable resistor.

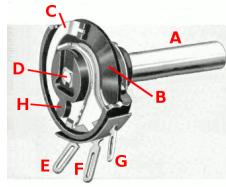


Fig 5.3 Cross section of a rotating pot

Rotating potentiometers are constructed with a resistive element (*B*) formed into an arc of a circle usually a little less than a full turn and a wiper (*C*) sliding on this element when rotated, making electrical contact. The resistive element can be flat or angled. Each end of the resistive element is connected to a terminal (*E*, *G*) on the case. The wiper is connected to a third terminal (*F*), usually between the other two ^[4].

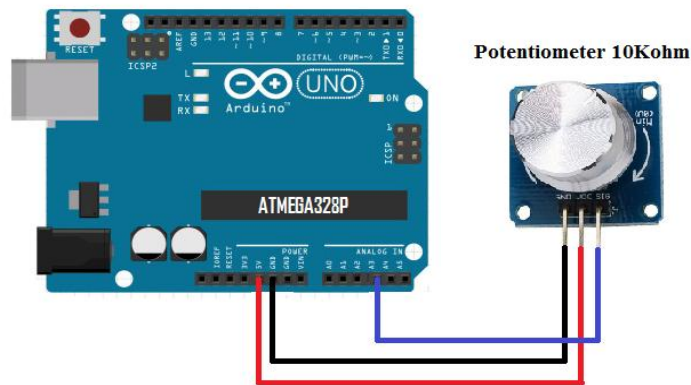


Fig5.4 Potentiometer Connection with Arduino

- VCC ->Arduino +5V pin
- GND ->Arduino GND pin
- Sig ->Arduino Analog Pin 3

The analog value corresponding to the current resistance offered by the potentiometer can be read using the method `analogRead()` and can be used for further calculations.

5.1.4 Ultrasonic Range Sensor HC-SR04

The HC-SR04 ultrasonic sensor uses sonar to determine distance to an object. It consists of the power supply pins Vcc and GND, and the sensor operation pins Trig and Echo. The Trig emits an ultrasound at 40,000 Hz, an 8 cycle sonic burst which will travel at the speed of sound and if there is an object or obstacle on its path, it will bounce back to the module and it will be received in the Echo pin. The Echo pin will output the time in microseconds that the sound wave traveled.

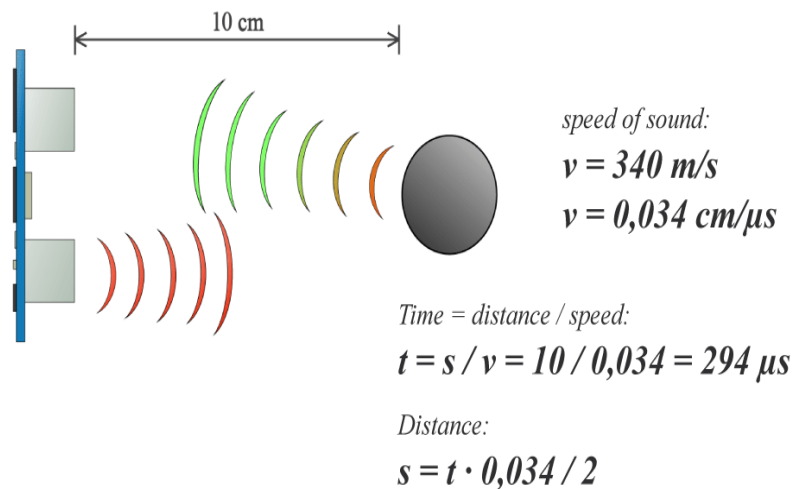


Fig 5.5 Working of HC-SR04

Considering the travel time and the speed of the sound you can calculate the distance. It offers non-contact range detection with high accuracy and stable readings from 2cm to 400cm.

The Circuit Connection for Ultrasonic Range Sensor HC-SR04 with Arduino is given as follows:

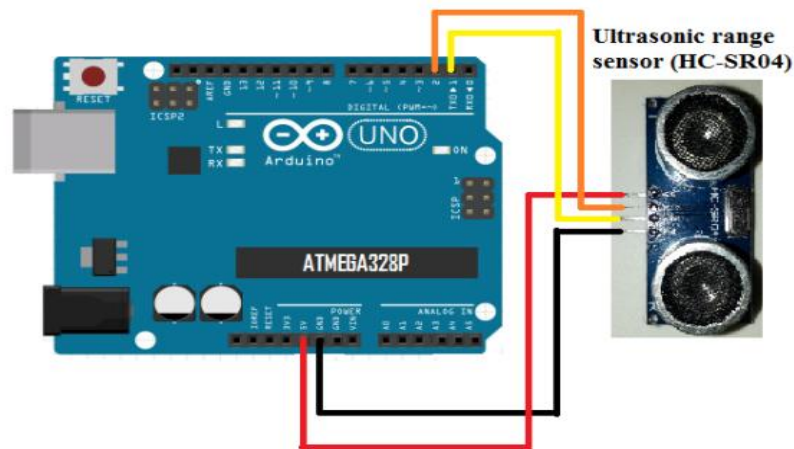


Fig 5.6 Ultrasonic Range Sensor Connection with Arduino

- VCC ->Arduino +5V pin
- GND ->Arduino GND pin
- Trig ->Arduino Digital Pin 2
- Echo ->Arduino Digital Pin 4

By reading the digital outputs from the Trig and Echo pins using `digitalRead()`, the board calculates the distance of the obstacle using the equations shown in Fig.

5.2 Output Design

The outputs of the real time clock DS3231, RPM sensor LM393, potentiometer, and ultrasonic range sensor HC-SR04 are stored in the SD card module from where it can be read and analysed to render graphs or other desired data formats using Microsoft excel, MATLAB etc.

5.2.1 Secure Digital Card Module

Here we are going to see about the connection of SD Card with our Arduino Uno board as shown below.

