Vol. 6, №2, 2022 ISSN 2520-6133

KHAZAR JOURNAL OF SCIENCE AND TECHNOLOGY (KJSAT)

[1] Also H. Issakhanly, H. Isakhanly, H. Isakhanli, H.A.Isayev, G.A.Isayev, or G.A.Isaev due to differences in transliteration. I use Hamlet Isayev in mathematics and science fields and Hamlet Isakhanli/Isaxanli in humanities.

Khazar University
41 Mehseti str., Baku AZ1096
Republic of Azerbaijan
Tel: (99412) 4217927
Website: www.kjsat.com

KHAZAR UNIVERSITY PRESS

Mündəricat

Evolution of Portable Devices

Ali Almasli

*Faculty of Engineering and Applied Sciences,
Khazar University, Azerbaijan*

Email: ali.almasli@khazar.org

Abstract

In 2007, the milestone of smartphones, Apple Company, under the leadership of Steve Jobs, released the first iPhone, where applications can be personalized. Although this device is a mobile phone, it made history as the first device with a large touch screen and an internet connection and added a new dimension to the market. In 2008, after the sales success of Apple, the Taiwanese HTC Company launched the first commercially named HTC Dream with the Android operating system, which is open-source code against Apple's IOS operating system iPhone, allowing development by other people and companies. The article examines the development path of mobile phones in the recent period.

Keywords: smartphone, IOS, apple, company, phone

Introduction

The dizzying developments in the smartphone market, and the development of mobile internet infrastructures in parallel with this, meet all the daily life needs of personal users (entertainment, shopping, education, health) in the world. Commercial companies, on the other hand, have started to transfer both their business processes and sales processes from desktop software to mobile software. In this way, people have the convenience of doing almost every operation that they can do with their portable computers, which they carry in their pockets, with their smartphones, that is, with the devices in which they have an internet connection, as an even simpler way of performing their transactions from portable computers and places where they can find internet service (Uğur and Turan, 2015).

Examples of these include banking transactions, video calls, smart home applications that can be managed from smartphones, and many other conveniences. In our country, the majority of the population and the increase in GSM operators have made it possible for smartphones to reach even low-income families, with the increase in service quality and internet speeds, as well as the decrease in tariff fees and the advancement of the information technology network all over the world. According to the data of mobile operators operating in Turkey, the subscription rate exceeding 80 million is an indication that everyone has at least one smart mobile phone in their pocket. The features and processing capabilities of smartphones have allowed doing all the daily operations on computers. As such, people perform many transactions in daily life, within a few minutes, thanks to mobile applications on smartphones. Shopping, entertainment applications and social networks can be followed on mobile devices; By purchasing e-newspaper and e-book applications from application markets, smartphones have become an indispensable part of humanity, not just a communication tool.

According to the mobile market report published by BTK in the third quarter of 2018; As of the end of September 2018, there are 80,637,671 mobile subscribers, including M2M subscribers, which corresponds to 98% of the investments made in Turkey (Information Technologies and Communication Board, 2018). There are 80 million 810 thousand 525 citizens in Turkey according to the last address-based census data made on 31 December 2017 (Turkish Statistical Institute, 2017:80). As it can be understood from the figures, it is a proof that there are more mobile subscribers than the population of Turkey and that mobile devices are used together with it. The most important factor was the increase in mobile line subscriptions, the development of the transaction capabilities of mobile devices, the decrease in prices to access levels, as well as the fact that GSM operators attach great importance to infrastructure investments in order to make their mobile internet infrastructures signal in almost every part of Turkey. 3G1 service, which was put into use in July 2009, reached 65,949,652 subscribers as of the end of March 2018. While it decreased to 8,662,636 at the end of September 2018; The number of 4.5G subscribers increased to 69,341,236. In Figure 1, the number of 2G, 3G and 4.5G mobile subscribers and their usage rates are compared by years. On the other hand, as of the end of September 2018, the number of M2M2 subscribers reached 5 million. (Information Technologies and Communication Board, 2018)

When all the report data are examined in general, we observe that mobile communication service providers in Turkey have developed their infrastructure and in parallel, mobile internet usage of mobile internet users has increased exponentially. With the increase in mobile internet speeds, and the widespread use of mobile devices, the companies that develop mobile applications are developing both within the market and within their own domestic markets, and the packaged software used on desktop computers is rapidly leaving its place for mobile applications.

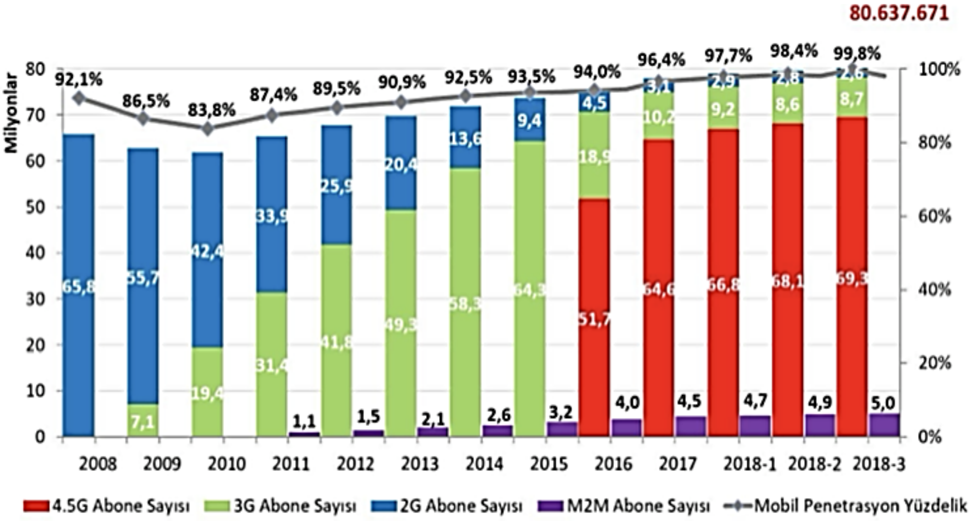


Figure 1: Total Mobile Subscribers and Comparison to Population

Source: BTK Turkey Electronic Communications Industry Quarterly Market Data Report 2018 3rd Quarter, p58

In fact, e-commerce companies, which sell most products and services, develop mobile applications or interfaces for accessing web pages with mobile devices and provide their customers with the service of accessing web pages and shopping in a shorter time than the time it takes to reach the web page by typing in the browser. To meet the needs of people, mobile applications are developed in almost every field. The main reason why smartphones and tablet computers are personal and so attractive to use is the existence of usage-specific mobile applications that can be loaded into them. It is known that the majority of mobile application developers are companies that develop operating systems.

Among the mobile applications that facilitate daily life, speed up and make mobile device use more functional, perhaps the most important is health applications (Demir 2016). Compared to the past, people live longer today and are conscious of a healthy life. The field of health is becoming an industry, as health investments require huge sums of money and satisfy its investors in this sense. Health tourism between countries is increasing exponentially every day. In addition to all these, together with the benefits of technology, health services are digitalized together with the benefits of technology, preventing transaction costs such as e-health and mobile health, and data repetitions are prevented to a large extent. It is aimed to provide more qualified health services and more qualified health personnel staff (Erol and Özdemir, 2014).

In light of all these, T.C. The e-Nabız "Personal Health Data Recording System" mobile application developed by the Ministry of Health allows people to follow up on their health data and to have more qualified information about the examination, treatment, health institution and personnel. All these are very important in terms of providing more transparent and higher quality healthcare services and minimizing transaction costs.

Evolution of Mobile Devices

Although smartphones, of which almost everyone has at least one, have been in our lives for more than ten years, the history of mobile applications and mobile phones is quite old. The ancestor of the smartphones, which we do most of our daily routine work today, has enabled this technology to take place in our lives as classical mobile phones in 1993, with the trials that enabled communication with a one-way transmitter in the Detroit Police Department in America in 1921. Mobile phones with classical Symbian operating systems were developed, and PDAs (Personal Digital Assistants), which we use today, was developed thanks to the mobile operating systems developed in them. The first PDA device is the IBM phone called Simon (HTTPS: mediatrend.mediamarkt.com.tr, 2018).

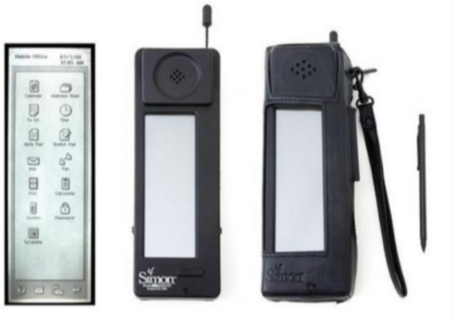


Figure 2: The First Smartphone PDA Produced by IBM

Source: https://mediatrend.mediamarkt.com.tr/akilli-telefon-tarihi/;2018

It is a Symbian operating system with a simple and limited interface developed by the Java company for mobile phones with the development of technology after the Simon device of BM company. The most well-known of the pioneering names of that period, which most of us are familiar with, is the Finnish company Nokia. It has achieved considerable success in the mobile phone market by using the Symbian operating system developed by the Java company in Java Language on its devices. Mobile devices of that day could connect to the internet, send e-mails and even access the internet. However, transactions were carried out with the mobile applications installed as the desired standard. This did not make the devices as attractive as they are today, users still preferred computers for communication. By 2005, the first foundations of today's popular use in the field of mobile devices and mobile applications were laid (https://www.webtekno.com, 2018).

These mobile devices are divided into two categories considering the historical chronology. They are simple devices compared to today, which allows the user to make phone calls and text messages with the Symbian operating system written in the Java programming language, the first of which is described as a mobile phone. The second one is the devices that we describe today as smartphones, which have a colourful, three-dimensional touch screen, contain mobile operating systems as well as the operational capability of the mobile phone, and where personalized mobile applications can be installed (Çakır and Demir, 2014).

When looking at the short history of mobility research, it is not possible to find a commonly accepted definition of the concept. In general, the term mobile is understood as the state of being able to move or be moved, and mobility is the ability to provide movement. The mobile can be anything physical or non-physical, tangible or intangible, as long as it moves or is moved. Therefore, mobile and mobility refer to a wide variety of situations and capabilities. Mobility is sometimes used as a physical movement and sometimes as an interaction between people using mobile technology to communicate with each other. Differently, mobility can be defined as the physical movement of people, as well as the remote interaction with other people or the movement of information resources. Cresswell (2006) separates the concepts of movement and mobility and describes the movement as “the act of displacement that allows objects, people and ideas to circulate between locations”.

Kaufmann et al. (2008) used the term spatial mobility by accepting mobility as the movement of people and objects in real and virtual space, expanding spatial mobility further, referring to one-way, irreversible migration or residential mobility, apart from only two-way travel and repeated daily mobility. According to them, spatial mobility should not be limited to the time between the starting and destination points, and should be seen as the structuring of social life and social integration. They also argue that it is necessary to talk about the social dimension of mobility, it is a very meaningful feature.

Bonns and Kesselring (2001) preferred a social but more restricted definition of mobility: "the human ability to carry out certain projects and plans in motion". They also talked about the new concept of motility2 - the capacity to be mobile. This concept, which is defined as the movement capacity of an animal or human organ in biology and medicine, has been interpreted by Kaufmann et al. According to them, motility also encompasses all the factors that define a person's capacity to be mobile (purpose, strategy, preferences, and skills) and can be expressed as the tendency to be mobile, which can vary in intensity from person to person. The difference between movement and motility can also be explained as follows: Movement indicates mobility performance, while motility indicates mobility potential.

