Smart Traffic Light Control on Edge in IOT-Regulated Intersections

SMART TRAFFIC LIGHT CONTROL ON EDGE IN IOT-REGULATED INTERSECTIONS

[1]EVELINA FORNO, [2]VITTORIO FRA, [3]MERT DOGAN ARISOY, [4]ZHOU CHENGHAN, [5]ANDREA SERGIO FERRARIS,[6]ENRICO MACII, [7]GIANVITO URGESE

Politecnico di Torino,Italy
E-mail: [6]enrico.macii@polito.it, [7]gianvito.urgese@polito.it

Abstract - Traffic is a well-known everyday problem that standard traffic lights controllers can struggle to deal with, especially in highly populated cities, resulting in congestion at the intersections and the consequent formation of queues.Smart traffic lights management, relying on Internet of Things (IoT) concepts and devices, may be adopted to mitigate this phenomenon. In this paper, we propose a Smart Intersection for Smart Traffic (SIST) regulated model using the max-pressure controller algorithm to dynamically modulate the duration of traffic lights, implemented on real-time embedded hardware and using data coming from local sensors and the IoT network.Compared to standard, fixed-duration control schemes, the dynamically IoT-regulated SIST model ensures overall reduction of the queue lengths, resulting in improved prevention of link overload by about 7% compared to the most favorable fixed-duration model.

Keywords - Smart traffic light control, IoT, Edge computing, Smart city

I. INTRODUCTION

In highly populated cities, traffic constitutes a well-known everyday problem affecting daily life activities, increasing the time spent in long vehicle queues. Similar consequences can impact pedestrians, who are subject to long waiting times if intersections are congested. In a wider perspective, especially in metropolitan areas where electrical vehicles do not represent a significant portion of the circulating fleet, traffic is a relevant source of pollution too, both in acoustic and in environmental terms. To mitigate these criticalities, smart environments constitute a promising design toolkit for more efficient management of the movement of people and vehicles [2, 15].With this aim, a straightforward starting point is represented by intersections, due to the possibility of directly acting on possible congestion by means of the traffic lights. By adopting a dynamic control of their duration, we can realize smart intersections with proper implementations of Internet of Things (IoT) concepts and devices [1, 6, 12].In this work, we propose a model of Smart Intersection for Smart Traffic (SIST) based on a locally-performed algorithm relying on real-time information collected through a combination of edge devices and cloud computing technologies. The max-pressure controller algorithm is employed on an embedded multi-core architecture running a real-time operating system; the implementation is then validated through simulations running on the Renode virtual development framework.

The impact of traffic as one of the problems experienced by cities can be evinced by the number of works carried out on this topic in recent years. Different approaches, focusing on specific aspects and presenting tailored solutions, can be found in literature. In [3], the Kerner three-phase traffic theory is applied to realize a synchronized system by establishing an Intelligent Transportation System (ITS) that provides automatic management of traffic lights. In [9], two schemes for secure, intelligent traffic light control are proposed, taking advantage of fog computing by implementing the Diffie–Hellman and the hash collision computational puzzles. Other works have also explored, besides hardware and software considerations, the possibility of accounting for the social impact of the involved vehicles, thus introducing the idea of assigning priority, for instance, to emergency vehicles, depending on social characteristics that can constitute suitable criteria [4, 5, 10]. Concerning the range of different strategies and methodologies to deal with traffic data and information, interesting solutions have also been proposed relying on image processing [11] and reinforcement learning [14, 16].

II. MATERIALS AND METHODS

We present now the theoretical and practical details of the proposed solution. In section II-A, we review two different techniques for traffic light control operations: the fixed-time controller and the max-pressure algorithm. In section II-B, we report the software and hardware details of the implementation.

A. Traffic lights control algorithms

There are two main approaches to traffic light control: the fixed-duration approach, which examines daily or hourly statistics in order to tune the expected traffic flow in each link of the intersection before deployment, and the dynamic-duration approaches, which continually adjust light durations in the field, based on data gathered from sensors or a network connection. In this work, we will focus on the Max-pressure algorithm for the implementation of dynamic light control.