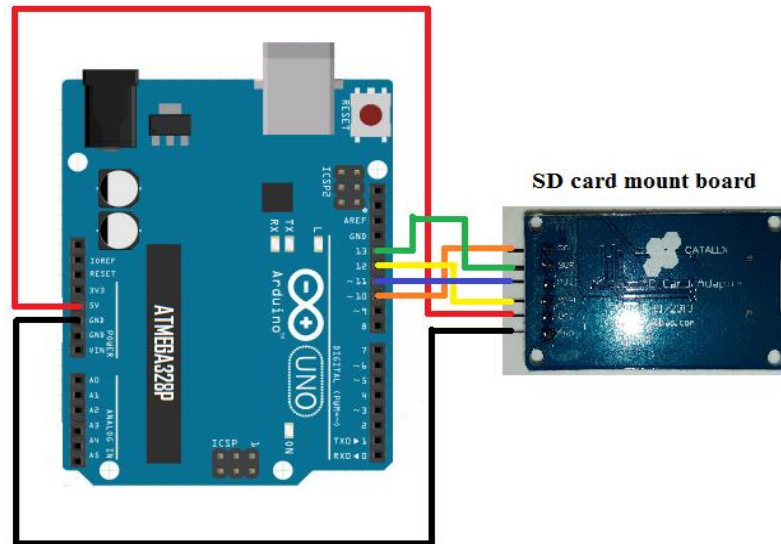


Fig 5.7 SD Card Connection with Arduino

- VCC ->Arduino +5V pin
- GND ->Arduino GND pin
- CS ->Arduino Digital Pin 10
- SCK ->Arduino Digital Pin 13
- MOSI ->Arduino Digital Pin 11
- MISO ->Arduino Digital Pin 12

The communication between the SD card and board uses the serial peripheral interface (SPI) protocol. After successful CS operation, the reading and writing of data can be done by using the corresponding methods to control master in slave out (MISO) and master out slave in (MOSI) pins respectively.

CHAPTER 6

SYSTEM ARCHITECTURE

For successful development of a product, its architecture has a very important role to play. Fig. below shows the overall basic block diagram of the system developed here.

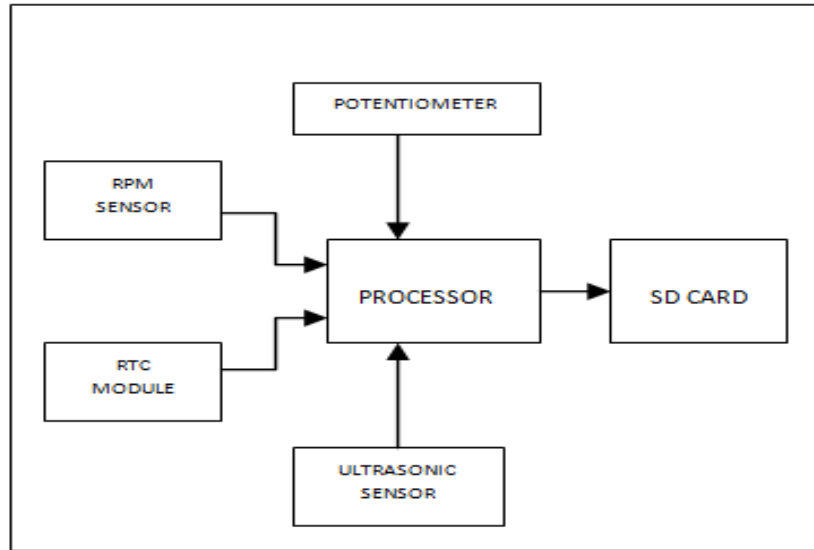


Fig 6.1 System Architecture

An embedded system is normally designed in 3 levels. These are functional, mapping and architectural levels. The mapping level consists of mapping the hardware/software to the sensors and their behaviour. And at the functional level, the capturing sensor behaviour and its verification is done. In architectural level, the hardware/software architecture is refined and linked to the complete architecture implementation and verification. Thus the schematic sketch given above is the portrait of our System.

