LOGI – Scientific Journal on Transport and Logistics
Vol. 12 No. 1 2021 DOI: 10.2478/logi-2021-0010

Smart City Decision Making System Based on Event-driven Platform

Andrej Saric[1], Ivona Zakarija[1], Vedran Batos[1]* and Srecko Krile[1]

[1]*University of Dubrovnik, Cira Carica 4, Dubrovnik 20000, Croatia; Email: andrej.saric@unidu.hr, ivona.zakarija@unidu.hr, vedran.batos@unidu.hr, srecko.krile@unidu.hr*

***Corresponding Author:** Vedran Batos

Received: 3 December 2020; Revised: 28 January 2021; Accepted: 1 February 2021; Published: 2 June 2021

Abstract: With the occurrence of rapid urbanization and intensive growth of different modes of transport, we face issues such as resource management, energy demand and lack of capacity due to overcrowding. To help with these issues we leverage technology and the Internet of Things (IoT) to develop smart cities. In this paper, we propose an innovative approach to development of new smart city event-driven platform with which we want to simplify data input, data transformation, and decision making processes. The platform uses events, labels and reliability factor to make decisions and trigger actions. This paper begins with an overview of the smart city framework and we review the characteristics of event-driven models. The event-driven platform is presented and discussed. The paper closes with our findings, suggestions, open issues, and future research possibilities.

Keywords: Smart city, transportation, IoT, decision making, event driven, software

1. Introduction

Rapid evolution in computing and technology allowed us to start a new way of life specially supported by intelligent transport and smart resource management. Technology is starting to take up new shapes and dimensions which has resulted in it becoming omnipresent and pervasive. Connectivity and ubiquitous computing enabled users to use their smart devices anywhere and anytime which had a huge impact on our society. Continuous connectivity enabled access to numerous data sources and real-time data transfer. By being able to connect to various systems and gather different types of information from users and Internet of Things (IoT) devices we are rapidly gathering huge amounts of data. Collecting information from multiple data sources enables proper real-time analysis; crucial in places where continuous adjustments to the environment are required. In this paper, we introduce a new event driven platform for decisions making and action triggering in smart city systems. The novel approach to development will allow us to avoid building or implementing monolithic systems and their pitfalls. Leveraging the novel design result in new model advantages reflecting to the various data fields, from input to analysis and decision making, including transformation, sharing and integrity. The remainder of this paper is organized as follows. Section 2 introduces smart cities and explains their importance in today's society. Section 3 explains IoT and its role in the development of smart cities. Section 4 discusses event driven model characteristics. Section 5 describes what the proposed event driven platform is and how it is applied on smart cities along with the technical specifications, concerns and possibilities for improvement. Section 6 concludes the paper with our findings, suggestions, open issues, and future research possibilities.

2. Smart Cities

UN reports state that by 2050 global percentage of people living in urban areas will be 68%. This number shows that accelerated migration from rural terrains into cities is happening and that we are transforming into a highly urbanized society. With this rapid rise in urbanization many problems are emerging such as: overpopulation, transportation, resource management, energy demand, gas emissions, and others.

To mitigate these problems a new framework emerged called "Smart City". This new framework aims to improve cities to become more efficient by having technology embedded across all city services. More about smart cities and their implementation can be found in [1] and [2]. By introducing technology into existing services which are consumed by citizens and visitors we can create a new value for them. It is important to stress out that in our cities we have citizens that use services daily, but in some highly touristic areas we also have tourists which will also consume some of the city's services in a certain timeframe. We need to make sure that our services can be consumed by anyone which means that we must make them simple to use, localized and efficient. Citizens are primary users and the core which is needed to transform a city into a smarter city. We need to inform and educate the citizens about advantages of becoming a smart city. By educating citizens and getting their support in the development of the smart city we will also gain access to additional data which is crucial for the development of citizen targeted services. To be able to introduce smart city framework in our cities we need technology. Technology is used to create smart services with new values for citizens. To be able to do that, we need to gather data which will be cleaned, transformed, analysed and used to make certain decisions. To be able to gather the amount of data that we need, we will use technology and the Internet of things.

