INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact factor: 4.295
(Volume 4, Issue 5)
Available online at: www.ijariit.com

Smart location clock

Vijay Balaji
vijaybalaji25121998@gmail.com
SRM Institute of Science and Technology, Chennai, Tamil Nadu

Hani Saravanan
hanisaravanan@gmail.com
SRM Institute of Science and Technology, Chennai, Tamil Nadu

Srivarshini Srinivasan
ssvarsh_98@yahoo.com
SRM Institute of Science and Technology, Chennai, Tamil Nadu

ABSTRACT

In this project, we have engineered a way to reconstruct the traditional location tracking software into a smart location clock. The location clock paves the way for an uncomplicated technique for location tracking. Using our system, users can trail the movements of any person or their family members who are connected to the clock. The project enables the trailing of people to the most frequently visited places. The clock embodies a symbolic representation of the frequently visited places of members in the family or the person in the place of numbers and each hand of the clock is designated to each person or family member. When a change in location is posted, the hand of the corresponding member shifts accordingly. This model or project is implemented with the help of a mobile application-"Contracts", Cloud MQTT, Google Firebase, Raspberry Pi 3 B, Paho Python, Node-Red, and a servo motor to shift the hands of the clock. This model can be used for vast applications.

Keywords— *Owntracks, MQTT cloud, Node red, Google firebase, Servo motor, Paho python*

1. INTRODUCTION

In today's modern era, technology is increasingly evolved at an exponential rate. The challenges we face today is entered and will be fundamentally different because they won't be solved by humans alone but through complex human-machine interactions. On the other hand, security is a trade-off, a balancing act between attacker and defender but it is more important to have it as a static one. Hackers can easily encrypt or decrypt into any system in no time and this has been a potential threat to almost each and every single person of this world. But, with the advent of the Internet of Things technology (IOT) and location-based sensors (such as GPS), we now have the ability for personal security solutions to include a location tracking capability. Rapid assistance from a danger or threat can now be a button push away. The capability to locate and track people and assets in real time is made possible with the Internet of Things technology. Google Maps can track our locations or send locations through a medium like social networks and we, in turn, can get the location from other people through those networks like Facebook, Whatsapp etc. But what if these social networks are tapped or the particular user device is tapped? Then, our location data can be easily misused or decrypted. For this very purpose, Smart Location Clock and the Display Device plays an important role in maintaining secured location data. Location tracking IoT solution starts with a location sensing device. This can be a hardware device built for that purpose or it could be an application installed on a smartphone which leverages the location sensing capability of the phone. The location sensor on the device attempts to triangulate the known points to provide a relative location. The location tracking device must have the capability to broadcast its location to a remote system (or Cloud platform) for processing. Depending on the data sent, the remote system will need to compute the location and translate it into something that can be understood by the consumer. Computing the location can be as straightforward as plotting geo-coordinates (both longitudinal and latitudinal). This system is very similar to the above process. OwnTracks is a simple and user-friendly application which serves the purpose of sending the user's coordinates directly to the cloud that is both latitudes and longitudes. This application can be downloaded from the Play Store which is free of cost available for both Android and Apple devices. The user's coordinates are directly sent to the Cloud MQTT with the help of Contracts. Here, MQTT or "Message Queuing Telemetry Transport is a Machine-to-Machine (M2M) Internet of Things" connectivity protocol, a method which can be used for communication between two or more machines. Cloud MQTT is an MQTT cloud used to store the latitudes and longitudes of the user and this is directly sent to Node-Red, a programming tool for wiring together hardware devices, APIs and online services in different ways. Various combinations of nodes are used together to plot the given latitudes and longitudes directly to the maps. The map with the plotted user's coordinates is then displayed on the display device or the system which determines the exact location of the user. On the other hand, the operation of the clock can be engineered with the help of the Raspberry Pi 3 B, couple of wires and a Servo Motor respectively. First, the required locations are designed on the clock accurately depending upon the user's choice along with insertion of a Stepper Motor at its center. The latitudes and longitudes can be sent from the cloud to Python script with the help of Paho module which is further sent to the Raspberry Pi's Python Script with the help of Google FireBase. And if condition compares the predefined latitudes and longitudes for each location with the user's latitudes and longitudes and if it matches, then the stepper motor rotates or moves to that point, imitating the design and the operation of a general clock. Given below is a detailed system architecture (Figure 1) and for the process which helps in understanding the flow.

