ACCIDENT DETECTION USING VEHICULAR AD-HOC NETWORK

Jisha P Abraham[1], Aathik T R[2], Arjun P Bhaskar[3], Reshma Revi[4]

[1]*Professor, Dept. of Computer Science & Engineering, Mar Athanasius College of Engineering Kothamangalam, Kerala, India*
[2,3,4]*Student, Computer Science & Engineering, Mar Athanasius College of Engineering Kothamangalam, Kerala, India*

---***---

Abstract - *The project builds an accident detection system based on the concept of Vehicular Ad-Hoc Network(VANET). VANET follows a basic concept which consists of a large web of vehicles and a wireless local area network(WLAN) technology that connects all the vehicles together to make communication possible. The network is built on the highway with the use of RF transmitters and receivers. The main aim of VANETs is to build an Intelligent Transport System (ITS). Multiple IoT sensors are embedded in each vehicle to detect the accident. This information is immediately broadcasted over the VANET, during which the vehicle nodes of the network act as relays to transmit the information over. Ultimately, the information reaches the central hub with the exact geographical location of the accident, where the information is processed and classified using machine learning techniques. Based on the severity of the accident, messages are sent to the nearest hospital, police station and other concerned authorities.*

Key Words: VANET, IoT Sensors, Data Classification, Machine learning

1.INTRODUCTION

The major cause of death in the event of an accident is due to the delay in emergency help. It becomes too difficult to get emergency help in case of vehicle failure and accident in no network area as the victim is unable to make contact with anyone. To find a solution to this major problem we have introduced VANET technology in this paper. VANET [Vehicle Ad-Hoc Network]is a technology that uses moving vehicles as nodes in a network to create an Ad-Hoc network. Each participating vehicle is considered as a wireless router or node, and this is allowing the vehicles approximately 100 to 300 meters of each other and create a network with a wide range. If any of the vehicles fall out of signal range or out of network, other vehicles join into the network, connecting the vehicles to each other, so that Vehicular Ad-Hoc network created. The Vehicular Ad-Hoc Network defines an intelligent way of using the vehicular network. The main objective of this project is to detect the accident and vehicle breakdown location automatically and provide the required emergency help on time.

2. IMPLEMENTATION

2.1 System Architecture

Every vehicle acts as a node in the network and each vehicle will consist of an Accident Detector inside it. The Accident Detector consists of an MPU 6050 which is basically an accelerometer and gyroscope sensor, a piezoelectric sensor, a flame sensor and a GPS module, all of which are connected to a microprocessor Arduino Uno. RF transmitters and receivers are also installed inside the detector to broadcast and receive the data from adjoining nodes. Hence a VANET is created. The assembly of sensors and the microprocessor inside each vehicle in the network is shown in Fig.-1. The components used for the project are :

• Arduino UNO: This is the most major component which is present in each and every node in the VANET. The Arduino controls and coordinates the inflow and outflow of data from and to the vehicles. The microcontroller is programmed so as to implement a section of the classification task to the vehicle end, where situations which are sure to be an accident are directly reported as so to the central hub by the microcontroller. The image of an Arduino UNO microcontroller is shown in Fig-2.

• Force Sensor: This is one of the main sensors incorporated into all the vehicles part of the network and its image is shown in figure 3. It is basically a piezoelectric sensor that comprises a thin flat surface that can detect the amount of force exerted on it. It gives out the analog values corresponding to the force applied on it to the microcontroller in that particular vehicle unit. The image of the A301 force sensor is shown in Fig-3.

• Gyroscope + Accelerometer (MPU 6050): The MPU 6050 gives the most critical data related to the vehicle nodes - analog values of various parameters such as acceleration, pitch, roll, yaw etc. These values play a major role in predicting the event of an accident. The pin diagram of the MPU 6050 Accelerometer+Gyroscope module is shown in Fig-4.

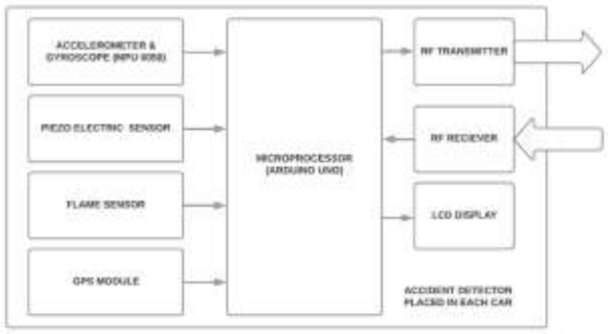

ACCELEROMETER & GYROSCOPE (MPU 6050)
PIEZO ELECTRIC SENSOR
FLAME SENSOR
GPS MODULE
MICROPROCESSOR (ARDUINO UNO)
RF TRANSMITTER
RF RECIEVER
LCD DISPLAY
ACCIDENT DETECTOR PLACED IN EACH CAR

