International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2017 IJSRCSEIT | Volume 2 | Issue 2 | ISSN : 2456-3307

Road Transport System with Ambulance Centric Intelligence

Ananthakumar Sethuramanujam, Deebhaghavel P.S.

Department of Electronics and Communication Engineering, Sri Eshwar College of Engineering, Coimbatore, Tamil Nadu, India

ABSTRACT

When ambulance is stuck in traffic, it leads to the wastage of precious time that could have been used to save the patient's life, otherwise. Given the technological advancements and the power of a microcontroller, we can contribute to the life of patients by automatically turning the traffic signal Green in favor of ambulance and thus the time that ambulance would have wasted if such technology has not existed, could be saved and we can reap the benefits of such technology by giving doctors more time to work on patients. The sole goal of the project is to help save the lives of people by completely eliminating the problem of ambulance having to stuck in traffic. We use Haversine formula to determine the traffic signal which was targeted by the ambulance driver. We input the GPS coordinates of the ambulance to the server which in turn takes out the nearest signal coordinates from the database and turns it green.

Keywords : Global Positioning System, Haversine, Intelligent Traffic System.

I. INTRODUCTION

Smart With the advent of Internet Of Things, the way we implement the solution to the problems has changed dramatically. We found that on average 2200 deaths a year in a metropolitan city [1, 2, 3]. The Japanese Company Fujitsu [5,6] is already developing a solution to build Human Centric Intelligent society [4, 7]. We have developed a novel system which uses Haversine formula [8] to implement a traffic system which will reduce congestion deaths.

The paper is organized as follows. Section II explains the Haversine formula which is the basis for this paper. Section III outlines the algorithm and flow of the system. Section IV presents the experimental results and finally Section V gives the conclusion.

II. METHODS AND MATERIAL

1. Mathematical Foundation

The haversine formula determines the great-circle distance between two points on a sphere given their longitudes and latitudes. Important in navigation, it is a special case of a more general formula in spherical trigonometry, the law of haversines, that relates the sides and angles of spherical triangles.

Haversine Formula:

$$a = \sin^2(\Delta\varphi/2) + \cos\varphi 1 \cdot \cos \Phi 2 \cdot \sin^2(\Delta\lambda/2)$$

$$c = 2 \cdot \operatorname{atan2}(\sqrt{a}, \sqrt{(1-a)})$$

$$d = R \cdot c$$

where φ *is latitude,* λ *is longitude,* R *is earth's radius (mean radius = 6,371km); The angles need to be in radians.*

In Java, Haversine law is implemented as follows:

```
Cursor c = db.rawQuery("select * from
gpsphonetable",null);

// gpsphonetable -  the table which has the details
of all the trafiic signal locations

        c.moveToFirst();

  while(!c.isLast())

        {double lat2=c.getDouble(0);

              double lng2=c.getDouble(1);

        doubleearthRadius = 3958.75;

        doubledLat = Math.toRadians(lat2-lat1);
```

```
        doubledLng = Math.toRadians(lng2-
lng1);

        double a = Math.sin(dLat/2) *
Math.sin(dLat/2) +

        Math.cos(Math.toRadians(lat1)) *
Math.cos(Math.toRadians(lat2))*
Math.sin(dLng/2) * Math.sin(dLng/2);

        double d = 2 * Math.atan2(Math.sqrt(a),
Math.sqrt(1-a));

        doubledist = earthRadius * d;

intmeterConversion = 1609;

floatdism = new Float(dist *
meterConversion).floatValue();

        if(i==0){

                val=dism;

                i++;

           }

        if(dism>val){

                c.moveToNext();}else {

                val=dism;

                latret=lat2;

                longitret=lng2;

                c.moveToNext();

        } }
```

This implementation is used in the Android code to run the system.

2. Algorithm

For easier comprehension we present the algorithm in the form of block diagram and flow diagram.

----------> BLOCK DIAGRAM ----------->

The ambulance is at the location (10.828272, 77.055007). The signal at (10.822477, 77.016144) is unaffected as it is not the traffic signal that the ambulance is targeting. Cloud computing plays a vital role in determining the correct traffic signal to which the message is to be sent. In this case, the traffic signal at (10.827465, 77.060404) is chosen, as cloud computing identifies that this is the one that ambulance driver is trying to turn green.

