International Journal of
Scientific Engineering and Technology Research

ISSN 2319-8885
Vol.03,Issue.13
June-2014,
Pages:2961-2964

www.semargroup.org,
www.ijsetr.com

Adaptive Cruise Control Using Google Maps API

ARJUN RAJAGOPALAN[1], MAHAK PATIL[2], ABHISHEK JOHN Z[3], PRASHANT RAJWANSHI[4]
Dept of Electronics and Telecommunication, Vishwakarma Institute of Technology, Pune, India.

Abstract: Adaptive Cruise Control (ACC) is a Driver Assistance System that can help to improve the safety of our roads by reducing the workload on the driver. By extending traditional Cruise Control (CC) with the ability to adjust its velocity based on traffic conditions, ACC can be used in many situations where traditional CC would be impractical. However, before the advantages of ACC can be leveraged, drivers must be able to trust it to perform safety even in the presence of faults. The goal of this project is to develop a hybrid system model of Adaptive Cruise Control and Mobile Communication which would help us keep a track on the vehicle with a WEB based Android Mobile Application.

Keywords: Android Application, Arduino Leonardo Microcontroller, Bluetooth Communication, Adaptive Cruise Control.

I. INTRODUCTION

Adaptive Cruise Control (ACC) is a Driver Assistance System that can help to improve the safety of our roads by reducing the workload on the driver. By extending traditional Cruise Control (CC) with the ability to adjust its velocity based on traffic conditions, ACC can be used in many situations where traditional CC would be impractical. However, before the advantages of ACC can be leveraged, drivers must be able to trust it to perform safety even in the presence of faults. The goal of this project is to develop a hybrid system model of Adaptive Cruise Control and Mobile Communication which would help us keep a track on the vehicle with a WEB based Android Mobile Application. Given below are the three main components or systems that we have developed:

- Mobile Application.
- Control System (Audrino based Control Unit).
- Automobile Hardware.

II. SOFTWARE AND HARDWARE IMPLEMENTATION

A .Mobile Application

An Android based mobile application is a program which can be run on a device which supports android. The android version on which the 'geoloc' application is developed is android 4.2.2 or 4.3 which is currently the latest versions on most android enabled devices. We have used a backend program named Page-Kite to use our local-host server as a Publically Accessed Server. PageKite creates a connection between your local-host server and the public Internet. Your local server gets a public name (something. pagekite. me) which anyone can connect to using a normal web browser. The PageKite back-end configures DNS for your server and establishes communication with our global pool of front-end relays. These relays have public IP addresses and are visible on the global Internet, which makes it possible for them to accept incoming requests on your server's behalf. These requests get forwarded over the tunnel to your server, and any replies travel back the same way. In technical terms, it is a dynamic, tunneled reverse proxy. When this program is executed it displays the IP address of every machine that accesses it.

The main purpose of development of this app is that once this device is enabled with the Adaptive Cruise Control (ACC), we should be able to track its every movement. This is done by using the GPS system of the phone. The GPS system of the phones returns to us the geostationary co-ordinates based on Longitude, Latitude and Altitude. These returned values are used to reverse Geo-trace to give us an exact location of the system. The location Rendering software that access the Google database and returns a "float" format of the respective values and these values are displayed on to the screen using a Text View. In the next part we use this returned values from GPS module of the phone and parse them on to the Google server for reverse Geo-coding and return the address of the values. The amount of detail in a reverse geo-coded location description may vary, for example one might contain the full street address of the closest building, while another might contain only a city name and postal code. The Geo-coder class requires a backend service that is not included in the core android framework. The Geocoder query methods will return an empty list if there no backend service in the platform.

We have provided a ABOUT menu where a synopsis of the application is given. Once the mobile device gets the location, the user is required to SET his destination using the menu button. The in-built set values are set to the Text View which are in-turn in the next clock cycle sent to the server. Now the User has to select the SEND TO BOARD Option.

