data

MDPI

Article

Congestion Adaptive Traffic Light Control and Notification Architecture Using Google Maps APIs †

Sumit Mishra [1,*], Devanjan Bhattacharya [2] and Ankit Gupta [3]

[1] Research Consultant, Learnogether Technologies Pvt. Ltd., Ghaziabad 201014, India
[2] Nova Information Management School, Universidade Nova de Lisboa, 1070-312 Lisbon, Portugal; dbhattacharya@novaims.unl.pt
[3] Department of Civil Engineering, Indian Institute of Technology (Banaras Hindu University), Varanasi 221005, India; ankit.civ@iitbhu.ac.in
* Correspondence: sumitmishra209@gmail.com; Tel.: +91-945-877-9491

† This paper is an extended version of "Mishra, S., Bhattacharya, D., Gupta, A., and Singh, V. R. (2018) "Adaptive Traffic Light Cycle Time Controller Using Microcontrollers and Crowdsource Data of Google Apis For Developing Countries", *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.* IV-4/W7, 83-90, https://doi.org/10.5194/isprs-annals-IV-4-W7-83-2018".

Received: 19 September 2018; Accepted: 12 December 2018; Published: 14 December 2018

Abstract: Traffic jams can be avoided by controlling traffic signals according to quickly building congestion with steep gradients on short temporal and small spatial scales. With the rising standards of computational technology, single-board computers, software packages, platforms, and APIs (Application Program Interfaces), it has become relatively easy for developers to create systems for controlling signals and informative systems. Hence, for enhancing the power of Intelligent Transport Systems in automotive telematics, in this study, we used crowdsourced traffic congestion data from Google to adjust traffic light cycle times with a system that is adaptable to congestion. One aim of the system proposed here is to inform drivers about the status of the upcoming traffic light on their route. Since crowdsourced data are used, the system does not entail the high infrastructure cost associated with sensing networks. A full system module-level analysis is presented for implementation. The system proposed is fail-safe against temporal communication failure. Along with a case study for examining congestion levels, generic information processing for the cycle time decision and status delivery system was tested and confirmed to be viable and quick for a restricted prototype model. The information required was delivered correctly over sustained trials, with an average time delay of 1.5 s and a maximum of 3 s.

Keywords: driver information system; real-time traffic signaling; road traffic congestion; Google Traffic API; agent-based traffic modeling

1. Introduction

Traffic congestion is a major problem that is growing exponentially in metropolitan cities due to the increasing demand for private vehicles combined with limited land resources. Traffic results in longer travel time and the waste of billions of human resource hours, waste of fuel, degradation of the environment, growing accident rates, and largely reduced service efficiency of roads. Traffic control targeted by an Intelligent Transportation System (ITS) helps to relieve congestion that is adding growing pressure to existing road infrastructure. An ITS system works by integrating information and communications technology (ICT) and electronic technologies with transportation infrastructures and vehicles. This integration can help in relieving traffic congestion, increasing safety, and reducing travel time and fuel overuse. Congestion-adaptive traffic light control is a pivotal factor for increasing the throughput of roads and reducing travel time [1,2].

In developing countries, nowadays, traffic lights mostly operate on fixed cycles or are manually controlled by a traffic inspector two or three times a day according to congestion characteristics. These manual and fixed solutions aim to sort out problems on road sections with low traffic flows, but, for the major sections, such solutions are not effective due to short temporal and spatial congestion changes. Also, this type of method introduces human evaluation errors and incorrect green-time balancing [3]. Further, the anxiety and stress of the driver also increase if they hit consecutive red lights [4,5]. Traffic congestion may also impose life-threatening scenarios due to psychological stress placed on the driver. The red light running (RLR) phenomenon that can cause an accident is a rash driving act. This RLR act mainly results from the frustration caused by short or long cycle lengths that the driver feels to be unjustified. Sometimes, too short a cycle length is adopted by authorities to deal with high traffic density. However, a short cycle length often fails to manage traffic queues of different adjoining roads on an intersection and may lead to waiting for more than two cycles before crossing the road junction.

With the advent of the technological era, wireless technologies for interconnectivity and data processing technologies, like big data on mobile and cloud computing, are evolving rapidly. This has engendered cyber-physical systems (CPSs) to alleviate public and industrial problems. CPSs are computational systems that combine many physical processes in a way that provides a reliable, evolvable, networked, and customizable facility for real-time needs [6]. In particular, cloud computing with CPS processes of scheduling, management, and control of resources in real-time allows complex systems, such as cloud-integrated manufacturing and cloud-integrated vehicles, to be deployed effectively. This has led to vehicular cyber-physical systems (VCPS), which aim to solve telematics applications that need decision-making and autonomous control [7,8]. The services and applications in a VCPS often form multiple end-to-end cyber-physical flows that operate in multi-layered environments [9]. For empowering CPS and VCPS, the Internet of Things (IoT) plays a major role. IoT is augmenting physical devices with capabilities like sensing, computing, and communication so that they can form a network and leverage the usage of the collective effort of networked objects [10]. Due to their powerful processing and endless application opportunities, these smart objects offer solutions to industrial problems. These smart objects of IoT can be connected to mitigate problems of transportation when joined with a CPS, which leads to a VCPS. This can be holistically defined as an ITS. It is only possible with the intervention of technologies that manage traffic light synchronization accurately with millisecond margins, along with many age-old ITS problems, like adaptive control, that can be solved with great ease.

Traffic congestion can be tackled by demand management of a given road intersection by adjusting a traffic light's cycle time according to the live congestion situation. Hence, real-time or near-real-time traffic data prediction is of prime importance. For collecting these traffic data, conventional methods using a WSN (Wireless Sensor Network) [11] and other sensors have limitations, such as coverage due to a sensor's fixed location, and cable-based connections increase the initial cost of implementation and maintenance [6]. There are several platforms and APIs (Application Program Interfaces) that leverage various sensors, a vehicular network, and crowdsourcing from various other technologies to piggyback on the data of live traffic status. The technique proposed here intends to use APIs provided by one of the major companies, i.e., Google, to monitor the tail of traffic congestion. Thus, for optimizing congestion, it eliminates the need for the initial infrastructure used for tracking vehicles, and, therefore, it cuts the cost involved with the deployment of the sensor network.

In this study, a method and system architecture were adopted for reducing the traffic density. The proposed method adjusts the knowledge-based cycle time of a traffic light in accordance with the congestion. Also, the system aims to inform the driver about traffic light status ahead of time so that they can be prepared mentally and manage speed according to the green signal time. This will also prevent stop-and-go events, which will save fuel and limit excess pollution that is emitted, as the ignition of an engine uses more fuel than continuous travel [12,13]. Moreover, the system proposed

must be fail-safe and heal itself after any temporary failure of the short wireless network. In the case of permanent failure, the situation can be easily monitored remotely by the authorities.

The remainder of the paper is outlined as follows. Section 2 comments on the main literature works and comparative studies on traffic monitoring techniques, traffic light status reporting systems, Google APIs, and a detailed analysis of various IoT technologies. Section 3 describes the system methodology and the proposed architecture of the overall scheme. Section 4 analyses and explains the scheme proposed here, with programming concepts involved in the innovative and dynamic traffic light time management. Section 5 presents a case study and the method's implementation, while Section 6 reports the conclusions.

2. Related Work and Comparative Technological Discussion

2.1. Traffic Congestion Acknowledgment Techniques and Traffic Light Status Reporting System

For working on congestion problems, database predictive or real-time knowledge of traffic is needed. This knowledge may come from any source. For knowledge gathering, traditional techniques include the use of sensors such as pneumatic tubes, automatic traffic counter/classifiers (ATCC), etc. However, these tubes not only have a short life but also lack the ability to detect parallel vehicle movement across multiple lanes. Another type of sensor is the induction loop, which needs road cutting for deployment. Also, it has issues like high maintenance costs and low accuracy in back-to-back traffic (slow moving traffic). Video/image processing for optical number plate recognition (ONPR) can also be adopted, but it is limited by its low accuracy on account of non-standardized number plates and visibility problems in foggy weather [14]. Currently, many traffic light systems use sensor node networks, like a WSN, to measure traffic density [15]. Although it is common, the deployment of a WSN needs substantial capital investment and involves considerable maintenance costs, and there are security issues that are a primary concern when using a WSN. A very good tabular-comparative discussion on different sensing technologies is found in [16].

In addition to infrastructure-based solutions, there are many other infrastructure-free vehicular networks (vehicle-to-vehicle (V2V)) discussed in the literature [1,2,17]. Vehicular cloud computing is a promising solution to meet the requirements of vehicular ad hoc network (VANET) applications and services. In light of this, the work in [18] lists the challenges, architectures, and future directions of the vehicular cloud paradigm V2V. When a V2V network is accessed via some infrastructure, i.e., vehicle-to-infrastructure (V2I), then knowledge of the real-time traffic situation can be obtained easily. In [19], a discussion on traffic prediction and estimation using both V2I and V2V communications is provided, along with the application of dynamic route choices. Also, several related works using fuzzy logic and rules for intelligent traffic light control can be found in the literature. Neural networks and learning, generic algorithms, and some hybrid techniques combining fuzzy logic and a generic algorithm are also used for controlling an isolated intersection traffic light [16–18]. So, this information illustrates that, in mixed heterogeneous traffic, a single algorithm or technique cannot be used; mixed techniques are discussed in [17].

