## AUTOMATED ALGORITHMIC ACCIDENT PREVENTION SYSTEM

#### A PROJECT REPORT

***Submitted by***

**GUNASEELAN K [REGISTER NO:211417104072] HARI HARAN B [REGISTER NO:211417104075]**

**SUDHARSAN K [REGISTER NO:211417104272]**

***in partial fulfillment for the award of the degree of***

### BACHELOR OF ENGINEERING

**IN**

#### COMPUTER SCIENCE AND ENGINEERING

**PANIMALAR ENGINEERING COLLEGE, CHENNAI-600123.**

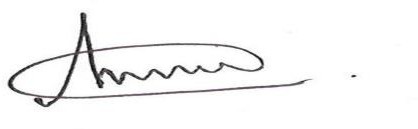
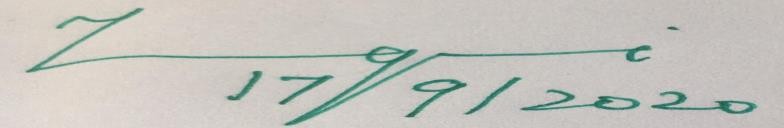
### ANNA UNIVERSITY: CHENNAI 600 025

**APRIL 2020**

### BONAFIDE CERTIFICATE

Certified that this project report **“AUTOMATED ALGORITHMIC ACCIDENT PREVENTION SYSTEM”** is the bonafide work of **“ GUNASEELAN K (2017PECCS396), HARI HARAN B (2017PECCS398), SUDHARSAN K (2017PECCS415) ”** who carried

out the project work under my supervision.



#### SIGNATURE SIGNATURE

**Dr.S.MURUGAVALLI,M.E.,Ph.D., Dr. S. BALAJI.,(Ph.D.), HEAD OF THE DEPARTMENT ASSISTANT PROFESSOR(GRADE-I),** DEPARTMENT OF CSE, DEPARTMENT OF CSE,

PANIMALAR ENGINEERING COLLEGE, PANIMALAR ENGINEERING COLLEGE, NAZARATHPETTAI, NAZARATHPETTAI,

POONAMALLEE, POONAMALLEE,

CHENNAI-600 123. CHENNAI-600 123.

Certified that the above candidate(s) was/ were examined in the Anna University Project Viva-Voce Examination held on 22.09.2020



#### INTERNAL EXAMINER EXTERNAL EXAMINER

**ACKNOWLEDGEMENT**

We express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We would like to extend our heartfelt and sincere thanks to our Directors

#### Tmt.C.VIJAYARAJESWARI, Thiru.C.SAKTHIKUMAR,M.E., and

**Tmt. SARANYASREE SAKTHIKUMAR B.E.,M.B.A.,** for providing us with the necessary facilities for completion of this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** for his timely concern and encouragement provided to us throughout the course.

We thank the HOD of CSE Department, **Dr. S.MURUGAVALLI , M.E.,Ph.D.,**

for the support extended throughout the project.

We would like to thank my **Project Guide Dr.S.BALAJI,(Ph.D.)** and all the faculty members of the Department of CSE for their advice and suggestions for the successful completion of the project.

K. GUNASEELAN,

B. HARIHARAN,

K. SUDHARSAN

## ABSTRACT

The main impact of this project is to minimize the number of accidents. Now-a-days, more number of accidents occurs during the time of Mist season and also the driver cannot able to predict the amount of brake applied by the before vehicle. So our proposed project will rectify these problems. The **UltraSonic Sensor** is used to gather the information of the Distance between the before vehicle or any obstacles and display the distance to the Driver using the LCD Monitor which is placed in front of Driver. This technique is useful for the Vehicle Drivers during Mist Season to analyze the distance of vehicles and prevent themselves from an Accident. The **LCD Display** should also be placed at the backside of Vehicle which displays distance between the current vehicle and previous vehicle using **VFDT ALGORITHM**. This displays about an distance information to the next upcoming vehicle. So that the drivers can able to predict and slow down the vehicles without putting any rash brakes. So this concept will be useful during all the Season to prevent the Accidents by using Rash Brakes.

Our Proposed project also useful to convey the message to the Parents when accident has occurred. This can be achieved by using the Crash Sensor, GPS Module, GSM Module. The **Crash Sensor** is used to pass the information to the GPS Module when an accident has occurred. The **GPS Module** will get an information about the Current Location where an accident had occurred and pass information to the GSM Module. The **GSM Module** will send the message with the desired Location to the registered Mobile Number. Thus the People can be saved by admitting in the Hospital within time.

### LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
| **4.** | **SYSTEM DESIGN** |  |
|  | 4.1. ER-Diagram | 25 |
|  | 4.2. Data Flow Diagram of Doctor | 28 |
|  | 4.3. Data Flow Diagram of Patient | 28 |
|  | 4.4. Use Case Diagram | 33 |
|  | 4.5. Class Diagram | 34 |
|  | 4.6 Sequence Diagram | 35 |
|  | 4.7 Timeline Chart | 36 |
| 5. | SYSTEM ARCHITECTURE |  |
|  | 5.1. System Architecture | 38 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
| **3.** | **SYSTEM ANALYSIS** |  |
|  | 3.1. Minimum Hardware Requirements | 13 |
|  | 3.2. Minimum Software Requirements | 13 |
| **4.** | **SYSTEM DESIGN** |  |
|  | 4.1. Data Dictionary of User | 26 |
|  | 4.2. Data Dictionary of Patient | 26 |
|  | 4.3. Data Dictionary of Doctor | 27 |
| **6.** | **SYSTEM TESTING** |  |
|  | 6.1. Test Case for User Module | 46 |
|  | 6.2. Test Case for Knowledge System | 47 |
|  | 6.3. Test Case for Doctor Module | 47 |

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | **ABSTRACT** | iv |
| **1.** | **INTRODUCTION** | 1 |
|  | 1.1 Overview | 2 |
| **2.** | **LITERATURE SURVEY** | 4 |
| **3.** | **SYSTEM ANALYSIS** | 10 |
|  | 3.1 Existing System | 11 |
|  | 3.2 Proposed system | 11 |
|  | 3.3 Requirement Analysis and Specification | 11 |
|  | 3.3.1 Input Requirements | 12 |
|  | 3.3.2 Output Requirements | 12 |
|  | 3.3.3 Functional Requirements | 12 |
|  | 3.4 Feasibility Study | 12 |
|  | 3.4.1 Technical Feasibility | 12 |
|  | 3.4.2 Economic Feasibility | 12 |
|  | 3.5 Hardware Environment | 13 |
|  | 3.6 Software Environment | 13 |
|  | 3.7 Software Specification | 14 |

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
| **4.** | **SYSTEM DESIGN** | 24 |
|  | 4.1. ER diagram | 25 |
|  | 4.2 Data dictionary | 25 |
|  | 4.3 Data Flow Diagram | 27 |
|  | 4.4 UML Diagrams | 29 |
| **5.** | **SYSTEM ARCHITECTURE** | 37 |
|  | 5.1 Architecture Overview | 38 |
|  | 5.2 Module Design Specification | 38 |
|  | 5.2.1 Module Explanation | 39 |
|  | 5.3 Program Design Language | 40 |
| **6.** | **TESTING** | 43 |
|  | 6.1 System Testing | 44 |
|  | 6.2 Unit Testing | 45 |
|  | 6.3 Test Cases & Reports / Performance Analysis | 46 |
| **7.** | **CONCLUSION** | 48 |
|  | 7.1 Conclusion and Future Enhancements | 49 |
|  | **APPENDICES** |  |
|  | A.1 Sample Screens | 50 |
|  | A.1 Sample Code | 54 |
|  | **REFERENCES** | 93 |

# CHAPTER 1 INTRODUCTION

### 1.1 OVERVIEW

Now-a-days most of the accidents are occurring during the Mist Season. During this season the Vehicle Drivers cannot able to have the clear vision of the before vehicles or road. Thus in majority of the time it may results in an Accidents.

There is no chance to stop the Vehicle by the Drivers during such situation because they may travel at high speed and they will see the Vehicle when reaches the before Vehicle most closely.

Our Proposed Model will intimate them when they drive the vehicle very closely to the another vehicle by displaying an information about the Distance between the Vehicles with the help of VFDT Algorithm (which alerts the user according to the driven speed) via LCD Display or LCD Monitor which will be placed infront of Driver Seat and also the LCD Display will be placed at the backside of the Vehicle which will be useful for the rear Vehicle driver to detect the speed of the front vehicle and make to manage their vehicle speed. It is used during all the time.