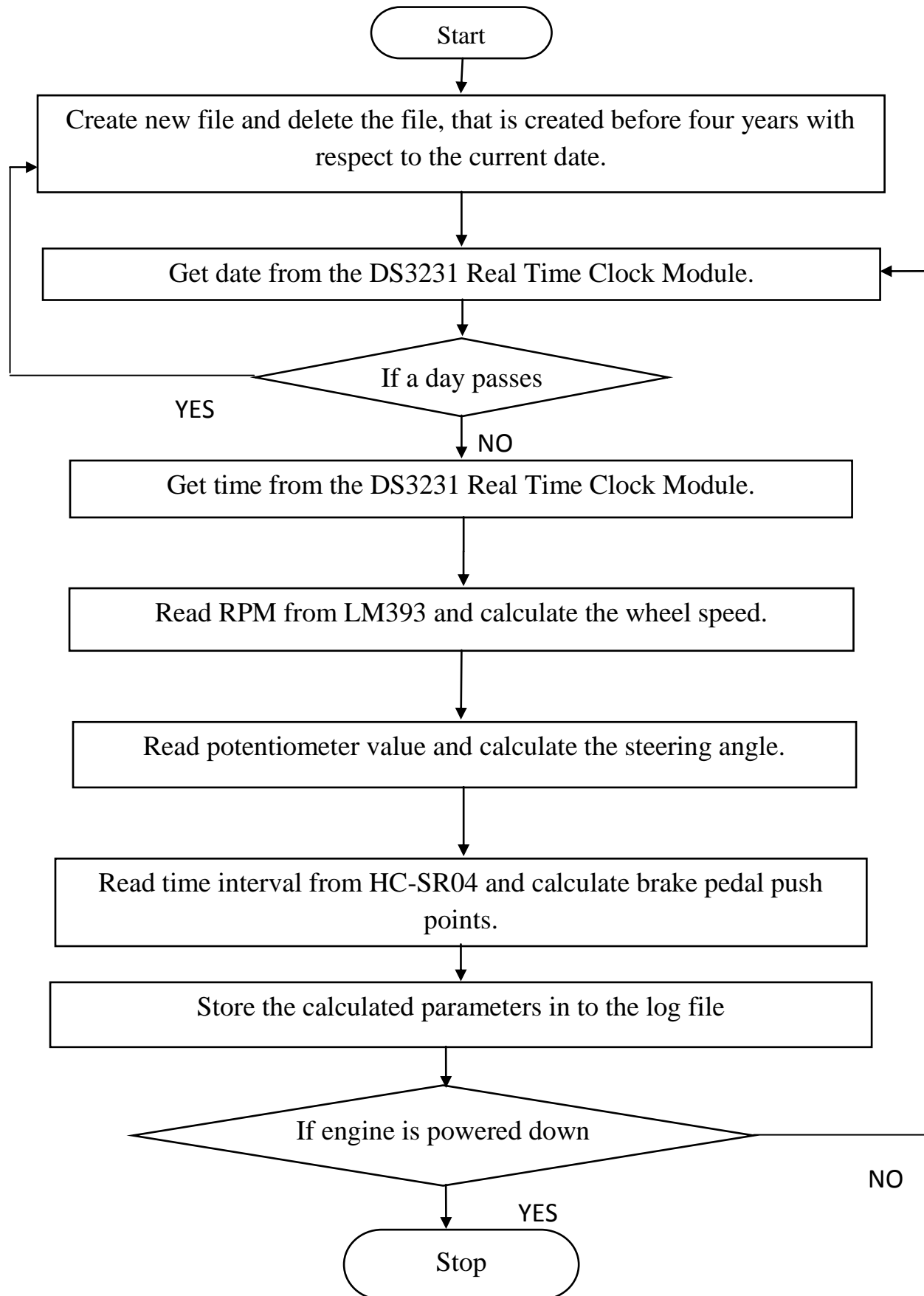
CHAPTER 7

SYSTEM IMPLEMENTATION

7.1 ALGORITHM

1. The EAIR module is given the supply voltage and is switched ON when the user starts the automobile.
2. The filename synthesis process.
 - 2.1 The filename for the file to be created is generated and stored as a string in the variable currentFilename.
 - 2.2 The filename for the file to be deleted is generated and stored as a string in the variable deleteFilename.
3. Next is the file handling process.
 - 3.1 With the currentFilename, the current log file with ‘.txt’ extension is created.
 - 3.2 With the deleteFilename, the log file which is generated four years before with respect to the currentFilename is deleted.
4. Then comes the process of reading the parameters.
 - 4.1 Read time from DS3231.
 - 4.2 Read RPM from LM393.
 - 4.2.1 Convert RPM value into linear speed.
 - 4.3 Read potentiometer value.
 - 4.3.1 Convert the read value into steering angle.
 - 4.4 Read time interval from HC-SR04.
 - 4.4.1 Convert it into distance.
 - 4.4.2 Map the distance value into respective brake pedal push points.
5. Log the data into memory module and save the file.
6. Repeat step 2 to step 5 while the car’s engine is in running state.
7. The process stops when the user powers down the engine.

7.2 FLOWCHART



Thus from the Algorithm and the flow chart given, the various steps to be followed while designing the EDR could be easily understood. Now let us see in detail about the working of our Project.

7.3 WORKING

The end product Black box would look like the photos given below.



Fig 7.1 Designed Black Box

The picture given below shows the circuitry present inside the designed Black box.



Fig 7.2 Internal Circuitry of Designed Black Box

We have formed a Simulator box which acts like a car giving inputs to the Black box that have been designed.

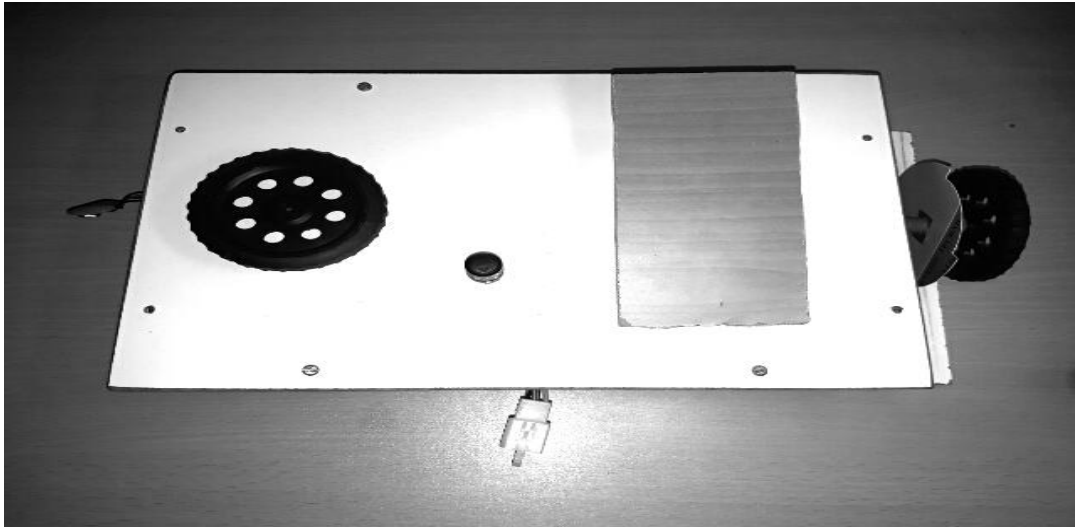


Fig 7.3 Simulator Box

Thus we are going to connect the Simulator box with the Black box designed so that the black box can obtain the inputs from the simulator box that acts like a car.

Below given picture shows the Entire Module.

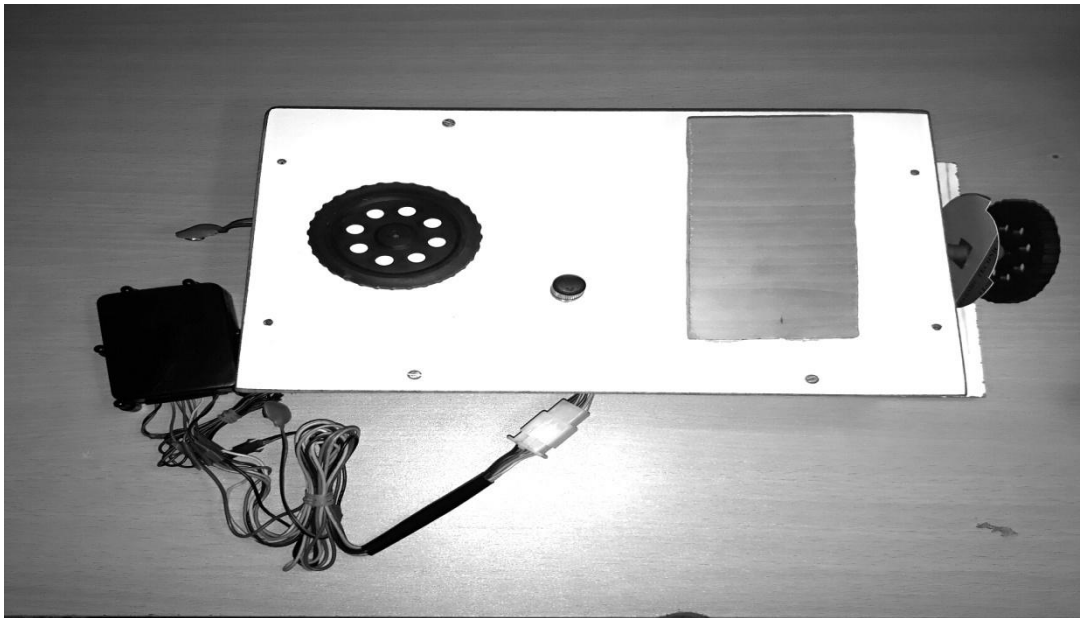


Fig 7.4 Entire Module

Thus the simulator provides the inputs to the Black box, the actual phenomenon behind the recording of parameters is listed as follows

The enhanced auto information recorder can be realized by integrating the above sensors with the processor board as per the following diagram.

Fig 7.5 Wiring of sensors with processor.

- The Real Time Clock module, the Microprocessor, and the SD card mount board are placed inside a crash proof console.
- The potentiometer is placed adjacent to the steering column in such a way, so as to measure steering angle.
- The RPM sensor is placed near the wheel shaft in such a way that it can read the motion of the encoder disc in order to measure the speed of the wheels.
- The ultrasonic range sensor is placed beneath the brake pedal in order to obtain its position.