The Wisdom Web of Things (W2T) vision relates and shows the interconnections between the social world, the physical world, and the cyber world [20–22]. This is a holistic view of understanding the interaction among humans, computers, and smart devices. The context-aware big data collected has helped in the planning of cities [23]. For this type of algorithm to work effectively, it is necessary to deploy huge processor resources, perform data mining, and enable very good user accessibility of good crowdsourced data. Google, Bing, and other data giants collect such crowdsourced data and provide the result in their APIs. Google Maps uses various sources for traffic data. This depends on the availability of hardware-intensive sensors, personalized network availability and anonymized traffic data, local road sensors, car/taxi fleets' private monitoring network, etc. Crowdsourced, anonymized traffic data are collected from people using the Google Maps application or other Google services on certain smartphones, including Android and personal digital assistant (PDA) devices.

Initially, the GPS functionality is set to *'applied'* by default to relay location data back to the Google server. GPS-determined locations are analyzed and transmitted by a large number of cellphone users. Using the data, user speeds along a stretch of road are calculated by Google to generate a live traffic map through the use of machine learning methods, to build predictive models for traffic.

While processing, Google filters out anomalies, such as vehicles whose characteristic is different from the nearby fleet, like if it is making frequent stops. When a threshold of users in a particular area is noted in a fleet, the color track changes on their map services. The solely mobile-based crowdsourced data is less reliable than the sensor-based and crowdsourced mixed data that Google uses for highway traffic. This increases the reliability of highway and main road statuses reported by Google compared with that of local streets. Presumably, this technological gap will shrink over time with the advent of smart cities and the adoption of Google Android-enabled smartphones. For the places where traffic tracking infrastructure cannot be placed due to either the huge capital investment required or wide area involved, the above-described method is a very good alternative and a solution that fits the bill. However, to be exact, there is no precise answer to the question of the accuracy of Google API results, as it changes with time, place, inputs to algorithms, and the regular updates to algorithms as well. However, in countries like the USA, accuracy can be very high and can be considered to be more than 90%.

A report on a survey done at 'the Place de la Nation', a plaza in southeastern Paris, stated that the comparative accuracy may vary and fluctuate [24]. Along with crowdsourced data from user PDAs, Google retrieves data from the cameras of road authorities, and Google Street-View cars have now driven more than 7 million miles in the USA. Other sources were also discussed by many high-ranking officials of Google Maps in an interview reported in the article "The Huge, Unseen Operation Behind the Accuracy of Google Maps" [25]. These data, viz. street width and others, are then fed to smart algorithms for prediction. However, any official authority can fail to provide accurate data in numbers since it varies; however, these data are used by independent transportation companies and other agencies and have turned out to be quite beneficial for them.

There are some works in the literature that use APIs from these sources to acquire and analyze data for predicting traffic jams [17,26]. Other works have aimed to present a lightweight web-based tool for assisting traffic engineers that provide an engineer-friendly way to interact with and explore traffic volume statistics [27,28]. Some works even leverage the use of social networks, namely, Twitter for road traffic prediction [29,30]. Moreover, there are several systems that use image and vision-based processing to detect the traffic light status from a distance for traffic light status reporting. One such system developed by researchers from MIT and Princeton University is SignalGuru [31], which relies on the collection of mobile phones to transmit status data, which are calculated using photos grabbed by an iPhone. Another system using images tries to solve the problem with a lower probability of error using a probabilistic location model of a traffic light [32]. These systems fail in low-visibility conditions, such as rain, fog, or snowfall.

There are very few associations summarized in the literature that have a direct interconnection with users, i.e., the driver and the road traffic-light management facility. Urban Traffic Management Control (UTMC) of Newcastle also worked on in-vehicle communication systems directly with the center. The device in an ambulance, which can be fixed on the vehicle's windscreen like a Sat-Nav, can detect traffic lights from 100 m away and can switch them to green, as well as inform the predictive speed to the other drivers, allowing the ambulance to adjust its speed so that it passes through a series of green light signals by avoiding the red signals [33]. For commercial purposes, Audi provides a great V2I solution for some cities that enables the car's communication with the infrastructure in selected cities across the USA. The On-board LTE data connection is used to link vehicles and the infrastructure of traffic technology services, including the server of the traffic management system [34].

2.2. Adaptive Traffic Light Cycle-Time Controller and Methods

The traffic congestion knowledge obtained by different methods is leveraged to control traffic lights. Adaptive traffic control systems—architectures and methodologies—attempt to bridge the gap, an effort primarily started by FHWA projects. Further, the ACS-Lite system developed by FHWA proves to be a cost-effective solution for applying adaptive control system technology to current, state-of-practice closed-loop traffic signal control systems [35,36]. The literature for adaptive control of traffic light systems is very rich, constituting of a variety of proposed work, such as that listed below [37]:

- Level 0 system: involves fixed-time and actuated control (TRANSYT, 1969, UK)
- Level 1 system: involves centralized control, offline optimization with more than 50 installations worldwide (SCATS, 1979, Australia)
- Level 2 system: involves centralized control, online optimization with more than 170 installations worldwide (SCOOT, 1981, UK)
- Level 3 system: involves distributed control, model-based with five installations in the USA (OPAC, RHODES, 1992, USA)
- Level 4 system: involves distributed self-learning control (MARLIN-ATSC, 2011, Canada)

There are papers that discuss algorithms, methodology, and related framework. For example, one work describes optimized traffic-signal controllers by statistical and prediction adaptive algorithms for dynamic queue length estimation using sensors [38]. The introduction of an adaptive signal control system with online signal timing updates along with real-time delay estimation is also reported in the literature [39]. Model predictive control (MPC) theory is also reported to be applied to a network-wide traffic controller [40]. Another work discusses the design of a system [41] which manages traffic light controllers by utilizing a traffic system communication algorithm (TSCA) and a traffic signal time manipulation algorithm (TSTMA). V2V and V2I using wireless communication between cars and a fixed node-based adaptive traffic light system are also deployed at intersections, as reported in [42]. Control algorithms combining fuzzy logic controller (FLC) and generic algorithms (GAs) are discussed in a literature review [43]. Further discussion on systematic instability analysis, higher performance, and controller design is presented in [44].

However, adaptive control for transportation is still not mature, and there is no agreement for universal models, e.g., the macroscopic and microscopic models that take average parameter values to calculate cycle time using signal system timing design software [45]. Optimization- and rule-based adaptive control strategies are two subcategories of adaptive control. Optimization-based strategies mainly deal with the computational process and performance. Artificial intelligence approaches, like reinforcement learning and others, are under development in the machine-learning community, and they offer key advantages in this regard. Optimization-based strategies incur complexity and fail to guarantee holistic optimal control, as they consist of many short-term optimizations based on different models. Rule-based strategies rely on preset rules, such as those embedded in signal controllers at some isolated intersections and the generalized adaptive signal control algorithms at other specified intersections [46,47].

We interviewed some traffic personnel (low-level, high-level, and traditional traffic light manufacturer). Interviewees highlighted that governments in developing countries like India want to deploy knowledge-based (rule-based) traffic light cycle timing along with agent control. This avoids computation complexity and helps to deal with the uncertainty of the dynamic road environment in developing countries. These knowledge-based data are pre-calculated using different traditional models and parameters for a given intersection. Also, agent-based technology is rapidly emerging as a powerful computing paradigm to cope with the complexity of dynamic distributed systems [48]. Traffic management systems integrated with mobile agent technology along with multi-agent systems have also been previously proposed [40]. Hence, from the entirety of the review given above, it is clear that a novel method is required for scenarios in developing countries, where investment is non-existent

but experimental solutions are welcome. This study aims to map congestion based on knowledge from an API and then implement adaptable cycle time along with providing the details on the architectural view of agent-based control and a traffic light status notification system to drivers. The present methodology and system compose the first architectural presentation of its kind, as revealed by the literature review.

2.3. Comparative Study and Analysis of Different Google APIs for Congestion Tracking

Google provides a live traffic status in several forms. The status can be represented by different color tracks in their map services or in the form of estimated time of arrival (ETA) at the destination in according to the congestion between the origin and destination. In this paper, we are concerned with the live traffic data for the desired route, which can be extracted both ways from the respective APIs.

Google Maps provides numerous JavaScript APIs, which have a vast variety of API functionalities for map editing, one of which is the traffic layer API [49] that gives the option to observe live traffic in four colors. With the help of the traffic layer API of Google Maps, we retrieved a map which has a lot of cluttered clusters, like water bodies, parks, building structures, local roads, etc. Google provides the *'MapTypeStyleFeatureType'* object specification and *'MapTypeStyleElementType'* object specification to manipulate the map [50]. So, Google gives the user the choice to edit roadmaps and manage the data, providing a good platform for a developer to experiment with the API. Google JavaScript API response data can be leveraged to extract congestion information by removing clutter and being left with color-coded traffic information on the map. Afterward, image processing is required to get the data on color pixel count, where clutter has certainly created a lot of distortion in the output. However, removing the clutter and reverse engineering the image to get the congestion status will create an error expected to be only about 2–3%, as the 'R, G, B' values of particular color pixels are not fixed and fall within a range.

To get the estimated time of travel/arrival, there is an API called *'Google Maps Distance Matrix API'* [36,51] which provides the travel distance and time for a matrix of origins and destinations. For calculating the ETA, the *'best_guess'* traffic model of the API can be used to specify the assumptions to use when calculating time in traffic. This model setting affects the value returned, after using a mixture of historical traffic conditions and live traffic, in the *'duration_in_traffic'* field. Live traffic becomes more important the closer the *'departure_time'* is to real time. The best estimation of travel time is predicted by extrapolating historical (time-of-day and day-of-week) traffic data to the future. This makes it easier to predict how long it will take to get somewhere. As defined, a delay in excess of that which is normally incurred under light or free-flow travel conditions can be taken as congestion. So, unacceptable congestion is travel time or delay in excess of an agreed-upon norm. These norms may vary due to the type of transportation facility and mode, geographic location, and time of the day. So, the travel time estimate is the better way to map the congestion while also being a very common congestion measurement parameter in the transportation research community. Also, Google ensures and approves of this type of data for non-commercial research usage [36,52].