This Model also useful when an accident occurs by passing an information to the corresponding user. When an accident occurs, the Crash Sensor in the vehicle will send an information and also intimates the GPS Module to gather current Location of the vehicle and sends it to the particular user via GSM Module.

# CHAPTER 2 LITERATURE SURVEY

**Charlotte Jacobe-de-Naurois, ChristopheBourdin, AncaStratulat”, “published on May 2019 in Science Direct Journal**

Charlotte Jacobe-de-Naurois and ChristopheBourdin, AncaStratulat specified that the principle of “DETECTION AND PREDICTION OF DRIVER DROWSINESS USING ARTIFICIAL NEURAL NETWORK MODELS”, is based on the continuous monitoring of the drivers drowsiness. This study aims to determine whether the standard sources of information used to detect drowsiness can also be used to predict when a given drowsiness level will be reached. In this Journal, the author uses the Deep Learning concepts of Neural Networks to predict the amount the driver’s drowsiness and also made the alert for the vehicle drivers when they have more drowsiness without their knowledge which helps to prevent the accidents during night travel. Moreover, we explore whether adding data such as

driving time and participant information improves the accuracy of the detection

and prediction of drowsiness.

#### Boddapati Venkata sai Padmaja, Venkata Ratnam Kolluru , Syam Sai Kota.

Boddapati, Ratnam Kolluru and Sai Kota in their research they have done an “IMPLEMENTATION OF VEHICLE MONITORING AND TRACKING SYSTEM USING NODE-MCU” using the IoT Concepts. This Journal is mainly concentrate on the monitoring of the Vehicle by using various sensors to predict the present condition of the vehicle. If any distraction had occurred in Vehicle, this module will able to identifies the location of the vehicle parts and sends the information about the status and current location of the vehicle to the particular user using the modules like NodeMCU, GSM SIM 900. This review results in the identification and transfer of the information about the status of vehicle.

#### [Sharmishta Desai](https://ieeexplore.ieee.org/author/37855762400), [Sourav Roy](https://ieeexplore.ieee.org/author/37086112979), [Brina Patel](https://ieeexplore.ieee.org/author/37086114897).

Sharmishta Desai and Sourav Roy, and Brina Patel has introduced the concept of VDFT Algorithm which is useful for the fast retrieval of Information from the Big Database which also react according to the number of records in the Database. In the era of Big Data where voluminous data is handled on a very large scale, traditional decision trees might be very time consuming and sometimes might even fail to work owing to its dataset size. Handling Big Data can also be a costly affair because of its high demand for memory and other hardware requirements. To the end of this paper, we have chosen a decision tree algorithm named Very Fast Decision Tree (VFDT) after comparing it with other decision tree algorithms like ID3 and C4.

**Arun Kumar.S, Madhu.S**

Arun Kumar and Madhu has published the journal on the topic of “A RESEARCH REVIEW ON AIRBAG IN AUTOMOBILE SAFETY SYSTEM” in which they explain about the working prototype of the Airbag. In this method, they explains that, the sensor would be placed at the front of the vehicle suppose if an accident has occurred then the sensor will intimate the system then the system will initiate the Airbag to be opened which prevents the people in the vehicle with less injury. This concept has been implemented mostly in all types of the car type vehicle.

# CHAPTER 3 SYSTEM ANALYSIS

### EXISTING SYSTEM

Existing System consist of only sensors to detect the distance between the previous vehicles or obstacles which always alerts the users even while passing the vehicles when the system works which irritates them always.

### PROPOSED SYSTEM

Our Proposed System consist of VFDT Algorithm which does not give alert always, it alerts when the speed of the vehicle is constantly high and the distance between vehicle decreases exponentially during Mist season. Our system involves LCD display which is placed at rear side of each vehicle which helps the upcoming vehicle to predict the distance between the current and prior vehicle and also they can able to manage the speed of vehicle according to it. Suppose an accident is occurred, then our system will gather the information about the current location of the vehicle and send an information to the corresponding registered users.

* + 1. Advantages
       1. Frequent calculation of distance and display it.
       2. Gives customer satisfaction.

### REQUIREMENT ANALYSIS AND SPECIFICAITON

The requirement engineering process of feasibility study, requirements elicitation and analysis, requirement specification, requirements validation and requirement management. Requirement elicitation and analysis is an iterative process that can

be represented as a spiral of activities, namely requirements discovery, requirements classification and organization, requirement negotiation and requirements documentation.

### INPUT REQUIREMENT

The input requirement at the base requires data from the environment using ultrasonic sensor to retrieve distance between the previous vehicle or any obstacle with the help of VFDT Algorithm. The input for the GPS Modules requires environment data that is the present location of the vehicle.

### OUTPUT REQUIREMENT

The output things necessary for these are display in the LCD Display for the purpose of the Vehicle Drivers and also the output will be transfer the collected information to the respective users with the help of GSM Module.

### FEASIBILITY STUDY

A feasibility study is carried out to select the best system that meets the performance requirements. The main aim of the feasibility study activity is to determine that it would be financially and technically feasible to develop the product.

### TECHNICAL FEASIBILITY

This is concerned with specifying the software will successfully satisfy the user requirement. Open source and business-friendly and it is truly cross platform, easily deployed and highly extensible.

### ECONOMIC FEASIBILITY

Economic analysis is the most frequently used technique for evaluating the

effectiveness of a proposed system. The enhancement of the existing system doesn’t incur any kind of increase in the expenses. Programming Language for Web-App development is open source and readily available for all users. Since, the project is runned in the Visual Studio Code and Apache Tomcat, hence it is cost efficient.

### MINIMUM HARDWARE REQUIREMENTS

|  |  |
| --- | --- |
| Processor | Core i3, 2.4 GHz |
| Hard disk | 500 GB |
| RAM | 4GB |
| Monitor / LCD Display | 14/15 inches Color |
| Sensors | Ultrasonic Sensor  Crash Sensor |
| Modules | Arduino UNO Board  GSM Module  GPS Module |

* 1. **SOFTWARE REQUIEMENTS**

|  |  |  |  |
| --- | --- | --- | --- |
| Development Environment | Embedded System | | |
| Software Used | Arduino | | |
| Algorithm Used | VFDT Algorithm | | |
| Technology Used | | | Cloud |

### SOFTWARE SPECIFICATION

#### EMBEDDED SYSTEM

An **Embedded System** is a system that has software embedded into computer-hardware, which makes a system dedicated for a variety of application or specific part of an application or product or part of a larger system.

An embedded system can be a small independent system or a large combinational system. It is a microcontroller-based control system used to perform a specific task of operation.

An embedded system is a combination of three major components:

* **Hardware:** Hardware is physically used component that is physically connected with an embedded system. It comprises of microcontroller based integrated circuit, power supply, LCD display etc.
* **Application software:** Application software allows the user to perform varieties of application to be run on an embedded system by changing the code installed in an embedded system.
* **Real Time Operating system (RTOS):** RTOS supervises the way an embedded system work. It act as an interface between hardware and application software which supervises the application software and provide mechanism to let the processor run on the basis of scheduling for controlling the effect of latencies.

Embedded C is **most popular programming language in software field for developing electronic gadgets**. Embedded C programming plays a key role in

performing the specific function by the processor.

1. **ARDUINO SOFTWARE IDE**

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board. The **Arduino Integrated Development Environment (**[**IDE**](https://en.wikipedia.org/wiki/Integrated_development_environment)**)** is a [cross-platform](https://en.wikipedia.org/wiki/Cross-platform) application (for [Windows](https://en.wikipedia.org/wiki/Windows), [macOS](https://en.wikipedia.org/wiki/MacOS), [Linux](https://en.wikipedia.org/wiki/Linux)) that is written in functions from [C](https://en.wikipedia.org/wiki/C_(programming_language)) and [C++](https://en.wikipedia.org/wiki/C%2B%2B_(programming_language)). It is used to write and upload programs to [Arduino](https://en.wikipedia.org/wiki/Arduino) compatible boards, but also, with the help of third-party cores, other vendor development boards. The source code for the IDE is released under the [GNU General Public License](https://en.wikipedia.org/wiki/GNU_General_Public_License), version 2.The Arduino IDE supports the languages [C](https://en.wikipedia.org/wiki/C_(programming_language)) and [C++](https://en.wikipedia.org/wiki/C%2B%2B) using special rules of code structuring. The Arduino IDE supplies a [software library](https://en.wikipedia.org/wiki/Software_library) from the [Wiring](https://en.wikipedia.org/wiki/Wiring_(development_platform)) 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](https://en.wikipedia.org/wiki/Cyclic_executive) program with the [GNU toolchain](https://en.wikipedia.org/wiki/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.