Fig -1: Accident Detector Units in Each Car

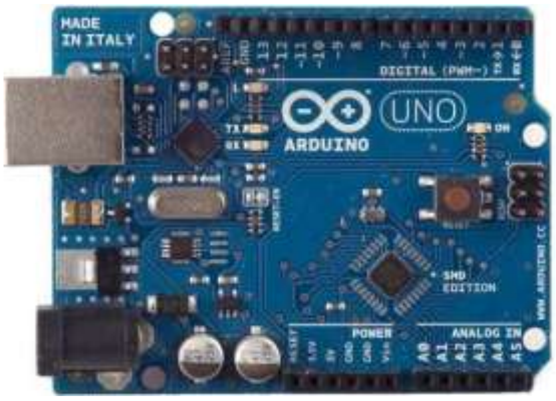


Fig -2: Arduino UNO

• RF Transmitter and Receiver: The signal gets broadcasted from the transmitter. It is amplified through transistors and passed to the oscillator which helps to get higher frequencies for transmission of longer distance. It is then passed to the tank circuit where isolation and storage of information will take place with the help of the RC circuit and once the signal gets processed in these stages it is broadcasted through an antenna. The RF receiver receives the broad-casted signal which is again amplified and involved in the stabilization process. It is then passed to the buffer, driver and relay circuit. The images of the transmitter and receiver are given in Fig-5.

Fig -3: Flexi Force A301 Sensor

Fig -4: MPU 6050

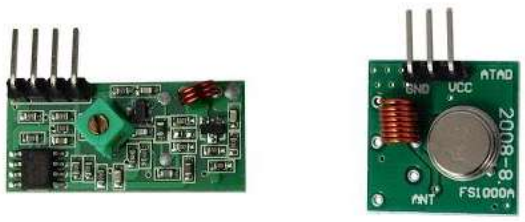


Fig -5: RF Transmitter & Receiver

• Fire Sensor: The fire sensor is part of the sensor units placed in each vehicle in the network, which is primarily used to obtain information regarding whether a fire has broken out somewhere inside the vehicle or not. A nod from this sensor at any time is taken to be as an accident and so, does not work in collaboration with the other sensors. The pin diagram of the fire sensor is shown in Fig-6.

• GPS Module (NEO 6M): The Neo 6M GPS Module is used to retrieve the location information about the vehicle in which it is fitted. The values retrieved from this module correspond to a complex combination of values that contain information regarding a variety of parameters such as latitude, longitude, speed, direction etc. This module has a horizontal accuracy of 2.5m, with a default baud rate of 9600. It has the antenna as a separate component with -161 dBm sensitivity. The image of a NEO 6M GPS module is shown in Fig-7.

Fig -6: Fire Sensor

Fig -7: NEO 6M GPS Module

The data information is transferred and received by multiple nodes and it is directed to a central hub where this data is processed. The block diagram of the central hub is given in Fig-8. The data received is processed and the accident is classified based on its severity and corresponding officials and authorities are informed about the location and severity of the occurred accident. The classification is done based on a pre-trained algorithm.

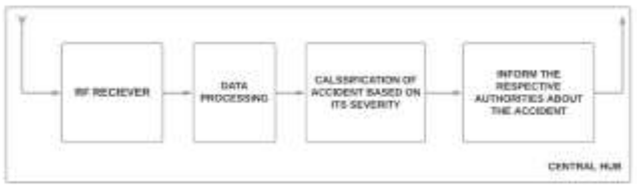

RF RECIEVER
DATA PROCESSING
CALSSIFICATION OF ACCIDENT BASED ON ITS SEVERITY
INFORM THE RESPECTIVE AUTHORITIES ABOUT THE ACCIDENT
CENTRAL HUB

Fig -8 : Block diagram of the Central Hub

3. DESIGN

As shown in Fig-1, each and every vehicle which is part of the VANET will consist of a series of sensors which produce outputs to a microcontroller in the form of an Arduino UNO chip. This chip collects and processes data from all the sensors such as the MPU 6050, NEO 6M GPS Module, A301 Force sensor, fire sensor and so on. Each vehicle, which we term as a 'node' in the adhoc network, will also communicate with each other and the central hub using RF transmitters and receivers. The flow of data during the event of an accident is shown in Fig- 9. Consider Car 1 (node 1) has met with an accident. It will transmit the alert message via the RF Module to Car 2 (node 2) which is in the same network. Again the alert message will be transmitted to the next moving cars, with each car acting as a relay. The process will continue until the car which is in the network area will receive this message. The alert message will finally be transmitted to the base station i.e. central hub.