3. System Methodology and Architectural Overview of the Proposed Method

The method proposed here builds on the usage of knowledge-based cycle time data [53]. Taking infrastructure-based knowledge and other parameters into account, the data of different cycle times and split times (time-of-day plan) are collected according to the congestion scenario for 830 different intersections in Delhi, India. In fact, similar datasets are available for other major cities, too. These cycle times or time-of-day plans are used in sets of hours according to congestion scenario patterns. However, congestion build-up can be treated as a stochastic process; therefore, the traditional period of *'set-of-hours'* is a very crude approximation. So, for tracking more appropriate near-real-time congestion scenario, the live crowdsourced data of the Google API are used. The proposed system informs the user about the status of the traffic light on the driving route. Due to the early diagnosis of the status of the next upcoming traffic light, it helps drivers to manage their speed from low to high but under

the safe speed limit so that, if possible, a user can pass through a traffic light without stopping. Also, if it is not possible to cross the traffic light without a stopping, then the user can slow the vehicle's speed. This will help the driver in two ways. First, from the early information, the driver can prepare their psychological mindset for the possibility of an upcoming red light, so they are not as likely to commit red light running and break traffic laws. Second, slowing the vehicle spreads the given traffic over the full-length infrastructure of the roads and thus decrease the traffic density at a particular site, which prevents traffic congestion and jams at traffic lights.

The proposed method performs a congestion status check at the point of deployment, i.e., from the center point of the road intersection up to the next traffic light in all directions of joined lanes, as presented in Figure 1. However, the congestion building curve suggests that to prevent choking, congestion must not accumulate to a defined level for a given infrastructure. If it crosses that level, it will take a long time to settle down to normal. Therefore, in the Google API request, we set the time input parameter as *'now'* (the actual time of request). Thus, it will manage the congestion from the origin up to the destination from *'now'* to until the destination is reached. The Google API responds to this query by providing an ETA considering the real-time and near-future data extracted by extrapolating previous records using their machine learning algorithm. Also, the API response string contains averaged ETA data. Afterward, the status-calculating algorithm defines the congestion status in three different quantized levels according to the processed data grabbed by Google's distance matrix API. The Google API response data provide the average time to travel between the origin and destination considering the congestion on the way. For a fair quantization and partitioning, three levels (1, 2, and 3) of traffic congestion statuses were chosen: no or low congestion, medium congestion, and high congestion, respectively. As the infrastructure knowledge-based data of each intersection for Delhi is mostly divided into three or four different cycle times according to congestion hours, we decided to use three levels in order to avoid any quantization error.

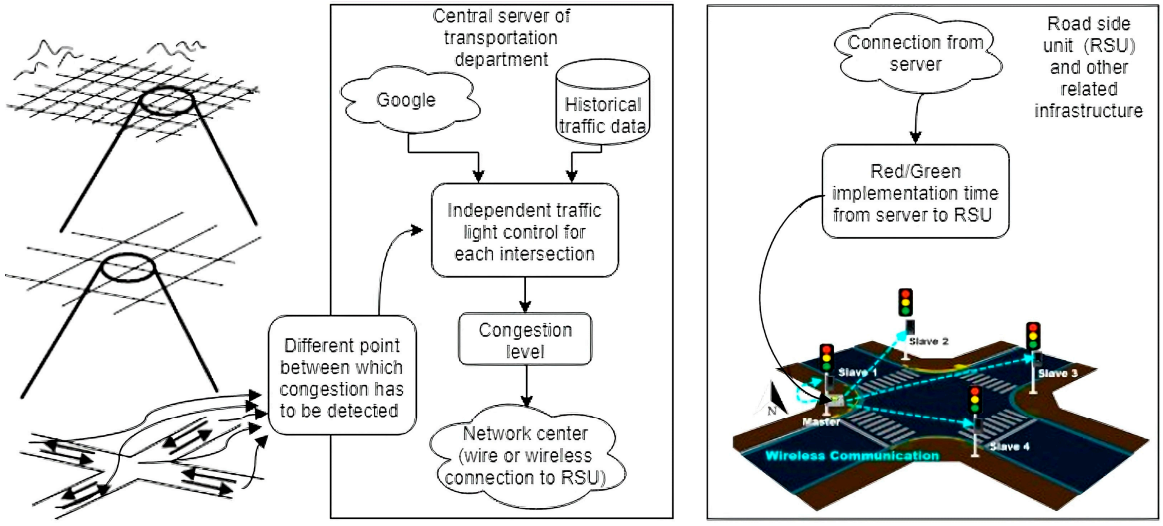


Figure 1. Proposed interconnected network.

Cycle time and split time infrastructure knowledge-based data for congestion statuses 1, 2, and 3 have to be loaded into the system. In order to deploy a change in the cycle time for a given road intersection, a small network of microcontrollers is needed. The central microcontroller server pulls the calculated data of the implementation time of each slave. After that, wirelessly, it transmits the decided cycle time and corresponding implementation time to each slave microcontroller. Each slave resides in each traffic light of an intersection. Further, the central microcontroller server is connected to the internet, as shown in Figure 1, and the slave microcontrollers are placed clockwise from the geographical north, starting from slave '1', so that the user can predict an upcoming slave in their path. This will help the user's device application to know the particular slave number for which

an associated time must be fetched from the server. Otherwise, the user's device application may also ping their approaching direction with the request, and the server itself will provide the required data. As discussed, the Google API does not publish its live congestion status data explicitly in digital form but mapping the response of the estimated travel time is possible. To map it, we followed a three-way process:

1. The origin is set at the central coordinates of the crossing with traffic light infrastructure, and the destination points are set to an appropriate distance on each track joined to the intersection.
2. All the differences 'D' (D = Estimated times to arrival provided by Google API − Averaged estimated time to arrival, grabbed by Google or provided by a road authority of each road lane) of each lane joining the intersection are added in accordance with the weight factor of each road to calculate the congestion value.
3. The calculated value is compared, as mentioned in Table 1, with the maximum congestion value for the same hour from the previous week.

The status-calculating algorithm takes the weight factor of each road, rated on a scale of '1', as input provided by the road authority personnel. Usually, there is no need for a weighting factor other than 1 to be uploaded for different lanes by a road officer, but it can be changed to a value near '1' in case the road authority feels that the Google API value is not correct. As presented in Equation (1), the congestion value is calculated by multiplying the weight factor with the difference between the initially grabbed estimated time to travel on each road and the averaged general time to travel. In Equation (1), 'n' is connected road number, and 'N' is the total number of connecting roads to an intersection. Also, a road authority may input an *'Average time to travel'* value instead of using the value provided by Google itself. Google keeps updating the value of averaged time to travel as per their algorithms. This averaged time to travel does not need to be changed very frequently, but it is updated once every 3–4 weeks for different places. In case traffic personnel want to provide more appropriate updates, they can do so. The origin-destination (O-D) API of Google is reliable, as found in a study [54]. So, there is no need to retrieve the average estimated time of arrival from the road authority, but it may help if there is any road infrastructure maintenance. For status comparison, the maxima and minima are grabbed by Google API from the previous week's data for each hour. This process involves extracting and comparing the congestion values to the maxima and minima for each hour, as stated in Table 1, to optimize the congestion condition on an hourly basis with concurrently small temporal cycle time adjustments in that given hour.

$$\text{Congestion Value} = \sum^{N} [\text{weight factor lane 'n' (Difference of estimated time for incoming lane 'n' + Difference of estimated time for outgoing lane 'n')}] \quad (1)$$

Table 1. Congestion status lookup table.

Congestion Value (CV)	Congestion Status	Type
CV ≤ (Maximum hourly congestion value of last week + 2 × Minimum hourly congestion value of last week)/3	1	No congestion
(Maximum hourly congestion value of last week + 2 × Minimum hourly congestion value of last week)/3 < CV ≤ (2 × Maximum hourly congestion value of last week + Minimum hourly congestion value of last week)/3	2	Medium Congestion
(2 × Maximum hourly congestion value of last week + Minimum hourly congestion value of last week)/3 < CV	3	High Congestion

In Table 1, the level of congestion is divided into three levels—low, medium, and high congestion—as Google uses the same crowdsourced data for four quantized levels (e.g., in Google's Java Map API, there are four color tracks representing different speed ranges). So, we divided the thresholds linearly

into three levels and ensured that quantization errors were not made. The manual parameter input may be of use later if these thresholds need to be changed manually in the server by a traffic agent, as the Google API, in some cases, inappropriately shows high congestion due to accidents, construction work, or a transport vehicle stalling or driving at a low speed on the roadside when the road is actually vacant and can handle a high traffic rate. Such scenarios of low-speed driving or blocking/stalling generally occur near industrial hubs. After the congestion status is calculated, the cycle time is updated accordingly as deemed by the road authority personnel. *Yellow Clearance*, *Red Clearance*, *Walk time*, and *Flashing Do Not Walk time* should not be changed [55].