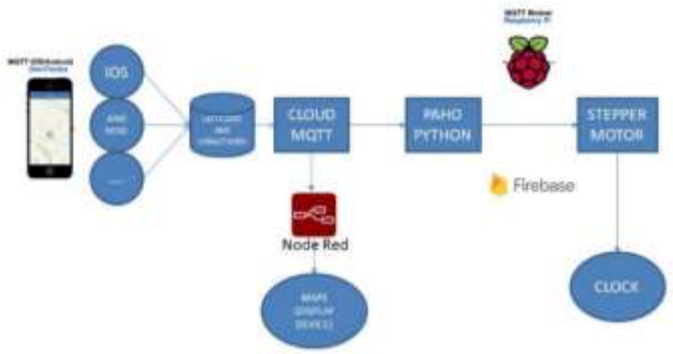


Fig. 1: System Architecture

2. PROBLEM STATEMENT

One of the rising problems these days is security ie the location data which can be easily tapped by the hackers. Hackers can track our location within seconds if our device or social networks are tapped by them. This can lead to misuse of location data and therefore less secure. Also, there are an only minimum number of solutions to send location data if there were no social networks. To send location data to specific people in a secured manner, the smart location clock device can be put into use. Placing it in a home, it can give the whereabouts of a person only if he/she is connected to the clock and the display device using Owntracks. The clock not only gives the appropriate locations but it also gives the exact location in the display device. A tap from the mobile application and the location is sent instantly to both the display device which maps it accordingly as well as the clock which points at the location thus enhancing the security of the location data of the user.

3. EXISTING SYSTEM

ETA Clocks (in-house movements) are analog wall clocks that tell you the location of your family members. The clock is made from oak plywood and laser-engraved with twelve destinations replacing the hour numerals respectively. Using the mobile applications for location tracking, the geolocation coordinates are sent to a secure server then received by the clock and the clock gets updated automatically. However, the exact location of a person will not be known using this ETA clock ie it cannot plot a location data on a map. A design of the ETA clock is represented below: (Figure 2)

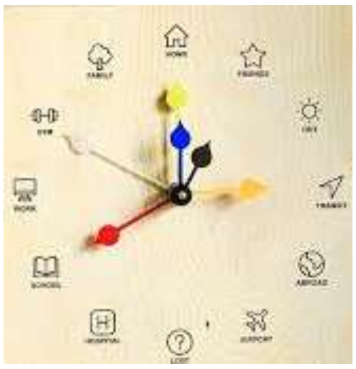


Fig. 2: ETA Clock

4. PROPOSED SYSTEM

The proposed system mainly consists of two modules- The Clock and the Display Device. In order to display the exact location in the map, it needs to map the coordinates or in other words -the latitude and longitude of a particular user. This is where OwnTracks application plays an important role. The OwnTracks application operates in one of two modes – MQTT and HTTP and the app are used according to the user's choice. In Private MQTT mode, IOS and Android app are configured to access the broker, and in Private HTTP mode, similar configurations are made but with an HTTP endpoint. But first for the OwnTracks to work, an instance of the plan "Cute Cat" must be created along with username and password in the Cloud MQTT console which helps to connect with the OwnTracks application. A server name, username, and password (where I represent the instance), port number will automatically be loaded by the MQTT Cloud in the "Details" Section. The flow can be seen in figure 3, 4, and 5.