The central Hub will classify the event based on the inputs it has received, into several classes distinct by the severity of an accident. After classification, an alert will be sent to the required stations, such as a service center in case of a vehicle break-down or hospital police-station in case of an accident. The power supply provided to the Arduino is 12V dc and it is provided from the car itself. The alert message generated will contain information like, vehicle owner's name, vehicle number, the current location of the vehicle and the mobile number on which the message is to be transmitted. The GPS module will provide the vehicle's current location which has met with an accident in no network area.

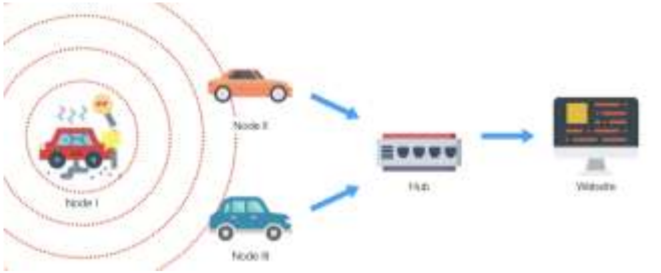
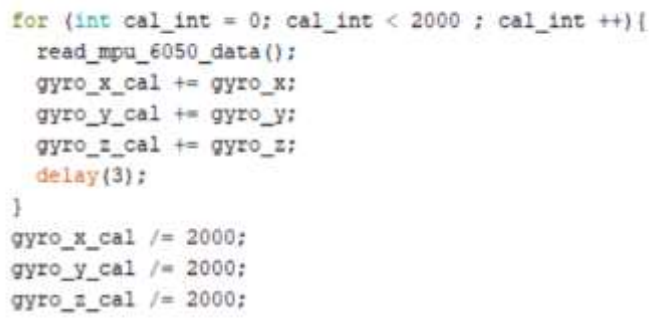


Fig -9: Flow of Data in the event of an accident.



```
for (int cal_int = 0; cal_int < 2000 ; cal_int ++){
  read_mpu_6050_data();
  gyro_x_cal += gyro_x;
  gyro_y_cal += gyro_y;
  gyro_z_cal += gyro_z;
  delay(3);
}
gyro_x_cal /= 2000;
gyro_y_cal /= 2000;
gyro_z_cal /= 2000;
```

Fig -10: Arduino Code for calibration of MPU 6050

The Arduino in each node is programmed so as to perform an initial processing of the data it receives, which in turn reduces the load on the central hub during its classification process. The microcontroller first calibrates the values it receives from the MPU 6050 Accelerometer+Gyroscope module, which mainly consists of the parameters such as pitch, roll, yaw etc. Calibration of these parameters is paramount due to the varying terrains and other conditions that the nodes may go through while the values are being taken in real-time, and its code is shown in Fig-10. Post calibration, realtime values are accessed and monitored to check for high variations in the data. A simple threshold is kept to detect whether a spike in the values of these parameters can only be formed due to an accident of some sort. In such cases, the affected node's microcontrollers immediately transmit this information along with a confirmation of an accident to it, which saves crucial time and processing at the central-hub end. The interfacing of this module with the UNO is shown in Fig-11. Similar to the MPU 6050, the NEO 6M GPS module, arguably the second most vital component in the node, also starts to spit data as soon as it is turned on. The data received from this module would be in the form of NMEA sentences, which is an abbreviation for the National Marine Electronics Association. This format is shown in Fig-12.

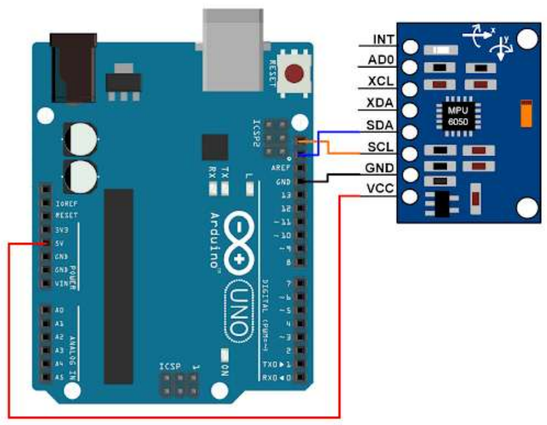
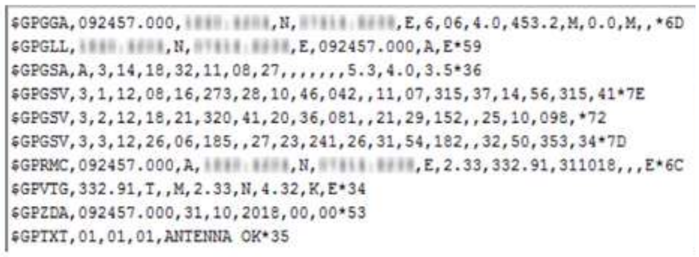