The congestion status is adjusted in order to apply the average effect of the whole area instead of the fact that the timing could be completely different based on the congestion level in each outgoing direction of the intersection. To account for this fact, we take the congestion weights (input by road personnel) of each connecting road at the intersection. The congestion value (CV) calculation includes mapping from the incoming traffic lane, as well as the outgoing lane from the intersection. The congestion level thus calculated will average out the different congestion levels on different connecting roads to generate the average level for selecting the appropriate cycle time out of the three levels. Also, this way, we do not have to account for all movements, including through, right turns, and left turns combined together, as the average effect of the whole area is considered. However, the timing allotted for different phases of different cycle times will be the same as that specified in the knowledge-based database of each intersection calculated by the traditional model. As per the knowledge-based data, the scope of this architecture enables the control of traffic lights independently for a given intersection. This independent nature of controlling traffic light autonomously limits the coordination maintenance of a green wave. Although the concept of coordination is of hardly any use in the erratic lane-free traffic of developing countries, after road infrastructure advancement of lanes and well-guided roads, the coordination issue could be addressed by advancing the framework.

4. Programming Analysis and Implementation-Level Details

The traffic data or actual situation is already available to us by the Google API. We used the 'Google Distance Matrix API' to get the estimated time of travel between the origin and the destination in accordance with the congestion. To get a live data feed, we have to hit the Google server every time we check the status. For this, we obtained a key that is provided by Google to each individual Google account. The service we use in our program comes as a free version. However, after limited usage, Google charges for the same. The system includes a third-party cloud server or transportation department server, a master as a gateway, and the slave microcontroller to control the traffic light. The cloud server or transportation department server is used for handling all the delicate and complex processing, as well as the calculations. A cloud server is also used to store the data and the parameters which are fed by authorized officer/agent. The effort of shifting the processing, as much as possible, to the cloud or remote server serves two purposes: (1) processing the complex algorithm on low resource edge devices creates a lot of heat, adversely affecting the lifetime of the edge devices [56]; (2) the cloud server is much more secure than edge devices and shifting the processing to the server secures it against any hacking or data manipulation threats.

In the server, two algorithms run side by side. One of the two algorithms is configured to send an HTTP request to the Google server for the API response. The request string for the API contains information about the format in which we are seeking the response—either JSON or XML—and the parameters, such as the travel mode, restrictions, and the origin and destination coordinates, along with the key. The received response is stored in the variable for further processing, along with some parameters needed that have to be fetched from the database. These are present on the server and are updated by a state transportation department officer/agent of the respective area. For this purpose, a user interface is provided to road officials. All the parameters for each lane connected to the intersection, e.g. the weight factor, average time to travel, split cycle time of each level, and inputs by road officials, are updated on the server. Using all of these input values, the congestion status will be

calculated as described. For the calculation of the time to implement the new cycle time in accordance with the calculated congestion status, the real time has to be fetched from the internet or worldwide web clock (GMT).

For large-scale WSNs, time synchronization is essential for networking. Many time-synchronization approaches that ensure high accuracy cause high communication overheads. Performance evaluation by simulating on NS-2 and realization were done on STM32W108 with a simple MAC protocol stack [57]. The experiment results show that each time-synchronization cycle lasts 10 s because this duration is sufficient for all the nodes in the network to synchronize and for all the necessary broadcasts to avoid data collision. So, to implement the new cycle time, a guard band time of 10 s has to be added to the current time to address the worst case and ensure (for large WSNs) that the whole process has been completed before the implementation of a new cycle time. The 'green distribution' of a given cycle time for different phases is specified in the knowledge-based data from the Delhi traffic police. So, after calculating the implementation time for each slave, it has to be posted on another database on the server for future use along with the corresponding slave numbers and the congestion value obtained from the congestion status calculation.

Also, the new cycle time and the time stamps of implementation, along with the corresponding slave number, have to be conveyed to 'master' using a publish-request protocol (Message Queuing Telemetry Transport—MQTT). After successful delivery, a delay, which is determined by an error report from the master, is given, and then the same process repeats for the day to provide data in accordance with the congestion. The whole process is presented in Figure 2a. The second algorithm that operates on the server checks the congestion value data and saves the maximum and minimum values for each hour in the weekly database, as described in Figure 2b. Further, the master is configured to receive traffic information related to a route and the congestion information involved; this information is sent by the third-party server using MQTT. If the new data are available, it will post the newly available data to each slave using short wireless communication, as presented in Figure 2c. If the master did not receive the required information from any of the slaves, the master will send an error report to the server. The server will delay the loop of the cycle time calculation for the largest cycle time so that the system can resynchronize.

The continuously monitored cycle time, along with other information sent by the master, is successfully grabbed by the slave via short wireless communication. The data are checked to verify whether it is updated or not. If the data are not updated, the previous cycle time is added to the previous implementation time to get a new implementation time; if the data are updated, all the parameters are saved to temporary variables. Regarding the fail-safe characteristics of the proposed system, the usage of the information flag and temporary variable will help to avoid random errors due to false packet grabbing or data updates during processing. Also, it will make the system immune to temporary internet connectivity failures and short wireless connection losses. After that, the real time is pulled from the RTC (Real Time Clock) to calculate the exact time of functioning as well as the countdown time for a given traffic light slave in accordance with different traffic phase patterns, as shown in Figure 3 [58]. At the appropriate time, the digital I/O pins are set high to light up the high-voltage lights using relays, and the countdown data are transmitted to the countdown board. The flow process is pictured in Figure 4a. In order to set up a fixed yellow or amber offset time, a simple algorithm for timing out a green signal can be implemented on a slave to save some time for offset. For example, to allow an offset of '5' s, if the green time is 'X' s for a given slave, then the timeout of digital I/O pins that trigger a green light is set to 'X − 5' s and, immediately, the triggering of I/O pins for an amber light is done for 5 s.

The traffic light countdown display board grabs the data from a slave microcontroller and transmits it accordingly to list the values for the counter register to update the countdown board microcontroller, as well as for subsequently signaling the anode of the display, as shown in Figure 4b. To implement an application that can detect the present and near-future traffic light status, the server has to provide this information directly or allow database access through an open access platform.

As the user traveling in the vehicle passes by the traffic lights, they get the information and status update via the mobile application running on their personal device assistants, such as an Android phone, iPhone, tablet, or an Internet-enabled dashboard of a vehicle. Though the impacts of the mobile phone and its emerging technology on ITS have been previously well examined [59], the fully loaded Android or any other OS in mobile devices will tremendously enhance the impact and optimization.

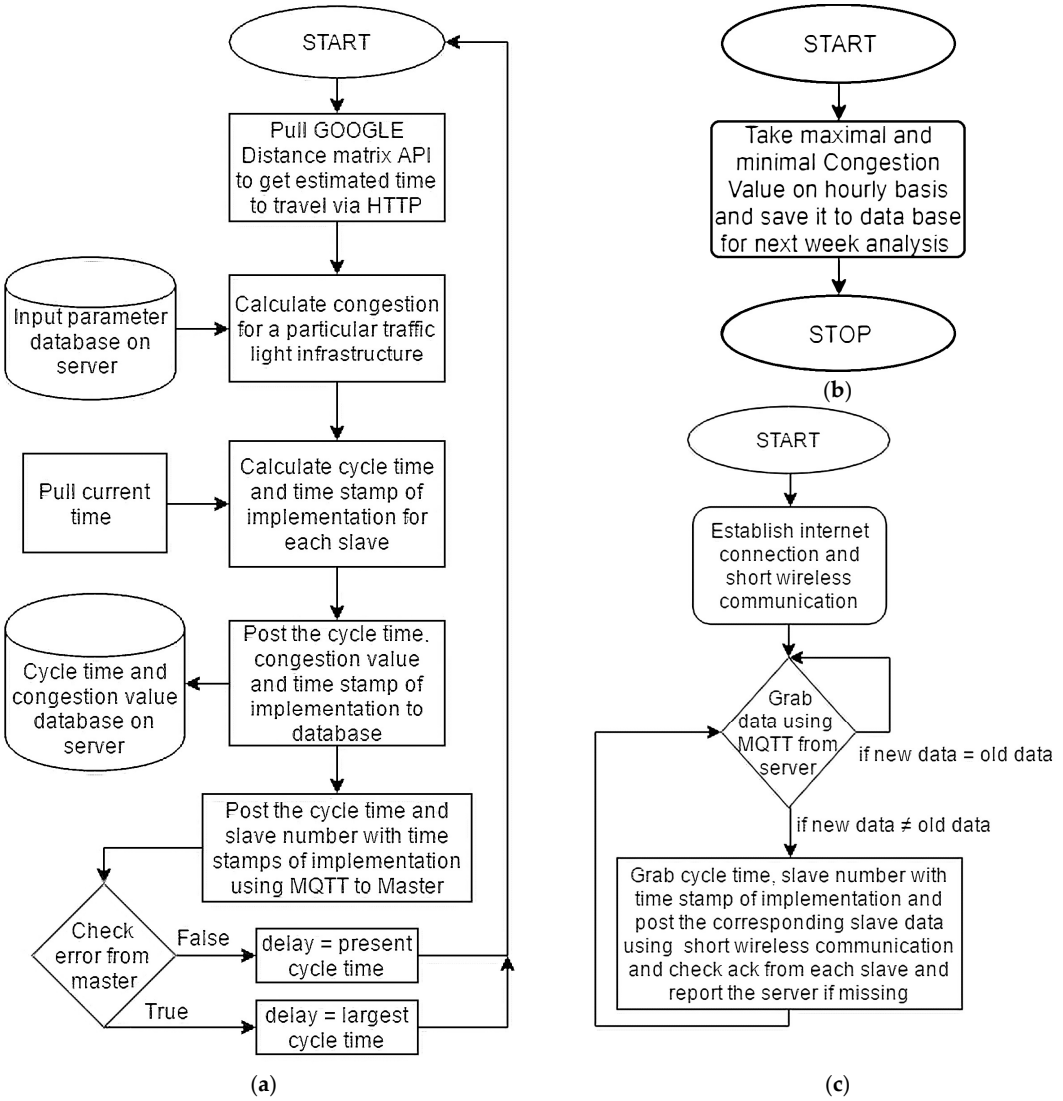


Figure 2. (**a**) Flowchart of the first algorithm running on the cloud server. (**b**) Flowchart of the second algorithm running on the cloud server. (**c**) Flowchart for the Master system design.