The processor is switched on and the logging of data starts when the user starts the automobile as per the algorithm specified in the section 7.1. At first the file name for logging of current data is generated the format 'DDMMYYYY'. For example if the current date is 27-10-1995, then the corresponding file name would be 27101995. Similarly the file name for the file to be deleted is also generated. For example if the current file name 271011995 then the respective file name for the file to be deleted would be 27101991 (i.e. the file which is generated four years from the current date). In this way the EAIR is designed to store past four year data from the current date in the form of a text file. The logged text file has the data for each second in one line as per the following syntax <time>,<speed>,<steering angle>,<brake pedal push point>. The time in the syntax of <HH:MM:SS>, brake pedal status is specified in the form of points ranging from 1 to 10 where 1 means no brake is applied while 10 indicates brake pedal is depressed to the maximum extent. The speed is recorded in km/h and the angle is recorded in degrees with 0 as the neutral position of the steering wheel, negative values indicating clockwise rotation and positive values indicating anti-clockwise rotation of the steering wheel. For analysis the text file can be interpreted into graphs. As the log has the details of speed, steering angle and amount of break applied recorded at each second, the condition of the vehicle can be predicted, provided the accident date and time.

CHAPTER 8

SYSTEM TESTING

8.1 SENSOR TESTING

Since this is a data logging process the input to the processor should be very accurate in order to produce the expected results. Hence the sensors which are used to collect data from the automobile should be tested before their actual implementation.

8.1.1 TESTING THE REAL TIME CLOCK MODULE DS3231

The purpose of this module is to provide date and time information for data logging process. The module is tested by initializing it with the current date and time and checking its working by reading the data from it after sometime to verify whether the date and time details are change automatically with the help of its backup battery supply.

8.1.2 TESTING THE LM393 SENSOR

The purpose of these sensors is to monitor whether there is an active optical conduction between its LED and photo diode terminals by which it gives out the logical HIGH and logical LOW signals as output. Its working is checked by interfacing it with a processor and reading its digital output while its line of conduction is periodically interfered by the arms of the encoder disc.

8.1.3 TESTING THE POTENTIOMETER

The purpose of this potentiometer is to give out the output values ranging from 0 to 1023 when its knob it's turned to set the resistance from minimum to maximum. Its working is checked by connecting it to a processor and reading its output which satisfies its following criteria, 0 for minimum resistance, 1023 for maximum resistance.

8.1.4 TESTING THE HC-SR04 SENSOR

The purpose of this sensor is to give a logical HIGH output when the ultrasonic sound pulse emitted from its trigger pin is received in its echo pin after hitting in the obstacle. Its working can be verified by reading its echo pin output which satisfies the following condition, logical 0 for no obstacle and logical 1 for presence of obstacle in its front face.

8.2 CODE TESTING

The following standard software testing procedures are used to test the EAIR processors code.

8.2.1 WHITE BOX TESTING

White box testing is sometimes called as glass box testing. It is a test case design that uses a control structure of the procedural design to drive the test cases. Using white box testing methods, the software engineer can drive test cases that

- Guarantee that logical decisions are on the true or false sides.
- Exercise all logical decisions are on the true or false sides.
- Execute all loops as their boundaries and within their operational bounds.
- Exercise internal data structure to assure the validity.

8.2.2 BLACK BOX TESTING

Black box testing focuses on the functional requirements of the software. That is the black box testing enables the software engineer to drive a set of input conditions that will fully exercise the requirements of the code.

Black box testing is not an alternative method for white box testing technique. Rather, it is a complementary approach that is likely to uncover different class of error. Black box testing attempts to find errors in the following categories:

- Interface errors.
- Performance errors in data structures.
- General performance errors.
- Initialization and termination errors.
- Incorrect and missing functions.

8.3 PERFORMANCE TESTING

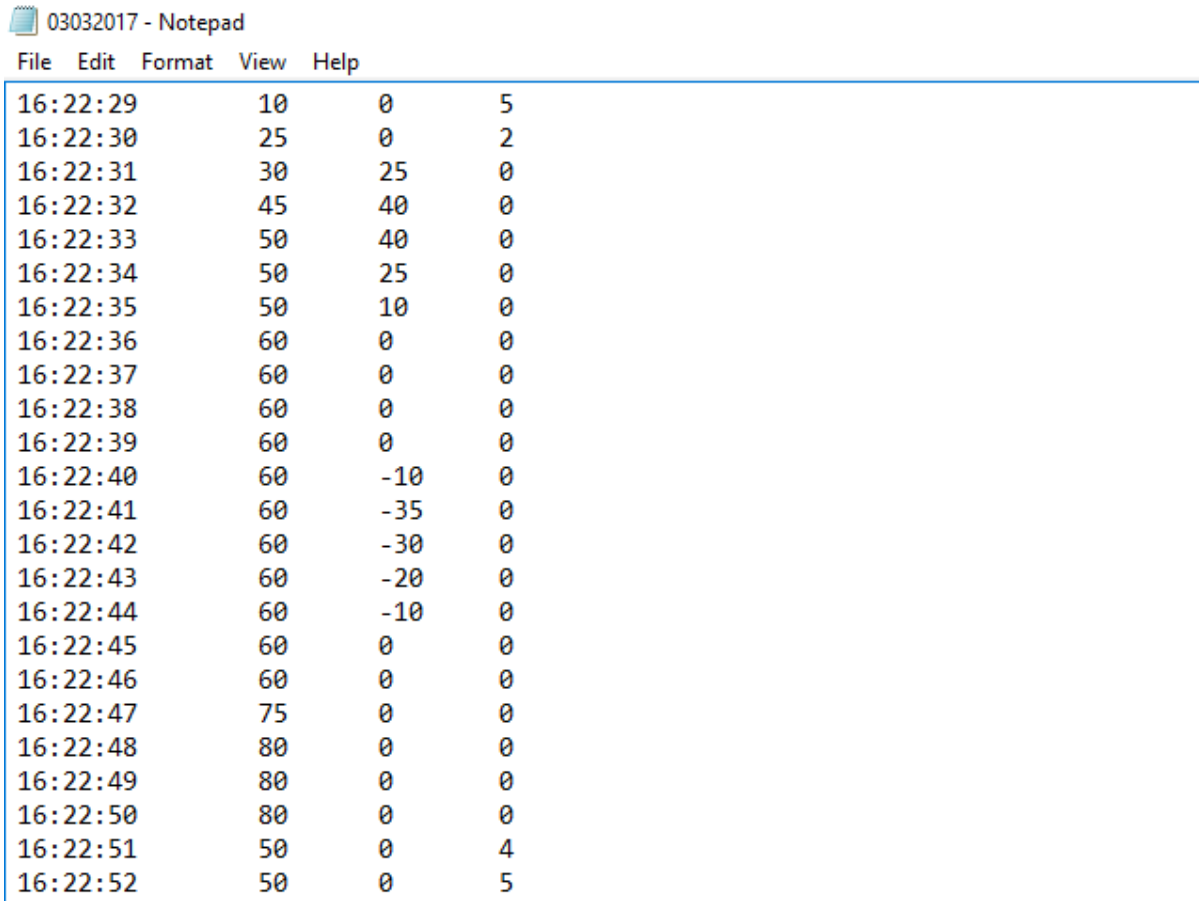
The EAIR processor is coded to perform basic file handling operations like create, open, write and delete a file. All these functions should be tested in real time application to ensure the perfect working of the device. For this all the sensors are integrated with processor and their corresponding outputs are read by the processor with specific delay between each input read operation before implementing the actual device in the automobile. Then the read parameters are processed and stored in the memory as a text file. The final text file output is verified for the condition of data being logged for each second and the file name is exactly matching with the current date of the logging process. This ensures the proper working of the device.