1. **VFDT Algorithm**

VFDT stands for “Very Fast Decision Tree” Algorithm.

A Hoeffding tree (**VFDT**) is an incremental, anytime decision tree induction **algorithm** that is capable of learning from massive data streams, assuming that the distribution generating examples does not change over time. Hoeffding trees exploit the fact that a small sample can often be enough to choose an optimal splitting attribute.

We have also proposed an algorithm for implementing VFDT on a Distributed Environment called Hadoop. This implementation can form a base for a large number of applications for handling Big Data. We have replaced the serial execution of VFDT algorithm by a series of Map and Reduce functions. We have also conducted an extensive analysis on various datasets which have proved our proposed algorithm to be more efficient in terms of time compared to the other existing decision tree models.

Proposes to implement and use the very fast decision tree (VFDT) algorithm can effectively perform a test-and-train process with a limited segment of data. In contrast with traditional algorithms, the VFDT does not require that the full dataset be read as part of the learning process thus reducing time. As a preemptive approach to minimizing the impacts of imperfect data streams, a data cache and missing-data-guessing mechanism called the auxiliary reconciliation control (ARC) is proposed to function as a within VFDT.

# CHAPTER 4 SYSTEM DESIGN

### ER DIAGRAM FOR DIABETES GUIDANCE SYSTEM

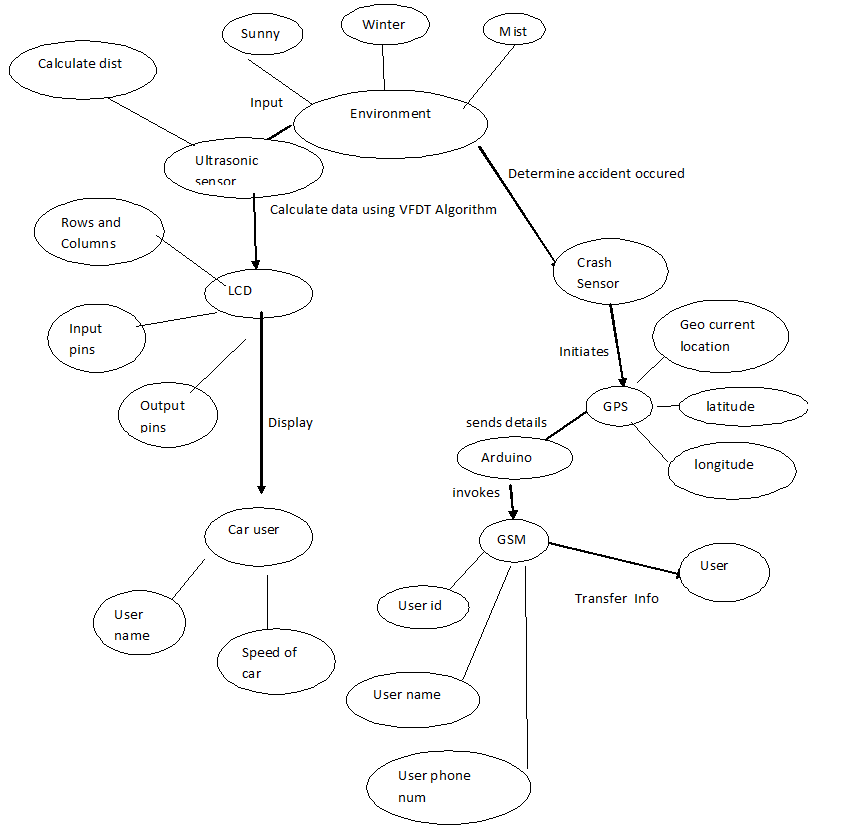
****

Fig 4.1 ER diagram

### DATA DICTIONARY

A data dictionary, or [metadata repository](https://en.wikipedia.org/wiki/Metadata_repository), as defined in the *IBM Dictionary of Computing*, is a "centralized repository of information about data such as meaning, relationships to other data, origin, usage, and format". [*Oracle*](https://en.wikipedia.org/wiki/Oracle_Corporation) defines it as a collection of tables with metadata. The term can have one of several closely related meanings pertaining to [databases](https://en.wikipedia.org/wiki/Database) and [database management systems](https://en.wikipedia.org/wiki/Database_management_system) (DBMS):

* A [document](https://en.wikipedia.org/wiki/Document) describing a database or collection of databases
* An integral [component](https://en.wikipedia.org/wiki/Software_component) of a [DBMS](https://en.wikipedia.org/wiki/Database_management_system) that is required to determine its structure
* A piece of [middleware](https://en.wikipedia.org/wiki/Middleware) that extends or supplants the native data dictionary of a DBMS

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **NAME** | **DATATYPE** | **VALUES** |
| 1 | Lidar Sensor | Integer | 87 |
| 2 | Crash Sensor | Integer | 0 or 1 |
| 3 | LCD Display | String | Message |
| 4 | Buzzer | Boolean | 0 or 1 |
| 5 | GPS | Latitude | 19.25484 N |
|  |  | Longitude | 68.25471 E |
| 6 | GSM | String | Message |

### DATA FLOW DIAGRAM

A picture is worth a thousand words. A Data Flow Diagram (DFD) is traditional visual representation of the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or combination of both. It shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communications tool between a systems analyst and any person who plays a part in the system that acts as the starting point for redesigning a system.

It is usually beginning with a context diagram as the level 0 of DFD diagram, a simple representation of the whole system. To elaborate further from that, we drill down to a level 1 diagram with lower level functions decomposed from the major

functions of the system. This could continue to evolve to become a level 2 diagram when further analysis is required. Progression to level 3, 4 and so on is possible but anything beyond level 3 is not very common. Please bear in mind that the level of details for decomposing function really depending on the complexity that function.

### UML DIAGRAMS

UML stands for Unified Modeling Language. It’s a rich language to model software solutions, application structures, system behavior and business processes. There are 14 UML diagram types to help you model these behaviors. Unified Modeling Language™ (UML®) is a standard visual modeling language intended to be used for

* modeling business and similar processes,
* analysis, design, and implementation of software-based systems

UML is a common language for business analysts, software architects and developers used to describe, specify, design, and document existing or new business processes, structure and behavior of artifacts of software systems.

Specification explained that process:

* provides guidance as to the order of a team’s activities,
* specifies what artifacts should be developed,
* directs the tasks of individual developers and the team as a whole, and
* offers criteria for monitoring and measuring a project’s products and activities.

UML is intentionally process independent and could be applied in the context of different processes. Still, it is most suitable for use case driven, iterative and incremental development processes. An example of such process is Rational Unified Process (RUP).UML is not complete, and it is not completely visual. Given some UML diagram, we can't be sure to understand depicted part or behavior of the system from the diagram alone. Some information could be

intentionally omitted from the diagram, some information represented on the diagram could have different interpretations, and some concepts of UML have no graphical notation at all, so there is no way to depict those on diagrams. For example, semantics of multiplicity of actors and multiplicity of use cases on use case diagrams is not defined precisely in the UML specification and could mean either concurrent or successive usage of use cases.

Name of an abstract classifier is shown in italics while final classifier has no specific graphical notation, so there is no way to determine whether classifier is final or not from the diagram.

#### List of UML Diagram Types

So, what are the different UML diagram types? There are two main categories; structure diagrams and behavioral diagrams. Click on the links to learn more about a specific diagram type.

#### Structure Diagrams

Structure diagrams show the things in the modeled system. In a more technical term, they show different objects in a system. Behavioral diagrams show what should happen in a system. They describe how the objects interact with each other to create a functioning system.

#### Class Diagram

Class diagrams are the main building block of any object-oriented solution. It shows the classes in a system, attributes, and operations of each class and the relationship between each class. In most modeling tools, a class has three parts. Name at the top, attributes in the middle and operations or methods at the bottom.

In a large system with many related classes, classes are grouped together to create class diagrams. Different relationships between classes are shown by different types of arrows.

#### Component Diagram

A component diagram displays the structural relationship of components of a software system. These are mostly used when working with complex systems with many components. Components communicate with each other using interfaces. The interfaces are linked using connectors. The image below shows a component diagram.