Kaufmann et al. (2008) considered mobility in three dimensions: movements, networks and motility, to overcome the limitations and boundaries of the concept of mobility and to make a more comprehensive definition of mobility.

For example, the telephone is also a movement with a starting point and a destination. Not only people but also objects, ideas and information can move. Networks can be defined as a system of movements. Technical networks (transportation, telecommunications, postal etc.) can be defined by the quality of the infrastructure and services and the access conditions of these services. On the other hand, social networks can be characterized as a set of institutionalized relationships. Motility is the social and spatial capacity of people, objects, ideas or information to act. Capacity encompasses any form of access by the individual or group, the skills they have to take advantage of those accesses and what they can do with those skills, as well as the opportunities for action they may or may not find appropriate. Mobility can be seen as a change of state between these three dimensions.

Situations that can occur between these three dimensions are examined below with examples: A person can move without being mobile: Movement in space does not change the person's state. For example, a businessman travelling around the world with conference centres in international hotels is a good example of this situation. Even though it changes geographically, its status does not change. His world of activities is not related to different environments. A person can be mobile even though he is not moving Internet, e-mail or skype users can be given as an example for this situation. A person talking on a mobile phone may be sitting at a table and not in motion, not every mobile is meant to be on the move. The fact that a book reader can go to the world of the characters of the book by imagining also indicates the state of being mobile without moving. One can be both on the move and mobile: we are talking about both virtual and physical mobility at the same time.

For example, talking on a cell phone while driving. The widespread use of mobile technology, which is considered a telecommunication revolution, has added another meaning to the term mobility. The electronic flow of intangible information, in other words, being mobile, has revealed the concept of virtual mobility. An electronically transmitted phone or e-mail constitutes a virtual extension of the human being in space. Messages sent to many recipients, not just one-to-one, create publicly accessible pieces of mobile information. The information received by the website can also be shown as an example of information mobility, thus virtual mobility.

Urry (1999) termed the virtual flow of information over the Internet as weightless travel and one-way public predetermined transmissions from television as imaginary travel. In the context of social mobility, sociological meanings related to the status transitions of individuals and groups in the social stratum have also been attributed to the term mobility by some scholars. Here, a relationship between social and spatial mobility is mentioned. The increase in social mobility also causes an increase in spatial mobility. The increase in social status brings with it the opportunity to use telecommunication services and to buy a car; therefore, it causes an increase in physical mobility and virtual mobility. On the other hand, wider information and virtual access can mean more physical movement and incentives and opportunities for an increase in social mobility. In other words, the greater the use of telecommunications in a place, the more social mobility there is.

Mobile technologies lead to a reduction in costs and expenses in responding to sudden and competing needs. Realizing many different mobile applications with a single device not only reduces hardware costs but also increases the user's familiarity with the device, its skills and the level of utilization. Location- and time-sensitive jobs are very good candidates for mobilization. Providing information at the point of full efficiency (transferring the right information at the right place and at the right time) increases work efficiency and decision-making efficiency.

One of the conveniences provided by mobile technology is its ease of use. Thanks to the ever-increasing functionality and user-friendly designs of mobile devices, their adoption and spread have been ensured. Mobile phones and portable computers have turned into a visual and public communication tool rather than a means of communication with their eye-catching designs, reduced size, weight, and the addition of many non-communication accessories such as games, calculators, radios, TV and camera.

The other side of the mobility coin, stability, cannot be considered apart from immobility, which can also mean immutability and stability. Mobility and stability, flows and resilience are complementary concepts. (Kellerman, 2006) The development and widespread adoption of mobile technologies has led to scientific research that previously focused on constraints, hierarchy, and shape, shifting to the topics of process, continuous communication, and mobility. Sociologists generally tend to think that movement and dynamism, which are considered new today, are good and old stability is bad, but it should not be forgotten that it is as equal in stability as the value given to mobility and its importance should not be ignored. Some scholars suggest building new social theories on mobility and stability and questioning the current relationship between old stability and new mobility. Today, there is a balance shift between mobility and stability created by the circulation of people and goods, information and ideas, and an example of this change is the disappearance of the traditional distinction between home and work positions, which are considered to be the most basic, with the development of communication technologies. Home and work activities are now intertwined with the use of the Internet and mobile phones. Houses have turned into home offices and the place of residence has begun to be seen as a mobile and temporary location. Some scientists even went further and suggested that cities would also be dissolved.

Urry (2004) states that observations on mobility and fixity point to complex socio-spatial differences, and complex relationships between people and places, in theorizing personal mobility. According to him, places and people are connected by activities, and these activities are carried out intermittently in these physical places. Activities should not be considered independent of places.

Conclusion

The importance of mobility today and the increasing variety and complexity of virtual mobility require that the role of immobility should not be underestimated. Although mobility is dominant in human activities, immobility can also be thought of as the stopping point in the mobile world, in other words, stagnation provides the formation of mobility. “The force of motion and mobility to create a field of flows can only be created by temporary balances.” (Massey, 2005:78) “If space allows movement, the place is also a stop; every stop in the movement allows the position to become the ground.” (Tuan, 1977). Human movements have a starting point and a destination point, and these points are fixed. These indicate immobile locations. For example, the locations of internet service providers and cell phone technology base stations are fixed. Those who provide services in telecommunications or other sectors that enable people to be physically or virtually mobile can be defined as immobile people.

References

Agutter, A.J. (1995). The linguistic significance of current British slang. Thesis (PhD). Edinburgh University.

Cakir, E. (2018). Multiple Criteria Decision-Making Methods in Electronic Document Management System (EBYS) Software Selection: A Municipality Example. Business, Economics and Management Research Journal, 1(1), p. 15-30.

Çakır, F., & Demir, N. (2014). A Study on Determining the Smartphone Purchase Preferences of University Students. Journal of Dokuz Eylul University Faculty of Economics and Administrative Sciences Vol. 29 No. 1, p. 213-243.

Doğan, R. Ö., Kayıkçıoğlu, T., & et al. (2018). Transfer and Storage of Electronic Health Records Using WCF Web Services. Journal of Suleyman Demirel University Institute of Science and Technology, 22(1), p. 232-236.

Dogrul, G. (2015). Analysis of Traffic Accidents with Association Rules. Gazi Journal of Engineering Sciences, 1(2), p. 265-283.

Erol, H., & Özdemir, A. (2014). Evaluation of Health Reforms and Health Expenditures in Turkey. Journal of Social Security, 4(1), p. 9-34.

Ertemel, A. V., & Çudin, D. (2017). Factors Affecting Smartphone Users' Paid App Purchase Intention. Istanbul Commerce University Journal of Social Sciences, 33(1), p. 153-171.

Göker, H., & Tekedere, H. (2018). Automatic Evaluation of Opinions on Fatih Project with Text Mining Methods. Journal of Information Technologies, 10(3), p. 291-299.

Guler, E. (2015). Gamification in Mobile Healthcare. Journal of Open Education Applications and Research, 1(2), p. 82-101.

Holland, M. (2004). Guide to citing Internet sources (2004) Available from: http://www.bournemouth.ac.uk/library/using/guide_to_citing_internet_sourc.html. Accessed 4 November.

Ilarslan, K. (2014). Journal of Academic Social Studies. Prediction of Stock Prices with the K-Nearest Neighbor Algorithm, 4(30), p. 375-392.

Işık, A. H., & Güler, I. (2010). Mobile Application Study in Telemedicine and Analysis of Mobile Communication Technologies. Gazi University Journal of Information Technologies Volume 3 Issue 1, p. 1-10.

Karataş, G., Akbulut, A., & et al. (2016). Security, Threats and Key Strategies in Mobile Devices. Istanbul Commerce University Journal of Science, p. 55-75.

Khashoggi, T., & Gokcen, H. (2014). Determination of E-Commerce Sites with Text Mining. Journal of Information Technologies, 7(1).

Kılıç, T. (2017). e-Health, Good Practice; Netherlands, p. 203-217.

Luenberger, D.G. (1973). Introduction to Linear and Nonlinear Programming. Addison-Wesley, California.

Ranger, A. T., Usta, E., & et al. (2016). Evaluation of the Graphical User Interface of Android Operating System Software in Terms of Visual Design Principles. Turkish Journal of Social Research, p. 781-791.

Rosen, J.B. (1961). The gradient projection method for nonlinear programming: Part II nonlinear constraints. SIAM J. Appl. Math., 9, 514-532.

Tezcan, C. (2016). An Innovative Perspective on Health: Mobile Health. TUSIAD. doi:TÜSİAD-T/2016-03/575

Tufekci, N., Yorulmaz, R., & Cansever, I. H. (2017). Digital Hospital. Journal of Current Researches on Health Sector, 143-156.

Uğur, N. G., & Turan, A. H. (2015). Acceptance and Use of Mobile Applications by University Students: Example of Sakarya University. Journal of Internet Applications and Management, 2(6), p. 63-79.

Khazar Journal of Science and Technology
Khazar University Press 2022

Volume 6 №2 2022, 15-31
DOI: 10.5782/2520-613.202.6.2.15

Resource Management in Internet of Things by Implementing Kruskal's and Floyd's Algorithms

Azar Alizada

Faculty of Engineering and Applied Sciences,
Khazar University, Azerbaijan
Email: agalizada00@gmail.com

Abstract

The idea of the Internet of Things (IoT) has received a lot of interest recently because it has the potential to transform our physical world into a meaningful information-filled digital cyber environment. The IoT devices are more computationally powerful and smaller in resources in terms of size, number, memory, bandwidth and energy consumption. This thesis will examine resource management in IoT. The Internet of Things (IoT) is a network that is made up of real-world items, including sensors, software, and other technologies integrated into them with the primary intention of connecting and exchanging data with other equipment and systems. Resource management is a critical component of IoT, as it ensures that the various devices and sensors are able to communicate and work together efficiently (I. Rašan, 2021). In this thesis, we will customize and implement two algorithms for resource management in IoT: Kruskal's algorithm and Floyd's algorithm. Kruskal's algorithm is an algorithm that looks for the most optimal solution, which is utilized to find the minimum spanning tree of a graph. The algorithm works by considering all the possible routes of the graph and selecting the one with the lowest weight. The minimum spanning tree is added to these possible routes, after which, the procedure is repeated until all the edges have been considered. Floyd's algorithm is an all-pairs shortest-path algorithm that is used to detect the best path in terms of length between all pairs of nodes in a certain graph. The algorithm works by considering all the nodes of the graph, and afterwards calculating the shortest route between every pair of nodes. We will compare and contrast the two algorithms, and evaluate their effectiveness by combining both to optimize resource management in IoT. As a result, Floyd's and Kruskal's algorithms are customized to detect and eliminate loops within devices to improve resource management in IoT devices. By using Floyd algorithm before Kruskal's implementation it can be determined if there is a cycle in the IoT resources. Floyd algorithm-discovered cycles can be eliminated by applying the Kruskal algorithm. This method allows fewer cables, tracks, and other components to be used in IoT devices. It can improve the efficiency and management of IoT resources by reducing costs, time, effort, and other factors.