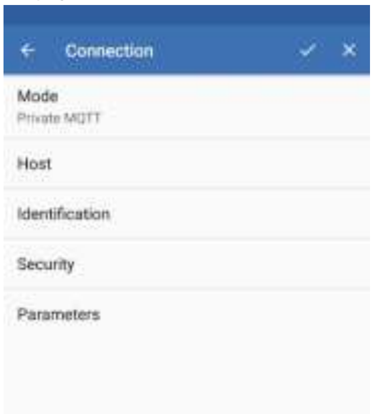
Fig. 3: Instance Creation

Fig. 4: Instances

Fig. 5: Instance Details

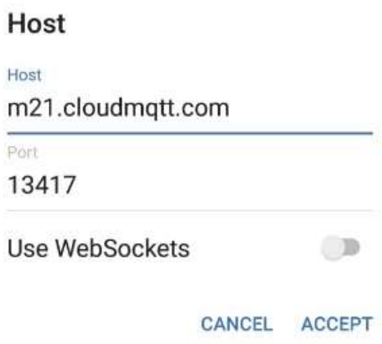
a) OwnTracks

For the Own track's application to work, four important things are to be checked- MQTT mode, Host, Port number, username, and password. The host here is the server name of the CloudMQTT. MQTT mode is private by default but it can be changed to the public which turns the location data to the public. The port number, host user from Cloud MQTT is copied and pasted. Device ID and Tracker Id is also given which is user-defined. Also, TLS ie Transport Layer Security is switched off which is optional. Once it is done, a tick mark is clicked at the upper right corner and the connection is established successfully. One of the excellent features of this application is that it can run on the background even after it is closed which makes it user-friendly. Now the latitude and longitude of the user are directly sent to CloudMQTT. The following flow can be seen in Figure 6, 7, 8, 9 and 10 respectively.



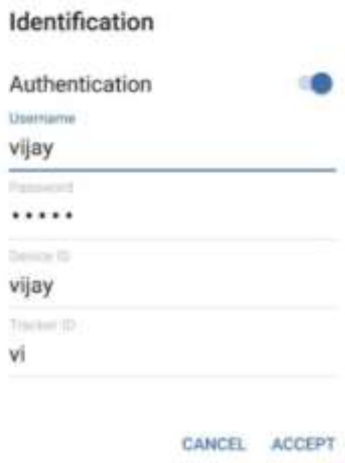
Mode
Private MQTT
Host
Identification
Security
Parameters

Fig. 6: Connection window



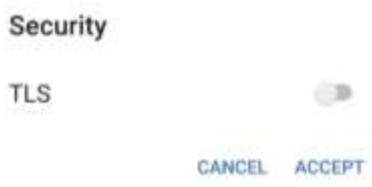
Host
Host
m21.cloudmqtt.com
Port
13417
Use WebSockets

Fig. 7: Host and Port Number



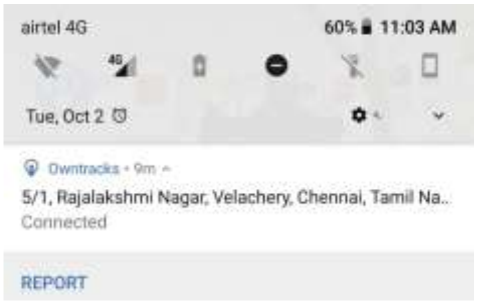
Identification
Authentication
Username
vijay
Password
•••••
Device ID
vijay
Tracker ID
vi

Fig. 8: Username, Password, Device ID, Tracker ID



Security
TLS

Fig. 9: TLS Disabled (Optional)



Tue, Oct 2
Owntracks • 9m
5/1, Rajalakshmi Nagar, Velachery, Chennai, Tamil Na..
Connected