This is will now open a new activity. This Activity deals with the communication section between the mobile device and the Arduino Board. The Arduino Board is connected with a HC05 Bluetooth module (Discussed in next section). The application is hardcoded with the MAC address of the Bluetooth Module (20:13:10:16:01:88). Once the mobile device is connected to the Bluetooth module, the user can control the car using the on display control panel. When a key is pressed it sends a assigned digit to the board where it is decoded and the necessary action is performed. This page also displays the route that is being taken. This is got from the webpage that we have designed to display all the necessary details.

We have provided to Buttons named:

- Test Environment
- Real World

The Test Environment Button sends a set of test values to the board so that we can use it for demonstration purpose. The Real World Button connects the mobile device to the server where it gets the distance and directions that need to be sent to the board. We get these values from the WEBPAGE that is discussed in the next section. Our App uses this feature provided by Google. Co to track the given co-ordinates and display this on an HTML page, with the help of a SQL table. This table consists of values which have been pushed into it by the application using JSON parser and PHP[10]. This table is then displayed on to an HTML page and thus we can reverse track it. The SQL table and the Geo-location map are displayed on to the page. The values from the last entry into the table are inserted as the input to the Geo-location Map and using a php code, it returns us the direction from source to destination. In the HTML page I have provided two Buttons.

- Send to Server
- Clear

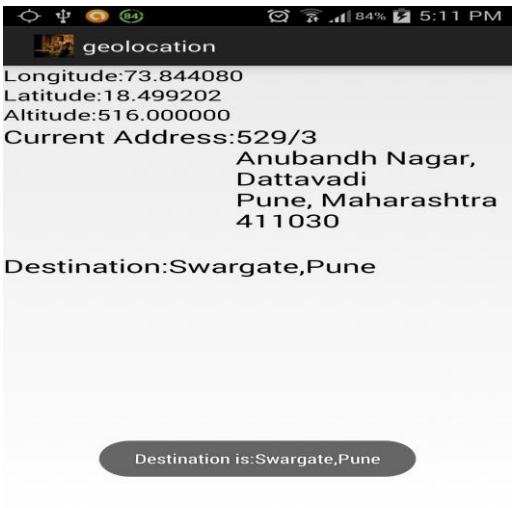

Figure1. Main page.

We have selected keywords such as ”left”, ”Right”, ”Continue”, these words are searched throughout the HTML page and saved into an array as and when they occur. Secondly, the distances of each directions is also calculated using a tag to find any value followed by the letters ”km”, and is stored in a array. This is then sent to a SQL table with each direction and distance as entries in the table, using the ”INSERT INTO” command. Every time the user logins to the app, he would be required to clear the table using the CLEAR table. As this just a BETA version of the app, it requires manual clearing of SQL table. Shown below is how the SQL table getting all the entries from the Map look like. When the Button is pressed it truncates the table using the” TRUNCATE TABLENAME”, so that new directions can be set in SQL table, to be sent to the phone again. Shown below are the screen shots of application figure 1 and 2.

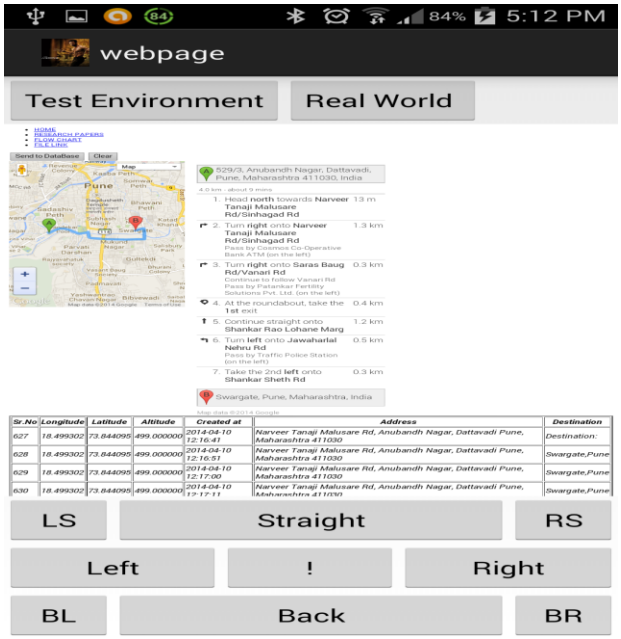

Figure2. Result from Webpage with control buttons.