Keywords: Sensor, embedded systems, network, cloud computing, technologies

Introduction

One of the most promising technologies of the present decade is the Internet of Things, defined as using intelligent, self-configuring things connected to a network and exchanging data by perceiving, responding to events, and interacting with the environment. Furthermore, if they have sensing technology and onboard computing components, unmanned mobile devices like drones have integrated into the overall "objects" participating in the IoT. These sensing and computational capabilities enhance network embedded intelligence in the recently introduced mobile Internet of Things and allow for the realization of complex tasks in a highly distributed manner, balancing load across infrastructure and significantly increasing communications' energy efficiency (Agrawal, Shashank and Dario Vieira, 2019). Mobile IoT networks are the consequence of their primary ability to move through space and time, which gives IoT monitoring requirements a new level of freedom.

In the article, first of all, the definition of the Internet of Things, its usage areas, its architecture and wireless communication technologies used in this field are explained. Then, with the existing studies, privacy and security analysis in the Internet of Things is done. Attacks against objects; physical, network and software cyber-attacks. Communication protocols used in this area are examined in terms of privacy and security and possible security gaps are determined. It describes the basic precautions that should be taken against security vulnerabilities. In addition, a new method is proposed to ensure the management of the nodes produced by the manufacturers from a single point, as well as to increase the security in the addition of new nodes to the existing networks, which is one of the most critical parts of information security on the Internet of Things, and to contribute to closing the security gaps that may occur. The proposed method is called cloud. For this method, a cloud application is first developed, and the nodes are managed from a single point. However, when a new node will join an existing network, the network coordinator will connect to the cloud system, will obtain the security information of the node to join and will verify whether the node is malicious or not. If the authentication is successful, the node joins the network. In case of failure to verify, the node cannot join the network.

To summarize all points talked about, IoT related products and gadgets produce an enormous amount of data that needs to be stored and easily accessed at any given moment which is made easier and cheaper via the use of the internet to connect to a cloud storage device. These cloud storage devices store the necessary data while communicating with the gadgets. However, cloud services should also provide a robust security system to combat against any cyber attacks which may lead to a handful of unwanted scenarios such as data being stolen and kept as ransom, or the attacker shutting down the service temporarily due to which the gadgets would not function. The cloud storage services must also provide a nimble way of storing and accessing data for efficiency purposes, the faster this process becomes the more accurate a device or a gadget may operate giving it the possibility to also analyze the data at a much more quick and accurate way. In addition, having a central network where all of the gadgets work is going to make it more straightforward to interact and manage the devices and all of its functionality (Evans, Martin, 2017). Moreover it will give the ability to interconnect different sensors and gadgets with each other to make certain actions easier.

Due to the enormous benefits that IoT provides on a daily basis, companies have sunk a massive amount of wealth on further developing the gadgets present in the field. This means that resource management has not received as much attention and is not up to date with the technology currently available. This leads to inefficiency which can cost a lot of money, not to mention the fact that better management could mean that servers for cloud based services could be made more compact while operating at the same if not better efficiency which would further lead into better space management for tech companies. Providing a solution for the above mentioned points would not only improve the quality of the services provided, but also solve most, if not all of the problems currently present in the resource management area of IoT.

Material and Method

Resource management in IoT

In order to enhance resource management in IoT devices, Floyd's and Kruskal's methods can be modified to locate and dispose of the loops within devices. It is possible to identify whether there is a cycle in the IoT resources by applying the

Floyd method prior to the deployment of Kruskal's method. The Kruskal method can be used to get rid of cycles found by the Floyd algorithm. Less wires, tracks, and other parts can be used in IoT devices thanks to this technique. By lowering expenses, spending less time and effort, and other variables, it can increase the management and efficiency of IoT resources. Resources and devices are dispersed throughout the IoT environment at an increasing rate. A vast variety of protocols, data formats, and physical sensing resources need to be managed in order to exploit the full potential of the deployed devices (Weber, Rolf H. and Romana Weber, 2020). This raises the issue of how effectively the resources of the devices can be controlled and provisioned.

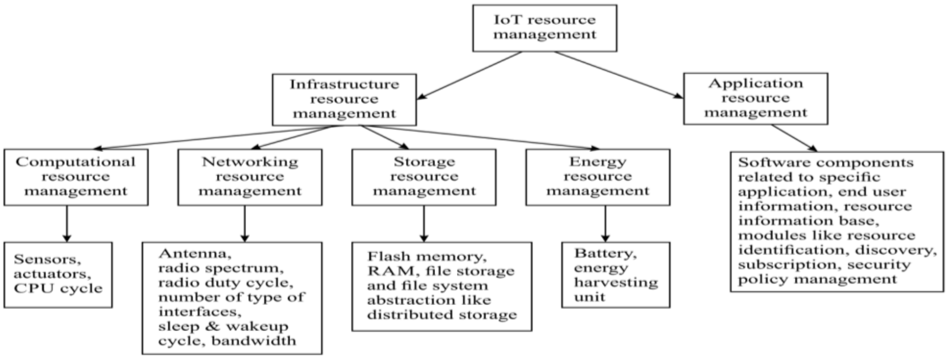

IoT resource management
Infrastructure resource management
Application resource management
Computational resource management
Networking resource management
Storage resource management
Energy resource management
Software components related to specific application, end user information, resource information base, modules like resource identification, discovery, subscription, security policy management
Sensors, actuators, CPU cycle
Antenna, radio spectrum, radio duty cycle, number of type of interfaces, sleep & wakeup cycle, bandwidth
Flash memory, RAM, file storage and file system abstraction like distributed storage
Battery, energy harvesting unit

Figure 1. Types of IoT resources.

Source: Bhajantri, Lokesh B, and Gangadharaiah S. "A Comprehensive Survey on Resource Management in Internet of Things." Journal of Telecommunications and Information Technology, vol. 4, no. 2020, 2020, pp. 27–43., doi:10.26636/jtit.2020.145220.

In order to address issues with IoT resource management, this study takes advantage of an algorithmic method that makes use of the Floyd and Kruskal algorithms. Kruskal's method is used to determine the shortest route between any two locations in a linked weighted graph. In contrast, Floyd's cycle detection method is used to spot pointless cycles. In order to address issues with IoT resource management, this study offers an algorithmic method that makes use of both of the above mentioned algorithms. These two methods have been combined in order to discover and eliminate loops inside certain devices, which will improve resource management in IoT devices. The quantity of cables, traces, and other components required in IoT devices can be decreased using this technique. Applying this strategy will result in savings in time, money, and effort, which could improve resource management in the IoT.

Each of these approaches has its own advantages and disadvantages. In this thesis, we will survey the state of the art in resource management in IoT, with a focus on the challenges that need to be addressed in order to achieve efficient resource utilization. We will discuss in the next chapter some of the perspectives that have been proposed to display these obstacles by implementing Kruskal's and Floyd's algorithms.

Implemented Algorithms

Floyd's Cycle Detection Algorithm

In computer science, a data graph is one that is used to train a model (cycle). If a pointer going on a road at two different speeds can display the same value here, then there is one similar to Floyd's Cycle Detection Algorithm here. It is the algorithm used to determine whether cycles exist in computer science graphs used to model data. This method, also known as Floyd's Cycle Detection Algorithm, identifies whether a loop is present if two speed pointers moving along a path might both show the same value. The Floyd's cycle finding method was developed by computer scientist Robert W. Floyd and introduced in 1967.

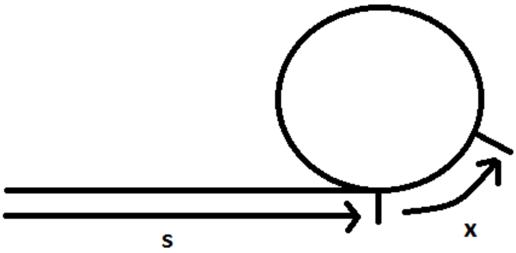


Figure 2. Floyd's Algorithm

Source: https://github.com/epomp447/Floyd-Warshall-Algorithm-Java-

The cycle's beginning is symbolized by the letter S.

S+x stands for, where s is the offset from the cycle's beginning where the two pointers meet.

The quicker pointer may have traveled an additional distance marked as s+x from the offset x location with the cycle after the slow pointer had to travel s+x. It returned to the same state and location with the offset x cycle thanks to the additional s+x distance (https://github.com/epomp447/Floyd-Warshall-Algorithm-Java-).

This simply specifies that the pointer will be from the cycle's start-position, offset 0, if the distance s+x moved from the specified offset accompanies the pointer to the offset x, and if x moved less from that specific position, or so s.

Floyd's cycle detection algorithm pseudo code:

```
function CycleExists(IoTDevices) {

    if(IoTDevices is empty)
        return 'No cycle found.';

    IoTDevicesSlowPointer = IoTDevices;
    IoTDevicesFastPointer = IoTDevices;
    while(IoTDevicesFastPointer != null && IoTDevicesFastPointer.nextIoTDevice != null)
    {
        IoTDevicesSlowPointer = IoTDevicesSlowPointer.nextIoTDevice;
        IoTDevicesFastPointer = IoTDevicesFastPointer.nextIoTDevice. nextIoTDevice;

        if(IoTDevicesSlowPointer == IoTDevicesFastPointer)
            return 'Cycle detected.';
    }
    return 'No cycle found in IoT resources.';
}
```

Kruskal's Algorithm

Kruskal's algorithm is an algorithm that looks for the most optimal solution, which is utilized to find the minimum spanning tree of a graph. The algorithm works by considering all the possible routes of the graph and selecting the one with the lowest weight. The minimum spanning tree is added to these possible routes, after which, the procedure is repeated until all the edges have been considered.

The detailed steps of the Kruskal method are shown below. The subset being grown is represented by the blue edges. The algorithm considers each edge in the order of its weight. Numbers next to the edges denote their weight. The edge that is being taken into consideration is indicated by an arrow at each level of the algorithm. If the edge connects two distinct forest trees,, the two trees are combined and the edge is added to the forest. If the edge creates a cycle, it is disregarded. As can be seen from the image, in step four (d), the edge with four weights is disregarded because it produces a cycle.

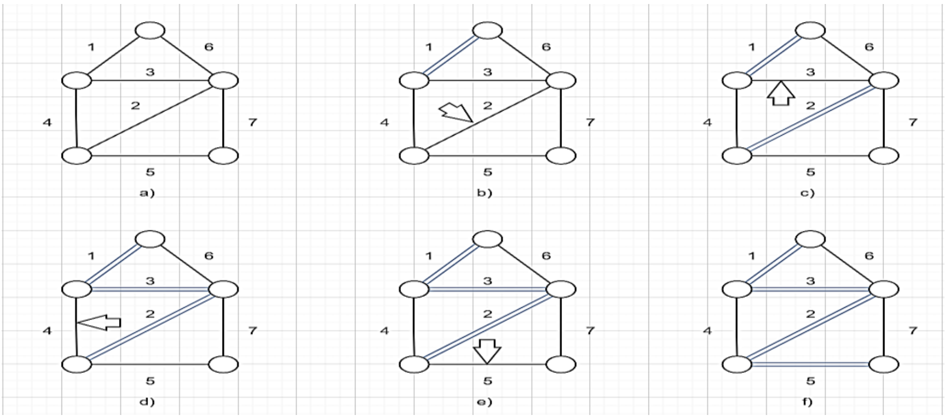


Figure 3. Occurrence of a cycle.