Fig -11: Interfacing of MPU 6050 with Arduino UNO.



```
$GPGGA,092457.000,[redacted],N,[redacted],E,6,06,4.0,453.2,M,0.0,M,,*6D
$GPGLL,[redacted],N,[redacted],E,092457.000,A,E*59
$GPGSA,A,3,14,18,32,11,08,27,,,,,,,5.3,4.0,3.5*36
$GPGSV,3,1,12,08,16,273,28,10,46,042,,11,07,315,37,14,56,315,41*7E
$GPGSV,3,2,12,18,21,320,41,20,36,081,,21,29,152,,25,10,098,*72
$GPGSV,3,3,12,26,06,185,,27,23,241,26,31,54,182,,32,50,353,34*7D
$GPRMC,092457.000,A,[redacted],N,[redacted],E,2.33,332.91,311018,,,E*6C
$GPVTG,332.91,T,,M,2.33,N,4.32,K,E*34
$GPZDA,092457.000,31,10,2018,00,00*53
$GPTXT,01,01,01,ANTENNA OK*35
```

Fig -12: Message Format of GPS Receivers (NMEA Sentences)

The NMEA standard is formatted in lines of data called sentences. Each sentence is comma-separated to make it easier to parse by computers and microcontrollers. These NMEA sentences are sent out at an interval called the update rate. NEO-6M GPS module updates this information once per second(1Hz frequency) by default. But you can configure it for up to 5 updates per second(5Hz frequency). NEO 6M GPS Module interfacing with Arduino UNO is shown in Fig-13.

The thin paper-like sensor called the force sensor is the one that tells the Arduino whether the body of the vehicle has experienced some force due to an impact on some part of its body. Due to its small size and flexibility, it is pretty easy to mount on each node and doesn't affect the performance of the vehicle in which it is fitted in any manner. The interfacing of the force sensor with the Arduino UNO microcontroller chip is shown in Fig-14. To measure the change in the analog inputs from the A301, we use a 10 M resistor which acts as a voltage divider. We can measure how much voltage is on the flexiforce using the analogRead and we will have our force reading. The amount of that 5V that each part gets is proportional to its resistance. So if the flexiforce and the resistor have the same resistance, the 5V is split evenly (2.5V) to each part. (analog reading of 512). But if the flexiforce is pressed on pretty hard, reading only 25K of resistance, the 1M resistor is going to soak up 40 times as much of that 5V. So the FSR would only get .12V. (analog reading of 25). If something is barely pressing on it, the flexiforce may be 5M of resistance, so the flexiforce will soak up 5 times as much of that 5V as the 1M resistor. So the flexiforce would get 4.2V (analog reading of 852).

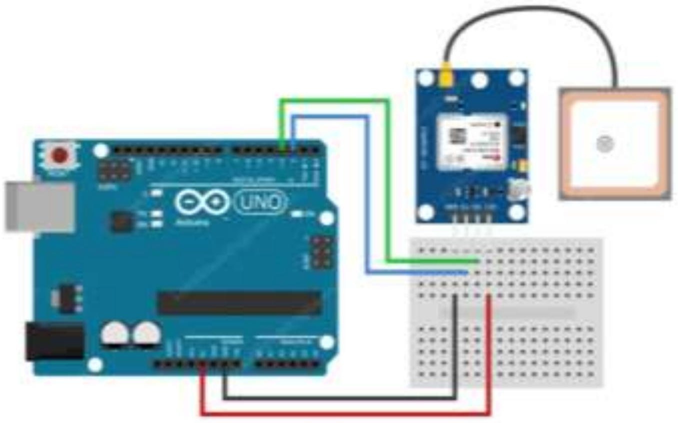


Fig -13: NEO 6M GPS Module interfacing with Arduino UNO

The interfacing of the 433 MHz transmitter and receiver with the Arduino UNO is shown in Fig-15. The communication is enabled using a library called RadioHead, which allows simple data transfer between Arduino boards. It's so versatile that it can be used to drive all sorts of radio communications devices, including our 433MHz modules. The RadioHead library encapsulates the data to be sent into a packet of data which includes a CRC (Cyclic Redundancy Check) and then sends it with the necessary preamble and header to another Arduino. If the data is received correctly, the receiving Arduino is informed that there is data available and proceeds to decode and action it.