1)Fixed-timing traffic light: This approach has long been used in most metropolitan intersections due to its simple implementation and efficacy in regulating traffic. It is based on the intuitive observation that the green light should last longer for the busiest link on the intersection than for the other links. The timing distribution for each of the 4 controllers in an intersection is calculated using previously collected data (e.g., the cumulative total number of vehicles crossing each intersection link) and programmed into the controllers before deployment. The green light durations, $G_{N,S}$and $G_{W,E}$, should be inferred from the ratio of queue lengths (N[t],S[t],W[t],E[t]) to cycle time (C_t[t]), according to Eqs. 1 and 2:

$$G_{N,S}[t] = \frac{\sum_{t=0} N[t] + S[t]}{\sum_{t=0} N[t] + S[t] + E[t] + W[t]} * C_t[t] \quad (1)$$

$$G_{W,E}[t] = \frac{\sum_{t=0} W[t] + E[t]}{\sum_{t=0} N[t] + S[t] + E[t] + W[t]} * C_t[t] \quad (2)$$

The summation of green light durations $G_{N,S}$and $G_{W,E}$should be equal to the cycle time, C_t[t]:

$$G_{N,S}[t] + G_{W,E}[t] = C_t[t] \quad (3)$$

In many situations, this simple calculation can provide suitable green light durations for each link. Timing distributions can also be adjusted to fit daily or hourly traffic loads; these will also be configured a priori and executed via a timer, without requiring any higher software functions. However, it can become inefficient when handling unforeseen or unbalanced amounts of traffic, and would require a manual intervention to update the programmed timings, should the need arise.

2) Dynamic-timing traffic light based on max-pressure algorithm: This solution proposes to stabilize the traffic flow depending on the capacity of each link and the queue lengths, i.e., the number of vehicles waiting at each link, with the aim of reducing waiting times at the intersection. The approach is based on work by Varaiya et al. [13], which does not require any data based on past or future demands; therefore, locallygathered information is sufficient.

The total cycle time for a traffic light C_n[t] can be expressed as the summation of active time and passive time. Active time (A_n[t]) corresponds to the green light duration G_n[t], during which traffic is allowed to flow; that is, vehicles can cross the intersection from one side to the other. The duration of passive time (P_n[t]) can be chosen according to what reason there is to stop the traffic flow at that particular point; while the typical reason is to allow pedestrians and other vehicles to cross the street, other elements, such as slowdown due to weather conditions, can be taken into account as a factor of passive time. In this study, we define passive time as the summation of the yellow (Y_n[t]) and red (R_n[t]) light durations.

$$C_n[t] = A_n[t] + P_n[t] \quad (4)$$

$$A_n[t] = G_n[t] \quad (5)$$

$$P_n[t] = Y_n[t] + R_n[t] \quad (6)$$

The minimum limit for the green light time, A_n[t] or G_n[t], is determined here to the need to let pedestrians cross the road. This limit corresponds to the longest time we allow a person to take when crossing from one side of the street to the opposite.

$$A_n[t] \geq A_{n,min}[t] \quad (7)$$

$$G_n[t] \geq G_{n,min}[t] \quad (8)$$

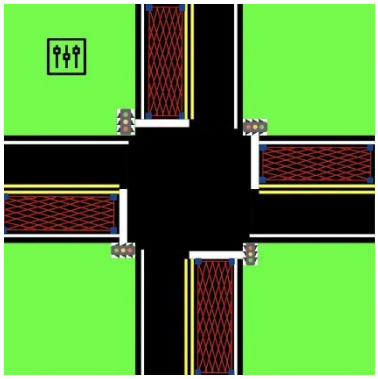


Fig. 1: A model of a 4-link intersection

For applying the max-pressure algorithm, we model an intersection as consisting of 4 links, as shown inFig. 1. Each link is described by a set of parameters, including length, width, turn permissions, etc.; these parameters determine the average capacity of the link, i.e., the number of vehicles it can accommodate at any given time. Depending on the capacity of each link and continuous data on the number of vehicles currently waiting on the link, a dynamic pressure inference can be made. The scientific definition of pressure is the ratio of force to the surface of the area where the force is applied; by analogy and simplification, the force term can be considered equivalent to the number of vehicles currently in the queue, and the surface area to the capacity of each link. According to the max-pressure algorithm, then, the pressure on a link z (P_z[t]) can be computed as in Eq. 9.