Finally, Floyd algorithm-discovered cycles can be eliminated by applying the Kruskal algorithm. Thus, the vertices' best and most efficient connections are discovered. This can help reduce the number of wires, traces, and other components inside IoT devices, which may improve Cost, time, and effort resource management in the Internet of Things.

Kruskal algorithm pseudo code:

```
     marked = true
Integer color[1000]
Input: Number of IoT Devices(total)
   FOR total number of IoT Resources(N):
      insert(locationX)
      insert(locationY)
      insert(Weight)
      graph[locationX][locationY] = marked
      weightGraph[locationX][locationY] = Weight
Output:
   FOR each node:
      DFS(currentIoTDevice)
```

```
    MSA(currentIoTDevice) //Regraphs the inputted graph so that there are no cycles
    MSA(Integer currentIoTDevice){
  FOR N number of IoT Resources:
    IF graph[currentIoTDevice][Nth IoT Resource] is equal to true & used[Nth IoT Resource] is not equal to true:
      MSAgraph[currentIoTDevice][Nth IoT Resource] = 1;
      fWeight(currentIoTDevice][Nth IoT Resource] = weightGraph[currentIoTDevice][Nth IoT Resource];
      MSA(i);
  }
  DFS(Integer currentIoTDevice){
    FOR N number of IoT Resources,:
      IF graph[currentIoTDevice][Nth IoT Resource] is an edge connecting the currentIoTDevice to the IoT Resource i:
        IF color[Nth IoT Resource] is not left completely:
          IF color[Nth IoT Resource] has been left but not completely:
            cycle is true
          DFS(Nth IoT Resource);
    color[currentIoTDevice] is completely left
}
print graph[N][N]
print MSAgraph[N][N]
print fWeight[N][N]
```

Customization of Kruskal's and Floyd's Algorithms

This study describes an algorithmic solution that employs the Floyd and Kruskal algorithms to address IoT resource management concerns. To enhance resource management in IoT devices, these two techniques have been integrated and used to find and remove loops inside specific devices. The only difference between the Kruskal's algorithm implementations used in this experiment is the method used to sort the graph's edges. As a result, we examine the time required by each sorting method and use it to determine which factor dominates the algorithm's total running time. By using Floyd algorithm before Kruskal's implementation it can be determined if there is a cycle in the given graph.

- Two pointers: fastPointer and slowPointer are initialized for looping through the nodes.

- The slowPointer advances one position, but the fastPointer advances two.

- The fastPointer moves twice as quickly as slowPointer. It travels twice as far as the slowPointer does.

- A loop is present if both pointers cross at some point. If the fastPointer reaches the end position then no loop exists.

- If a loop is found then by using Kruskal's algorithm the edges that create the loop can be removed.

Floyd's and Kruskal's methods can be modified to locate and dispose of the loops within devices. It is possible to identify whether there is a cycle in the IoT resources by applying the Floyd method prior to the deployment of Kruskal's method. The Kruskal method can be used to get rid of cycles found by the Floyd algorithm. In order to address issues with IoT resource management, this study takes advantage of an algorithmic method that makes use of the Floyd and Kruskal algorithms. Kruskal's method is used to determine the shortest route between any two locations in a linked weighted graph. In contrast, Floyd's cycle detection method is used to spot pointless cycles. . Less wires, tracks, and other parts can be used in IoT devices thanks to this technique. By lowering expenses, spending less time and effort, and other variables, it can increase the management and efficiency of IoT resources. Resources and devices are dispersed throughout the IoT environment at an increasing rate (Atzori, Luigi, Antonio Iera, and Giacomo Morabito, 2020). In order to address issues with IoT resource management, this study offers an algorithmic method that makes use of both of the above mentioned algorithms. These two methods have been combined in order to discover and eliminate loops inside certain devices, which will improve resource management in IoT devices. The quantity of cables, traces, and other components required in IoT devices can be decreased using this technique. Applying this strategy will result in savings in time, money, and effort, which could improve resource management in the IoT.

Flowchart for customization of Kruskal's and Floyd's Algorithms is given below:

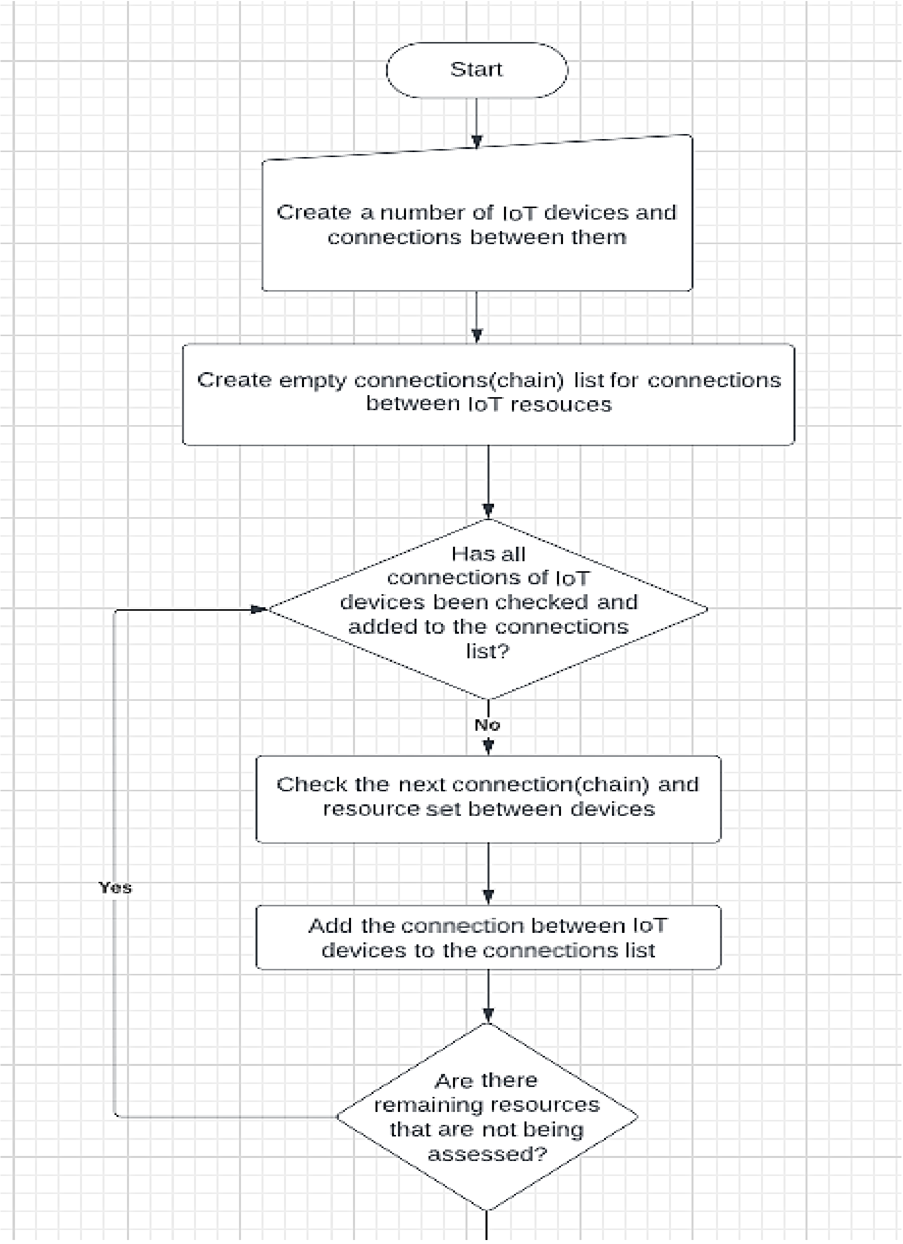

Start
Create a number of IoT devices and connections between them
Create empty connections(chain) list for connections between IoT resouces
Has all connections of IoT devices been checked and added to the connections list?
No
Check the next connection(chain) and resource set between devices
Add the connection between IoT devices to the connections list
Are there remaining resources that are not being assessed?
Yes

Diagram continues with the following page

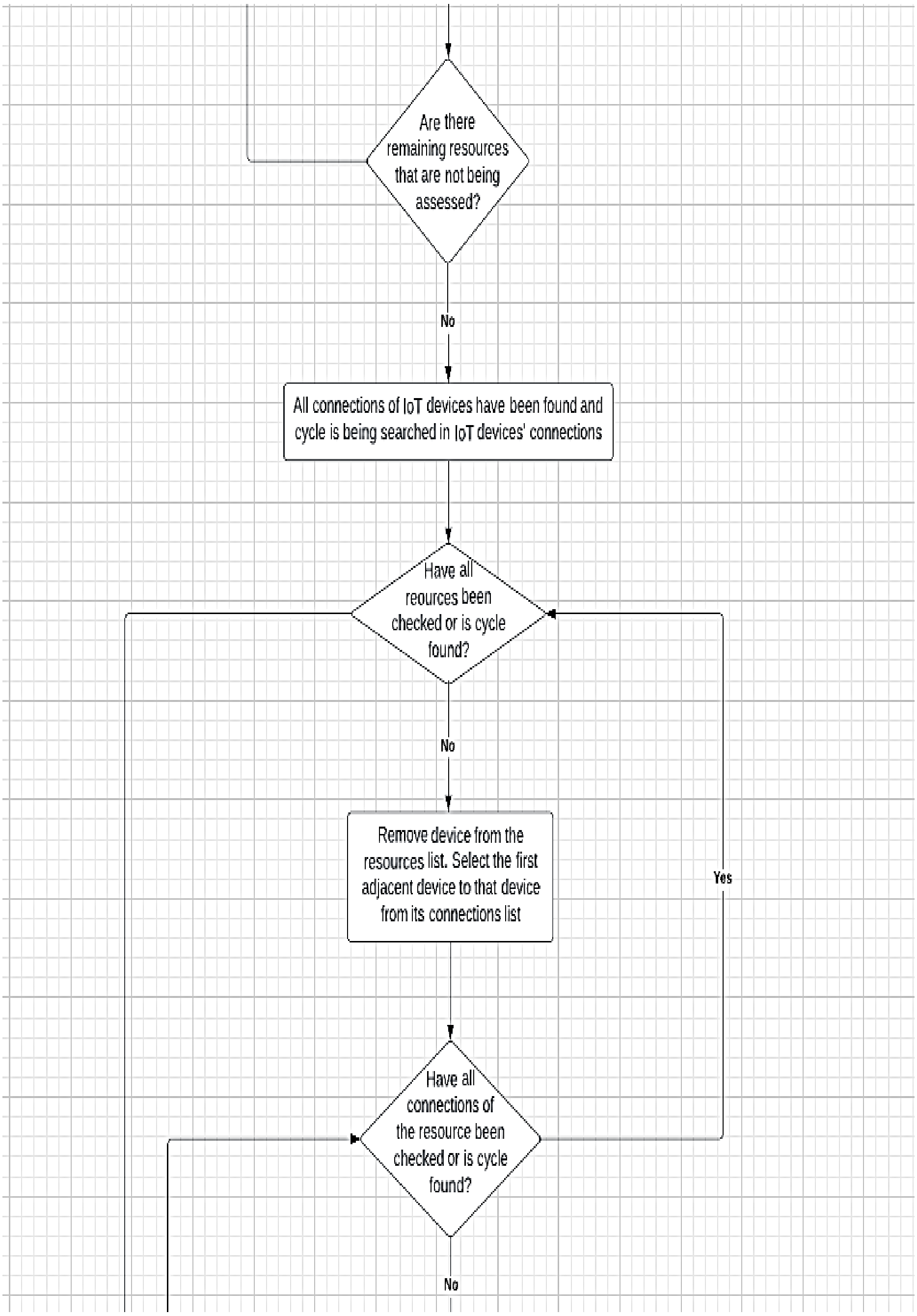
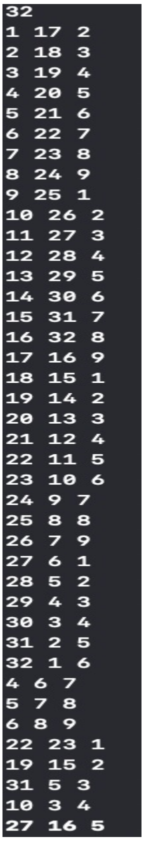