Fig. 10: Connection Window with Location

b) CloudMQTT

CloudMQTT has managed Mosquitto servers in the cloud**.** Mosquitto implements the Message Queuing Telemetry Transport protocol, MQTT, which provides lightweight methods of carrying out messaging using a publish/subscribe message queueing model (figure 11). The instance which was created at Cloud MQTT is now accessed by Contracts application. If the OwnTracks app and Cloud MQTT are not connected properly, an error is showed indicating that the connection was not established successfully. In the case of an MQTT, one device can publish data to another device and the third device can subscribe to it thereby getting the data flow. MQTT allows both publishing and subscribing of topics which consist of the messages. Once the CloudMQTT is ready, the report is clicked from the app and if the connection is successful, the Cloud MQTT receives the user's latitude and longitude information (figure 12). The data can then be seen in the web socket UI in the CloudMQTT which consists of the topic, message dialog box for sending messages along with the received messages on the right-hand side. The topic is of the form own tracks/username/device ID and the received messages are in the JSON format.

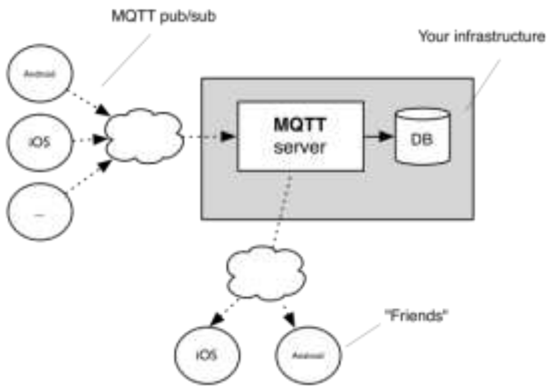


Fig. 11: Basic Flow of MQTT Cloud and MQTT client

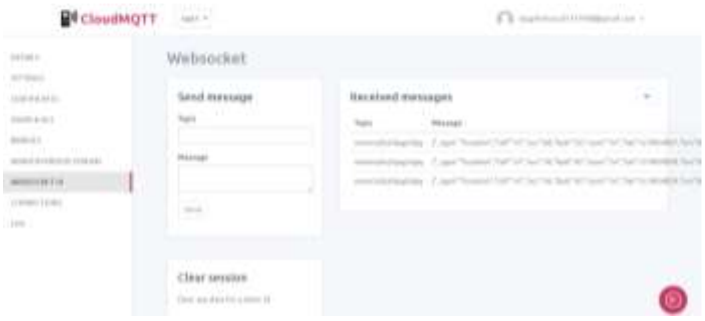


Fig. 12: Location Data of the user

c) Node-Red

Node-RED is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways. It provides a browser-based editor that makes it easy to wire together flows using the wide range of nodes in the palette that can be deployed to its runtime in a single click. Plotting the coordinates in the map is done with the help of node red. There are different types of nodes which are available in the Node library each serving a unique purpose. The basic concept behind Node-Red is to convert the given JSON string to Javascript Object consisting of only the latitudes and longitudes of the user and it is then directly fed into the world map node which plots the location. This can be duplicated or sent to devices such as tab, phone etc which acts as the display device. OwnTracks node can be downloaded from the node library which contains the hostname and port number as its parameters. Once it is done, hostname and port number are copied from CloudMQTT and pasted on the Own tracks node which connects the node directly to the CloudMQTT. Then it flows through the json node which converts the given location data to Javascript Object. Change node is used to remove the unwanted parameters thus containing only the latitudes and longitudes of the user. Finally, it is sent to the debug node and to the World map node wherein debug node displays the given location data in the debug messages console and the World map node plot the given latitudes and longitudes of the user. Once it is done, Deploy button is hit which starts the flow and then Ctrl+Shift+M is used to open the map. The world map shows the exact location of the user in the form of a small circle. The flow can be shown in fig. 13 and the map with the exact location of the user can be seen in figure 14.