3. Internet of Things

Internet of things plays an important role in the development of smarter cities. IoT provides physical everyday objects with vision, hearing, logic and ability to communicate which allows them to provide us with a certain amount of automatization based on the collected data. By communicating with each other these devices create flows of data and allow more precise decision making to happen; especially in the area of emergency responses where human based decision making might be biased or difficult. To integrate IoT platform into a city we need to invest into the infrastructure and devices such as sensors and microcontrollers. In the existing cities, we need to adjust the infrastructure as much as we can while in case of building new cities, we need to start preparing the infrastructure immediately so that it can support new technologies. With advances in areas of ubiquitous computing, pervasive computing and communication technologies we are able to create IoT devices that can be integrated in every area of our everyday life. IoT services and application categorization samples can be found in [3]. To improve on the existing services that are based on real-time interaction with IoT devices we need to bring analytics and data generation down to the source of data. To be able to do this we need to use edge computing [4] which will allow us to move data processing to the edge of the network which are IoT devices. This will optimize our system in terms of resources that are being used in the cloud service.

4. Event Driven Model Characteristics

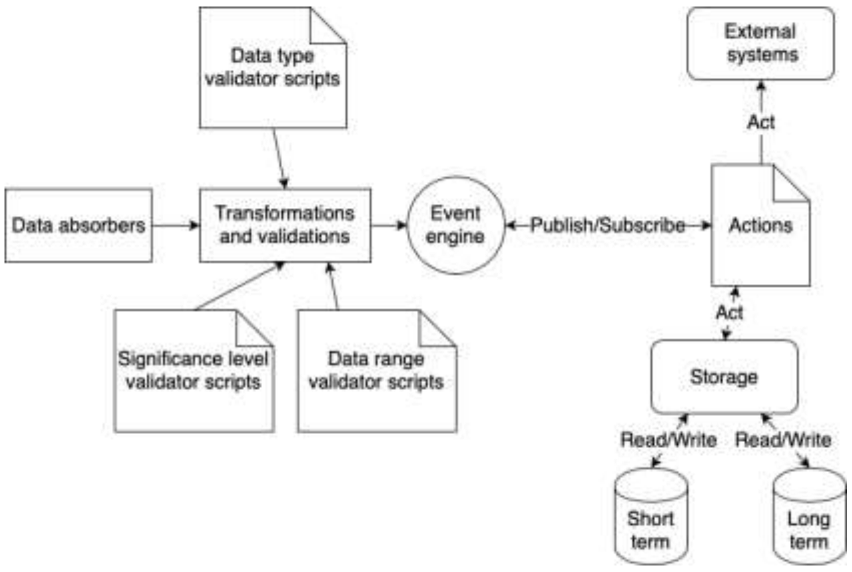
Event-driven models are focused on producing and managing events. It is a very flexible architectural model in which event generators do not need to know who will be receiving the event and what will be done with the event. Characteristics of event-driven models are:

- Fast at adapting to change which allows us to create more responsive architecture
- Fault tolerant, allowing working continuity of the components if one of the components of the system fails, enabling robust distributed systems design
- Enabled triggering of event generations by external sources
- Enabled analysis, transformation and further forwarding of events within a system

We believe that these characteristics make the event-driven model a suitable choice for smart city platform architecture. By using the event-driven model's fast adaptive characteristics we can develop a robust distributed platform for smart cities and their subsystems. An example of event-driven architecture in smart cities can be found in [5].