The receivers of the vehicles are kept on at all durations. In case of a detected accident, the receivers are turned off and the transmitter is switched on. Ultimately the data reaches the Central Hub and is transferred to the SQL database with the help of an Ethernet Shield. It enables the Arduino to send and receive data from anywhere in the world with an internet connection.

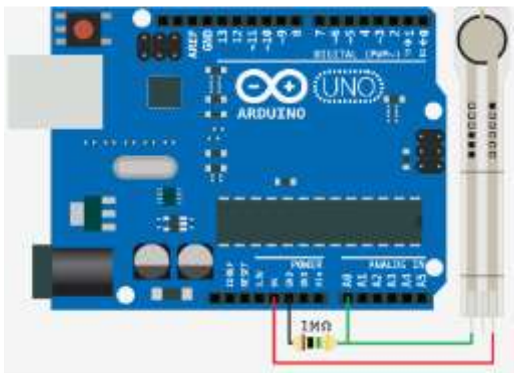


Fig -14: FlexiForce A301 Sensor interfacing with Arduino Uno

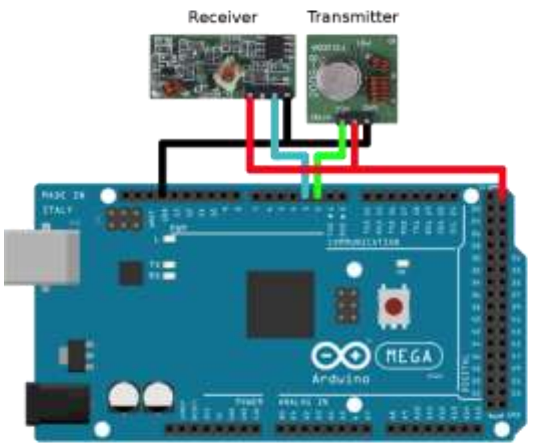


Fig -15: 433 MHz Transmitter and Receiver interfacing with Arduino Uno

The SQL Database is implemented using XAMPP online web server. XAMPP is a free and open-source cross-platform web server solution stack package that mainly consists of the Apache HTTP Server, MariaDB database, and interpreters for scripts written in the PHP and Perl programming languages. This data is then used to predict the severity of the accident. A Machine learning model called Naive Bayes Classification Model is used to predict the severity of the accident. This algorithm is simple to implement and has the ability to classify a large set of instances into their proper class-label and is thus, efficient. It also allows multiple classes which is required for this scenario. It uses conditional probability, using the Naive Bayes Theorem. Let (x1, x2, . . . , xn) be a feature vector and y be the class label corresponding to this feature vector. Applying Bayes' theorem shown in Fig-16 :

$$P(y|x_1, ..., x_n) = \frac{P(y)P(x_1, ..., x_n|y)}{P(x_1, ..., x_n)} \qquad (1)$$

Fig -16: Bayes Theorem

When dealing with continuous data, a typical assumption is that the continuous values associated with each class are distributed according to a normal (or Gaussian) distribution. So we use Gaussian Naive Bayes algorithm for classification. The probability density of the normal distribution is shown in Fig-17:

$$f(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} {}^{-\frac{(x-\mu)^2}{2\sigma^2}} \qquad (2)$$

Where

- 'μ' is the mean or expectation of the distribution,
- 'σ' is the standard deviation, and
- 'σ^2' is the variance.

The model is pre-trained with the help of a known dataset.

Fig -17: Probability Distribution Function

The model is pre-trained with the help of a known dataset. Python 3.6 along with the libraries Pandas and scikit-learn is used to implement the training model. Pandas library is used to access the dataset in '.csv' format. Scikit-learn library has GaussianNB(), which implements the Gaussian Naive Bayes algorithm for classification. The class labels which would be allotted to each and every case of an accident are Critical, Major and Minor. Once the classification is done, concerned authorities and help services will be notified about the accident and various important details regarding the accident as well.

REFERENCES

[1]] Sandyashree, U. Kumar- "Design and implementation of VANET for accident identification using WSN technology" ISSN: 0976-1353 Volume 14 Issue 2- APRIL 2015.

[2] C. Dahat, M. Thakur, R. Sahare, R. Ramtekkar "VANET for Emergency Services and Accident Detection", International Journal of Engineering Science and Computing, March 2017.

[3] Arduino Interfacing, https://forum.arduino.cc.