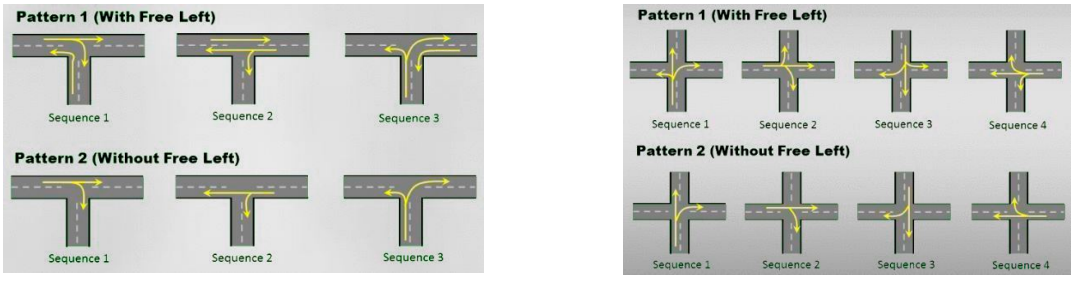


Figure 3. (**a**) Three-road traffic patterns and sequence. (**b**) Four-road traffic patterns and sequence.

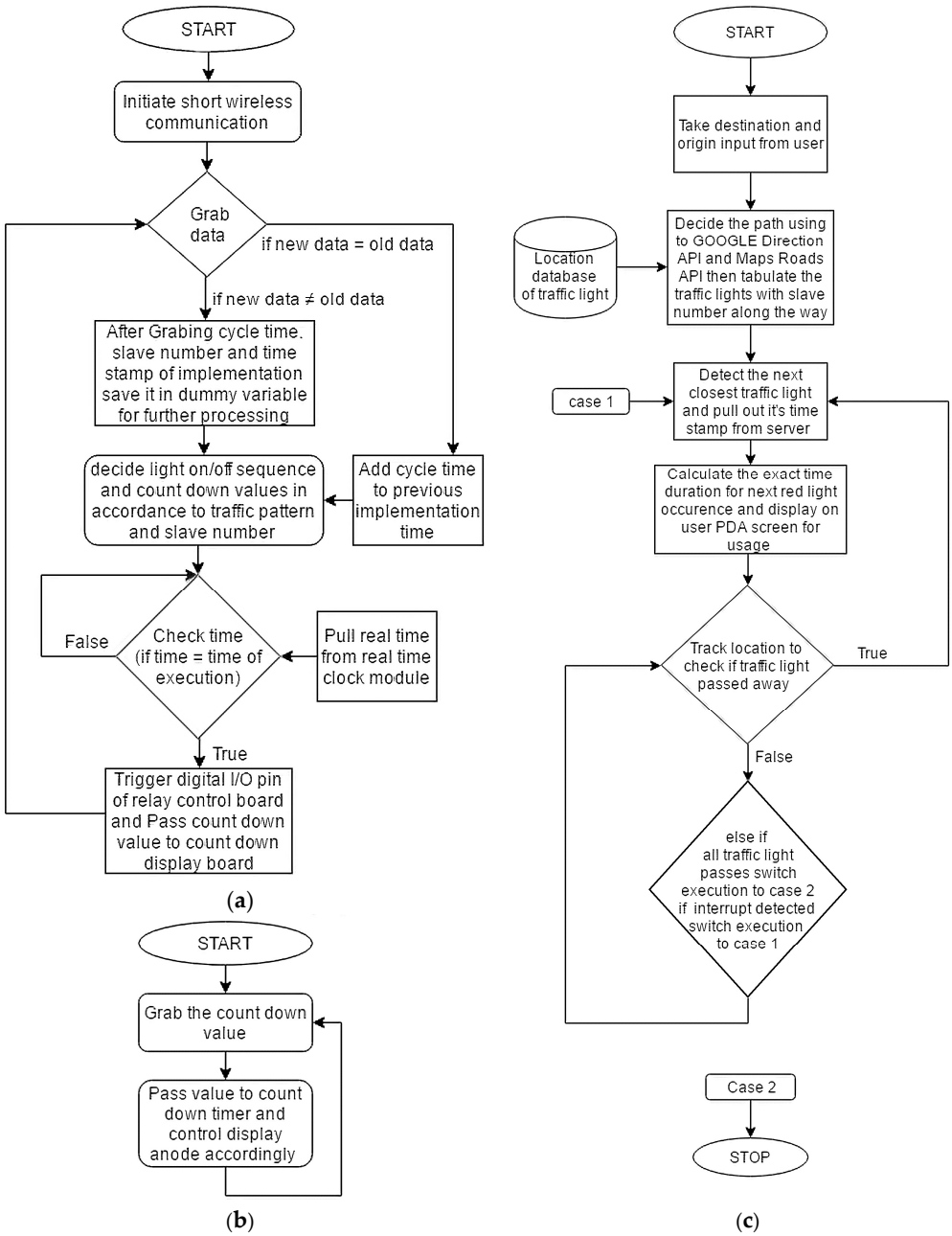


Figure 4. (**a**) Flowchart of the process running on the slave microcontroller. (**b**) Flowchart of the process running on the countdown display board attached to the slave microcontroller. (**c**) Flowchart of the application process running in the user's personal digital assistant (PDA).

The program running on PDA application works by taking the destination and origin coordinates or location from the user as the input. After that, it gets the path status using other Google Maps APIs, mainly the Maps Roads API [36,60] and Direction API [61] for tracking, and, in so doing, it detects the closest traffic light that is next on the route. For this traffic light, the location database has to be checked, and the location of the slave number on the route can be tracked since they are planted clockwise from the geographical north, starting from 1. After that, the recent time stamp for that traffic light is grabbed from the server and displayed to the user so that they can manage their speed accordingly. Location tracking will occur concurrently so that if one traffic light is passed, the program itself iterates the loop for the next traffic light on the way. The process is self-repeating until the user gives the overriding demand for a status check. The process ends when all the traffic lights have been passed. The whole process is summarized in Figure 4c.

5. Case Study

5.1. Four-Way Intersection Location for Survey

For verifying the congestion status mapping, we collected the data from the Google API in two different megacities in India—Bangalore and Delhi—for four four-way intersections in each city. The intersections chosen, tabulated in Table S1 (data provided in Supplementary Material), are such that they have varying congestion and have volume-to-capacity ranges of 1:1, 1:2, 1:3, and 1:5, in general. The lane length from the central joint of the four-way intersection is taken such that either it is within a range of 200–750 m or it is measured up until it is further joined by a major road. The coordinate mentioned has to be varied a little bit while sending the request to the API so that it responds with the O-D ETA, covering the given lane only once. Hence, we inputted eight sets of O-D coordinates for each intersection after carefully checking them manually on Google Maps.

5.2. Data Grabbing from Distance Matrix API

For the analysis, data retrieval, and calculating the congestion value and hence the congestion level, we wrote a Java program. The program asks for the time to run and the time to iterate the process, and the origin-destination set is input in a text file. All the O-D sets for which data were collected have to be in a single comma-separated values (CSV) file, separated by the character 'u'. So, we send a '64' request in one iteration, eight for each intersection.

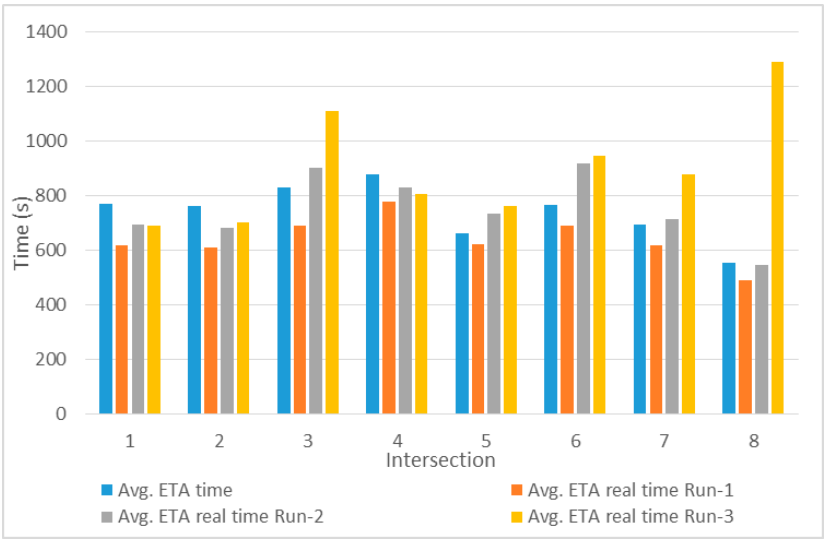
The main API request from the program to Google server is -
https://maps.googleapis.com/maps/api/distancematrix/json?origins=aaaa,bbbb&destinations=cccc,dddd&departure_time=now&key=YOUR_API_KEY
The response is-
{"destination_addresses": ["aaaaaa"],
"origin_addresses": ["ccccccc"],
"rows":
[{"elements":[{"distance":{"text":"0.6 km", "value": 616}, "duration":{"text":"5mins", "value": 307},
"duration_in_traffic": {"text": "8 mins", "value": 465}, "status": "OK"}] }],
"status": "OK"}