5. Proposition of a New Event Driven Model

Today we analyse huge amounts of data and try to achieve automatization. To be able to fully automate smart cities we need to consider both predictable factors and unpredictable factors. Cities are highly unpredictable because multiple factors can influence their services. To minimize the effects of the unpredictability we propose a new event driven platform that will enable real-time data processing, data sharing and decision making. To be able to predict certain behaviors of any subject we need data that is gathered, validated, processed and transformed to recognize patterns. These patterns are events that have occurred in a certain sequence and can be used for predictions. Building blocks of our proposed model are as follows (Fig. 1).



Data type validator scripts
External systems
Act
Data absorbers
Transformations and validations
Event engine
Publish/Subscribe
Actions
Act
Significance level validator scripts
Data range validator scripts
Storage
Read/Write
Read/Write
Short term
Long term

Fig. 1 Event driven model diagram. Source: authors

Data can be gathered from various software solutions, scripts, parsers, IoT devices. That data uses different protocols and structures which means it can be in many forms and formats. To solve the issue of unstandardized data we propose using data absorbers. Data absorbers gather raw world data from external data sources such as IoT devices and web so that we can provide our platform with the data needed to make decisions. By creating internal protocols and formats we can enable creation of data absorbers and ensure operability of the system. The absorbers are independent isolated scripts that can be enabled and disabled. Advantage of creating independent scripts for data absorbers is that we can use third party libraries which will create isolated dependencies that will not influence the whole system. Using set protocols and predefined data formats makes it easier to debug and support these scripts. Example of a data absorber would be a Python script that scrapes web to get weather data or a smart parking system that tracks the number of vacant parking spots.

5.1 Transformation and Validation Layer

In this layer, we already have the data in a common format which we propose to be Binary JSON (BSON) [6], or MessagePack [7] which supports many programming languages and is small in size. More about object serialization techniques and formats can be found in [8]. To be able to work with incoming data we propose writing individual isolated scripts that will act as validators and can be plugged in and executed when needed. This approach provides an advantage because it allows us to use different programming languages and their built-in functionalities, also we can easily integrate third party libraries into our validators without making the whole layer dependent on them. For some validations, we might need to do additional calls to some external script or a web service. In case of invalid data it will be discarded and event will be logged so that we can diagnose the problem. Potential problems that can provide us with invalid data can occur in data sources or data absorbers and by registering these issues early in the pipeline, we can easily fix them. We propose three types of validators to be used: data type, data range and significance level validator. To achieve full system components interoperability these types will have predefined data input and output protocols.

Data Type Validator ensures that data coming into the system has correct data types to be able to preserve high data quality and consistency. Example in case if we were measuring temperature would be that we would check if the value is a decimal number. Another example would be if we were counting the number of people that entered some building; value would have to be an integer.

Data Range Validator provides range validation. Depending on the use case we might want to avoid processing the data which is out of range and which could potentially negatively influence our data set. To avoid this, we need to validate data value based on whether it fits certain predefined criteria. Example in case of the temperature is that we would validate if the decimal number is within a certain acceptance range based on the unit that temperature was provided in.

Significance Level Validator is used to approve data significancy. To avoid the potential risk of processing heaps of massive wasteful data we use significance level to decide which data will be discarded or kept. Example would be if we are collecting temperature data and we measure a change in data of +/-0.5 which is below our predefined significance level of +/-1. Data would be ignored and discarded and in case that measured data is greater or equal to +/-1; we will process it and send it forward to other components of the system.

5.2 Events

Once we have data which is cleaned, transformed and structured we can create events. Events are used to broadcast messages and trigger actions between all components of our system. Events need to be descriptive so that we can understand what they do; and they need to contain structured data that will be easy to consume by other components in our system. We propose the following structure for an event. Every event has five root fields: label identifier, data source, trail, reliability factor and meta data. Label Identifier signifies a specific meaning of the label, so that the other components in the system know how to consume and utilize the data. Examples of the labels would be “temperature”, “num_vacant_parking_spots”, “humidity”, etc. Data Source holds the identifier of the source of the data which can be a process, external system, user or a device. Trail is a part of every event which is updated as the event flows through the system. It is an ordered list of script identifiers through which this event data was generated, validated, transformed, and shaped. The advantage of having this information is that other components of the system can use it to trace back where did this event originated from and which scripts processed it. Based on this information components can decide what to do with the event. Reliability Factor is considered as a weight of an event. Based on this value an event can trigger actions. Reliability factor is calculated at transformation and validation level and can be based on the source, time, sentiment and redundancy. Meta-data presents the data associated with the event. It is provided by data sources which is then transformed to follow unified protocol and format. Example in case of the temperature would be meta-data object with fields such as: timestamp, value, location.