#### Deployment Diagram

A deployment diagram shows the hardware of your system and the software in that hardware. Deployment diagrams are useful when your software solution is deployed across multiple machines with each having a unique configuration. Below is an example deployment diagram.

#### Package Diagram

As the name suggests, a package diagram shows the dependencies between different packages in a system. Check out this wiki article to learn more about the dependencies and elements found in package diagrams.

#### Composite Structure Diagram

Composite structure diagrams are used to show the internal structure of a class. For a detailed explanation of composite structure diagrams, click here.

#### Use Case Diagram

As the most known diagram type of the behavioral UML diagrams, use case diagrams give a graphic overview of the actors involved in a system, different functions needed by those actors and how these different functions interact.

It’s a great starting point for any project discussion because you can easily identify the main actors involved and the main processes of the system. You can create use case diagrams using our tool and/or get started instantly using our use case templates.

#### Activity Diagram

Activity diagrams represent workflows in a graphical way. They can be used to describe the business workflow or the operational workflow of any component in a system. Sometimes activity diagrams are used as an alternative to State machine diagrams. Check out this wiki article to learn about symbols and usage of activity diagrams.

#### Sequence Diagram

Sequence diagrams in UML show how objects interact with each other and the order those interactions occur. It’s important to note that they show the interactions for a scenario. The processes are represented vertically, and interactions are shown as arrows. This article explains the purpose and the basics of Sequence diagrams. Also, check out this complete Sequence Diagram Tutorial to learn more about sequence diagrams. You can also instantly start drawing using our sequence diagram templates.