5.3. Result Achieved

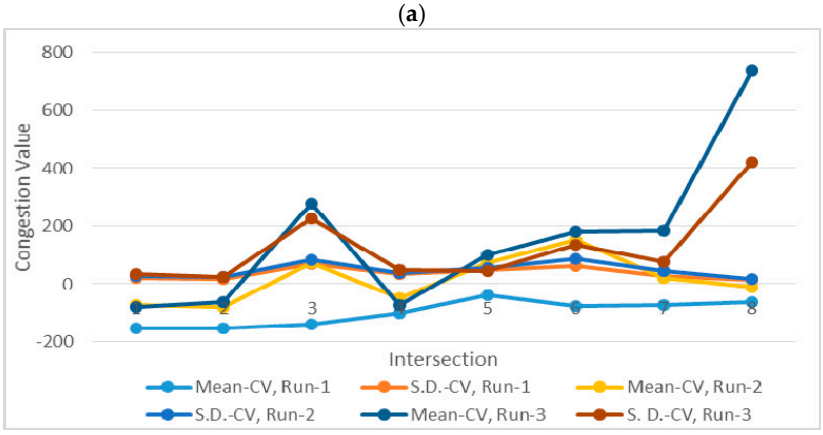
For calculating the congestion value and status, we extracted the maximum and minimum values and averaged the data for 1 week to find the average ETA. We sampled the data for 3 h a day in three 1-h intervals (sampled every 2 min within each interval): 08:00–09:00, 14:00–15:00, and 20:00–21:00. Finally, the same process was repeated for the same weekday, the following week, e.g., for a Monday, the samples collected from the three software runs are 30, 29, and 30, respectively. Eight different intersections were chosen that vary in congestion. Afterward, the congestion values were calculated; the complete data analysis is tabulated in Table S2 (data provided in Supplementary Material).

It can be easily seen from Figure 5a that intersections 1, 2, and 4 are less congested than they should be as per the average road speed data. Further, Figure 5b illustrates that in the morning, even though there is less congestion depicted by the average value for CV (Mean-CV), the traffic condition fluctuates as depicted by the standard deviation. The standard deviation plots of all three runs have comparable ranges, as is the actual CV value spread for all three different times. Hence, to consider congestion as an hour-dependent variable is a wrong assumption. The methodology adopted is adaptive to the congestion status for that particular hour, hence, the congestion status calculated is compared with that single hour in which data were grabbed. Therefore, most of the time, the congestion status is varying, as shown in Figure 5c, where intersections 1, 2, and 4 are less congested, and hence have high numbers of the level-1 status. For the afternoon, in general, level 1 and 2 are more prevalent; however, the average congestion in the afternoon is more than that in the morning for all intersections, as shown

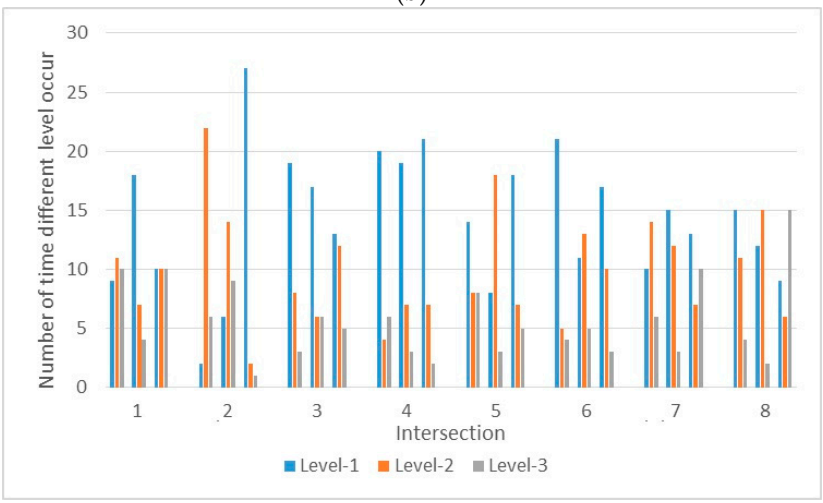
in Figure 5b. In the afternoon, the traffic value fluctuates more than it does in the morning, and, hence, the status value is scattered for levels 1, 2, and 3, with a greater inclination to 1 and 2. In the evening, the congestion is the worst and the most fluctuating congestion condition. Hence, for the evening, the status values are mostly scattered.



(**a**)



(**b**)



(**c**)

Figure 5. (**a**) Variation of ETA values submission of all lanes connected to a particular intersection for run-1, 2, 3 and general (average) conditions. (**b**) Mean and standard deviation plot for runs 1, 2, and 3. (**c**) Distribution of different congestion levels for runs 1, 2, and 3 for eight different intersections.

The congestion value calculated is in respect to an hour and, therefore, provides good insight into the temporal congestion values. The congestion values extracted for the different hours, e.g., for the afternoon and evening, are spread around with relatively the same median values for some instances, and, therefore, there are cases for which the congestion in the afternoon is more than that of the average evening, even though, traditionally, evening congestion is considered to be worse. So, the cycle time is adjusted in accordance with the status value. This will prevent congestion from forming on a short-term basis. For testing in a confined environment, hardware was implemented using limited open source platforms which imitate the full ITS system. The structure of the hardware prototype is presented in Figure 6.

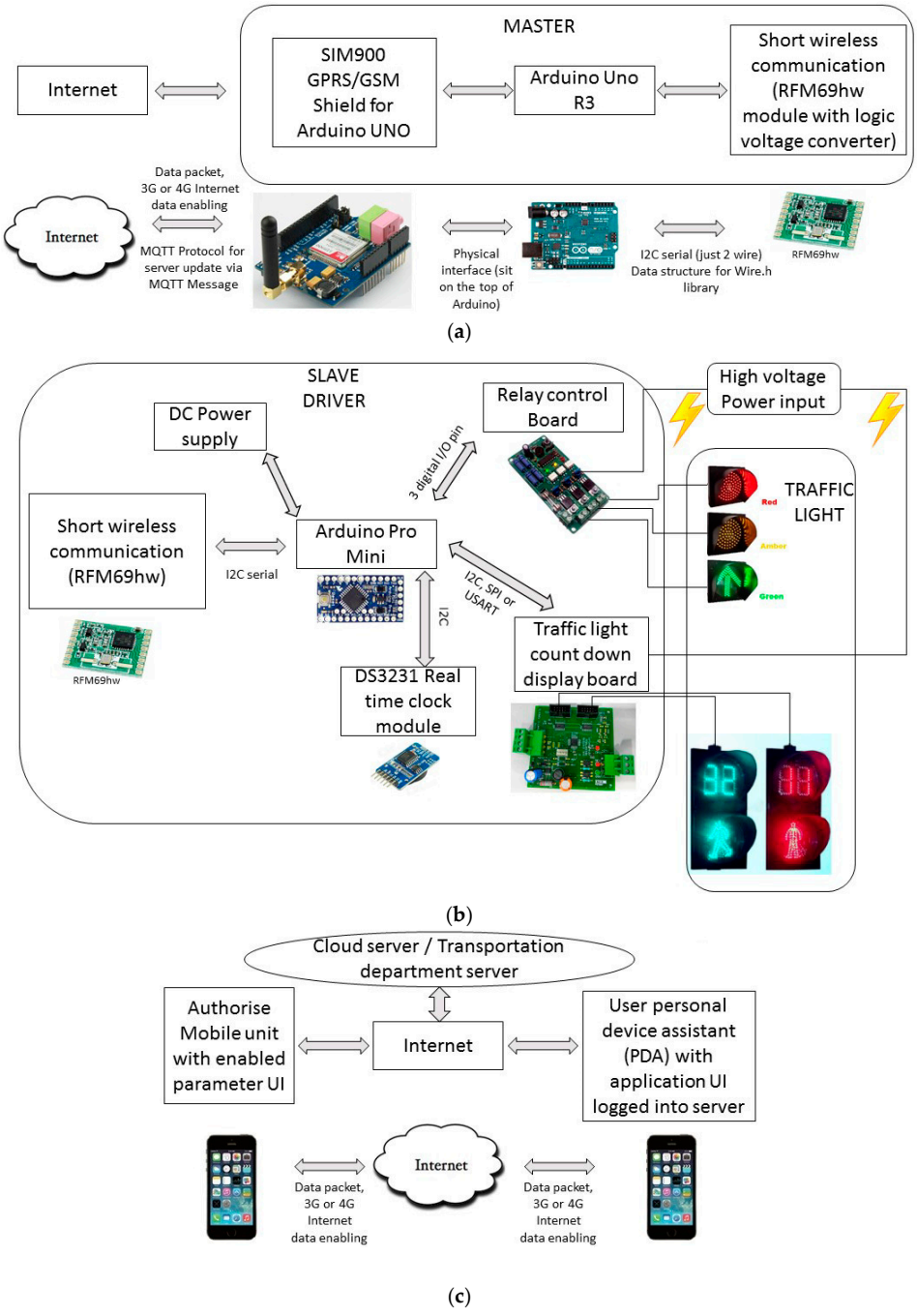
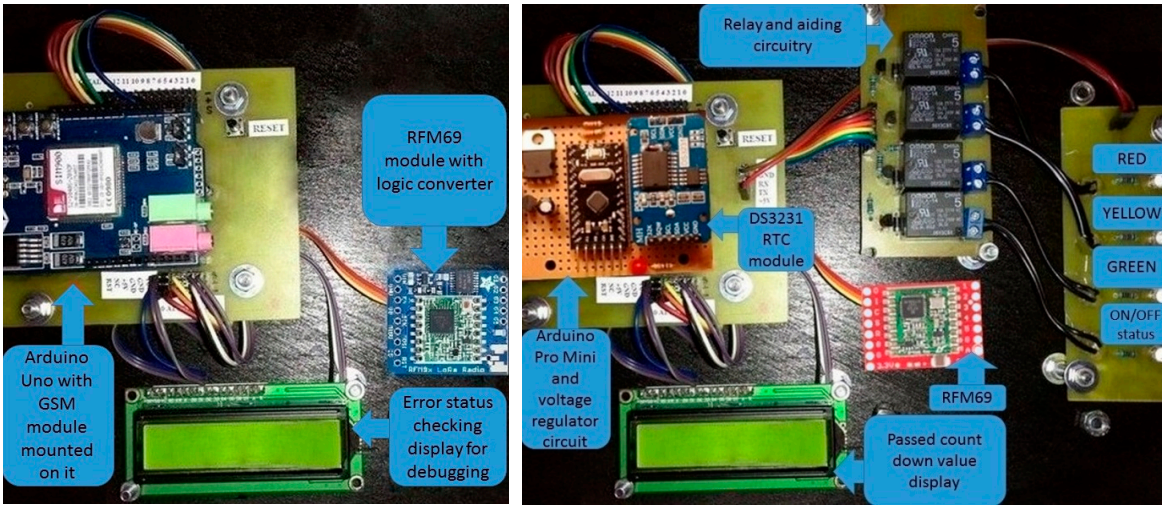