The above block diagram uses cloud for processing the location data but it is undesirable sometimes as we cannot be certain that all the times the server will respond fast. To overcome this, the database is incorporated locally in the android app eliminating the need for cloud computing.

----------> DATA FLOW DIAGRAM ----------->

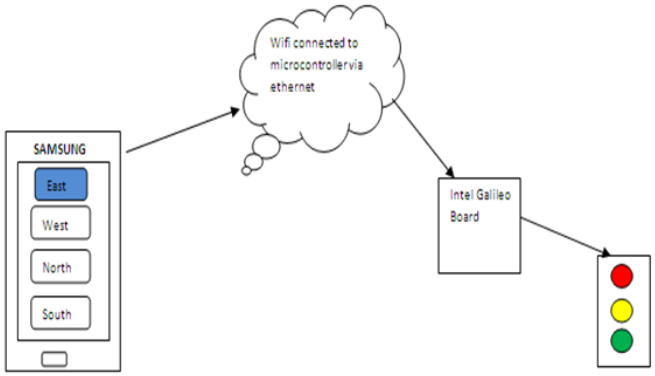

Figure 1.

III. RESULTS AND DISCUSSION

The following components are used in our project.

1. LED – 12 (Red, Yellow, Green each 4)
2. Resistor -12 (330 ohm)
3. SIM 900A GSM module and a Vodafone SIM card with number
4. An android app that has four options (East, West, North and South indicating the direction that the ambulance is traveling) and Google API facility
5. A Java script that processes the location of ambulance and sends the message to corresponding SIM card
6. GR- KAEDE board

The controller program is simple and just calls eight functions namely,

1. east(),
2. east_south_yellow(),
3. south(),
4. south_west_yellow(),
5. west(),
6. west_north_yellow(),
7. north(),
8. north_east_yellow()

The functions at Odd number positions operate in such a way that the green LED in the direction that the function name bears glows and the other directions

glow red.

The functions at Even numbered positions have two directions in their name. The first direction is meant to glow Green LED and the second direction is meant to glow Yellow LED.

For example, east_south_yellow()- here east is first direction and south is second direction.

This way the traffic signal operates continuously. The project demands the microcontroller to respond to the message received by SIM card and change the signal Green in the direction that the message instructed to. Hence GR-KAEDE board should be able to read GSM now and then. To facilitate this, a function named read() is used. It reads the message from SIM card and acts accordingly as shown in the source code. Once the message is read, the message is flushed off. To be more accurate and facilitate fast response, the frequency of checking the GSM message is kept as high as 30000 times during the signal wait period. It is done with the help of using nested for loop, where the outer one corresponds to number of seconds and the inner one corresponds to milliseconds. Since there are 1000 milliseconds in a second, 30 seconds equals 30000 milliseconds. The read() function is used inside the inner 'for' loop and hence it runs 30000 times in a span of 3o seconds evoking faster response to ambulance.

Android app is created with Android Studio and Google API functionality is imparted. Thus when the ambulance driver clicks any one of the options available, the name-value pair along with the latitude, longitude of the ambulance is sent to the server.

As seen in the code, we have layout creation part which lays out the android app as shown below:

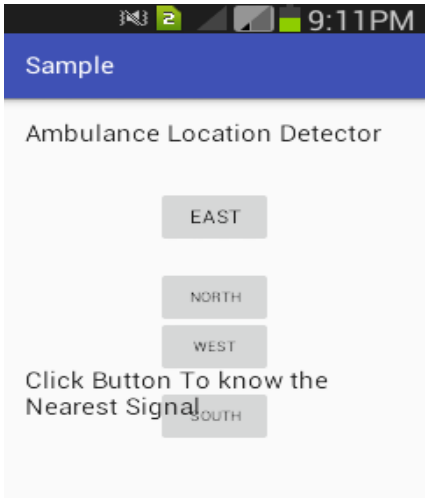

Figure 2. The Andriod app

For each of the buttons east, west, north and south, I created four onclicklistener events which respond to the clicks of the button. When east is clicked, E is set to value identifier and gps location finder function is called whose name is LatLng. Then this function returns the latitude, longitude values of the location to the calling function and sets the values in the variable GPS. Then we make a HTTPrequest with the function AndriodHTTPRequestsactivity. We are passing the values to the server side script using this function. The values sent are GPS location and the value identifier that holds the direction value.