CHAPTER 9

SAMPLE SCREEN OUTPUTS

9.1 TEXT FILE OUTPUT

The snapshot of the Text Output file is given as follows:



The image shows a Notepad window titled "03032017 - Notepad". The menu bar includes "File", "Edit", "Format", "View", and "Help". The text content is a table of sensor data with four columns: a timestamp, and three numerical values. The data spans from 16:22:29 to 16:22:52.

16:22:29	10	0	5
16:22:30	25	0	2
16:22:31	30	25	0
16:22:32	45	40	0
16:22:33	50	40	0
16:22:34	50	25	0
16:22:35	50	10	0
16:22:36	60	0	0
16:22:37	60	0	0
16:22:38	60	0	0
16:22:39	60	0	0
16:22:40	60	-10	0
16:22:41	60	-35	0
16:22:42	60	-30	0
16:22:43	60	-20	0
16:22:44	60	-10	0
16:22:45	60	0	0
16:22:46	60	0	0
16:22:47	75	0	0
16:22:48	80	0	0
16:22:49	80	0	0
16:22:50	80	0	0
16:22:51	50	0	4
16:22:52	50	0	5

Fig 9.1 Sample text files output.

Thus from the Text file output it is very much clear that the outputs from various sensors have been processed and are recorded as shown above.

9.2 TIME VERSUS SPEED ANALYSIS

The snapshot of the graph with Time Versus Speed is given as follows:

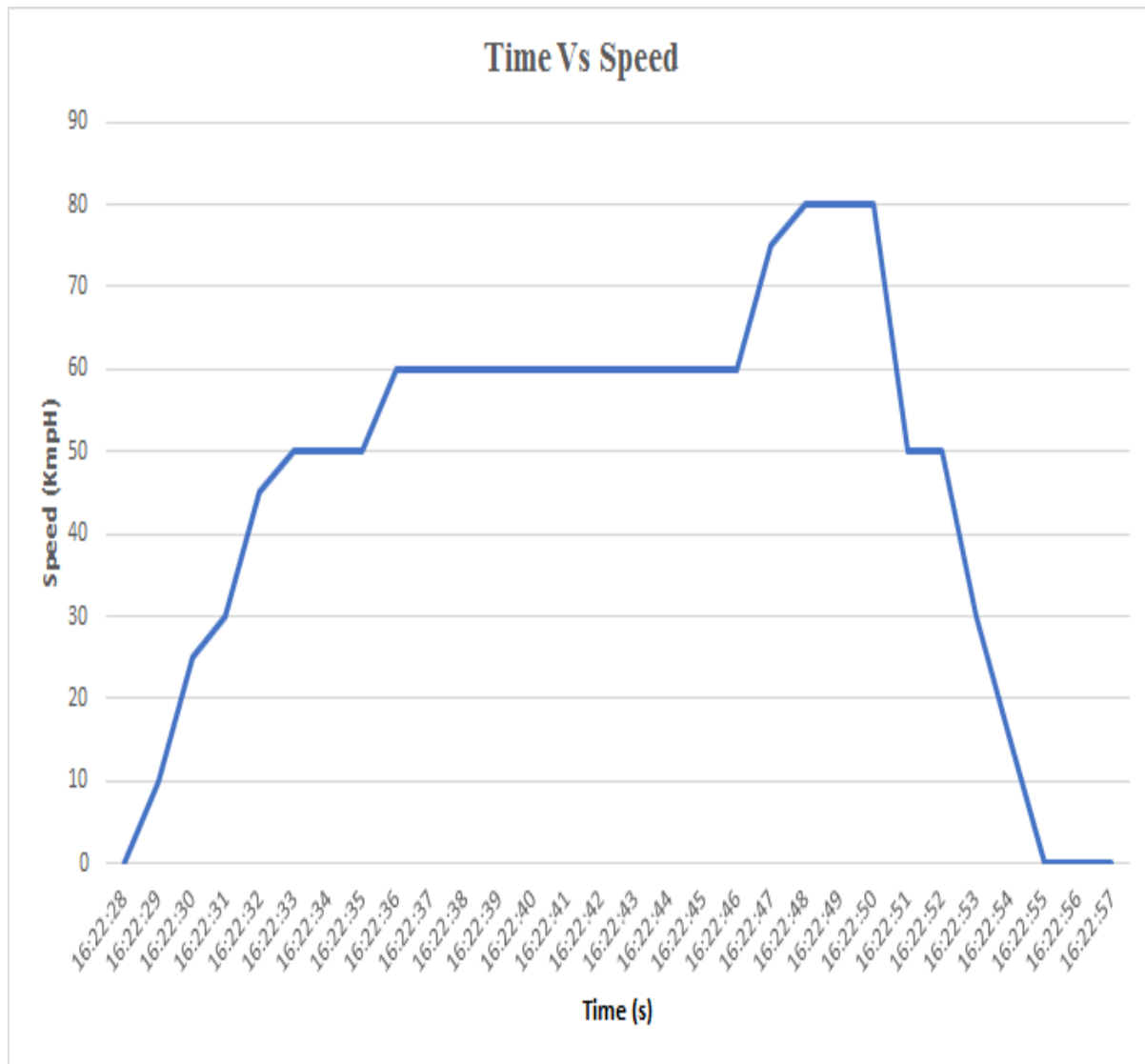


Fig 9.2 Time Vs Speed.

From the graph shown above we can infer the fact that as and when time varies the speed also varies. Thus from the outputs one could easily detect the abnormalities if present.