**USECASE DIAGRAM**

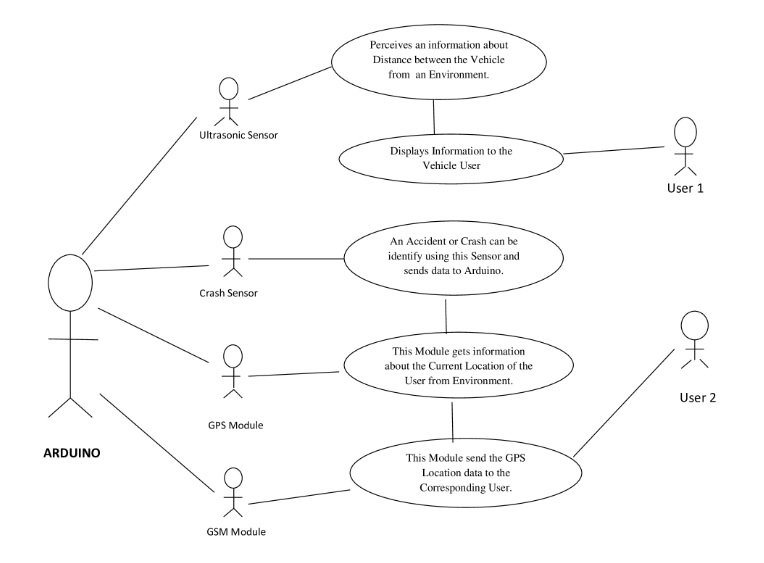
****

Fig 4.4 Use case diagram

**CLASS DIAGRAM**

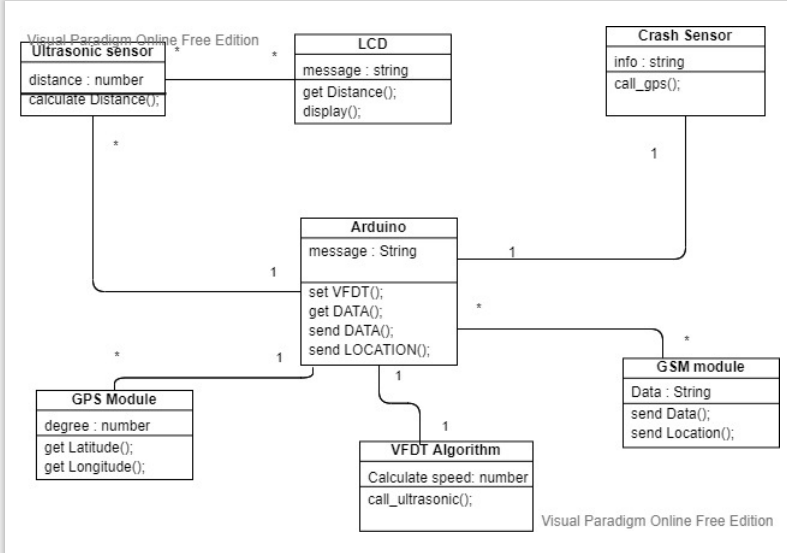
****

Fig 4.5 Class diagram

**SEQUENCE DIAGRAM**

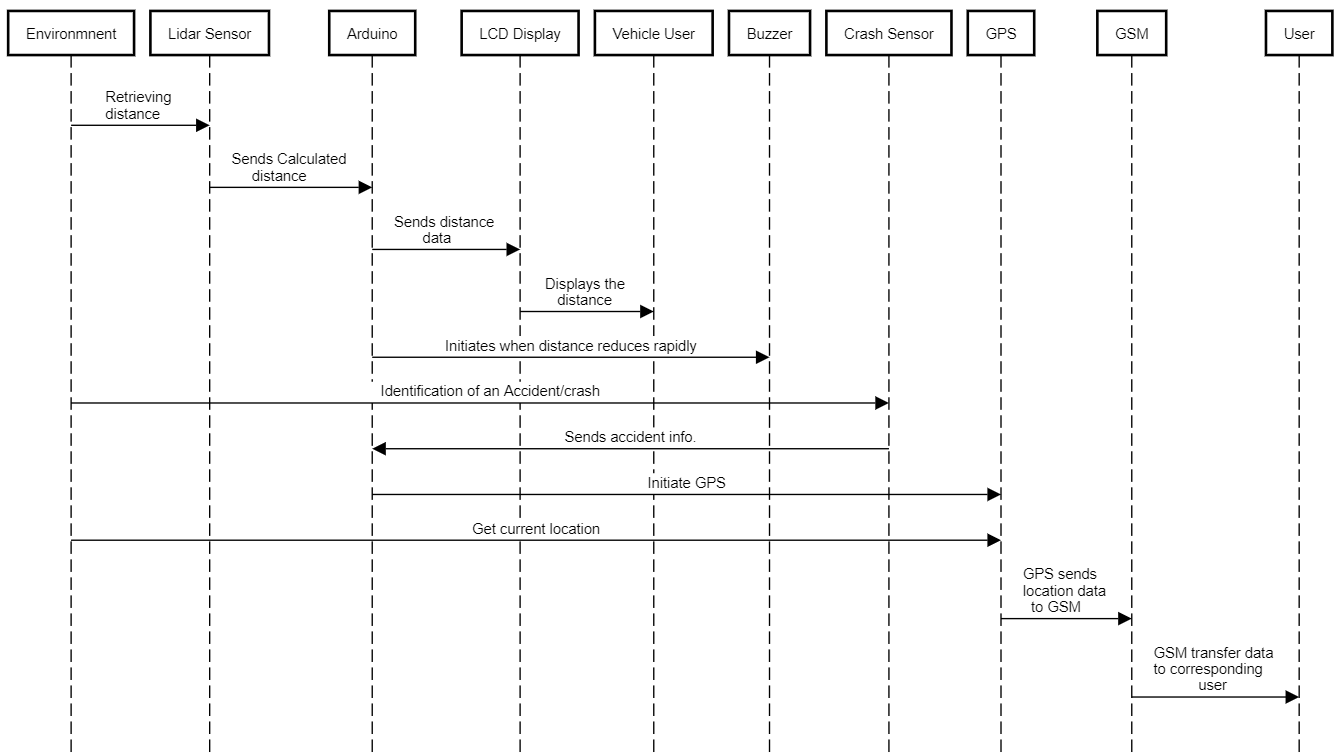


Fig 4.6 Sequence diagram

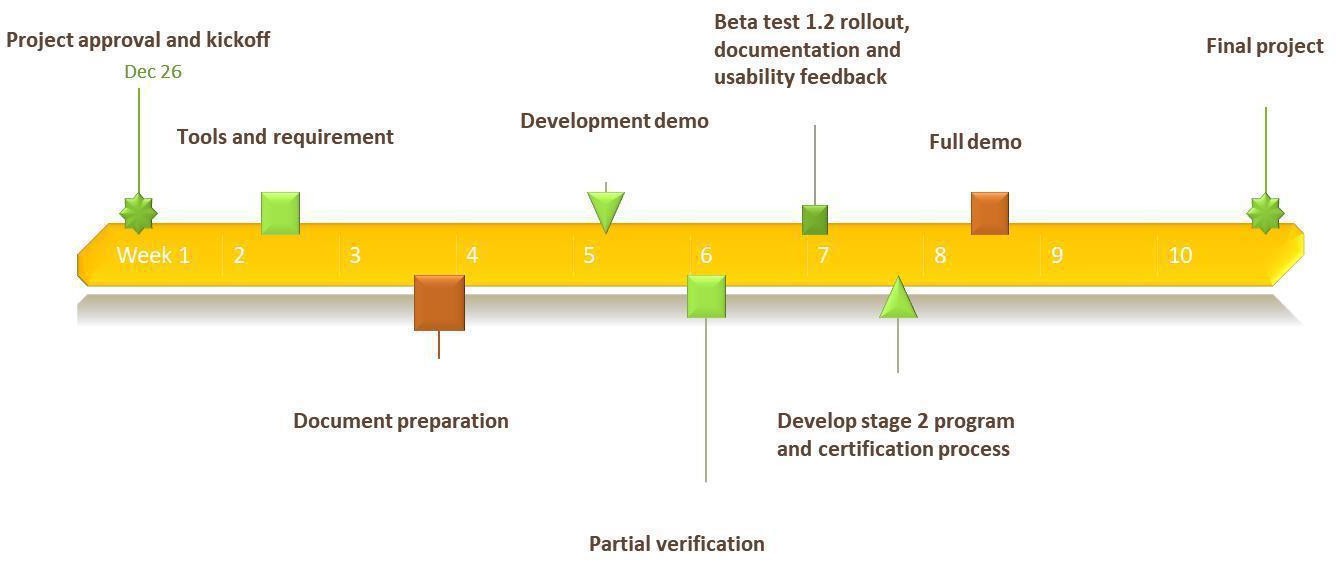


Fig 4.7 Timeline for diabetes guidance system

### CHAPTER 5 ARCHITECTURE

#### SYSTEM ARCHITECTURE

System architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system.

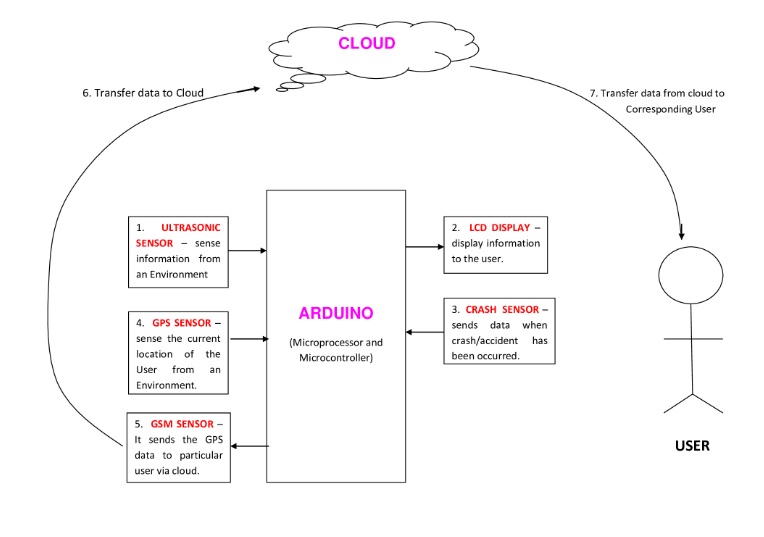


Fig 5.1 System Architecture

#### SYSTEM MODULE

The Accident Prevention System contains three modules functions namely:

* + 1. Arduino UNO Board
    2. Ultrasonic Sensor, Crash Sensor
    3. Global Positioning System (GPS)
    4. Global System for Mobile Communication (GSM Module)
    5. Liquid Crystal Display (LCD)

#### MODULES EXPLAINATION:

1. **Arduino UNO Board:**

#### The Arduino Uno is a ****microcontroller board**** based on the ATmega328. It has 20 digital input/output pins (of which 6 can be used as PWM outputs and 6 can be used as analog inputs), a 16 MHz resonator, a USB connection, a power jack, an in-circuit system programming (ICSP) header, and a reset button.

#### The Arduino Uno can be powered by USB cable or directly supplying 9-12v from the barrel jack. The circuitry operates at 5v dc which in case input more than that is regulated with the help of 7805 voltage regulator. The [7805 voltage regulator ic](https://eeeproject.com/7805-voltage-regulator-short-description-2/) is used regulate the voltage supplied to the arduino board and manage it through processor and other elements.

#### Arduino Uno can detect the surroundings from the input. Here the input is a variety of sensors and these can affect its surroundings through controlling motors, lights, other actuators, etc. The ATmega328 microcontroller on the Arduino board can be programmed with the help of an Arduino programming language and the IDE (Integrated Development Environment). [**Arduino projects**](https://www.elprocus.com/arduino-boards-electronics-and-electrical-engineering-projects/) can communicate by software while running on a PC.

#### (ii) Ultrasonic and Crash Sensor

Ultrasonic sensors measure distance by using ultrasonic waves.  
The sensor head emits an ultrasonic wave and receives the wave reflected back from the target. Ultrasonic Sensors measure the distance to the target by measuring the time between the emission and reception.

The distance can be calculated with the following formula:

### Distance L = 1/2 × T × C

### Crash sensors collect the data necessary to make decision when an accident occurs. Crash sensors measure how quickly a vehicle slows down in a frontal crash or accelerates to the side in a side-impact crash. Some vehicles are equipped with a sensing system designed to detect the onset of a rollover crash. Frontal crash sensors may be located in the front of the vehicle near the engine, in the passenger compartment, or sometimes in the electronic control unit (ECU).

**(iii) LCD DISPLAY / MONITOR**

Arduino LCD Display The LCD (Liquid Crystal Display) is a type of display that uses the liquid crystals for its operation. Here, we will accept the serial input from the computer and upload the sketch to the Arduino. The characters will be displayed on the LCD.

### LCDs allowed displays to be much thinner than cathode ray tube ([CRT](https://whatis.techtarget.com/definition/cathode-ray-tube-CRT)) technology. LCDs consume much less power than LED.  The liquid crystals in an LCD produces an image using a backlight.

### Overall, it is useful to display the data analyzed by the Sensor.

### GPS Module

### GPS (Global Positioning System) is a device that is capable of receiving information from [GNSS](https://en.wikipedia.org/wiki/GNSS) satellites and then to calculate the device's geographical position. It is one of the [global navigation satellite systems](https://en.wikipedia.org/wiki/Satellite_navigation) (GNSS) that provides [geolocation](https://en.wikipedia.org/wiki/Geolocation) and [time information](https://en.wikipedia.org/wiki/Time_transfer) to a [GPS receiver](https://en.wikipedia.org/wiki/Satellite_navigation_device) anywhere on or near the Earth where there is an unobstructed line of sight to four or more GPS satellites. The GPS does not require the user to transmit any data, and it operates independently of any telephonic or internet reception, though these technologies can enhance the usefulness of the GPS positioning information.

### GSM Module

### A**GSM module or a GPRS module is a chip or circuit that will be used to establish communication** between a mobile device or a computing machine and a **GSM or GPRS system**. This module is useful to send the message to the Registered number (which should be configure in GSM Module)

#### OVERVIEW OF AN ARCHITECTURE

#### The Arduino plays a major role in this project which works based on the IoT Domain. The Arduino is the set of Microprocessor and Microcontroller. The Microcontroller is used to access and control of the entire operation in the system. The Arduino works with the help of the power supply of minimum 9V.

#### 

#### The Ultrasonic Sensor which is connected to an Arduino acts as an Input digital value. This sensor will always sense the distance between the current and previous vehicle from an Environment. The calculated value of distance must be send to the LCD Display via Arduino. The LCD Display continuously displays the distance between the vehicle to the driver.

#### This system will gives an alert via buzzer to the driver, only when the speed

#### of the vehicle is high and when the distance between the vehicle decreases exponentially. This can be detected with the help of VFDT (Very Fast Decision Tree) Algorithm. The LCD Display also placed at rear side of each vehicle which will displays the distance between the current vehicle and previous vehicle. Thus it helps the upcoming vehicle to predict the distance and control their vehicle.