5.3 Actions

Events are forwarded to the action scripts. Action scripts decide what to do based on the event data. Action script is able to do the following tasks: store, act, and ignore. Store task is able to store event. Action script might have different conditions to trigger an action. Conditions might be: multiple events must happen to trigger script action, and / or, action can only be executed at a specified date or time. To be able to conform to these two conditions, the script can store the event in the short-term storage of the system and wait for a trigger. Stored event must have a defined expiry time after which it can be discarded from the short-term storage. Act task is triggering an action script that can act if the event fulfills predefined conditions specified in the script. Action script then acts based on a specific predetermined action or in conjunction with other events. Examples of potential actions are: data processing, system wide alerting, triggering external APIs, sending data to front end client apps, transforming and saving data to long-term storage. Ignore task is able to ignore event based on the label identifier, trail, or reliability factor.

5.4 Event Engine

Internal communication will use publish-subscribe protocol while keeping in mind the limitations described in [9]. We propose a central source which would be a broker that would be in charge of receiving and distributing events. Pub-sub clients can both publish events or subscribe to receive certain events. This is a lightweight solution because broker does not store data; broker just distributes it among subscribers. With this approach, we are lessening the strain on the network because in the request-response model we must request for the data from the broker and then the broker must send a response back to us. By using pub-sub model we want to avoid making requests every time one of the components needs data and lessen the strain on the system. To achieve publish-subscribe communication we propose using MQTT [10], ZeroMQ [11,12] or SignalR [13]. Example of an event flow can be found in (Fig. 2).

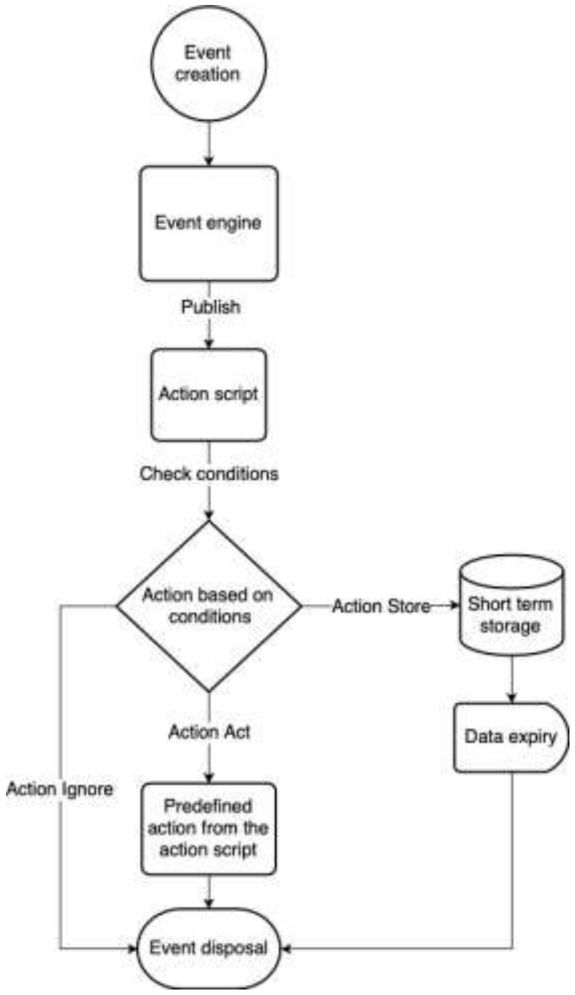


Fig. 2 Event lifecycle flowchart. Source: [10-13]