$$P_z[t] = \left[\frac{X_z[t]}{X_{z,max}} - \sum_{w}^{\beta} \frac{_{i,w} X_w[t]}{X_{w,max}} G_{n,j}[t] \right] S_z \quad (9)$$

where:

P_z[t]	Pressure on link z
X_z[t]	Queue Length of link z
Xz,max	Capacity of link z
βi,w	Ratio of vehicles turning from ito w
X_w[t]	Output stream
βi,wXw[t]	Output stream considering turn ratios
Xw,max	Maximum output stream
S_z	Saturation rate

The saturation rate indicates the time needed for a vehicle to exit the intersection. Given the simplifying

assumption that all output links exiting the network have infinite capacity and do not encounter any output blockage, the second term of Eq. 9 can be canceled out. In this case, the pressure on link z, $P_z[t]$, is simply the queue's flow rate multiplied by its corresponding saturation rate, S_z.

$$P_z[t] = \left[\frac{X_z[t]}{X_{z,max}}\right] S_z \quad (10)$$

After calculating the pressures on each link, the algorithm can distribute the green light durations for the current network cycle. As this dynamic operation mode attempts to reduce accumulated pressure on the links, we can state that pressure and green light duration for a link are directly proportional.

Opposite direction pairs, i.e., north-south, east-west, are assigned the same timings; thus, opposite link pressures must be considered as a conjugate when distributing the green light durations. The final distribution formula is provided in Eqs. 11 and 12.

$$G_{N,S}[t] = \left[\frac{P_N[t] + P_S[t]}{P_N[t] + P_S[t] + P_W[t] + P_E[t]}\right] C[t] \quad (11)$$

$$G_{E,W}[t] = \left[\frac{P_E[t] + P_W[t]}{P_N[t] + P_S[t] + P_W[t] + P_E[t]}\right] C[t] \quad (12)$$

B. Hardware and software implementation

We simulated a complete design for an embedded controller implementing the smart traffic light controller on the Renode virtual development platform, which allowed us to validate the proposed system in specifically defined conditions. As target hardware, we selected the HiFive Unleashed board, based on a 64-bit RISC-V instruction architecture.

The controller unit has multi-core technology and is suitable for real-time performance; the fast and expandable RAM is sufficient to accommodate all necessary data coming from the IoT network or generated by the max-pressure algorithm task. The choice of architecture also opens up future expansions of this work, as the selected hardware has been proven suitable for running a full-fledged operating system [7] and image detection powered by deep learning [8]. Connection to the IoT network can be accomplished through the board's Gigabit Ethernet MAC interface; for the purpose of this study, we used a function to emulate the process that receives suggestions from the network. Finally, the board's GPIO pins are used to control the LEDs and emulate the input coming from sensors counting the vehicles arriving at the intersection.FreeRTOSwas the selected operating system. It is a real-time operating system that provides task separation and isolation and enables safety-critical operations.

In order to run our simulation, we created six tasks in total:

1) Init task: Initializes the RTOS and creates all other tasks.

2) Max-pressure task: Based on the max-pressure controller algorithm (described in section II-A2), this task runs once per cycle; it takes queue lengths as input and gives green light durations as output. Queue lengths are saved as a "snapshot" at the end of each half cycle. Then, the algorithm calculates pressures and green light durations. The Max-pressure task also logs past data into a memory buffer: it triggers the Network task on each cycle where the buffer is filled, and the Network task can then access the data stored in the buffer and send it to a cloud server.

3)Fixed-mode Task: Sets fixed green light durations, and exists as a fallback in case of failure of the Max-pressure task. The system must be able to face any critical unexpected issue (such as broken sensors) and switch to a predefined standard behavior. If the network becomes unavailable and the max pressure algorithm cannot exchange logs with the cloud server, this task ensures that the system remains safe and coherent. In this work, we also used the fixed mode task as a reference for comparison with the Max-pressure task.