#### 

#### Since the ultrasonic sensor covers the distance only upto 11meter. We have used the Lidar (Light Detection and Rangin) sensor which diverse the range upto 400 meters. It is a method for determining [ranges](https://en.wikipedia.org/wiki/Ranging) (variable distance) by targeting an object with a [laser](https://en.wikipedia.org/wiki/Laser) and measuring the time for the reflected light to return to the receiver.

#### 

#### 

#### The Crash Sensor connected to an Arduino which should be placed at the front of the vehicle. It will be activate when an accident or crash has occurred. When it is activated, it sends information to the GPS (Global Positioning System) Module via an Arduino. The GPS Module will get the current Location of the vehicle and pass the latitude and longitude of location to the GSM Module.

#### 

#### The GSM (Global System for Mobile Communication) which is used to send the retrieved information from the GPS about Location to the corresponding user.

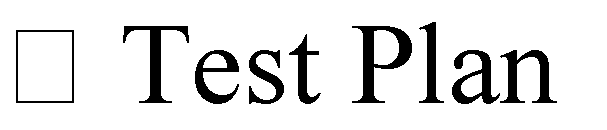
**CHAPTER 6 TESTING**

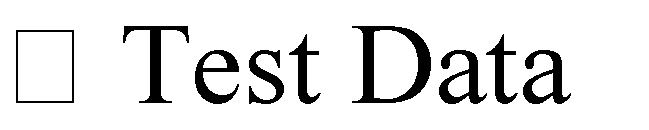
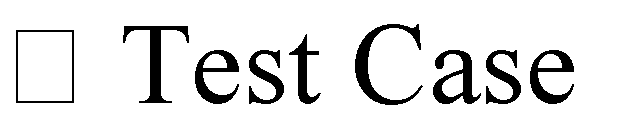
#### SYSTEM TESTING

The testing approach document is designed for Information and Technology Services’ upgrades to PeopleSoft. The document contains an overview of the testing activities to be performed when an upgrade or enhancement is made, or a module is added to an existing application. The emphasis is on testing critical business processes, while minimizing the time necessary for testing while also mitigating risks. It’s important to note that reducing the amount of testing done in an upgrade increases the potential for problems after go-live. Management will need to determine how much risk is acceptable on an upgrade by upgrade basis.

System testing is simply testing the system as a whole; it gets all the integrated modules of the various components from the integration testing phase and combines all the different parts into a system which is then tested. Testing is then done on the system as all the parts are now integrated into one system the testing phase will now have to be done on the system to check and remove any errors or bugs. In the system testing process the system will be checked not only for errors but also to see if the system does what was intended, the system functionality and if it is what the end user expected.

There are various tests that need to be conducted again in the system testing which include:





If the integration stage was done accurately then most of the test plan and test cases would already have been done and simple testing would only have to be done in order to ensure there are no bugs because this will be the final product. As in the integration stage, the above steps would need to be re-done as now we have

integrated all modules into one system, so we have to check if this runs OK and that no errors are produced because all the modules are in one system.

#### Unit Testing

In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures are tested to determine if they are fit for use. In object-oriented programming, a unit is often an entire interface, such as a class, but could be an individual method. Unit tests are short code fragments created by programmers or occasionally by white box testers during the development process. Ideally, each test case is independent from the others. Substitutes such as method stubs, mock objects, fakes, and test harnesses can be used to assist testing a module in isolation. Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended.

#### TEST CASES

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Cas e Id | Test Cases | Priorit y | Test Case  Description | Expected Results | Actual Results | Pass/Fai l |
| TU0 1 | Information Retrieval | A | Gather information from an environment.  Using Lidar sensor | Passes laser on the target and infromation | Information gathered | Pass |
| TU0 2 | Calculate the distance | A | The sensor should calculate the distance between the current and previous vehicle | Get the calculated distance | Distance Calculated | Pass |
| TU0 3 | Crash Sensor | A | The Crash sensor will used to intimate the user when the accident has been occured | Notification must be done when crash or accident done | The driver receives the message | Pass |
| TU0 4 | Alerts the user | A | When the speed of vehicle is high and distance between vehicle decreases exponentially. | The model should alerts the user | The user has been alerted | Pass |

Table 6.1 Test Cases for Lidar and Crash Sensor

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Case Id | Test Cases | Priorit y | Test Case Description | Expected Results | Actual Results | Pass/Fa il |
| TE0 1 | GPS Module invoke | A | The GPS Module should be invoke by an Arduino when the crash sensor intiates. | The initiation process of GPS | Initiation done successfully | Pass |
| TE0 2 | Retrieve Location | A | The GPS Module should sense the current location with latitude and longitude from an environment | The sense of current location of user | The location has been sensed | Pass |
| TE03 | Information should be shared | A | The GPS Module should send the location which has got to the GSM Module via Arduino. | Transfer of the location details | The Location send has done` | Pass |

Table 6.2 Test Cases for GPS Module

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Case Id | Test Cases | Priority | Test Case  Description | Expected Results | Actual Results | Pass/Fail |
| TC01 | Gets data from Arduino | A | The GSM should gets the information from Arduino which has been sent by GPS Module. | GSM should get location info | The info has been retrieved | Pass |
| TC02 | Sends data to user | A | The GSM should send the data to the corresponding user which has been registered. | Message should send to user. | The message has been sent successfully. | Pass |

Table 6.3 Test Cases for GSM Module

## CHAPTER 7

**CONCLUSION AND FUTURE ENHANCEMENT**

#### CONCLUSION

This guidance system provides the required and useful information about the distance between vehicle to the present vehicle driver during mist season and to upcoming vehicle driver during everytime via the LCD Display which reduces the number of accidents. This proposed model also alerts the driver when the distance between vehicle reduced exponentially or rapidly. Suppose an accident or crash has been occurred our system will gets information about location from an environment and transfer it to the corresponding user via GSM.

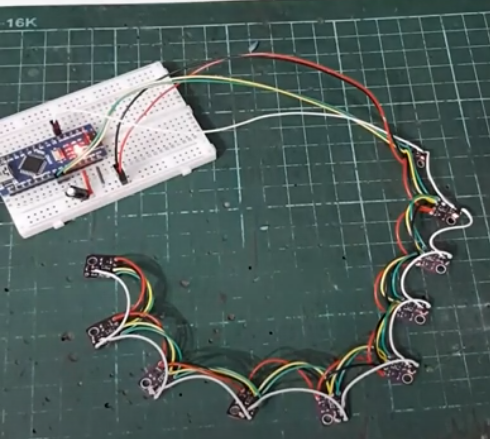
#### FUTURE ENHANCEMENT

In the further enhancement we are planning transfer the data/information to the corresponding user via inbuilt Wi-Fi after implementing more security measures while transferring data via cloud. In addition to that we have planned to observe the changes of speed of the vehicle via GPS which reduce the number of components.