Are there remaining resources that are not being assessed?
No
All connections of IoT devices have been found and cycle is being searched in IoT devices' connections
Have all reources been checked or is cycle found?
No
Remove device from the resources list. Select the first adjacent device to that device from its connections list
Yes
Have all connections of the resource been checked or is cycle found?
No

Figure 4. Flowchart for cycle detection and deletion by implementing Kruskal algorithm after finding cycle by Floyd algorithm

Simulation of the Customized Algorithms with Scenario

This image shows an example of an input made by a user. There are 32 nodes in this example, the first digit inputted on each line shows the node from which the connection between the nodes is made(edge(s)), and the second value on each line is the node that the edge is directed towards/connected to. The third numeral represents the weights of each edge from one vertice to another. Hence combinations of these edges between these vertices create a directed graph, which in turn can be used to derive different sorts of information or outcomes.



```
32
1 17 2
2 18 3
3 19 4
4 20 5
5 21 6
6 22 7
7 23 8
8 24 9
9 25 1
10 26 2
11 27 3
12 28 4
13 29 5
14 30 6
15 31 7
16 32 8
17 16 9
18 15 1
19 14 2
20 13 3
21 12 4
22 11 5
23 10 6
24 9 7
25 8 8
26 7 9
27 6 1
28 5 2
29 4 3
30 3 4
31 2 5
32 1 6
4 6 7
5 7 8
6 8 9
22 23 1
19 15 2
31 5 3
10 3 4
27 16 5
```

Figure 5. Inputted Graph of Example as a list

The 4 images shown on the next pages are the inputs entered by the user in a weighted graph form, multidimensional array form, the output of the input without cycles and comparison table of the task execution. The first multidimensional graph is the graph that had been input in picture 5. The second multidimensional figure is the output of the input graph without the edges that cause cycles.

The image below is the input entered by the user converted into a directed graph which is much easier for the user to understand. In this specific example, there are 8 cycles and 32 nodes.

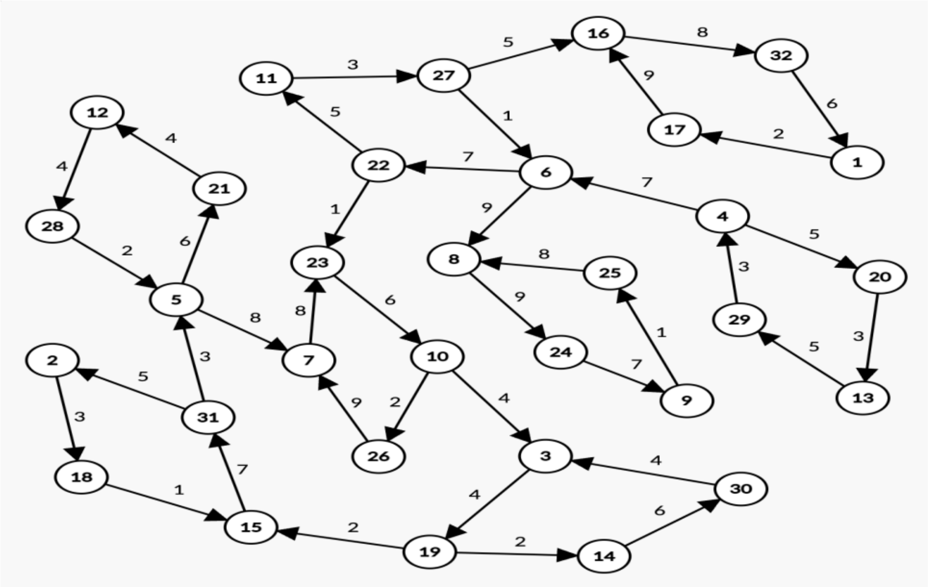
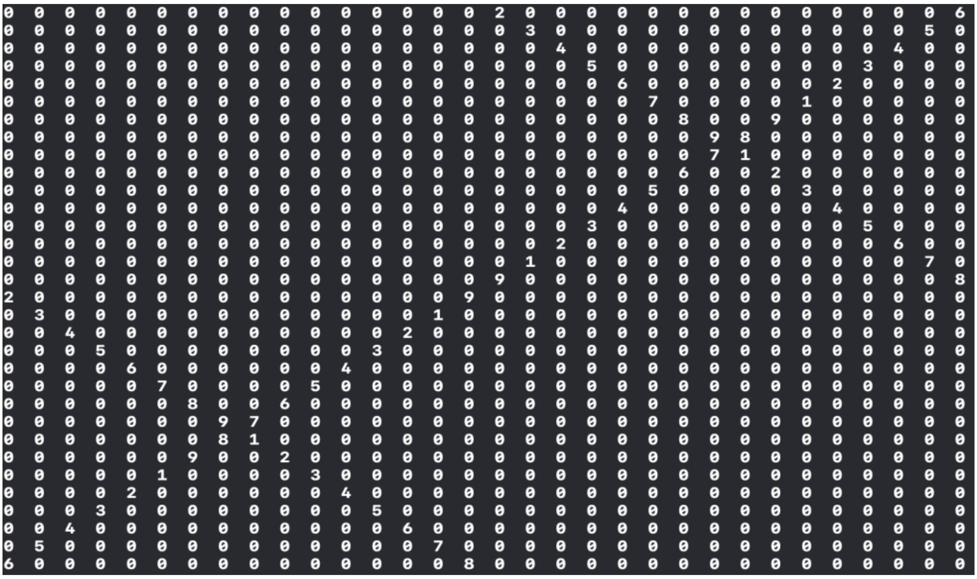
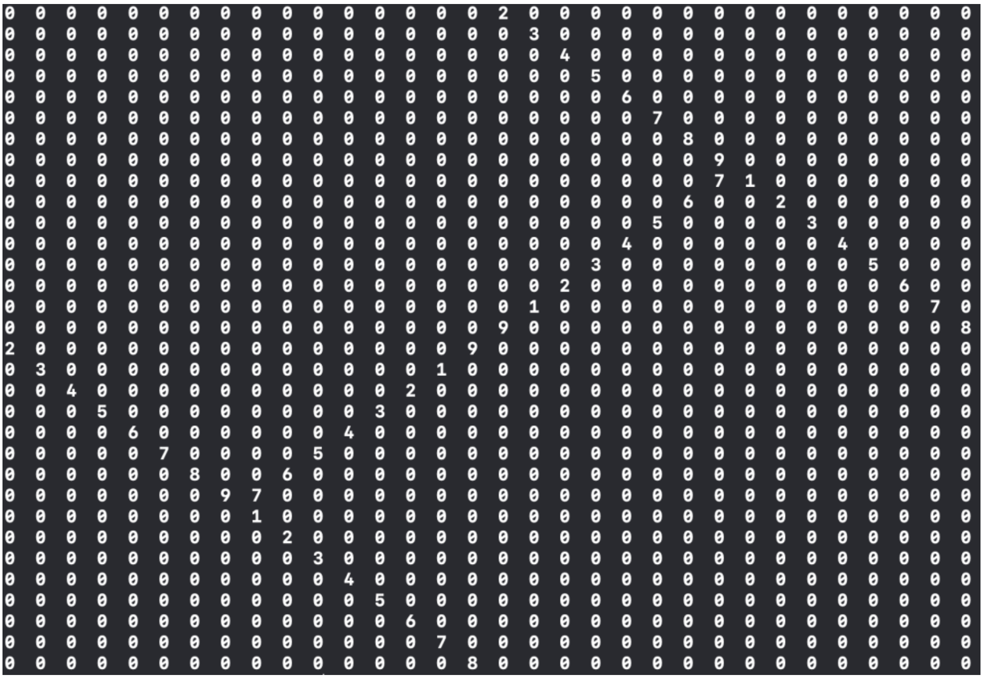


Figure 6. Directed graph



```
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   6
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   3   0   0   0   0   0   0   0   0   0   0   0   0   5   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   4   0   0   0   0   0   0   0   0   0   0   4   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   5   0   0   0   0   0   0   0   0   3   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   6   0   0   0   0   0   0   2   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   7   0   0   0   0   1   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   8   0   0   9   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   9   8   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   7   1   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   6   0   0   2   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   5   0   0   0   0   3   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   4   0   0   0   0   0   0   4   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   3   0   0   0   0   0   0   0   0   5   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   2   0   0   0   0   0   0   0   0   0   0   6   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   7   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   9   0   0   0   0   0   0   0   0   0   0   0   0   0   0   8
2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   9   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   3   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   4   0   0   0   0   0   0   0   0   0   0   2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   5   0   0   0   0   0   0   0   0   3   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   6   0   0   0   0   0   0   4   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   7   0   0   0   0   5   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   8   0   0   6   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   9   7   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   8   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   9   0   0   2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   1   0   0   0   0   3   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   2   0   0   0   0   0   0   4   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   3   0   0   0   0   0   0   0   0   5   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   4   0   0   0   0   0   0   0   0   0   0   6   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   5   0   0   0   0   0   0   0   0   0   0   0   0   7   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
6   0   0   0   0   0   0   0   0   0   0   0   0   0   0   8   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

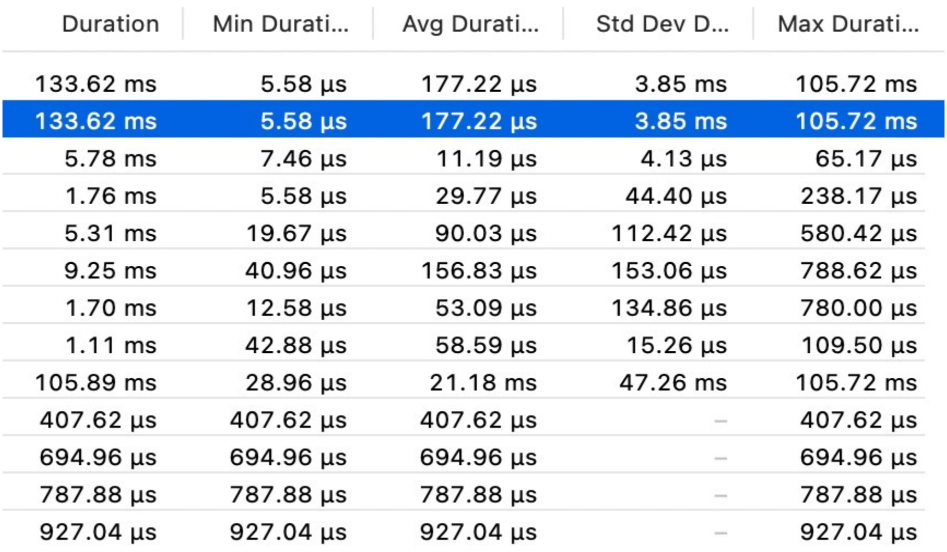
Figure 7. Multidimensional Array Form of Input List



```
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   3   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   4   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   5   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   6   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   7   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   8   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   9   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   7   1   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   6   0   0   2   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   5   0   0   0   0   3   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   4   0   0   0   0   0   0   4   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   3   0   0   0   0   0   0   0   0   5   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   2   0   0   0   0   0   0   0   0   0   0   6   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   7   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   9   0   0   0   0   0   0   0   0   0   0   0   0   0   0   8
2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   9   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   3   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   4   0   0   0   0   0   0   0   0   0   0   2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   5   0   0   0   0   0   0   0   0   3   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   6   0   0   0   0   0   0   4   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   7   0   0   0   0   5   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   8   0   0   6   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   9   7   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   3   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   4   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   5   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   6   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   7   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   8   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

Figure 8. Outputted Graph of the Given Example without the Cycles

Table 1. statistical parameters like duration, etc.