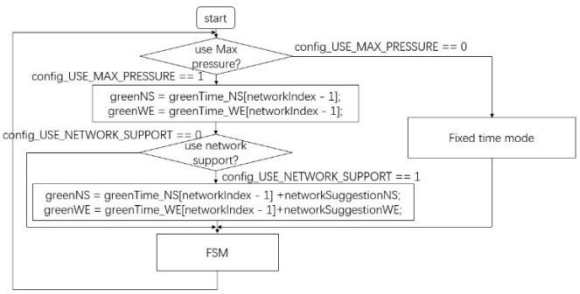


Fig. 2: Flowchart of the Traffic Lights task.

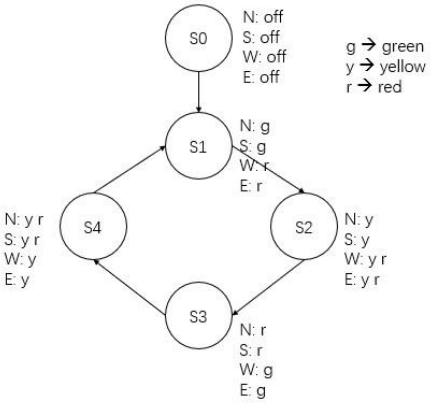


Fig. 3: Structure of the Traffic Lights task FSM.

4) Queue-length task: Detects the number of vehicles waiting in a queue. This task is always running.

Let us consider the intersection in Fig. 1, housing 4 links: for each link, an input and an output sensor detect the number of vehicles stationing between the top and bottom ends of the link. We used GPIO buttons to simulate these sensors. When a vehicle enters the link, the sensor triggers the system to increment the variable monitoring the link's queue length.

5)Traffic Light Task: This task displays the traffic light colors by operating a GPIO. The task's flowchart is shown in Fig. 2. A configuration variable determines whether the Max-pressure task will be activated from suspension, otherwise, the Fixed-mode task will execute. The calculated timings will be filled into global variables greenNSand greenWE. The GPIO manipulation routine can be represented by an FSM (Finite State Machine) consisting of 4 states, as shown in Fig. 3; the time for each state is controlled by the aforementioned global variables. In S1, the north-south green light and east-west red light are turned on. S2 is the period of yellow light. S3 is the reverse state of S1, where the east-west green lightand the north-south red light are turned on. Similarly, S4 is the reverse state of S3.

6)Network task: The abstraction of a sample network data transmission. It accesses the logs stored in internal memory buffers and prints the data to the terminal. Its output comprises the queue lengths, pressures, and green light durations; these logs can be used for machine learning or artificial intelligence-based studies in order to predict congestion and research further improvements. The network task may also receive operation data from other intersections, enabling future scenarios for intelligent traffic regulation based on real-time data sharing: for instance, the master controller can reduce or increase input flow from one intersection to another.