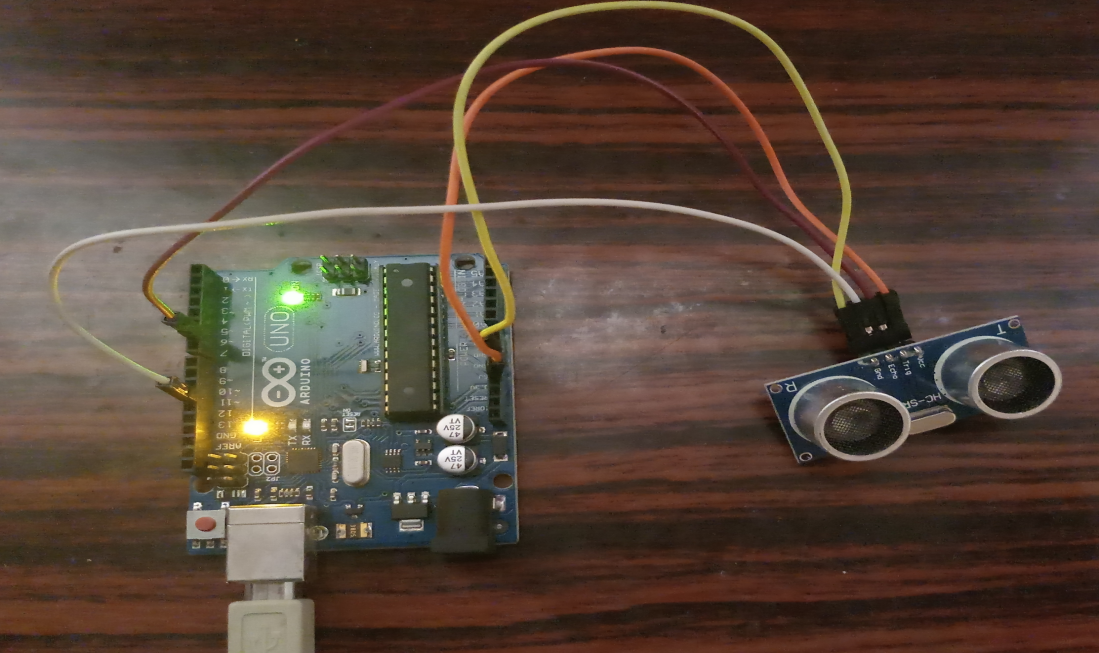
**APPENDICES**

**A 1. SAMPLE SCREENS**

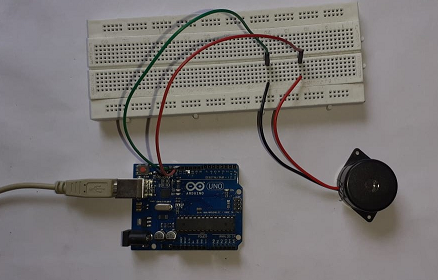
1. **Lidar Sensor**

****

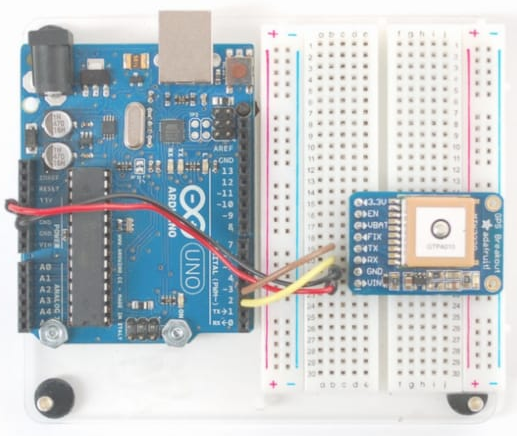
1. **Ultrasonic Sensor**



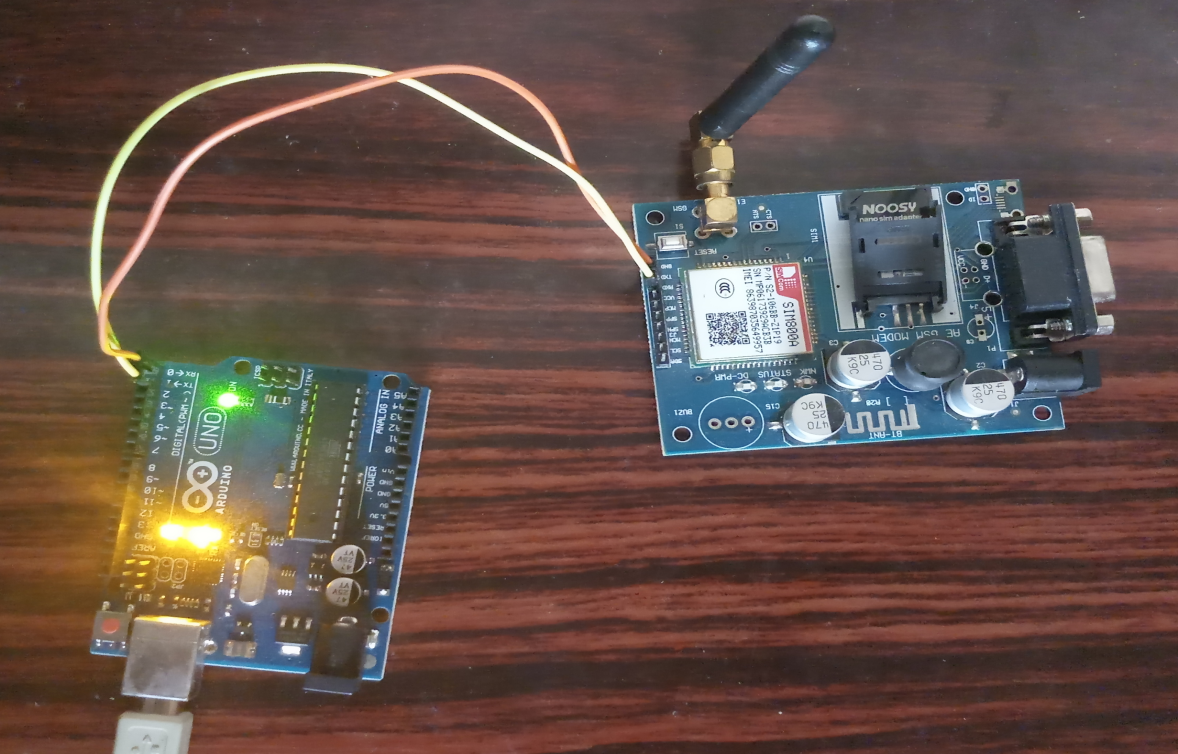
1. **Buzzer**



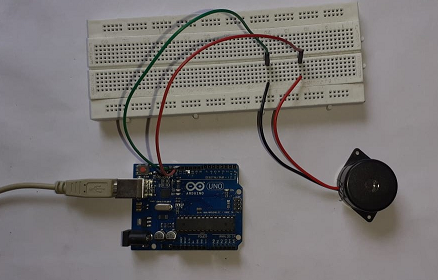
1. **GPS Module**



**(v) GSM Module**



1. **BUZZER**



### A2. SAMPLE CODE

1. **Ultrasonic Sensor with speed calculation**

#include <SoftwareSerial.h>

int trigger\_pin = 3;

int echo\_pin = 4;

int buzzer\_pin = 5;

int time;

int predef; //pre defined magnitude of speed, which a vehicle could manage to break with ease

float distance1,distance2;

float caution1,caution2,caution3; //caution for informing required condition as per our algorithm

//here the caution defines respective speed which is (SPEED=DISTANCE/TIME)

int distanceCm,distanceInch;

void setup ( ) {

Serial.begin (9600);

pinMode (trigger\_pin, OUTPUT);

pinMode (echo\_pin, INPUT);

pinMode (buzzer\_pin, OUTPUT);

}

void loop ( ) {

digitalWrite (trigger\_pin, HIGH);

delayMicroseconds (10);

digitalWrite (trigger\_pin, LOW);

time = pulseIn (echo\_pin, HIGH);

distance1 = (time \* 0.034) / 2;

Serial.println("Distance = ");

Serial.println(distance);

delay(1000)

digitalWrite (trigger\_pin, HIGH);

delayMicroseconds (10);

digitalWrite (trigger\_pin, LOW);

time = pulseIn (echo\_pin, HIGH);

distance2 = (time \* 0.034) / 2;

Serial.println("Distance = ");

Serial.println(distance);

delay(1000)

digitalWrite (trigger\_pin, HIGH);

delayMicroseconds (10);

digitalWrite (trigger\_pin, LOW);

time = pulseIn (echo\_pin, HIGH);

distance3 = (time \* 0.034) / 2;

Serial.println("Distance = ");

Serial.println(distance);

delay(1000)

digitalWrite (trigger\_pin, HIGH);

delayMicroseconds (10);

digitalWrite (trigger\_pin, LOW);

time = pulseIn (echo\_pin, HIGH);

distance4 = (time \* 0.034) / 2;

Serial.println("Distance = ");

Serial.println(distance);

delay(1000)

caution1=distance2-distance1;

caution2=distance3-distance2;

caution1=distance4-distance3;

if (caution1<predef && caution2<predef && caution3<predef )

{

digitalWrite (buzzer\_pin, HIGH);

delay (2000);

}

else {

|  |
| --- |
|  |
|  |  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

digitalWrite (buzzer\_pin, LOW);

delay (2000);

}

}

1. **LIDAR SENSOR**

#define NUM\_OF\_SENSORS 9

VL53L0X sensor[NUM\_OF\_SENSORS];

#define STARTING\_ADDRESS 0x53

uint8\_t address[NUM\_OF\_SENSORS];

#define STARTING\_CHIP\_XSHUTN\_PIN 2

uint32\_t readDataTimer;

#define READ\_DATA\_DURATION 24

uint16\_t distance[NUM\_OF\_SENSORS];

#define DEBUG\_PRINT\_TO\_SERIAL

#define MAX\_RANGE 1000

void setup()

{

Serial.begin(115200);

Serial.println("Starting sketch - Fuzzy Radar - Raw Data Serial Output.");

Wire.begin();

//Initialize the I2C address array.

uint8\_t addressOffset = 0; //Avoid using default address 0x52 as new address.

for (uint8\_t index = 0; index < NUM\_OF\_SENSORS; index++)

{

if ((STARTING\_ADDRESS + index) == 0x52) addressOffset = 1;

address[index] = STARTING\_ADDRESS + index + addressOffset;

}

/\* Chip shutdown in now controlled by XSHUTN, using a NMOS inverter.