5.5 Storage

Actions

Actions can also trigger data saving. Data that will enter the system will be contained within a predefined structure but we expect that meta-data which will hold readings, location, etc. might not be flexible enough so we want to allow the system to accept any kind of structure within meta-data field. For data storage, we propose using non-relational (NoSQL) databases because they do not enforce a strict schema, have flexible data modeling and provide excellent scalability. A good comparison of NoSQL databases can be found in [14] and a comparison for NoSQL and MySQL can be found in [15].

For specific cases where we have different requirements we propose hybrid approach as explained in [16].

5.5.1 Long-term

Data in long-term storage can and should be used for historical analysis and future predictions. This data will also be used by client facing apps and systems.

5.5.2 Short-term

Short-term storage should be used for data that does not have high importance over a long period of time and cannot be used for prediction and learning. Information that will be stored here are the ones that have relevance only within a certain timeframe after which they can be disposed of.

For short-term storage, we propose Redis [17] because it can be used as a database, cache, message broker and it has great runtime performance as stated in [18].

5.6 External Systems

We need to provide outlets for data to flow into external systems and hooks for external systems to request the data from the platform. To provide operability between the platform and external systems we propose the creation of frameworks and libraries that will provide an API interface for systems to interact with the platform. Functionalities that frameworks and libraries should provide, are as follows.

5.6.1 External System Registration

We need to provide a way for the external systems to register on to the platform. We propose registration to be done using unique system identifiers. After the registration system is provided with access which can be revoked in case of system integrity and security being compromised.

5.6.2 Data Retrieval

There is a need to allow external systems to retrieve the data of interest to them. Based on the system we can expect that the data will be used for decision making, triggering activities, presenting data to clients. Two options of data retrieval are prosed: manual – happens when triggered, and scheduled – happens in predefined intervals.

5.6.3 Data Provider

If an external system provides the platform with the data using this functionality it means that it can be categorized as a data source. Data provided by the system will pass through data absorbers as explained earlier in the article. System needs to conform to platform’s standards and data structures by implementing predefined structural formats.

There are potential pitfalls that we need to avoid such as one external system providing heaps of data and flooding the platform which can result with making the data biased and integrity compromised. To avoid this, we propose using security mechanism of API keys [19] and OAuth [20]. Examples of OAuth being used in IoT networks can be found in [21] and an evaluation of OAuth can be found in [22]. Activities in AAL (Ambient Assisted Living) [23-25] has been analyzed.

API keys and OAuth would allow the platform to disable access for certain data providers. To decide which data providers, need to be disabled, we can use overall reliability factor of that providers historical data; or we can use the frequency in which the provider generates the data.

5.6.4 Trigger Creation

Allows external systems to create triggers based on which they can react and make decisions.

Example is an irrigation system creating a forecast trigger that will activate once the platform receives information with high reliability factor that there will or will not be rain. This trigger can start a process on the external system which will then check the current ground humidity and turn on the sprinklers if there will be no rain. The advantage of this approach is that in this example forecast data comes from the smart city platform which has multiple data sources with validated data; meanwhile the forecast data that irrigation system is gathering on its own might be from a single data source where data integrity may be compromised, or data could be irrelevant.

6. Conclusion

There is no doubt that the role of smart cities will become crucial in the years to come. Because of this we need to be able to create safe and error-tolerant smart city platforms which can integrate other smart city services as their subsystems. In this article, we argue that using event-driven models is suitable for creation of smart city platforms because of its flexibility, responsiveness and fault tolerance that enables us to build robust distributed platforms. We discussed potential issues that need to be resolved and have proposed potential solutions and technologies. Authors believe that with this research they have shown that event-driven model is suitable for smart city platform architecture. Future research will focus on security and disaster recovery aspects of the proposed model alongside with the actual implementation on a small scale.