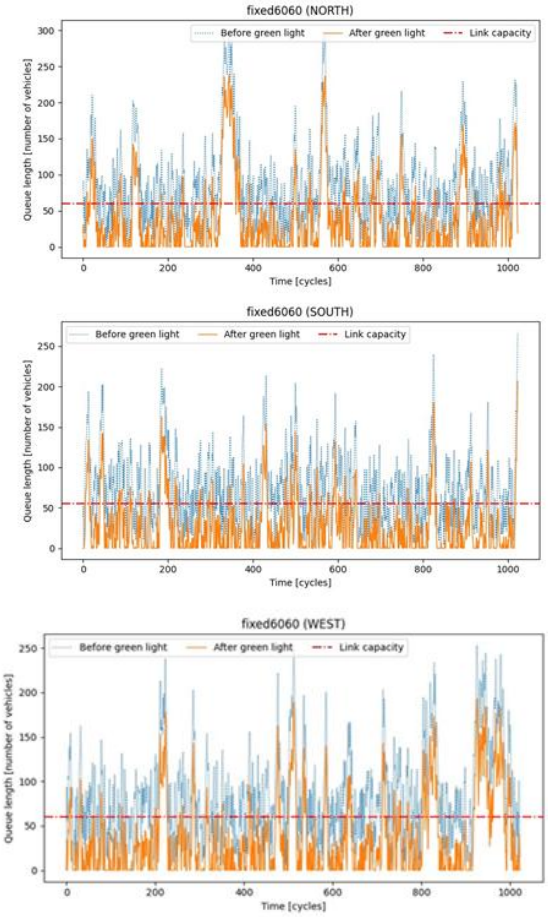
III. RESULTS AND DISCUSSION

We performed traffic simulations in order to evaluate the performance of the two examined solutions: the fixed-timing traffic light and the dynamic-timing traffic light based on the max-pressure algorithm. We present an analysis based on data logs created by the Network task, as explained in section II-B.Because the main goal of traffic control algorithms is to reduce queue lengths (and, consequently, waiting times), the main performance metric taken into account is the queue length at each link. In the following plots, we visualize with a dotted blue line the queue sizes taken as a snapshot just before the green light starts, and with a continuous orange line the queue sizes just before the red light comes back on. The capacity of the link is displayed as a horizontal red line.The simulation model is based on an irregular intersection model: each link is modeled as having a different capacity. The input stream, i.e., the number of vehicles entering a link within each cycle, is randomly generated from a uniform distribution within the interval [2,100]. The link capacities and random distribution are chosen so as to enable the possibility for a link to become overloaded within a single cycle. Furthermore, we make the simplifying assumptions that no new vehicles reach the intersection during a green light, and that vehicles can exit the intersection at a rate of 1 vehicle per second; cycle time $C_t[t]$ is assumed to be constant and equal to 120 seconds.

A. Fixed Mode (NS:60, WE:60)

As a first test, we limited the system to using only Fixed Mode, by setting equal green light durations on each intersection link. The 120-second cycle time is split into symmetric intervals of 60-second green light, and 60-second red light; the simulation is then run for 1024 cycles. The queue lengths for cycle time are displayed in Fig. 4.

Results show sustainable queue growth in each link for the given input distribution. On average, this setting managed to keep the queues within link capacity 73% of the time and prevented unbounded growth in the queues. Using this kind of symmetric timing works well if, like in our experiment, vehicle arrival times have a uniform random distribution.



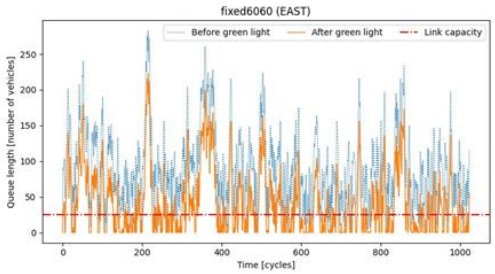


Fig. 4: Evolution in time of the queue lengths for the 4 intersection links when applying Fixed Mode (NS:60, WE:60).

B. Fixed Mode: NS:70, WE:50

In this simulation, we distribute fixed green light durations in an asymmetric way: 70 seconds for the North and South links and 50 seconds for the West and East links. These cycle splits would be proportional to the relative capacity of the links, as per Eqs. 1 and 2. However, given a uniform random distribution of the vehicle arrival times, this choice puts increased stress on the West and East links (which are undersized with respect to the rest) and demonstrates the consequences of suboptimal timing settings.

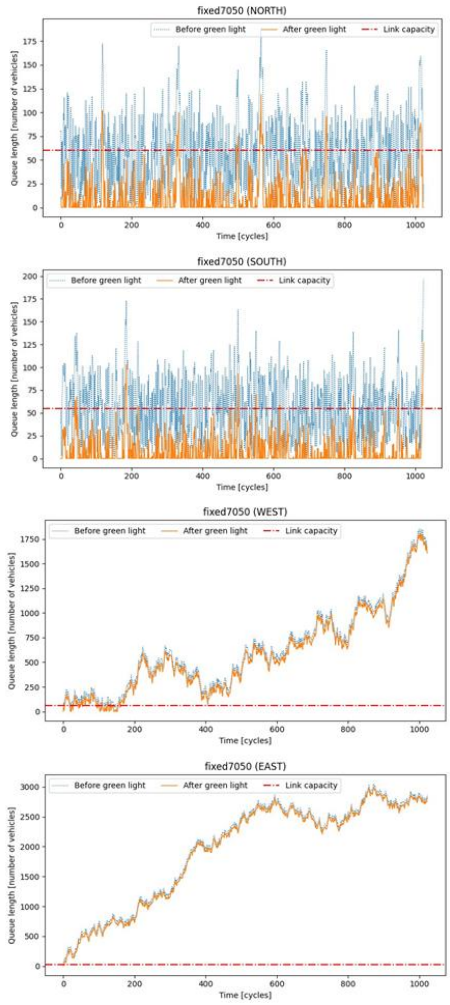


Fig. 5: Evolution in time of the queue lengths for the 4 intersection links when applying Fixed Mode (NS:70, WE:50).