9.3 TIME VERSUS BRAKE PEDAL STATUS ANALYSIS

The snapshot of the graph with Time Versus Brake Pedal Status is given as follows:

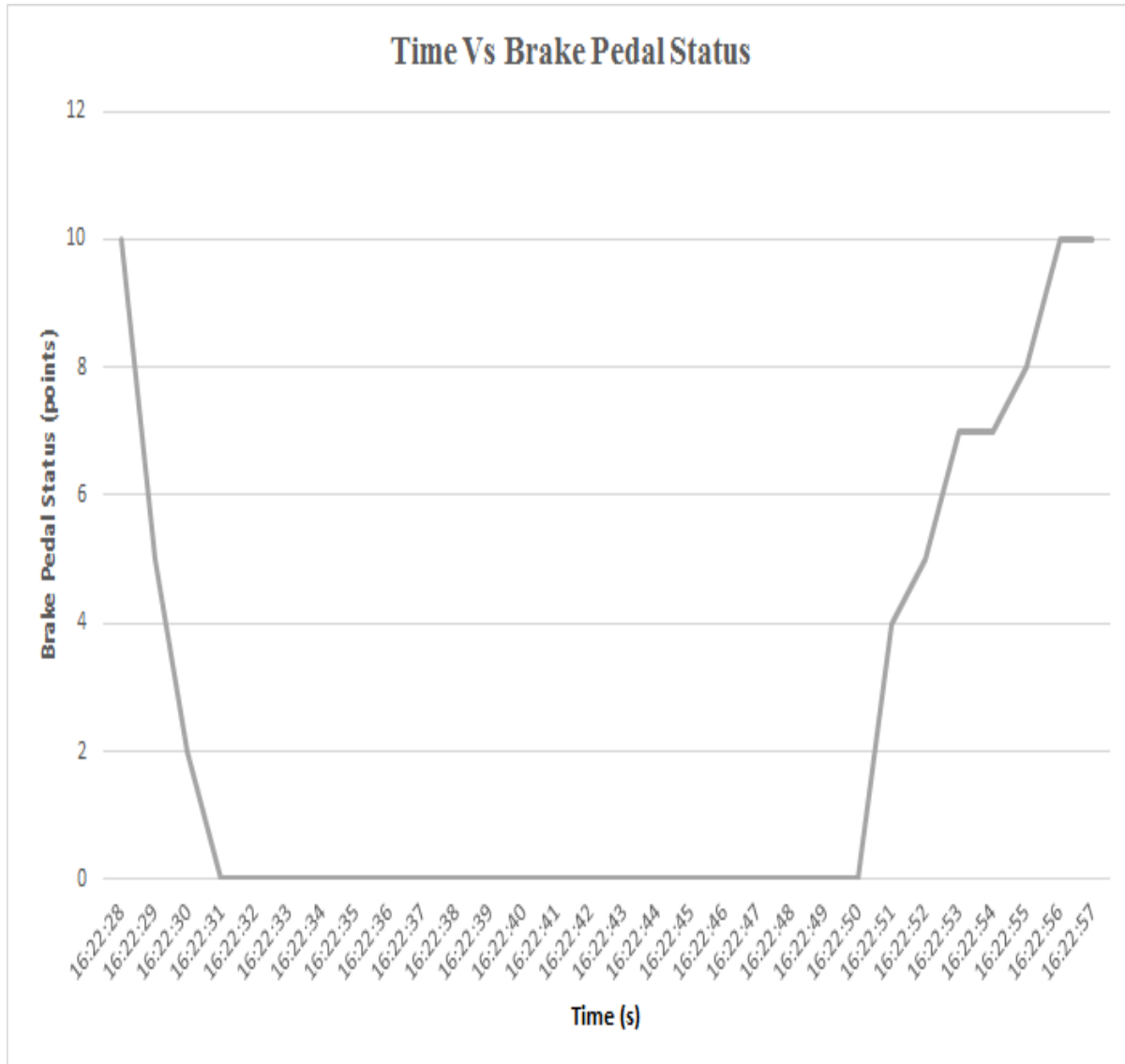


Fig 9.3 Time Vs Brake Pedal Status.

The range of the brake pedal status have been set from 0 to 10. And from the above given output it is clear that for varying time, the varying brake pedal status have been recorded on a real time basis.

9.4 TIME VERSUS STEERING ANGLE ANALYSIS

The snapshot of the graph with Time Versus Steering Angle is given as follows:

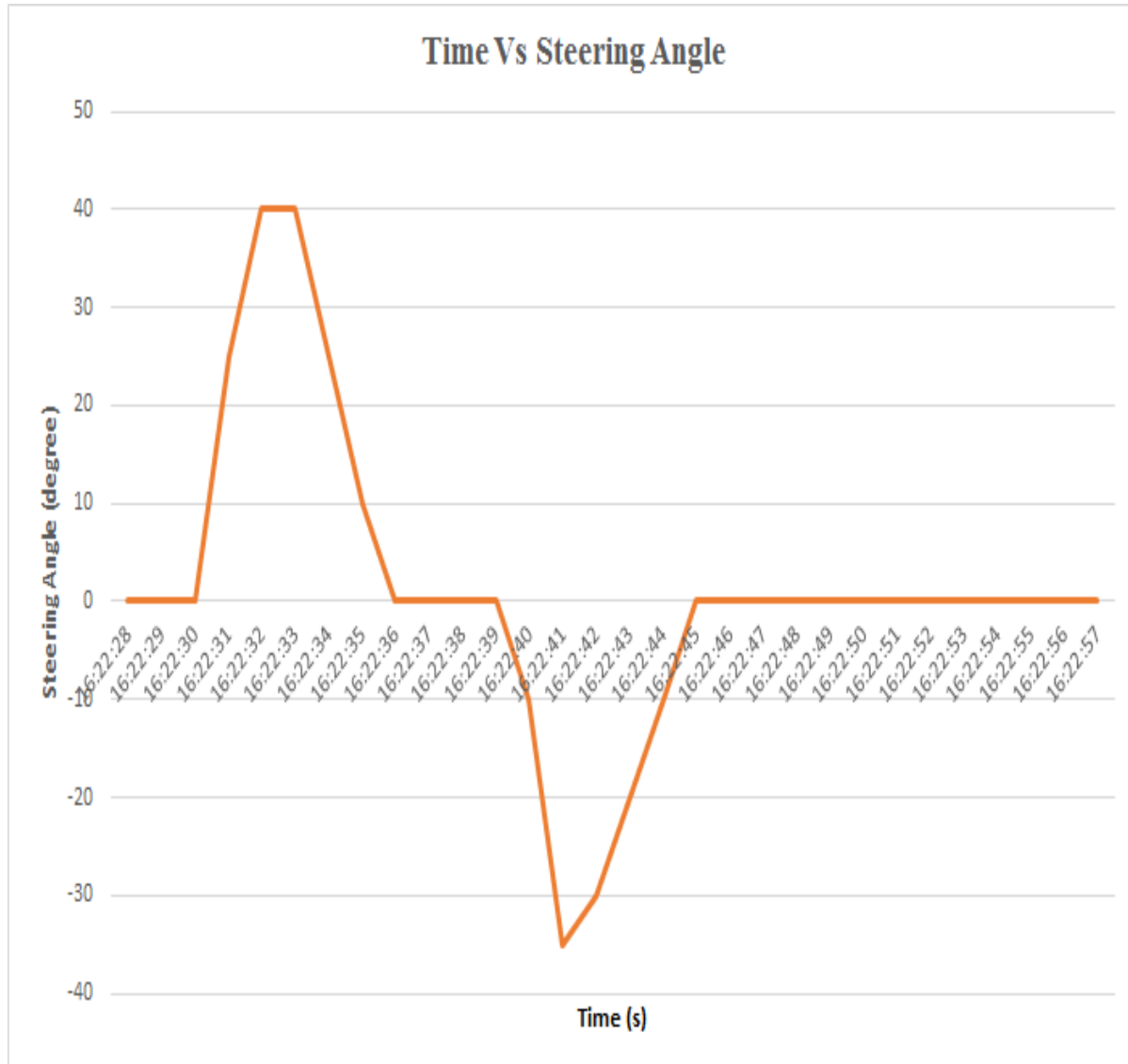


Fig 9.4 Time Vs Steering Angle.

For varying time, varying Steering angle has been recorded as shown above.

9.5 TIME VERSUS SPEED AND BRAKE PEDAL STATUS ANALYSIS

The snapshot of the graph with Time Versus Speed and Brake Pedal Status is given as follows:

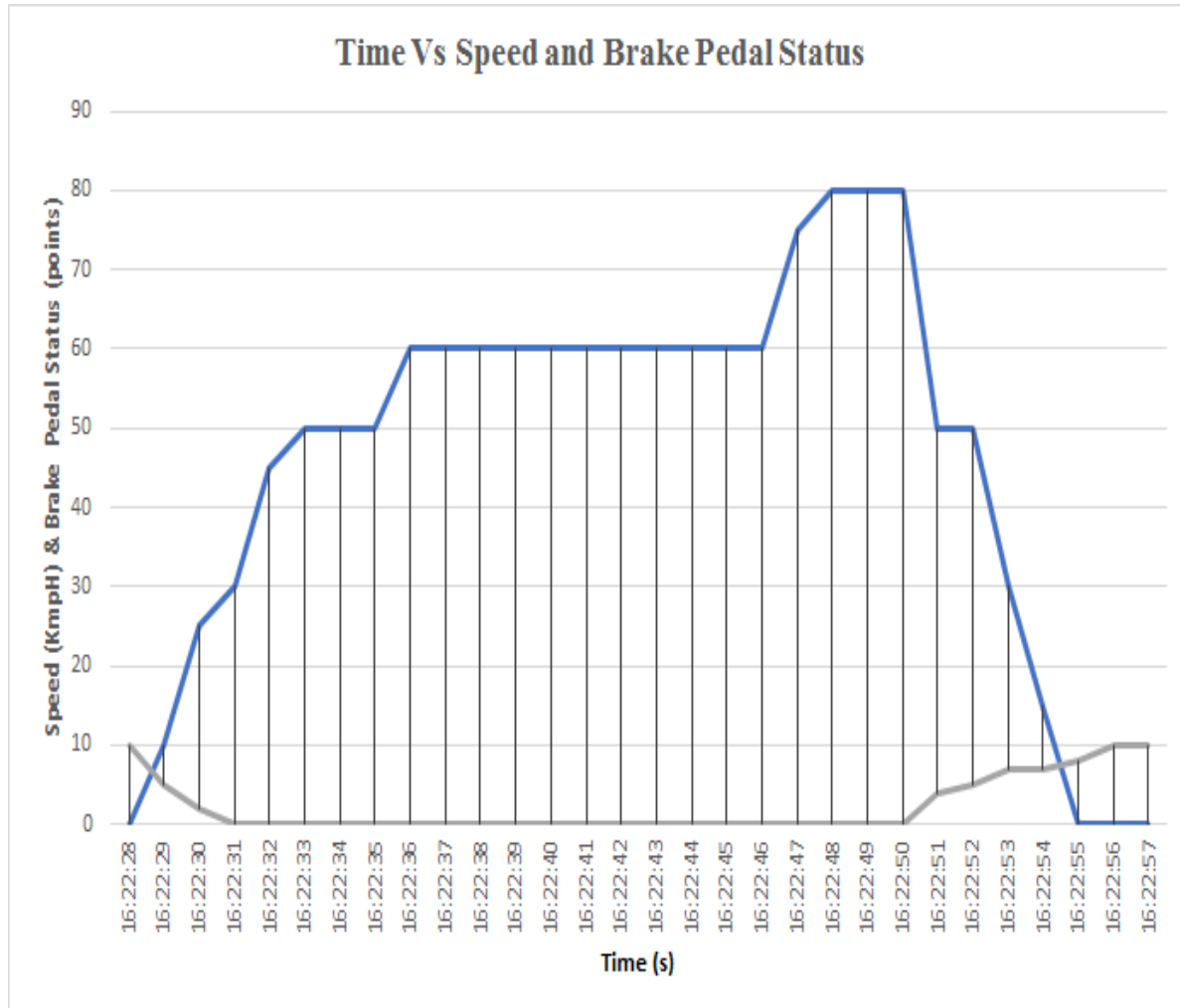


Fig 9.5 Time Vs Speed and Brake Pedal Status.

This graph shows the variations for time corresponding to the Speed and Brake pedal Status.

Thus all the recorded values have been interpreted as Graphs for easy understanding .

CHAPTER 10

CONCLUSION

The ultimate objective of this particular project is to record the information that can be used for post-accident analysis, find the cause of the accident and improve vehicle standards accordingly to prevent similar accidents in the near future. It was done by measuring certain parameters of the vehicle which includes the event's date and time, steering angle, brake pedal status and wheel speed in running condition. The data collected can be used for identifying false insurance claim, faults in the vehicles, driving behaviour of the driver and helps in aiding police investigations of the causes of accidents. This is also used to prevent future accident due to similar causes, which can help save a lot of lives in the future.

10.1 CHALLENGES ENCOUNTERED

At the initial stage, the selection of appropriate sensors was little tedious. Formulating the Wiring Diagram, Interfacing the Sensors to the Board, Developing the bug free code was back breaking. After several steps of planning the right assembling of the entire module was found out.

10.2 UNIQUE FEATURES

In this project, an economical Auto Information Recording system with built-in security features has been designed and implemented. Portability, low power consumption and low cost are the most important factors brought into the system by the usage of the embedded intelligent Processing module. These are some of the unique features of this project. Besides other benefits it proves to highly a cost cutting system and also easily affordable.

The proposed work is an epitome of crash forensics.