Abbreviations: AAL (Ambient Assisted Living), API (Application Programming Interface), BSON (Binary JSON), IoT (Internet of Things), MQTT (Message Queuing Telemetry Transport), NoSQL (Not Only Structured Query Language), OAuth (Open Authentication)

References

[1] Mohanty, S, Choppali, U. & Kougianos, E. (2016). Everything you wanted to know about smart cities: The Internet of things is the backbone. IEEE Consumer Electronics Magazine 5(3), 60-70. Retrieved June 6, 2020, from PubMed database on the World Wide Web: http:// www.pubmed.gov. DOI: 10.1109/mce.2016.2556879.

[2] Pellicer, S., Santa. G., Bleda, A., Maestre, R., Jara, A & Skarmeta, A. (2013). A Global Perspective of Smart Cities: A Survey. 2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing. Retrieved July 14, 2020, from PubMed database on the World Wide Web: http:// www.pubmed.gov. DOI: 10.1109/imis.2013.

[3] Gigli, M & Koo, S. (2011). IoT. Internet of Things: Services and Applications Categorization, Advances in Internet of Things 01(02), 27-31. Retrieved April 20, 2020, from PubMed database on the World Wide Web: http:// www.pubmed.gov. DOI: 10.4236/ait.2011.12004.

[4] Shi, W., Cao, J., Zhang, Q., Li, Y. & Xu, L. (2016). Edge Computing: Vision and Challenges. IEEE Internet of Things Journal 3(5), 637-646. Retrieved Mai 18, 2020, from PubMed database on the World Wide Web: http:// www.pubmed.gov. DOI: 10.1109/jiot.2016.2579198.

[5] Filipponi, L., Vitaletti. A., Landi. G., Memeo, V., Laura, L. & Pucci, P. (2010). Smart City: An Event Driven Architecture for Monitoring Public Spaces with Heterogeneous Sensors. Fourth International Conference on Sensor Technologies and Applications. Retrieved December 6, 2006, from PubMed database on the World Wide Web: http:// www.pubmed.gov. DOI: 10.1109/sensorcomm.2010.50.

[6] BSON (Binary JSON): Specification. (2019). Bsonspec.org, Retrieved June 11, 2020, from PubMed database on the World Wide Web: http:// www.pubmed.gov. http://bsonspec.org/spec.html.

[7] GitHub. (2019). Retrieved June 11, 2020, from PubMed database on the World Wide Web: http:// www.pubmed.gov. https://github.com/msgpack/msgpack/blob/master/spec.md.

[8] Maeda, K. (2011). Comparative Survey Of Object Serialization Techniques And The Programming Supports. International Journal Of Computer And Information Engineering 5(12), 1488-1493. Retrieved June 5, 2020, from https://publications.waset.org/15057/comparative-survey-of-object-serialization-techniques-and-the-programming-supports.

[9] Happ, D. & Wolisz, A. (2016). Limitations of the Pub/Sub pattern for cloud based IoT and their implications. 2016 Cloudification of the Internet of Things (CIoT), 2016. Retrieved December 6, 2006, from PubMed database on the World Wide Web: http:// www.pubmed.gov. DOI: 10.1109/ciot.2016.7872916.

[10] MQTT Version 3.1.1. (2019). Docs.oasis-open.org. Retrieved June 15, 2020, from https://docs.oasis-open.org/.

[11] Hintjens, P. (2013). ZeroMQ: Messaging for Many Applications. 1st ed. Sebastopol, CA: O'Reilly Media. Retrieved on June, 11, 2020, from https://www.oreilly.com/library/view/zeromq/9781449334437/.

[12] 23/ZMTP ZeroMQ RFC. (2019). Rfc.zeromq.org, Retrieved June 11, 2020 from https://rfc.zeromq.org/spec:23/ZMTP.