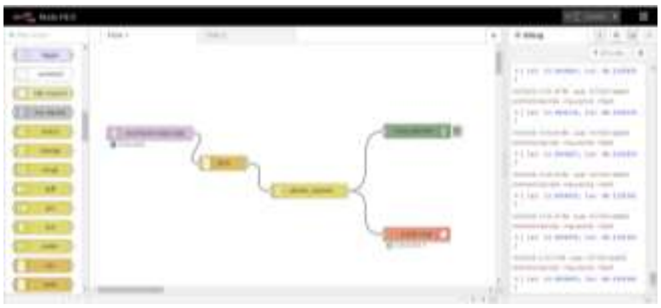


Fig. 13: Node-Red Flow

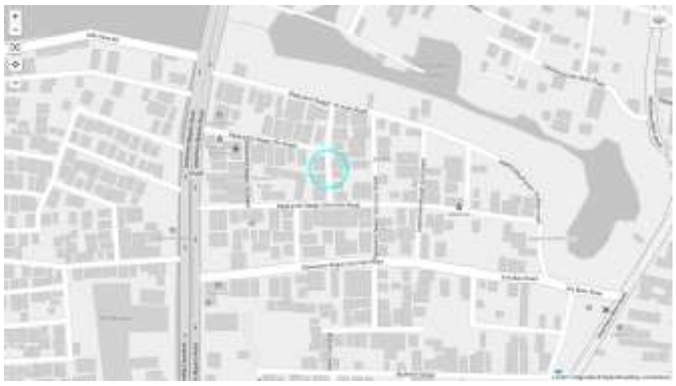


Fig. 14: WorldMap with user's location

d) Paho Package

To send the coordinates to the Raspberry Pi python script, the first step is to install the paho package in the system Python. It provides a client class which enables applications to connect to an MQTT broker to publish messages and to subscribe to topics and receives published messages. It also provides some helper functions to make publishing one-off messages to an MQTT server very straightforward. It supports Python 2.7.9+ or 3.4+, with limited support for Python 2.7 before 2.7.9. The paho module connects directly to the MQTTcloud using the following command.

"connect(host ,port=1883, keepalive=60)" Here, host is the hostname of the remote broker and port is network port of the server host to connect to. The hostname and the port number is directly copied from CloudMQTT to the Python Script. This process connects the python script CloudMQTT. In the code, there are two functions – on_connect and on_message where the former helps to connect to the cloud and subscribe to the topic whereas the latter helps in displaying the message. Once it is done, a message pops up showing that the connection was successful. Then the JSON String is edited by importing the JSON module and it is converted to a dictionary where the latitudes and longitudes can be extracted separately. They are stored in separate variables which are then printed out to the Python Shell. The following output is obtained (figure 15).

Fig. 15: Coordinates of the user (lat,lon)

e) Google Firebase

It is a software development platform owned by Google that reduces the hassle of establishing proper infrastructure, provides analytics about user behavior, and facilitates rapid application deployment. Firebase provides a realtime database and backend as a service. The service provides application developers an API that allows application data to be synchronized across clients and stored on Firebase's cloud. It also enables integration with Android, IOS etc. The database is also accessible through a rest API and bindings for several JavaScript Frameworks. The rest API uses the server sent events protocol, which is an API for creating HTTP connections for receiving push notifications from a server. It provides a user-friendly feature and enhances data security. Now, this module is used to send the location data to the cloud which is then retrieved by the Raspberry Pi 3B. Google Firebase can be called using the Firebase package in Python. To send data to the cloud, the variables containing the latitudes and longitudes are imported and it is sent to the firebase. GET and POST is the two major methods used for accomplishing this particular task. The POST is used to post the data directly to the firebase and GET is used to acquire the data from the firebase and this is used by the Raspberry Pi in order to receive the location data. To access firebase, a new project id is created which in turn generates an API key which is used by the python script to access the particular project id. Project id with Web API key can be seen in figure 16. If the connection is successful, along with the methods, Firebase receives the latitudes and longitudes of the user. It is then sent to the Raspberry Pi's python script using the GET method which acquires the coordinates.