Fig. 5 shows that, while the North and South links experience lessened pressure, the East and West links quickly reach unbounded growth and cannot adequately process the queue. In fact, unbalanced fixed timings can lead to catastrophic results in the event of unforeseen traffic situations. On average, this setting was able to maintain the queues below capacity only 50% of the time, as the N-S links spent most cycles below capacity while the W-E links remained overloaded at all times.

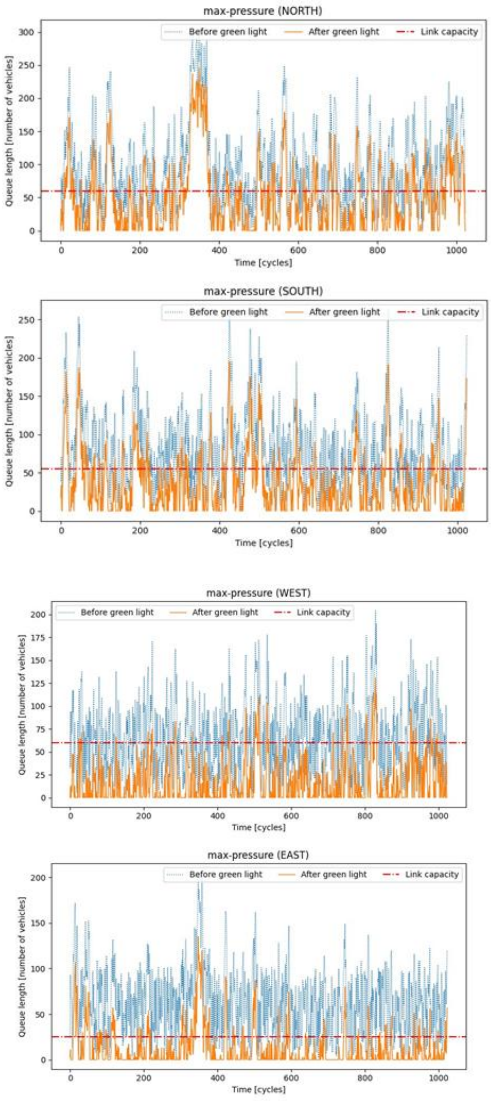


Fig. 6: Evolution in time of the queue lengths for the 4 intersection links when applying the Max-Pressure algorithm.

C. Max-Pressure Mode

For this final simulation, we again executed 1024 cycles leaving the devised system in Max-pressure mode. Results are displayed in Fig. 6. The max-pressure algorithm drew the best overall result, with average queue lengths staying below link capacity for 80% of the time. The green light durations are also instantaneously updated, taking into account the current vehicle load and local capacity. An interesting effect can be observed in the North link at

around the 300th cycle: as the system is exposed to overload, the max-pressure controller eliminates the effect of the disturbance and manages to stabilize the system. In the same way, the East link, which has the smallest capacity, is managed more efficiently and spends more time within capacity than with the symmetric Fixed Mode (83% rather than 49%). A summary of the time spent under capacity for each link using the three different algorithms is reported in Table I.

Algorithm	N	W	E	S	Avg.
Fixed (60, 60)	80%	78%	49%	84%	73%
Fixed (70, 50)	97%	7%	0.1%	98%	50%
Max-pressure	67%	93%	83%	76%	80%

TABLE I: Percent of cycles spent within capacity for each link, given different timing algorithms

IV. CONCLUSION

We have presented a design for an embedded Smart Intersection for Smart Traffic controller based on a real-time operating system kernel. The system is able to receive input data from local sensors as well as the IoT network in order to determine the traffic congestion in different links of the intersection, and implements dynamic traffic light timing based on the maxpressure algorithm, with the possibility to switch to a fixed timing mode as a fallback. The proposed SIST, although a simple prototype in the current version, demonstrated the capability to mitigate the problem of increasing queue lengths in congested intersections. The capabilities of the system can be further enhanced, for example, by extending the input sensor network with integrated data streams coming from different sources such as cameras or GPS signals from voluntary user participation in the Google Maps API. The selected hardware is also powerful enough that traffic analysis and algorithm tuning features may be added to the system through the TinyML paradigm. All these tools open the way smart traffic light controllers that can be deployed without previous knowledge of traffic conditions, and that are able to adapt and handle unforeseen variations in the traffic flow.