[13] dotnet/aspnetcore. (2020). GitHub. Retrieved June 11, 2020, from https://github.com/dotnet/aspnetcore/blob/master/src/SignalR/docs/specs/HubProtocol.md.

[14] Amghar, S., Cherdal, S. & Mouline, S. (2018). Which NoSQL database for IoT Applications?. 2018 International Conference on Selected Topics in Mobile and Wireless Networking (MoWNeT). Retrieved from PubMed database on the World Wide Web: http:// www.pubmed.gov. DOI: 10.1109/mownet.2018.8428922.

[15] Phan, T., Nurminen, J. & Di Francesco, M. (2014). Cloud Databases for Internet-of-Things Data. 2014 IEEE International Conference on Internet of Things(iThings), and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom). Retrieved June 5, 2020, from PubMed database on the World Wide Web: http:// www.pubmed.gov. DOI: 10.1109/ithings.2014.26.

[16] Braulio, L., Moreno, E., de Macedo, D., Kreutz, D. & Dantas, M. (2018). Towards a Hybrid Storage Architecture for IoT. 2018 IEEE Symposium on Computers and Communications (ISCC). Retrieved Retrieved June 10, 2020, from PubMed database on the World Wide Web: http:// www.pubmed.gov. DOI 10.1109/iscc.2018.8538474.

[17] Redis Protocol specification – Redis, Redis.io. (2019). Retrieved June, 11, 2020, from https://redis.io/topics/protocol.

[18] Puangsaijai, W. & Puntheeranurak, S. (2017). A comparative study of relational database and key-value database for big data applications. 2017 International Electrical Engineering Congress (iEECON). Retrieved from PubMed database on the World Wide Web: http:// www.pubmed.gov. DOI:10.1109/ieecon.2017.8075813.

[19] Farrell, S. (2009). API Keys to the Kingdom. IEEE Internet Computing, Vol. 13(5), 91-93. Retrieved June 5, 2020, from PubMed database on the World Wide Web: http:// www.pubmed.gov. DOI: 10.1109/mic.2009.100.

[20] Leiba, B. (2012). OAuth Web Authorization Protocol. IEEE Internet Computing, vol. 16, no. 1, 74-77. Retrieved June 5, 2020, from PubMed database on the World Wide Web: http:// www.pubmed.gov. DOI: 10.1109/mic.2012.11.

[21] Emerson, S., Choi, Y., Hwang, D. & Kim, K. (2015). An OAuth based authentication mechanism for IoT networks. 2015 International Conference on Information and Communication Technology Convergence (ICTC). Retrieved June 8, 2020, from PubMed database on the World Wide Web: http:// www.pubmed.gov. DOI: 10.1109/ictc.2015.7354740.

[22] Darwish, M. & Ouda, A. (2015). Evaluation of an OAuth 2.0 protocol implementation for web server applications. 2015 International Conference and Workshop on Computing and Communication (IEMCON). Retrieved June 12, 2020, from PubMed database on the World Wide Web: http:// www.pubmed.gov. DOI: 10.1109/iemcon.2015.7344461.

[23] Chen, L. & Nugent, C.D. (2019). Human Activity Recognition and Behaviour Analysis. Switzerland: Springer International Publishing.

[24] Zahid, H., Mahmood, T., Morshed, A. & Sellis, T. (2019). Big data analytics in telecommunications: literature review and architecture recommendations. IEEE/CAA Journal of Automatica Sinica 7(1), 18-38. DOI: 10.1109/JAS.2019.1911795.

[25] Larcher, L., Stroele, V., Dantas, M. & Bauer, M. (2020). Event-Driven Framework for Detecting Unusual Patterns in AAL Environments. 2020 IEEE 33rd International Symposium on Computer-Based Medical Systems (CBMS), 28-30 July 2020 (pp. 309-314). Rochester, MN, USA: IEEE.