Fig. 16: Project Id with Web API key

Fig. 17: Location data in Firebase

f) Servo Motor

The Python Script of the Raspberry Pi contains the location data which is to be sent to the Servo Motor. Raspberry Pi is an open-source microcontroller consisting of 40 open GPIO pins. Here GPIO stands for "General Purpose Input/Output", which means these pins can either send electrical signals to drive hardware or receive them and read sensor data. The clock hand is placed on top of the servo motor imitating the exact design of the clock. A servo motor is a type of DC motor that, upon receiving a signal of a certain frequency can rotate itself up to an angle. The way a servo motor reads the information is being sent by using an electrical signal called PWM. PWM stands for "Pulse Width Modulation" which just means sending ON electrical signals for a certain amount of time, followed by an OFF period, with repeated hundreds of times a second. The amount of time the signal is on sets the angle the servo motor will rotate too. In most servos, the expected frequency is 50Hz or 3000 cycles per minute. Servos will set to 0 degrees if given a signal of .5 ms, 90 when given 1.5 ms, and 180 when given 2.5ms pulses. This translates to about 2.5-12.5% duty in a 50Hz PWM cycle. PWM signals are sent from one GPIO pin on the RPi, and it is powered from the GPIO board, so three wires will run from the servo to the Raspberry Pi. The Circuit Diagram can be seen in figure 19. Also, the servo motor is placed in the center of the clock for free movement. The main concept is that the servo motor must move after checking the predetermined coordinates matches the user's coordinates. If they are, then the clock hand moves to that particular place or position in the clock design. For example, if the user's coordinates and the predefined coordinates are destined to home and if they are equal, it means that the user is present at home and hence the hand moves towards "HOME". There can be 12 different locations and depending on that, predefined coordinates for each location must be assigned. If it does not satisfy at least one of the any given locations, then the user is termed as "LOST" which can be seen in the given design of the clock. Also, all the python scripts are automated to keep track of the location data. For proper implementation, a python script is written which uses an if condition to compare the predefined coordinates with the user's coordinates, thus giving the appropriate location and the exact location at the same time. The Servo motor and the clock can be seen in figure 18 and figure 19 respectively.

Fig. 18: Servo Motor

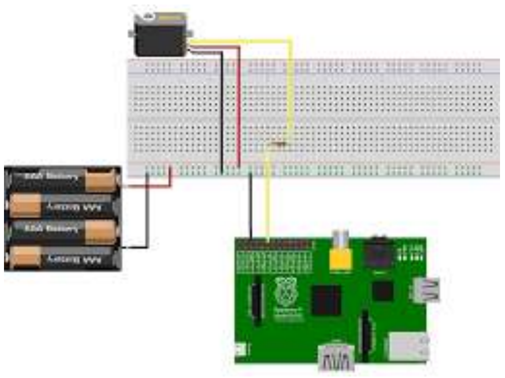


Fig. 19: Circuit Diagram

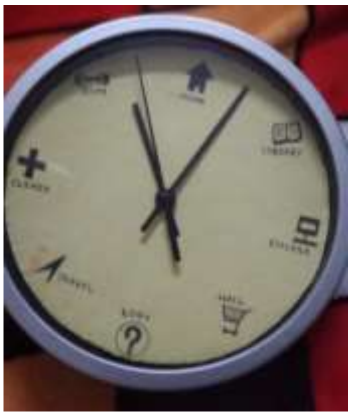


Fig. 20: Smart Location Clock

5. FUTURE RESEARCH

The Smart Location Clock is a prototype clock which displays the appropriate user location. Also, various MQTT clients can be easily connected to the cloud. In the near future, multiple hands can be fitted onto a single clock to keep track of multiple location data including publishing and subscribing of data. The clock can also be miniaturized into watches, allowing it to track location data when in motion. This can be used for various applications. On the other hand, the display device can be remodeled to LED screens, workspace computers for an easy and wide range of map view.