Duration	Min Durati...	Avg Durati...	Std Dev D...	Max Durati...
133.62 ms	5.58 µs	177.22 µs	3.85 ms	105.72 ms
133.62 ms	5.58 µs	177.22 µs	3.85 ms	105.72 ms
5.78 ms	7.46 µs	11.19 µs	4.13 µs	65.17 µs
1.76 ms	5.58 µs	29.77 µs	44.40 µs	238.17 µs
5.31 ms	19.67 µs	90.03 µs	112.42 µs	580.42 µs
9.25 ms	40.96 µs	156.83 µs	153.06 µs	788.62 µs
1.70 ms	12.58 µs	53.09 µs	134.86 µs	780.00 µs
1.11 ms	42.88 µs	58.59 µs	15.26 µs	109.50 µs
105.89 ms	28.96 µs	21.18 ms	47.26 ms	105.72 ms
407.62 µs	407.62 µs	407.62 µs	–	407.62 µs
694.96 µs	694.96 µs	694.96 µs	–	694.96 µs
787.88 µs	787.88 µs	787.88 µs	–	787.88 µs
927.04 µs	927.04 µs	927.04 µs	–	927.04 µs

The first one from the left is the duration of the task, and the second from the left shows the minimum time it should take to solve a task like this. Avg duration shows how much time it would usually take to solve a task like this, Std Dev is the library used, and Max duration means the maximum amount of time needed to run the program. In all test cases, the results came back in less than 150 ms

Results and Discussion

In result, as it seems in the above example all user inputs have been tested and calculation time of all inputs has been shown in the tables for given scenario. For better understanding the inputs and test results have been converted into graphs. By using Kruskal's and Floyd's algorithms the cycles inside graphs have been found and depicted in the images, output is obtained without cycles which were given in input. As a result, utilizing this method can help decrease the number repetitions/cycles in terms of wires, traces, and other components inside IoT devices, which may boost IoT resource management in terms of cost, time, and effort.

In the IoT, there are progressively more resources and devices being distributed into the environment. In order to take advantage of the deployed devices, a wide range of protocols, data formats, and physical sensing resources must be controlled (Wilkinson, Glenn, 2019). This study introduced an algorithmic solution that employs the Floyd and Kruskal algorithms to address IoT resource management concerns. The shortest path in a connected weighted graph between any two points is found using Kruskal's approach. On the other side, Floyd's cycle detection technique is employed to identify unnecessary cycles. This study describes an algorithmic solution that employs the Floyd and Kruskal algorithms to address IoT resource management concerns. To enhance resource management in IoT devices, these two techniques have been integrated and used to find and remove loops inside specific devices. This method allows for a reduction in the number of wires, traces, and other components used in IoT devices.

Graph theory, a branch of discrete mathematics, introduces the concept of Kruskal's algorithm (Lokesh B. Bhajantri and Gangadharaiah, 2022). This technique finds the shortest path between any two points in a linked weighted graph. This method converts a given graph into a forest by considering each node as a separate tree. Floyd's cycle detection algorithm, often known as the Hare-Tortoise algorithm, is a pointer algorithm that traverses the sequence using just two pointers and two distinct speeds. The loop in a linked list can be found using this approach. It employs two pointers, one of which moves twice as quickly as the other. In this research these two algorithms have been combined and employed to detect and eliminate loops inside given devices in order to improve resource management in IoT devices. By using this technique, the quantity of wires, traces, and other components inside IoT devices can be reduced.

References

Abou-Zahra, Shadi, Judy Brewer, & Michael Cooper (2019). Web Standards to Enable an Accessible and Inclusive Internet of Things (IoT). The 14th Web for All Conference on The Future of Accessible Work - W4A '17 Proceedings, pages 1-4ACM Press, New York, New York, USA.

Agrawal, Shashank & Dario Vieira (2019). A Survey on Internet of Things, Abakós 1. (2).DOI: 10.5752/P.2316-9451.2013v1n2p7

Al-Fuqaha, Ala, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, & Moussa Ayyash (2019). Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. IEEE Communications Surveys & Tutorials, vol. 17(4), pp. 2347–2376.

Al-Sarawi, Shadi, Mohammed Anbar, Kamal Alieyan, & Mahmood Alzubaidi (2017). Internet of Things (IoT) Communication Protocols: Review, 8th International Conference on Information Technology, 2018. (ICIT). IEEE. pp. 685–90

Atzori, Luigi, Antonio Iera, & Giacomo Morabito (2020) in 2017 8th International Conference on Information Technology, Internet of Things (IoT) Communication Protocols: Review (ICIT). IEEE. Pp. 685–90

Bravo, Jose, Ramon Hervas, Salvador W. Nava, Gabriel Chavira, & Carlos Sanchez. n.d. Towards Natural Interaction by Enabling Technologies: A Near Field Communication Approach. in Constructing Ambient Intelligence. Berlin, Heidelberg: Springer Berlin Heidelberg. Pp. 338–51

Centenaro, Marco, Lorenzo Vangelista, Andrea Zanella, & Michele Zorzi (2020). Long-Range Communications in Unlicensed Bands: The Rising Stars in the IoT and Smart City Scenarios. IEEE Wireless Communications 23(5):60–67.

Doppler, Klaus, Mika Rinne, Carl Wijting, Cassio Ribeiro, & Klaus Hugl (2019). Device-to-Device Communication as an Underlay to LTE-Advanced Networks. IEEE Communications Magazine 47(12):42–49.

Duan, Kang-Kang & Shuang-Yin Cao (2020). Emerging RFID Technology in Structural Engineering – A Review. Structures 28:2404–14.

Evans, Martin (2017) Arduino in Action.

Geng Wu, S. Talwar, K. Johnsson, N. Himayat, & K. D. Johnson (2017). M2M: From Mobile to Embedded Internet. IEEE Communications Magazine 49(4):36–43.

González, Gustavo Ramírez, Mario Muñoz Organero, & Carlos Delgado Kloos (2008). Early Infrastructure of an Internet of Things in Spaces for Learning. Pp. 381–83

Gubbi, Jayavardhana, Rajkumar Buyya, Slaven Marusic, & Marimuthu Palaniswami (2018). Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions. Future Generation Computer Systems 29(7):1645–60.

He, Debiao, Neeraj Kumar, & Jong-Hyouk Lee (2019). Secure Pseudonym-Based Near Field Communication Protocol for the Consumer Internet of Things. IEEE Transactions on Consumer Electronics 61(1):56–62.

Hodges, Steve, Stuart Taylor, Nicolas Villar, James Scott, Dominik Bial, & Patrick Tobias Fischer (2019). Prototyping Connected Devices for the Internet of Things. Computer 46(2):26–34.

Lokesh B. Bhajantri & Gangadharaiah (2022). A Comprehensive Survey on Resource Management in the Internet of Things. Journal of telecommunications and information technology 4/2022

Rašan, I. (2021). Digitalization in the role of humanization or dehumanization of modern society, 44th International Convention on Information, Communication and Electronic Technology (MIPRO), pp. 1460–1465.

Villa-Henriksen, Andrés, Gareth T. C. Edwards, Liisa A. Pesonen, Ole Green, & Claus Aage Grøn Sørensen (2020). Implementation, Applications, Challenges, and Potential of the Internet of Things in Arable Farming. in Biosystems Engineering.191:60–84

Weber, Rolf H. & Romana Weber (2020). Internet of Things. Berlin, Heidelberg: Springer Berlin Heidelberg.

Welbourne, Evan, Leilani Battle, Garret Cole, Kayla Gould, Kyle Rector, Samuel Raymer, Magdalena Balazinska, & Gaetano Borriello (2009). RFID Ecosystem Experience: Building the Internet of Things Using RFID. IEEE Internet Computing, 13(3).48–55

Whitmore, Andrew, Anurag Agarwal, & Li Da Xu (2019). The Internet of Things—A Survey of Topics and Trends. Information Systems Frontiers 17(2):261–74

Wilkinson, Glenn (2019). Digital Terrestrial Tracking: The Future of Surveillance. DefCon22

https://github.com/epomp447/Floyd-Warshall-Algorithm-Java-

Khazar Journal of Science and Technology
Khazar University Press 2022

Volume 6 №2 2022, 31-55
DOI: 10.5782/2520-613.202.6.2.31

Recognition of Handwritten Azerbaijani Letters using Convolutional Neural Networks

Heydar Rzayev, Amir Masoud Rahmani,
Behnam Kiani Kalejahi

Faculty of Engineering and Applied Sciences,
Khazar University, Azerbaijan
Corresponding author: heydar.rzayev@khazar.org

Abstract

Technology advancements have made it possible to fill out documents such as petitions and forms electronically. However, in some circumstances, hard copies of documents that are difficult to share, store, and save due to their rigid dimensions are still used to preserve documents in the conventional manner. It is crucial to convert these written documents into digital media because of this. From this view point, this goal of this study is to investigate various methods for the digitalization of handwritten documents. In this study, image processing methods were used to pre-process the documents that were converted to image format. These operations include splitting the image format of the document into the lines, separating them into words and characters, and then classifying the characters. Convolutional Neural Networks, which is used for image recognition, is one of the deep learning techniques used in classification. The Extended MNIST dataset and the symbol dataset created from the pre-existing documents are used to train the model. The success rate of the generated dataset was 88.72 percent.

Keywords: Image processing; character recognition; handwriting recognition; deep learning; convolutional neural network.

Introduction

In the modern era, with the advancement of technology, more and more documents are being filled in the computer environment. Nevertheless, there are many documents that are still on the paper. Transferring these to the digital environment is done with the assignment of one or more people. To eliminate manual work, handwriting recognition systems have been developed to automatically transfer such documents to the digital environment. Handwriting recognition is the identification of letters, numbers and symbols written on media such as paper, tablet and phone by computer systems and making them meaningful.

Handwritten characters are more difficult to distinguish than optical characters. Therefore, the accuracy rates in current studies are below 90%. There are many variations that affect the recognition of handwriting (e.g., natural variations, word size, defects in the image). These variations cause some difficulties in the recognition of words or characters. Despite these difficulties, many studies have been carried out and continue to be conducted in this area.