REFERENCES

[1] W. Alajali, S. Gao, and A.D. Alhusaynat. Fog computing based traffic and car parking intelligent system. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 11945 LNCS:365–380, 2020.

[2] Y. Alsaawy, A. Alkhodre, A.A. Sen, A. Alshanqiti, W.A. Bhat, and N.M. Bahbouh. A comprehensive and effective framework for traffic congestion problem based on the integration of iot and data analytics. Applied Sciences (Switzerland), 12(4), 2022.

[3] RghiouiAnass, Hernafi Yassine, and Bouhorma Mohammed. Iot for its: A dynamic traffic lights control based on the kerner three phase traffic theory. International Journal of Computer Applications, 145(1):40–48, Jul 2016.

[4] O. Barzilai, A. Giloni, N. Voloch, and O.L. Steiner. Auction based algorithm for a smart junction with social priorities. Transport and Telecommunication, 21(2):110–118, 2020.

[5] O. Barzilai, N. Voloch, A. Hasgall, and O.L. Steiner. Real life applicative timing algorithm for a smart junction with social priorities and multiple parameters. 2019.

[6] S. Dahbour, R. Qutteneh, Y. Al-Shafie, I. Tumar, Y. Hassouneh, and A.A. Issa. Intelligent transportation system in smart cities (itssc). Advances in Intelligent Systems and Computing, 868:1157–1170, 2018.

[7] M. Numan Ince, Joseph Ledet, and MelihGunay. Building an open source linux computing system on risc-v. In 2019 1st International Informatics and Software Engineering Conference (UBMYK), pages 1– 4, 2019.

[8] Shalini K, Abhishek Kumar Srivastava, Surendra Allam, and DilipLilaramani. Comparative analysis on deep convolution neural network models using pytorch and opencvdnn frameworks for identifying optimum fruit detection solution on risc-v architecture. In 2021 IEEE Mysore Sub Section International Conference (MysuruCon), pages 738– 743, 2021.

[9] Jian Liu, Jiangtao Li, Lei Zhang, Feifei Dai, Yuanfei Zhang, Xinyu Meng, and Jian Shen. Secure intelligent traffic light control using fog computing. Future Generation Computer Systems, 78, 02 2017.

[10] H. Nurwarsito and H. Nugroho. Implementation of smart traffic light prototype using mqtt protocol for emergency vehicles. pages 38–43, 2021.

[11] B. Siripatana, K. Nopchanasuphap, and S. Chuai-Aree. Intelligent traffic light system using image processing. pages 14–18, 2021.

[12] Y. Tashtoush, M. Al-refai, G. Al-refai, N. Zaghal, D. Darweesh, and O. Darwish. Dynamic traffic light system to reduce the waiting time of emergency vehicles at intersections within iot environment. International Journal of Computers, Communications and Control, 17(3), 2022.

[13] Pravin Varaiya. Max pressure control of a network of signalized intersections. Transportation Research Part C: Emerging Technologies, 36:177–195, 2013.

[14] X. Wei, J. Zhao, L. Zhou, and Y. Qian. Broad reinforcement learning for supporting fast autonomous iot. IEEE Internet of Things Journal, 7(8):7010–7020, 2020.

[15] W. Wen. A dynamic and automatic traffic light control expert system for solving the road congestion problem. Expert Systems with Applications, 34(4):2370–2381, 2008.

[16] Q. Wu, J. Wu, J. Shen, B. Yong, and Q. Zhou. An edge based multiagent auto communication method for traffic light control. Sensors (Switzerland), 20(15):1–16, 2020.

★★★