XSHUTN is not pulled up nor pulled down.

Only the first chip is controlled by arduino pin

\*/

Serial.println("Set chip 0 into reset mode.");

pinMode(STARTING\_CHIP\_XSHUTN\_PIN, OUTPUT);

digitalWrite(STARTING\_CHIP\_XSHUTN\_PIN, HIGH);//set chip 0 into reset mode. All subsequent chips should go into reset mode as well.

Serial.println("All status LEDs should be off.");

//delay(2000);

Serial.println("Now configuring the sensors. LED should light up one by one.");

//delay(1000);

for (uint8\_t index = 0; index < NUM\_OF\_SENSORS; index++)

{

Serial.print("Configuring chip ");

Serial.println(index);

//Bring one chip out of reset mode

if (index == 0)

{

//First chip

digitalWrite(STARTING\_CHIP\_XSHUTN\_PIN, LOW);//Enable first chip

}

else

{

//Subsequent chips, index = 1,2,3,4...

sensor[index - 1].setGPIO(LOW); //Enable chips other than the first chip

}

delay(5);//Required for VL53L0X firmware booting.

Serial.print(" - Reset I2C address to ");

Serial.println(address[index]);

sensor[index].setAddress(address[index]);

Serial.println(" - Initialize the sensor.");

sensor[index].init();

sensor[index].setTimeout(500);

//delay(1000);

}

Serial.println("Radar array configuration completed.");

//Start continuous reading mode.

for (uint8\_t index = 0; index < NUM\_OF\_SENSORS; index++)

{

Serial.print("Start continuous ranging mode for chip ");

Serial.println(index);

sensor[index].startContinuous(20);

}

}

void loop()

{

if (millis() - readDataTimer >= READ\_DATA\_DURATION)

{

//The time required for new data is around 21-22 ms.

readDataTimer = millis();

readData();

#ifdef DEBUG\_PRINT\_TO\_SERIAL

printDataToSerial();

#endif

}

}

void readData()

{

for (uint8\_t index = 0; index < NUM\_OF\_SENSORS; index++)

{

distance[index] = sensor[index].readReg16Bit(sensor[index].RESULT\_RANGE\_STATUS + 10);

if (distance[index] > MAX\_RANGE) distance[index] = 0;

}

}

void printDataToSerial()

{

for (uint8\_t index = 0; index < NUM\_OF\_SENSORS; index++)

{

if (distance[index] != 0)

{

Serial.print((distance[index] < 10 ? "0" : ""));

Serial.print((distance[index] < 100 ? "0" : ""));

Serial.print((distance[index] < 1000 ? "0" : ""));

Serial.print(distance[index]);

}

else

{

Serial.print("----");

}

Serial.print(" ");

}

Serial.println();

}

**(iii) LCD DISPLAY**

#include<LiquidCrystal.h>

LiquidCrystal lcd(0,1,8,9,10,11);

const int pingPin=3;

const int echoPin=4;

long duration;

int distanceCm,distanceInch;

void setup() {

// put your setup code here, to run once:

lcd.begin(16,2);

pinMode(pingPin,OUTPUT);

pinMode(echoPin,INPUT);

}

void loop() {

// put your main code here, to run repeatedly:

digitalWrite(pingPin,LOW);

delayMicroseconds(2);

digitalWrite(pingPin,HIGH);

delayMicroseconds(10);

digitalWrite(pingPin,LOW);

duration=pulseIn(echoPin,HIGH);

distanceCm=duration\*0.034/2;

distanceInch=duration\*0.0133/2;

lcd.setCursor(0,0);

lcd.print("Distance ");

lcd.print(distanceCm);

lcd.print("cm");

delay(1000);

lcd.setCursor(0,1);

lcd.print("Distance ");

lcd.print(distanceInch);

lcd.print("inch");

}

**(iv) GPS with GSM**

#include <SoftwareSerial.h>

#include <TinyGPS.h>

int state = 0;

const int pin = 5;

float gpslat, gpslon;

TinyGPS gps;

SoftwareSerial sgps(11, 12); //ports

SoftwareSerial sgsm(9, 10); //ports

void setup()

{

sgsm.begin(9600);

Serial.begin(9600);

sgps.begin(9600);

delay(100);

}

void loop()

{

sgps.listen();

while (sgps.available())

{

int c = sgps.read();

if (gps.encode(c))

{

gps.f\_get\_position(&gpslat, &gpslon);

}

}

if (Serial.available()>0) {

switch('s')

{

case 's':

sgsm.listen();

sgsm.print("\r");

delay(1000);

sgsm.println("AT+CMGF=1");

delay(1000);

sgsm.println("AT+CMGS=\"+916382389xxx\"\r");

delay(1000);

sgsm.println("Emergency!!! The vehicle with vehicle number xxxx met with an accident");

sgsm.println("The location of the accident is as follows");

sgsm.print("Latitude :");

sgsm.println(gpslat, 6);

sgsm.print("Longitude:");

sgsm.println(gpslon, 6);

sgsm.println("AT+CMGS=\"+91101\"\r");

sgsm.println("Emergency!!! The vehicle with vehicle number xxxx met with an accident. Please send the necessary ambulance support to the following location ASAP");

sgsm.print("Latitude :");

sgsm.println(gpslat, 6);

sgsm.print("Longitude:");

sgsm.println(gpslon, 6);

delay(1000);

sgsm.println((char)26);

delay(10000);

break

case 'r':

sgsm.println("AT+CNMI=2,2,0,0,0");

delay(1000);

break;

}

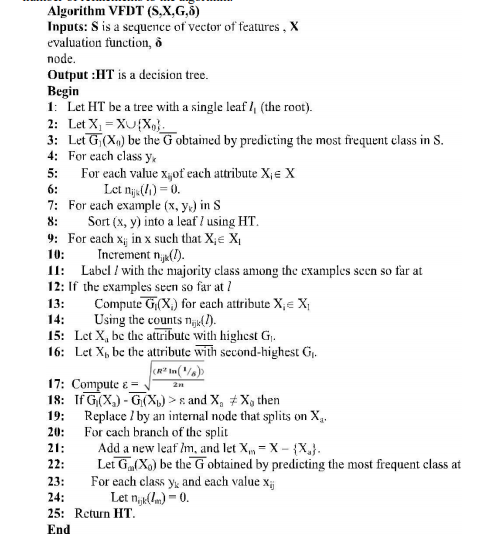
if (sgsm.available()>0) {

Serial.weite(mySerial.read());

}

}

1. **VFDT ALGORITHM**

****

### REFERENCES

# Charlotte Jacobe-de-Naurois, ChristopheBourdin, AncaStratulat”, “DETECTION AND PREDICTION OF DRIVER DROWSINESS USING ARTIFICIAL NEURAL NETWORK MODELS”, published on May 2019 in Science Direct Journal. <https://ieeexplore.ieee.org/document/8391218>

1. Boddapati Venkata sai Padmaja, Venkata Ratnam Kolluru , Syam Sai Kota, “IOT BASED IMPLEMENTATION OF VEHICLE MONITORING AND TRACKING SYSTEM USING NODE MCU”, published on April 2019 in International Journal of Innovative Technology and Exploring Engineering (IJITEE).
2. B Devi, S Sai bavatharini, “VEHICULAR SENSING AND AUTOMATIC BRAKING FOR ACCIDENT AVOIDANCE ON THE JUNCTIONS AND BLIND CURVES”, published on July 2018 in International Journal of Pure and Applied Mathematics
3. A.Anusha “Vehicle Tracking and Monitoring System to Enhance the Safety and Security Driving Using IoT” 2017 International Conference on Recent Trends in Electrical, Electronics and Computing Technologies (ICRTEECT), July 2017
4. [Sharmishta Desai](https://ieeexplore.ieee.org/author/37855762400), [Sourav Roy](https://ieeexplore.ieee.org/author/37086112979), [Brina Patel](https://ieeexplore.ieee.org/author/37086114897), “Very Fast Decision Tree (VFDT) algorithm on Hadoop”, published on August 2016, in IEEE journal.