10.3 FUTURE ENHANCEMENT

We aim, in a future work, to achieve further enhancements like incorporating other variety of sensors to record more and more critical parameters to develop an advanced version of this for the welfare of the society.

We also look forward to implement additional innovations in this system to make it highly intelligent like directly uploading the recorded parameters on the web portal designed. And even the mode of communication could be made wireless in the near future.

CODING

The code for EAIR processor is as follows.

```
#include<SPI.h>

#include<SD.h>

#include<DS3231.h>

DS3231 rtc(SDA,SCL);

const int csPin=10, rpmPin=7, trigPin=6, echoPin=5, potPin=3;

String fileName, DMY, loopDMY, curDate, curYear, curMonth, delYear,
delFileName, extension = ".txt";

long duration, interval, centi, brakePoint;

double secForOneRotation , microSec, milliSec, secs, wheel_speed;

int curYearValue, delYearValue, potValue, degree, rpm;

File logFile;

void deleteFile(String delFileName)
{
    if(SD.exists(delFileName)) // Checking existence of file to be deleted.
    {
        SD.remove(delFileName); // Deleting the file.
    }
    else{
        Serial.println("Error deleting file : File not Found ");
    }
}

String nameFile()
```

```

{
    DMY = rtc.getDateStr();
    curDate = DMY.substring(0,2);
    curMonth = DMY.substring(3,5);
    curYear = DMY.substring(6,10);
    fileName = curDate + curMonth + curYear + extension ;
    curYearValue = curYear.toInt();
    delYearValue = curYearValue - 4;
    delYear = String(delYearValue);
    deleteFile(delFileName);
    return fileName;
}

void setup()
{
    Serial.begin(9600);
    pinMode(csPin,OUTPUT);
    pinMode(9,OUTPUT);
    digitalWrite(9,HIGH);
    if(SD.begin())
    {
        Serial.println("SD card ready to use.");
    }
    else
    {

```

```

        Serial.println("Error : Initializing SD card. ");
    }
    pinMode(potPin,INPUT);
    pinMode(trigPin,OUTPUT);
    pinMode(echoPin,INPUT);
    pinMode(rpmPin,INPUT);
    pinMode(9,OUTPUT);
    digitalWrite(9,HIGH);
    rtc.begin();
    fileName = nameFile();
}
void loop()
{
    if(DMY != loopDMY) // Checking whether a day passed.
    {
        fileName = nameFile(); // Requesting for new file name
    }
    delay(200);
    microSec = pulseIn(rpmPin, HIGH);
    if (microSec<= 0) // Checking if the wheels are idle with engine ON condition.
    {
        secForOneRoation =0;
    }
    milliSec = microSec*0.001;

```



```

secs = microSec*0.000001;

secForOneRoation =secs*4; // Calculating time to complete one rotation.

rpm = 60/ secForOneRoation; // Calculating RPM.

wheel_speed = 2*3.14*0.033*rpm*0.06; // Converting RPM to wheel speed.

delay(200-milliSec);

potValue = analogRead(potPin);

degree = map(potValue, 0, 1023, -125, 125); // Calculating steering angle

delay(200);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);

centi = duration / 29 / 2;

brakePoint = map(centi, 3, 10, 10, 1);

brakePoint = constrain(brakePoint, 0, 10); // Calculating brake pedal push point.

delay(200-0.000002);

logFile = SD.open(fileName, FILE_WRITE); // Creating or opening a log file.

if (logFile)
{
logFile.print(rtc.getTimeStr()); // Writing time to the log file.

logFile.print(",");

logFile.print(wheel_speed); // Writing wheel speed to the log file.

logFile.print(",");

```

```
logFile.print(brakePoint);// Writing brake pedal push point to log file.  
logFile.print(",");  
logFile.print(degree); // Writing steering angle to the log file.  
logFile.println(); // Moving cursor to new line.  
logFile.close(); // Closing the file after logging one line of data.  
  
}  
  
else  
  
{  
Serial.println("ERROR: Failed to open file.");  
  
}  
  
delay(200);  
  
}
```

REFERENCES

- [1] Abdelmjid Saka, Fatima Ezzahra Saber, Mohamed Ouahi, “Vehicle Dynamics and Steering Angle Estimation Using a Virtual Sensor”, Engineering, Systems and Applications Laboratory National School of Applied Sciences (ENSA), USMBA Fez, Morocco.
- [2] Accident Statistics :

<http://www.firstpost.com/india/mumbai-pune-expressway-accident-is-a-reminder-of-the-state-of-road-safety-in-india-2826294.html>
- [3] Alan German, Dainius J. Dalmotas, Jean-Louis Comeau, “Event Data Recorders in Toyota Vehicles”, D.J. Dalmotas Consulting, Inc.
- [4] Antonio C. Canale, Rosalvo T. Ruffino, “Study Of The Dynamic Weight On The Axle And Wheels Of An Automobile Using The Static Equilibrium Or Center Of Gravity Range Of The Vehicle”, SAE Technical Paper 931648, 1993, doi:10.4271/931648.
- [5] Arduino : <https://en.wikipedia.org/wiki/Arduino>
- [6] Bing-Fei Wu, Chung-Hsuan Yeh, Ying-Han Chen, “ Driving behaviour based event data recorder ,” in IET Intelligent Transport Systems (Volume:8, Issue: 4, June 2014).
- [7] Carl Ragland, Chip Chidester, John Brophy, John Hinch, “Evaluation of Event Data Recorders in full systems crash test.”, Paper No: 05-0271, National Highway Traffic Safety Administration, United States.
- [8] Caroline Crump, Douglas Young, Robyn Brinkerhoff, Sarah S. Sharpe, “Accelerator-to-Brake Pedal Transition Movements during On-Road Stopping in an Older Population”, SAE Technical Paper 2017-01-1396, 2017, doi:10.4271/2017-01-1396.
- [9] Event Data Recorder – various informations about it and it’s types

[:https://en.wikipedia.org/wiki/Event_data_recorder](https://en.wikipedia.org/wiki/Event_data_recorder)

- [10] Mahalaxmi Bhille, Murugesh Gorajanal, Priyanka Mannikeri, Venkatesh Nayak, Vikas Deshpande, "AUTO – FIR ," in *Proc. of the 2015 IEEE International Advance Computing Conference (IACC)* ,B.M.S. College of Engineering Bangalore, India.
- [Online]. Available: <http://ieeexplore.ieee.org/document/6824012>
- [11] Mr David Connolly, "Event Data Recorder As A Forensic Tool", Dublin Institute of Technology (DIT), Proceedings of the ITRN2014, University of Limerick .
- [12] Pacheco .J .E ., E. López , "Monitoring driving habits through an automotive CAN network," in *Electronics, Communications and Computing (CONIELECOMP)*, 2013 International Conference on Electronics, Communications and Computing.