Figure 6. (**a**) Block diagram for hardware implementation of the Master's connection. (**b**) Block diagram for hardware implementation of the Slave driver's connection. (**c**) Data flow between the user's PDA, authorized person's PDA, the server, and the internet.

The time delay to publish the data on the user's HTML is 1.5 s on average, while the maximum is 3 s, depending on random parameters for each hardware run. This is a very minimal and reliable time delay for live data feeds. The limited prototype system is presented in Figure 7. The more expert-level "Green Light Optimal Speed Advisory" (GLOSA) application [62,63] can also be assessed by directly linking it with the database without any visibility problems reported for the image processing technique. As previously claimed, this can reduce CO_2 emission and fuel consumption by 20.3%, on average [31]. Audi also claims that the consistent use of traffic light informative systems could lead to a 15% reduction in carbon dioxide emissions and 900 million liters (238 million gal) of saved petrol annually in Germany alone [64].



(**a**) (**b**)

Figure 7. (**a**) Hardware implementation of the Master's connection. (**b**) Hardware implementation of the Slave driver.

6. Conclusions and Future Scope

In this paper, we propose a dual-application system framework that is intended to provide a traffic light timing adjustment solution. The methodology is based on analyzing traffic density information using the Google API in real time and near future, as well as providing information that conveys the traffic light status to the driver. The proposed traffic condition data collection solution is infrastructure-free and is thus scalable. The system implemented aims to obtain the status of the next traffic light in minimal time just after crossing the traffic light before it, given a certain route. Based on the traffic light status, the driver can decide to drive more smoothly to maintain a continuous journey by adjusting their speed, which helps to improve the flow of traffic and significantly reduces carbon dioxide emissions and fuel consumption. Hence, using this system divides congestion equally among links between traffic lights, and the network as an entity is completely utilized. The technology can be easily incorporated into the smart city concept and can save resources for any country, such as reducing the time of workers in traffic and preventing the damage and harm from accidents caused by RLR, as RLR is less likely to occur due to the method's stabilizing effect on the psychology of the driver.

This whole system can be of much more use to transport engineers and traffic authority personnel to make spatial location-based reports using the database by acquiring accurate location logs of vehicles. This database can also be used to form an evacuation plan in case of emergency, optimize routes for public vehicles, detect the need for increasing the yellow (crossing) time, and much more.

Supplementary Materials: The following are available online at http://www.mdpi.com/2306-5729/3/4/67/s1.

Author Contributions: S.M. worked on driving the methodology, performed data analysis, and wrote the paper; D.B. worked on analysis, literature review, and paper writing—review and editing; A.G. worked on analyzing the results and paper writing—review and editing.

Funding: This research received no external funding.

Acknowledgments: Authors want to thank the anonymous reviewers for their expert comments. Also, one of the authors, Sumit Mishra, wants to acknowledge Learnogether Technologies Pvt. Ltd. for facilitating development by availing expert insights on targeted technologies.

Conflicts of Interest: Authors declare no conflicts of interest.

References

1. Georgios, K.; Altintas, O.; Ekici, E.; Heijenk, G.; Jarupan, B.; Lin, K.; Weil, T. Vehicular networking: A survey and tutorial on requirements, architectures, challenges, standards and solutions. *IEEE Commun. Surveys Tutorials* **2011**, *13*, 584–616. [CrossRef]
2. Lu, N.; Cheng, N.; Zhang, N.; Shen, X.; Mark, J.W. Connected vehicles: Solutions and challenges. *IEEE IoT J.* **2014**, *1*, 289–299. [CrossRef]
3. Stoffers, K.E. Scheduling of traffic lights—A new approach. *Transp. Res.* **1968**, *2*, 199–234. [CrossRef]
4. Allos, A.E.; Al-Hadithi, M.I. Driver behaviour during onset of amber at signalised junctions. *Traffic Eng. Control* **1992**, *33*, 312–317.
5. Van der Horst, A.; Richard, A.; Wilmink, A. Drivers' decision-making at signalised intersections: An optimization of the yellow timing. *Traffic Eng. Control* **1986**, *27*, 615–617.
6. Shu, Z.; Wan, J.; Zhang, D.; Li, D. Cloud-integrated cyber-physical systems for complex industrial applications. *Mobile Netw. Appl.* **2016**, *21*, 865–878. [CrossRef]
7. Yue, X.; Cai, H.; Yan, H.; Zou, C.; Zhou, K. Cloud-assisted industrial cyber-physical systems: An insight. *Microprocess. Microsyst.* **2015**, *39*, 1262–1270. [CrossRef]
8. Wan, J.; Zhang, D.; Sun, Y.; Lin, K.; Zou, C.; Cai, H. VCMIA: A novel architecture for integrating vehicular cyber-physical systems and mobile cloud computing. *Mobile Netw. Appl.* **2014**, *19*, 153–160. [CrossRef]
9. Kumar, N.; Singh, M.; Zeadally, S.; Rodrigues, J.J.; Rho, S. Cloud-assisted context-aware vehicular cyber-physical system for PHEVs in smart grid. *IEEE Syst. J.* **2017**, *11*, 140–151. [CrossRef]
10. Vujović, V.; Maksimović, M. Raspberry Pi as a Sensor Web node for home automation. *Comput. Electr. Eng.* **2015**, *44*, 153–171. [CrossRef]
11. Collotta, M.; Conti, V.; Scatà, G.; Pau, G.; Vitabile, S. Smart wireless sensor networks and biometric authentication for real time traffic light junctions management. *Intern. J. Intell. Inf. Datab. Syst.* **2013**, *7*, 454–478. [CrossRef]
12. Pandian, S.; Gokhale, S.; Ghoshal, A.K. Evaluating effects of traffic and vehicle characteristics on vehicular emissions near traffic intersections. *Transp. Res. Part D Transp. Environ.* **2009**, *14*, 180–196. [CrossRef]
13. Smit, R.; Brown, A.L.; Chan, Y.C. Do air pollution emissions and fuel consumption models for roadways include the effects of congestion in the roadway traffic flow? *Environ. Model. Softw.* **2008**, *23*, 1262–1270. [CrossRef]
14. Klein, L.A. *Sensor Technologies and Data Requirements for ITS*; Artech House Publishers: Norwood, MA, USA, 2001.
15. Pascale, A.; Nicoli, M.; Deflorio, F.; Dalla Chiara, B.; Spagnolini, U. Wireless sensor networks for traffic management and road safety. *J. Intell. Transp. Syst. IET* **2012**, *6*, 67–77. [CrossRef]
16. Nellore, K.; Hancke, G.P. A survey on urban traffic management system using wireless sensor networks. *Sensors* **2016**, *16*, 157. [CrossRef] [PubMed]
17. He, Z.; Guan, W.; Ma, S. A traffic-condition-based route guidance strategy for a single destination road network. *Transp. Res. Part C Emerg. Technol.* **2013**, *32*, 89–102. [CrossRef]
18. Mekki, T.; Jabri, I.; Rachedi, A.; ben Jemaa, M. Vehicular cloud networks: Challenges, architectures, and future directions. *Veh. Comm.* **2017**, *9*, 268–280. [CrossRef]
19. Wan, J.; Liu, J.; Shao, Z.; Vasilakos, A.V.; Imran, M.; Zhou, K. Mobile crowd sensing for traffic prediction in internet of vehicles. *Sensors* **2016**, *16*, 88. [CrossRef] [PubMed]
20. He, Z.; Cao, B.; Liu, Y. Accurate real-time traffic speed estimation using infrastructure-free vehicular networks. *Int. J. Distrib. Sensor Netw.* **2015**, *11*, 530194. [CrossRef]
21. Zhong, N.; Ma, J.H.; Huang, R.H.; Liu, J.M.; Yao, Y.Y.; Zhang, Y.X.; Chen, J.H. Research challenges and perspectives on wisdom web of things (W_2T). *J. Supercomput.* **2013**, *64*, 862–882. [CrossRef]