Fortunately, in some languages such as Chinese, Arabic and Japanese, the handwriting recognition systems are quite good. However, on the contrary, there are not 2 enough studies in the field of Azerbaijani handwriting recognition. The level of success in the studies carried out is quite low.

There are two methods in handwriting recognition systems: online and offline. In the online method, the two-dimensional coordinates of the writer's strokes are used while writing on tablets or special screens. The second method is called the offline method, where the text is used as a picture. This is a well-known problem, and many methods can be used to recognize the image.

In this study, offline handwritten texts were translated into picture format, and the recognition process was carried out. Since Azerbaijani has an infinite dimensional vocabulary, character-based recognition was used in the study. Instead of a separate algorithm, both the feature extraction and recognition processes of the characters were done with a single method, which is the Convolutional Neural Network (CNN) method.

The first step is to create a model for recognition and then train this model. A model has been created, and training has been made for the recognition of the characters. The EMNIST (NIST, 2017, April) data set was used in the training of the model which is obtained from the Kaggle site. The EMNIST dataset contains a large number of handwritten letters and numbers. All the characters in the EMNIST dataset were trained in 2D with the CNN method. The data is given as a direct input to the CNN model without any feature extraction on the characters beforehand. The distinctive features of the characters were made with the CNN method during the training of the model, and the recognition was carried out in this way.

The second stage is to obtain the characters from the documents and perform the recognition process on these characters. The data used in the recognition process was created from documents written by many people around us. Then, a preprocessing step was done on these documents, and their characters were separated. And then a recognition process was performed using the model trained with the CNN method.

Handwriting Recognition

Handwriting recognition is the conversion of handwritten text on other devices, such as paper documents, photographs, and touch screens, into an understandable digital format. It is examined under two main headings as online and offline handwriting recognition. The online handwriting recognition is the process of converting handwritten text into a digital format while the user is writing. The offline handwriting recognition is the process of converting handwritten text into digital format after the user has finished writing.

Online Handwriting Recognition

Online handwriting recognition (OHWR) is a system for the automatic processing of writing using a hand or pen by tablets or special screens. In OHWR, the size of the vocabulary has a direct impact on the performance of the system. As the number of words increases, the recognition speed and performance of the system decrease (Bilgin Taşdemir, 2018).

Many methods and approaches have been found about this system since the beginning of the 1960s. These methods can be studied in three classes: understanding of text written with a first-class screen and stylus. Verification of the person's identity from signatures in second-class handwriting. Third grade is to use the neuromotor features of handwriting to design systems for education and rehabilitation (Plamondon & Srihari, 2000).

In OHWR systems segmentation process is used to get the basic structure of character recognition. Two methods are used. The first method is line detection (Hennig et al., 1996). Thanks to this method, word segmentation and adjustment of parts that do not contain text are done. The second method focuses on splitting the input into individual characters or even sub-characters. The biggest problem with splitting words is determining the beginning and end of individual characters. The most common approaches used for this are unsupervised learning (Hébert et al., 1999), (Plamondon & Srihari, 2000) and data-based knowledge-based methods (Hennig et al., 1997).

Offline Handwriting Recognition

Offline handwriting recognition is a system that detects characters on a document. When using this system, first the document is digitized by transferring it to the computer environment. Then, the document is divided into paragraphs, sentences and words, respectively (Şekerci, 2007)0. Offline handwriting recognition is divided into a holistic or analytical strategy:

- Holistic Strategy: It is processed directly on the word without slicing. The longer the word, the lower the recognition rate.

- Analytical Strategy: It is processed on the document after slicing it to paragraphs, sentences, words, or characters. This strategy in character recognition follows a certain sequence of operations. These are (Srihari et al., 2001):

o Pre-processing

o Segmentation

o Feature extraction

o Classification

o Post-processing

Pre-processing

The pre-processing stage consists of the operations such as normalization, noise removal, and reference line determination. Noise-removal can be achieved by techniques such as filters, noise models, and morphological operations. Normalization is the process of standardizing characters by removing differences in typefaces. Detecting and using reference line is also important because it prevents some characters from mixing with each other. For example, while the characters 'g' and '9' may be confused with each other, when their position relative to the reference line is considered, the confusion of these two characters with each other will be eliminated (Steinherz et al., 1999), (Şekerci, 2007).

Segmentation

The segmentation phase is the process of separating words into letters, characters and numbers. One popular method is to split the document first into lines, then words and lastly into characters. Another method used is to first divide the document into as small pieces as possible and then combine them using the Hidden Markov Models method (Hidden Markov Model-HMM) (Arica & Yarman-Vural, 2001), (Şekerci, 2007).

Feature extraction

The feature extraction is an important part of the handwriting recognition process. It is used to determine which pixels are relevant and which aren't. Different methods can be used at this stage. These methods are: applying transformations such as Fourier and Wavelet, histogram or projection-based methods, or defining letters as a set of simple shapes such as lines, 8 curves, and corners (Arica & Yarman-Vural, 2001).

Classification

After the feature extraction, we classify characters or numbers with the data we have. Many algorithms have been used for the classification process:

- K-NN
- libSVM
- Artificial Neural Networks (ANN)
- Convolutional Neural Networks (CNN)
- Bayesian Classifier

K-NN, LibSVM, and Artificial Neural Network algorithms are algorithms used for character recognition treated as an analytical strategy. CNN is the algorithm used both as a holistic strategy and for character recognition.

K-NN

K-Nearest Neighbours (K-NN) algorithm is a machine learning algorithm used in clustering problems. The performance of the algorithm is affected by the number of k clusters, the values of the initially selected cluster centres and the similarity measurement criteria (Zouhal & Denoeux, 1998). The choice of the value of k is important in the K-NN algorithm. The K value is usually chosen as 1. Having a K value of 1 may cause problems in recognizing similar characters. An example of this is the recognition of the letters 'e' and 'c'. Therefore, the K value is chosen as 3 for the clustering of lowercase letters and as 5 for the clustering of uppercase letters (Bektaş et al., 2016).

libSVM

It is a library developed for Support Vector Machines (SVM). Unlike SVM, which is used to estimate two-class data, it allows the classification of multi-class data (Bektaş et al., 2016).

ANN

ANNs mimic the structure of a living organism's nervous system. The process involves receiving numerous pieces of information, analysing them individually and gauging their relative importance in relation to others—then making decisions based on these factors by choosing an appropriate output action. It is used for solving complex problems that cannot be solved by conventional, linear approaches alone. A simple ANN model is shown in Figure 1.

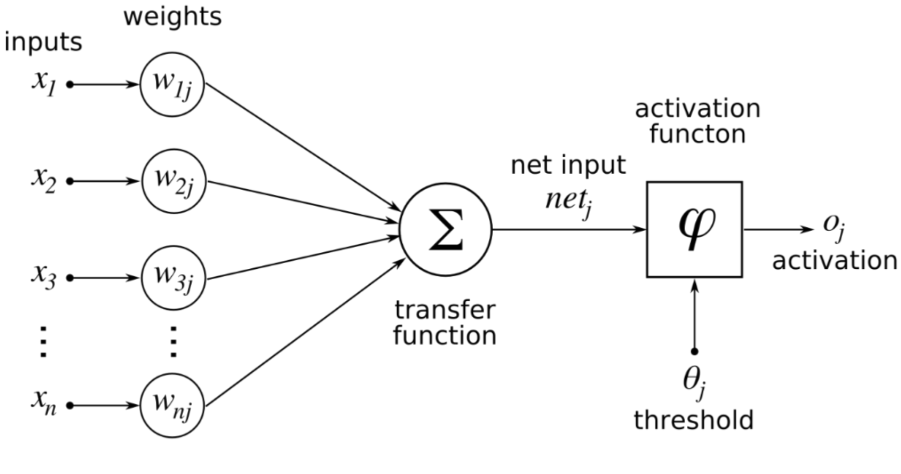


Figure 1. ANN

CNN

Convolution Neural Network (CNN) is a deep learning architecture. The algorithm takes place in two stages. In the first stage, the image, which is given as the input value to the algorithm, is processed in the convolution layer and the important features of the image are extracted in this layer and thrown into a matrix. In the second stage, classification is performed on the matrix obtained in the convolution layer by using the multiple artificial neural network algorithm. The character recognition steps of the CNN algorithm are shown in Figure 2.

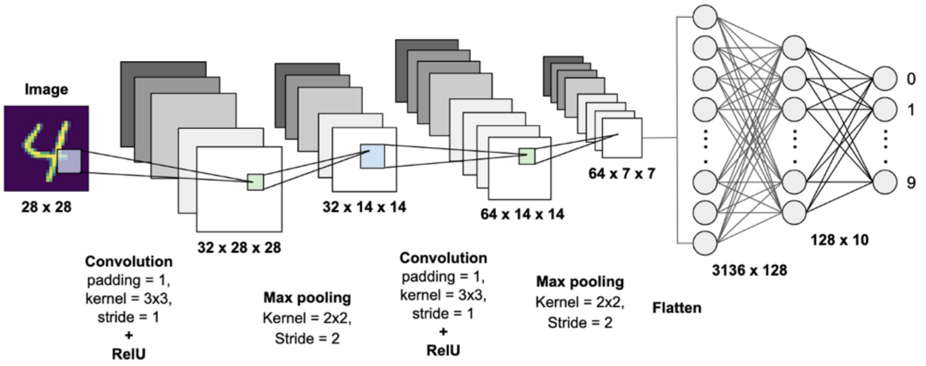


Figure 2. Character Recognition with CNN

Bayesian Classifier

Naive Bayes classifier is a classification method developed by taking Bayes probability theorem as an example. It is an approach that calculates the probability that a new data belongs to which of the existing classes, using the entire dataset for a new classification.

The probabilities of a new data sample X with no class value are calculated using Eq. (1) for each class using the Bayesian classifier. Considering that there are N classes, S = (s_1, s_2, …, s_n). For each data in the dataset, data m-dimensional feature vectors; It is shown as X = (x_1, x_2, …, x_m).

$$P\left(\frac{s_i}{X}\right) = \frac{P\left(\frac{X}{s_i}\right) * P(s_i)}{P(X)} \qquad (1)$$

The class that X data belongs to is found by using Eq. (2).

$$X \in s_i \rightarrow P\left(\frac{S_i}{X}\right) = maxP\left(\frac{S_i}{X}\right); i \in |S| \qquad (2)$$

Post-processing

Post-processing is used to increase the accuracy of classification. There are many methods for verification. One of them is the verification of letters using databases containing combinations of 2 or 3 letters. Another method is to increase the accuracy of the sentence by using a high-level formal grammatical model. The main idea behind the post-processing is to improve the accuracy of the result by using additional information from other sources such as additional datasets or knowledge about the problem domain.

Material and Method

Technical Information

The hardware information of the computer used for the training and testing of the model developed in the study is given in Table 1.

Table 1. Computer hardware information

Processor:	Intel Core i5-7360U, 2.30 GHz, 4 MB
Graphics card:	Intel Iris Plus Graphics 640
Memory:	8 GB
Persistent memory:	1 TB
Operating system:	Ubuntu 14 LTS

Python 2.7 was used as the programming language in the software part. OpenCv Library was used for image processing, Tensorflow and Keras libraries were used for artificial intelligence.