B. Control Unit

When the Arduino is powered up, the sensor calibrates itself during its first read cycle. The sensor uses this stored information to range a close object. It is important that objects not be close to the sensor during this calibration cycle. The best sensitivity is obtained when it is clear for fourteen inches, but good results are common when clear for at least seven inches. If an object is too close during the calibration cycle, the sensor may then ignore objects at that distance. These readings are sent to the Arduino through a serial port. This reading is then checked with a predefined value (by programmer). We have set it to six inches. If the read value is greater than the set threshold value, only then the next steps will be executed. If the condition is not met, the sensor will repeat the ranging process till the requirements are met. When a reading greater than the threshold is received, it reads data from the android application which sends directions via Bluetooth. Along with directions, it also sends the distance that has to be travelled by the car. When this data is received, the Arduino generates appropriate instructions to the motors i.e. if it receives LEFT, the Arduino sends signals to the otors to turn left etc. If at

any point, the minimum conditions are not met, the car stops running. This process repeats till data is available from the Bluetooth. Below are the flowcharts (figures 3 to 5) for the various functional entities of the project:

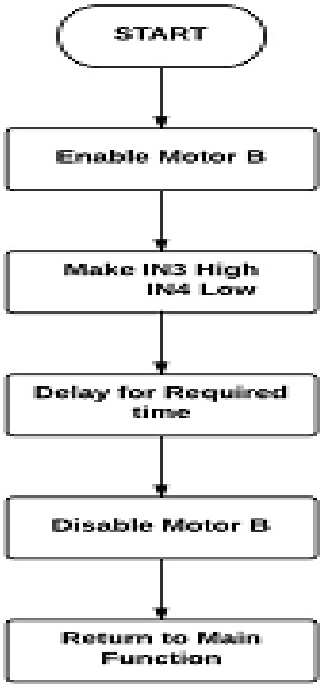

Figure3. Move Straight.

Figure4. Move right.

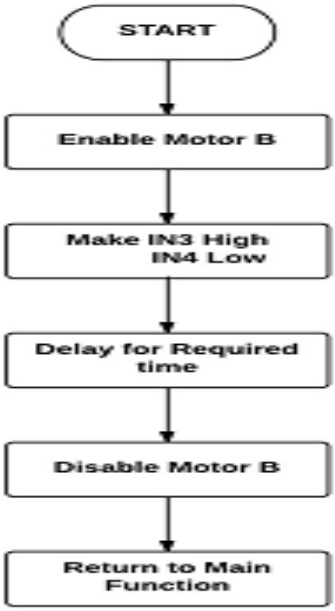

Figure5. Move Left.

C. Automobile Hardware

A model car was used in the project which we bought from a local store. It is a 1:10 model car of a monster truck which works on a 9.6V rechargeable battery. After buying the car, we dismantled it entirely. We kept the chassis and wheel arrangement as it was provided and mounted the Arduino Leonardo along with the peripherals on the arrangement. After various trials, we decided that using the already provided battery pack is the best option as it could be recharged over time. We connected both the motors to the Arduino board and mounted the Bluetooth module on the PCB as well. Alignment and placing of the sensor was crucial. With a misalignment of even 1 degree, the errors in calibration of the sensor were tremendous. Due to relatively small size of the model car, we placed the sensor on the top middle portion of the model car as shown in figure 6.

Figure6. Final Model.

III. RESULTS

Once the final model was finalized, we created a test environment to see the working of the model. Firstly, we created a test environment in the android application which followed a set of predefined instructions:

- Go straight for 4 meters.
- Turn right and continue for 4 meters.
- Turn left and continue for 4 meters.
- Go reverse for 4 meters.