22. Guo, B.; Zhang, D.; Yu, Z.; Liang, Y.; Wang, Z.; Zhou, X. From the internet of things to embedded intelligence. *World Wide Web* **2013**, *16*, 399–420. [CrossRef]
23. Dobre, C.; Xhafa, F. Intelligent services for big data science. *Future Generat. Comput. Syst.* **2014**, *37*, 267–281. [CrossRef]
24. Sanchot, M. How Accurate Is Google's Popular Time Function? Available online: https://www.linkedin.com/pulse/how-accurate-googles-popular-time-function-mélissa-sanchot/ (accessed on 19 September 2018).
25. Miller, G. The Huge, Unseen Operation Behind the Accuracy of Google Maps. Available online: https://www.wired.com/2014/12/google-maps-ground-truth/ (accessed on 19 September 2018).
26. Tostes, A.I.J.; de LP Duarte-Figueiredo, F.; Assunção, R.; Salles, J.; Loureiro, A.A. From data to knowledge: City-wide traffic flows analysis and prediction using bing maps. In Proceedings of the 2nd ACM SIGKDD International Workshop on Urban Computing (UrbComp'13), Chicago, IL, USA, 11–14 August 2013; p. 12.
27. Juntunen, T.; Kostakos, V.; Perttunen, M.; Ferreira, D. Web tool for traffic engineers: direct manipulation and visualization of vehicular traffic using Google maps. In Proceedings of the 16th International Academic MindTrek Conference (MindTrek 2012), Tampere, Finland, 3–5 October 2012; pp. 209–210.
28. Logi, F.; Ritchie, S.G. A multi-agent architecture for cooperative inter-jurisdictional traffic congestion management. *Transp. Res. Part C Emerg. Technol.* **2002**, *10*, 507–527. [CrossRef]
29. Singh, B.S.R. Real time prediction of road traffic condition in London via twitter and related sources. Master's Thesis, Middlesex University, London, UK, 2012.
30. Kwak, D.; Kim, D.; Liu, R.; Iftode, L.; Nath, B. Tweeting traffic image reports on the road. In Proceedings of the 2014 6th International Conference on Mobile Computing, Applications and Services (MobiCASE), Austin, TX, USA, 6–7 November 2014; pp. 40–48.
31. Koukoumidis, E.; Peh, L.S.; Martonosi, M.R. Signalguru: Leveraging mobile phones for collaborative traffic signal schedule advisory. In Proceedings of the 9th ACM International Conference on Mobile Systems Applications and Services, Bethesda, MD, USA, 28 June–1 July 2011; pp. 127–140.
32. Levinson, J.; Askeland, J.; Dolson, J.; Thrun, S. Traffic light mapping, localization, and state detection for autonomous vehicles. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China, 9–13 May 2011; pp. 5784–5791.
33. Knapton, S. Gadget which Turns All Traffic Lights Green Trialled in UK. Available online: http://www.telegraph.co.uk/news/science/11512274/Gadget-which-turns-all-traffic-lights-green-trialled-in-UK.html (accessed on 19 September 2018).
34. Audi MediaTV. Online Traffic Light Information. Available online: https://www.audi-mediacenter.com/en/audimediatv/video/online-traffic-light-information-2418 (accessed on 19 September 2018).
35. Luyanda, F.; Gettman, D.; Head, L.; Shelby, S.; Bullock, D.; Mirchandani, P. ACS-Lite algorithmic architecture: Applying adaptive control system technology to closed-loop traffic signal control systems. *Transp. Res. Rec. J. Transp. Res. Board* **2003**, *1856*, 175–184. [CrossRef]
36. Mishra, S.; Bhattacharya, D.; Gupta, A.; Singh, V.R. Adaptive Traffic Light Cycle Time Controller Using Microcontrollers and Crowdsource Data of Google Apis For Developing Countries. *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.* **2018**, *2018*, 83–90. [CrossRef]
37. Xin, W. A New Architecture Adaptive Traffic Signal Control in A Data-Rich Environment. Ph.D. Thesis, Polytechnic Institute of New York University, New York, NY, USA, 2014.
38. Athmaraman, N.; Soundararajan, S. Adaptive predictive traffic timer control algorithm. In Proceedings of the 2005 Mid-Continent Transportation Research Symposium, Ames, IA, USA, 18–19 August 2005.
39. Liu, H.; Oh, J.S.; Recker, W. Adaptive signal control system with online performance measure for a single intersection. *Transp. Res. Record J. the Transp. Res. Board* **2002**, *1811*, 131–138. [CrossRef]
40. Lin, S.; De Schutter, B.; Xi, Y.; Hellendoorn, H. Efficient network-wide model-based predictive control for urban traffic networks. *Transp. Res. Part C Emerg. Technol.* **2012**, *24*, 122–140. [CrossRef]
41. Yousef, K.M.; Al-Karaki, M.N.; Shatnawi, A.M. Intelligent traffic light flow control system using wireless sensors networks. *J. Inf. Sci. Eng.* **2010**, *26*, 753–768.
42. Gradinescu, V.; Gorgorin, C.; Diaconescu, R.; Cristea, V.; Iftode, L. Adaptive traffic lights using car-to-car communication. In Proceedings of the IEEE 65th Vehicular Technology Conference, Dublin, Ireland, 22–25 April 2007; pp. 21–25.
43. Feng, G. A survey on analysis and design of model-based fuzzy control systems. *IEEE Trans. Fuzzy Syst.* **2006**, *14*, 676–697. [CrossRef]

44. Odeh, S.M.; Mora, A.M.; Moreno, M.N.; Merelo, J.J. A hybrid fuzzy genetic algorithm for an adaptive traffic signal system. *Adv. Fuzzy Syst.* **2015**, *2015*, 11. [CrossRef]
45. Jiao, P.; Li, Z.; Liu, M.; Li, D.; Li, Y. Real-time traffic signal optimization model based on average delay time per person. *Adv. Mechan. Eng.* **2015**, *7*. [CrossRef]
46. Li, H.; Prevedouros, P.D. Traffic adaptive control for oversaturated isolated intersections: Model development and simulation testing. *J. Transp. Eng.* **2004**, *130*, 594–601. [CrossRef]
47. Abdulhai, B.; Pringle, R.; Karakoulas, G.J. Reinforcement learning for true adaptive traffic signal control. *J. Transp. Eng.* **2003**, *129*, 278–285. [CrossRef]
48. Guerrero-Ibanez, A.; Contreras-Castillo, J.; Buenrostro, R.; Marti, A.B.; Muñoz, A.R. A policy-based multi-agent management approach for intelligent traffic-light control. In Proceedings of the 2010 IEEE Intelligent Vehicles Symposium (IV), San Diego, CA, USA, 21–24 June 2010; pp. 694–699.
49. Google Developers. Traffic Layer. Available online: https://developers.google.com/maps/documentation/javascript/examples/layer-traffic (accessed on 19 September 2018).
50. Google Developers. MapTypeControlOptions. Available online: https://developers.google.com/maps/documentation/javascript/3.exp/reference#MapTypeControlOptions (accessed on 19 September 2018).
51. Google Developers. Distance Matrix. Available online: https://developers.google.com/maps/documentation/distance-matrix/intro (accessed on 19 September 2018).
52. Permissions. Available online: https://www.google.co.uk/permissions/geoguidelines.html (accessed on 19 September 2018).
53. Delhi Traffic Police. Traffic Signals. Available online: https://delhitrafficpolice.nic.in/be-road-smart/signals/ (accessed on 19 September 2018).
54. Wang, F.; Xu, Y. Estimating O–D travel time matrix by Google Maps API: Implementation, advantages, and implications. *Ann. GIS* **2011**, *17*, 199–209. [CrossRef]
55. Signal Timing. Available online: https://en.wikipedia.org/wiki/Signal_timing (accessed on 19 September 2018).
56. Srinivasan, J.; Adve, S.V.; Bose, P.; Rivers, J.A. The Impact of Technology Scaling on Lifetime Reliability. In Proceedings of the 2004 International Conference on Dependable Systems and Networks (DSN), Florence, Italy, 28 June–1 July 2004; p. 177. [CrossRef]
57. Qiu, T.; Chi, L.; Guo, W.; Zhang, Y. STETS: A novel energy-efficient time synchronization scheme based on embedded networking devices. *Microprocess. MicroSyst.* **2015**, *39*, 1285–1295. [CrossRef]
58. Traffic Signal Timing Manual. Available online: http://ops.fhwa.dot.gov/publications/fhwahop08024/chapter4.htm (accessed on 19 September 2018).
59. Zhao, Y. Mobile phone location determination and its impact on intelligent transportation systems. *IEEE Trans. Intell. Transp. Syst.* **2000**, *1*, 55–64. [CrossRef]
60. Google Developers. Roads Api. Available online: https://developers.google.com/maps/documentation/roads/intro (accessed on 19 September 2018).
61. Google Developers. Directions API. Available online: https://developers.google.com/maps/documentation/directions/ (accessed on 19 September 2018).
62. Stevanovic, A.; Stevanovic, J.; Kergaye, C. Green light optimized speed advisory systems: Impact of signal phasing information accuracy. *Transp. Res. Rec. J. Transp. Res. Board* **2013**, *2390*, 53–59. [CrossRef]
63. Katsaros, K.; Kernchen, R.; Dianati, M.; Rieck, D. Performance study of a Green Light Optimized Speed Advisory (GLOSA) application using an integrated cooperative ITS simulation platform. In Proceedings of the 7th International Wireless Communications and Mobile Computing Conference, Istanbul, Turkey, 4–8 July 2011; pp. 918–923. [CrossRef]
64. Audi USA. Audi Announces the First Vehicle to Infrastructure (V2I) Service—The New Traffic Light Information System. Available online: https://www.audiusa.com/newsroom/news/press-releases/2016/08/audi-announces-first-vehicle-to-infrastructure-service (accessed on 19 September 2018).