6. CONCLUSION

In this project, Smart Location Clock was built to keep track of a particular user along with ease of access with user-friendly options. It was done using simple platforms like Node-Red, Google FireBase, Python etc. A deeper research led us to an array of devices that make use of the technology completely in a different way, thereby offering us not just convenience but solutions to some of the most complex concerns as well ie security in this case. There are infinite solutions to the Internet of things which finds applications in several fields, including electrical, medical, information technology, universal space research, and so on. To understand better about IOT, we must know about M2M, which stands for the machine to machine, and machine to mobile connections that are used for connecting electrical and electronic devices, and other various systems intelligently. Each application is built or upgraded in different ways serving a unique purpose, thereby paving the way to human-machine evolution.

7. REFERENCES

[1] Goel and V. Gruhn, "Fleet Monitoring System for Advanced Tracking of Commercial Vehicles", Proceedings of the 2006 IEEE International Conference on Systems, Man, and Cybernetics (SMC 2006), pp. 2517-2522, Taipei, Taiwan, 08.10.2006-11.10.2006.

[2] Chia-Hung Lien, Chi-Hsiung Lin, Ying-Wen Bai, Ming-Fong Liu, and Ming-Bo Lin, "Remotely Controllable Outlet System for Home Power Management," Proceeding of 2006 IEEE Tenth International Symposium on Consumer Electronics (ISCE 2006), St. Petersburg, Russia, pp. 7-12, June 28-July 1, 2006.

[3] E. D. Kaplan, Understanding GPS: Principles and Applications, Artech House Publishers, ISBN 0890067937, February 1996.

[4] Junaid Ali, Shaib Nasim, Taha Ali, Naveed Ahmed and Syed Riaz un Nabi, "Implementation of GSM-based Commercial Automobile Tracker Using PIC 18F452 and Development of Google Earth Embedded Monitoring Software" Proceedings of 2009 IEEE student conference on

Research and development(SCOReD 2009), 16-18 Nov 2009, UPM Serdang, Malaysia,

[5] M. Mcdonald, H. Keller, J. Klijnhout and V. Mauro, “Intelligent Transport Systems in Europe: Opportunity for Future Research” World Scientific Publishing Company, ISBN 981270082X, 2006.

[6] Muhammad Ali Mazidi, Janice Gillespie, Mckinlay, Rolin D., “ The Microcontroller in Embedded System: using Assembly and C,”

[7] Real-Time Vehicle Tracking System using GSM and GPS Technology-An Anti-theft Tracking System. Available from: https://www.researchgate.net/publication/266412980_Real_Time_Vehicle_Tracking_System_using_GSM_and_GPS_Technology-An_Anti-theft_Tracking_System

[8] Scott Pace, Gerald P. Frost, Irving Lachow, Dave Frelinger, Donna Fossum, Don Wassem, Monica M. Pinto. The Global Positioning System, Assessing National Policies, Appendix B: G PS History, Chronology, and Budgets, monograph/report products, Rand Corporation. http://www.rand.org/pubs/monograph_reports/MR614/MR614.appb.pdf

[9] “Temex Times Galileo”, GPS World, Feb. 1st, 2006 http://www.gpsworld.com/gpsworld/article/articleDetail.jsp?id=300299

[10] Crowley (1998), “Virtual logistics: transport in the marketspace”, International Journal of Physical Distribution & Logistics Management, Vol. 28 No. 7, pp. 547-574.

[11] DELHI Transportation Corporation case study (2007), http://www.cmcltd.com/case_studies/transportation/GPS_System _Case_Study_DTC.pdf

[12] General Information Document of GPS-based Fleet Management and Tracking Systems (2008), Exaterra Inc., Ottawa, Canada: Canadian company based in Ottawa.

[13] Hariharan, Krumm, Horvitz (2005), “Web-Enhanced GPS”, School of Information and Computer Sciences, University of California, Irvine, USA.