The received values are then processed by PHP Script using Haversine formula. The haversine formula determines the signal that the ambulance driver intents to change. Then MySQL database is queried to the phone number corresponding to the selected signal location. Then the message specifying the direction value is sent to the SIM card in the GSM module at the signal which in turn controls the flow of the signal.

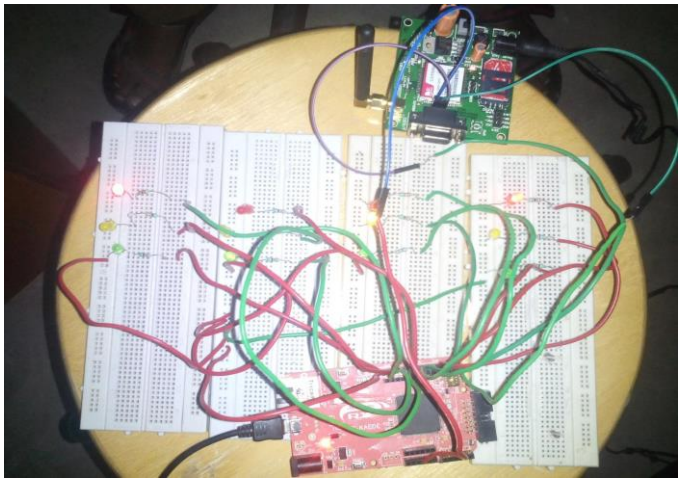

Figure 3. Experimental Setup

IV.CONCLUSION

We have implemented a system that turns the traffic signal on a per the arrival of ambulance. This system is the first step in creating the future smart cities. We have tested our system multiple times at different locations and it is performing up to the mark.

V. REFERENCES

[1]. https://www.ft.com/content/40774fc6-76b5-11e6-bf48-b372cdb1043a

[2]. http://oecdobserver.org/news/fullstory.php/aid/647/Road_congestion:_What_s_the_deal_.html

[3]. http://www.who.int/gho/road_safety/mortality/en/

[4]. https://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=3&cad=rja&uact=8&ved=0ahUKEwj7hI-3nNDSAhWKH5QKHa4BDFoQFggvMAI&url=http%3A%2F%2Fwww.fujitsu.com%2Fglobal%2Fvision%2Fhuman-centric-innovation%2F&usg=AFQjCNEGlIVMeeJQOurcyOHoe0W6gzhNcA&bvm=bv.149397726,d.dGo

[5]. http://www.fujitsu.com/global/vision/human-centric-ai/

[6]. https://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=5&cad=rja&uact=8&ved=0ahUKEwj7hI-3nNDSAhWKH5QKHa4BDFoQFgg8MAQ&url=http%3A%2F%2Fsuperuser.openstack.org%2Farticles%2Ffujitsu-artificial-intelligence%2F&usg=AFQjCNFUozrAg2hyeLRlN3oHP6b77NyBug

[7]. https://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=5&cad=rja&uact=8&ved=0ahUKEwiJ4qT3nNDSAhWHpZQKHRwgBNkQFgg_MAQ&url=http%3A%2F%2Fwww.sciencedirect.com%2Fscience%2Fbook%2F9780123747082&usg=AFQjCNFTe4svb0XPV40-OVk1JPlyzCwYng

[8]. https://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=18&cad=rja&uact=8&ved=0ahUKEwii2a-WndDSAhWIKpQKHWuZAI0QFgiHATAR&url=http%3A%2F%2Fwww.igismap.com%2Fhaversine-formula-calculate-geographic-distance-earth%2F&usg=AFQjCNHWEvWiOdwHRlzKZ73-oi1Dx99vXg