We then formulated the maximum distance the motor should run for so as to maintain a safe distance from obstacles in front of it, it is given by:

$$\text{Distance\_max} = (0{:}8 \text{ X Distance\_input}) \quad (1)$$

This formula is just based on the test environment, as the distance would increase the multiplying factor of 0.8 would also need to be varied with respect to the distance. When the Arduino was plugged into a computer and this test environment was sent via Bluetooth, the following results figure 7 were obtained We concluded that the test conditions were successfully received and further data processing was also done satisfactorily. Proceeding further, we attached the Arduino to the model car and verified all the connections. Again the test conditions were sent to the Arduino and the car moved accordingly. We then concluded that the model works in compliance with the input data.

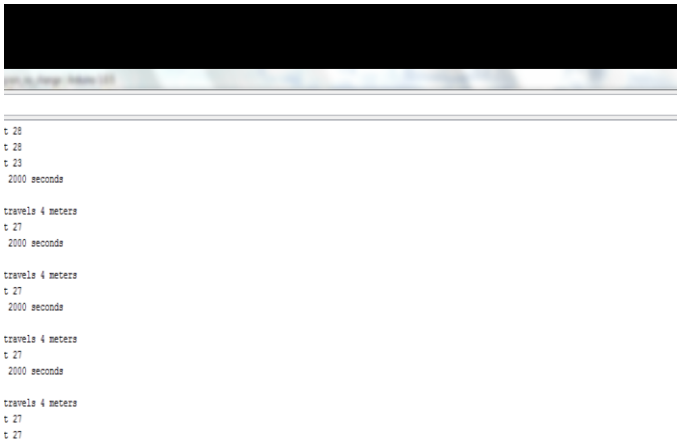

Figure 7: Final Result.

IV. CONCLUSION

We had initial divided ourselves into three groups:
Stage 1: Mobile Application Design.
Stage 2: Control System (Audrino based Control Unit) design
Stage 3: Integration.

We have successfully completed all the above stages as discussed with our guide. The Mobile Application is complete with successful outcomes and with desired outputs. The Mobile application was connected successfully with the automobile hardware via Bluetooth. Control unit is programmed for the TEST ENVIRONMENT, REAL WORLD environment details have been sent to the board successfully, but the control unit would require a different program with minor changes in the receiving pattern. Future scoop of his project is tremendous as it would ease the burden of driving in traffic. The next step would be to integrate it on a real car and also to find a faster and efficient way of communication between the Mobile Device and Control Unit.

V. ACKNOLEDGEMENT

We the students of Fourth year Electronics and telecommunication Engineering would like to express our heart-felt gratitude to our guide, Prof. Suhas Bhise for helping us out in this project by constantly proving the insight needed for the same, keeping our morale high when faced with any problem and encouraging us to always go the extra mile. We would also like to thank him for giving us the opportunity to do this project that adds to our knowledge of Electronics and Computer Science as well. Last but not the least, we would also like to thank our college Vishwakarma Institute of Technology for providing us with fully-equipped labs, which were used by us for successfully completing our project.

VI. REFERENCES

[1] Methodology for accessing ACC behavior, IEEE, 2008.
[2] Safe design methodology for ACC with GPS, Labbani and Ruttn, 2007.
[3] Han and Yi, 2006.
[4] Nissan Manufacturing Manual.
[5] Ford manual.
[6] Schlottman-Peters.
[7] Ogata, Katsuhiko. System Dynamics. 1998. Prentice-Hall, Inc. Upper Saddle River, New Jersery.
[8] http://howstuffworks.lycoszone.com/cruise-control.htm.
[9] http://developers.android.com.
[10] http://www.mybringback.com/tutorial eries/12992/ and roimmysql-php- json-part-2- setting-up-a- xampp-server-and-mysql.
[11] www.pagekite.com.
[12]HRLV-MaxSonar- EZ Series High Resolution, Precision, Low Voltage Ultra Sonic Range Finder MB1003, MB1013, MB1023, MB1033, MB1043.
[13] IEEE 802.15.1.
[14] Ultrasonic Doppler for Distance Measurement,IRE TRANSACTIONS OF ULTRASONIC ENGINEERING.
[15] http://msdn.microsoft.com/en-us/library/hh156513.aspx.