Dataset

Training dataset

The data set used in the training of the model was taken from the site called Kaggle. The EMNIST dataset contains handwritten letters and numbers consisting of 28 x 28 pixel images. There are six different panels provided in this dataset. A brief summary of the dataset (NIST, 2017, April) is as follows:

- EMNIST ByClass: 814,255 characters. 62 class value.
- EMNIST ByMerge: 814,255 characters. 47 class value.
- EMNIST Balanced: 131,600 characters. 47 class value.
- EMNIST Letters: 145,600 characters. 26 class value.
- EMNIST Digits: 280,000 characters. 10 class value.
- EMNIST MNIST: 70,000 characters. 10 class value.

Looking at Table 2, a data summary of six different partitions of the EMNIST dataset is given. Looking at the chart, how much data is reserved for testing and training, the total number of data and its availability for validation are given.

Table 2. EMNIST dataset and organization

Name	Category	Training	Test	Verification	Total
BY_CLASS	62	697,932	116,323	No	814,255
BY_MERGE	47	697,932	116,323	No	814,255
BALANCED	47	112,800	18,800	Yes	131,600
DIGITS	10	240,000	40,000	Yes	280,00
LETTERS	37	88,800	14,800	Yes	103,600
MNIST	10	60,000	10,000	Yes	70,000

Test dataset

This data set was created with documents written by many people around us. It is a data set created for recognizing handwritten characters on paper. The dataset contains 10 documents. These documents are handwritten by different people and transferred to the computer environment by scanner or camera.

Handwriting Recognition with Developed Deep Learning Algorithm

The processes performed in this study are divided into the following steps. The applications in each step are developed using the Python language. The steps of the developed application are shown in Figure 3.

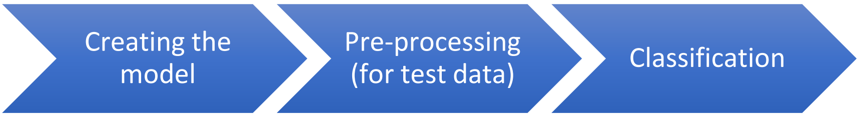


Figure 3. Steps of the developed application

Creating the model

In the study, the deep learning method was used to classify the characters. A model was created using the Convolutional Neural Network architecture of deep learning and the classification process was made using this model.

Deep Learning

Deep learning performs feature extraction with the help of multiple nonlinear layers. All consecutive layers take the output of the previous layer as input (Deng & Yu, 2014), (Şeker et al., 2017). Deep learning is basically a method based on learning based on the values of the pixels that best represent the data. In other words, it works by extracting the most important traits of the character in the image and classifies it by using these traits. This method is called the feature extraction method. In deep learning, different algorithms are used to extract the best pixels on the image instead of manually extracting data.

Looking at the history of deep learning, the first algorithm for supervised deep-feed multilayer perceptrons was introduced by Ivakhnenko and Lapa in 1965 (Ivakhnenko & Lapa, 1966). In this study, the best features in the layers are determined by statistical methods and sent to the next layer. Backpropagation is not used to train the networks end-to-end.

Second, deep learning architecture was proposed by Fukushima in 1979. It was developed by being inspired by the nervous system of vertebrates. Fukushima's networks, similar to today's networks, contain multiple bending and pool layers (Fukushima et al., 1983).

Although deep learning architectures have emerged in previous years, a successful deep neural network application has been developed for the first time by Yann LeCun et al. on mailbox texts (LeCun et al., 1989), (Şeker et al., 2017). After this work, Yann LeCunn applied convolutional networks together with backpropagation to classify handwritten digits (MNIST) using the “LeNet” network (Le Cun et al., 1989). With this study, the first operations in character recognition were started.

Today, with the developing technology, the interest in this field has increased. The term deep learning was first introduced by Igor Aizenberg et al. in 2000 (Aizenberg et al., 2000).

Later, in this area, Geoffrey Hinton described how to train a multilayer feedforward neural network and how to do a supervised back propagation method in his article in 2006 (Hinton, 2007)0.

With the development of computers, the acceleration of GPU and CPU, and training of deep networks without pre-training have emerged. Ciresan et al. used this method in applications such as traffic signs, medical imaging and character recognition (Ciresan et al., 2011).

Krishevsky, Sutskever and Hinton used similar architectures in 2012 and in their work using GPU, they developed the "dropout" layer to reduce memorization and used this method in their applications (Hinton et al., 2012)0.

From the past to the present, many applications have been developed with deep learning and continue to be developed. In this process, a lot of work has been done in the field of deep learning and different architectures have been developed. There are 6 architectures of deep learning. These are:

- Convolutional Neural Networks
- Recurrent Neural Networks
- Long / Short Term Memory
- Restricted Boltzmann Machines
- Deep Belief Networks
- Autoencoders and Denoising Autoencoders

Convolutional Neural Networks: Convolutional neural networks (CNN) are a different type of multi-layer perceptron (MLP). The first CNN network in history is the LeNet architecture founded by Yann LeCun in 1988 (Le Cun et al., 1989).

CNN algorithms include image and audio processing (Mushtaq et al., 2020), (Su et al., 2019), natural language processing (NLP) (Akhtyamova et al., 2017), (Sun et al., 2019)0, biomedical image processing (Cho et al., 2020), (Momeni et al., 2018). It is a deep learning algorithm that has the best classification success, especially in the field of image processing.

- CNN architecture processes data in various layers. The layer structure of the CNN model is as in Figure 4. These layers (Ergin, 2018, October) are:
- Convolutional layer – Used to extract features on the image.
- Non-Linearity layer – ReLU layer - It is the layer where the activation process is located. It uses a nonlinear function in the system.
- Pooling (Down-sampling) layer – Size reduction and compatibility check
- Flattening layer – Prepares one-dimensional vector data for Standard Neural Network.
- Fully-Connected layer – Uses Standard Neural Network for classification.

The Convolution layer is used to detect features on the image. This is done by using low-level or high-level filters that are smaller than the size of the image to extract the feature on the image in this layer. These filters are usually matrices of odd numbers. By moving these filters over the image and using matrix multiplication, features are tried to be detected. Zero values are added so that the picture does not lose its original size after the filters.

The Non-Linearity layer comes after the Convolution layers. This layer is also known as the activation layer and one of the activation functions is used. The purpose of this layer is to prevent the model from learning negative values or failing to grasp some features by using activation functions. Usually as an activation function; Nonlinear functions such as ReLU, tanh and sigmoid are used. Since the Rectifier (ReLU) function gives the best results in terms of speed, this function is mostly used in training.

The pooling layer is added between successive convolution layers. This layer is used to reduce its dimensionality. Thanks to this layer, unnecessary features are removed and focus on important features is ensured. In CNN models, two different pooling techniques are generally used as Max (Maximum) and Average (Average). In the first technique, Maxpooling, a filter is first created and this filter is moved over the picture and takes the largest number in the area it covers. In the second technique, Averagepooling, a filter is created again and this filter is moved over the picture to get the average number of pixels in the area it covers.

The Flattening layer is the layer that prepares the input data for the Fully Connected Layer. It takes the input data of neural networks as a one-dimensional array. These data are the data in the matrices from the Convolution and Pooling layers. The flattening layer converts this data to a one-dimensional array.

The Fully Connected Layer receives the data as processed in the Flattening layer and converts it into a one-dimensional array and performs the learning process with the classical neural network method. During the training, the Dropout method is used to prevent problems such as under fitting, unnecessary memorization and over fitting.

Training the Created Model

For the deep learning algorithm in the classification step, an application developed in Python using the Keras library was made. The Balanced part of the EMNIST dataset was used for training. The layers of the model are also created using the Convolutional neural network architecture. Layers of the developed model; It consists of 4 CONV2D layers, 4 MaxPooling2 layers, 1 Flatten layer, 1 Dropout layer and 2 Dense layers. The output of the layer structure of the model is shown in the application developed in Figure 4.

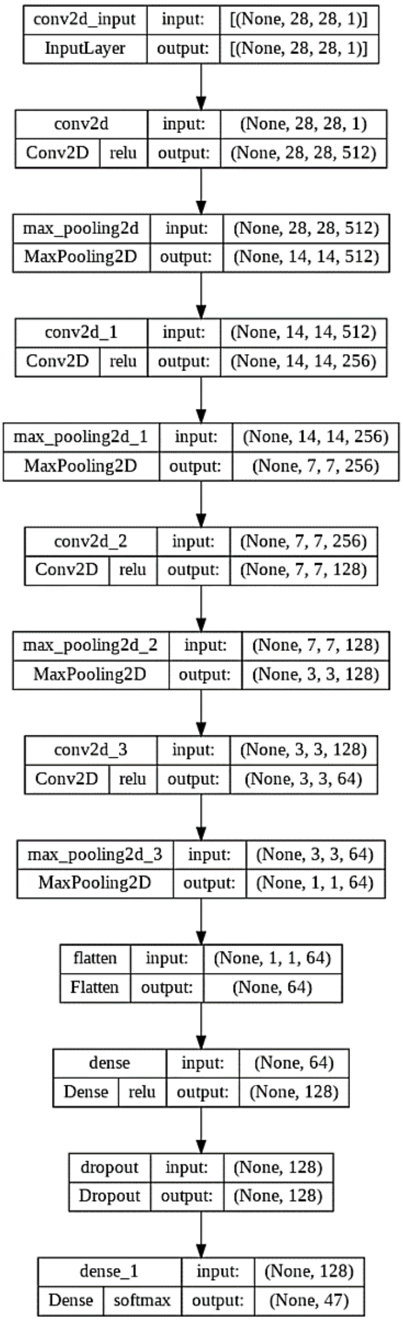

conv2d_input | InputLayer | input: [(None, 28, 28, 1)] | output: [(None, 28, 28, 1)]
conv2d | Conv2D | relu | input: (None, 28, 28, 1) | output: (None, 28, 28, 512)
max_pooling2d | MaxPooling2D | input: (None, 28, 28, 512) | output: (None, 14, 14, 512)
conv2d_1 | Conv2D | relu | input: (None, 14, 14, 512) | output: (None, 14, 14, 256)
max_pooling2d_1 | MaxPooling2D | input: (None, 14, 14, 256) | output: (None, 7, 7, 256)
conv2d_2 | Conv2D | relu | input: (None, 7, 7, 256) | output: (None, 7, 7, 128)
max_pooling2d_2 | MaxPooling2D | input: (None, 7, 7, 128) | output: (None, 3, 3, 128)
conv2d_3 | Conv2D | relu | input: (None, 3, 3, 128) | output: (None, 3, 3, 64)
max_pooling2d_3 | MaxPooling2D | input: (None, 3, 3, 64) | output: (None, 1, 1, 64)
flatten | Flatten | input: (None, 1, 1, 64) | output: (None, 64)
dense | Dense | relu | input: (None, 64) | output: (None, 128)
dropout | Dropout | input: (None, 128) | output: (None, 128)
dense_1 | Dense | softmax | input: (None, 128) | output: (None, 47)

Figure 4. Description of the layers of the model used in the study

The descriptions of the layers of the model used in this study are given below:

- CONV2D: It is the Convolution layer. It is the layer used to detect features on the image.

- MaxPooling2: The pooling layer is added between successive convolution layers. This layer is used to reduce dimensionality.

- Flatten: The Fatting layer is the layer that prepares the input data for the Fully Connected layer.

- Dropout: It is used to prevent under fitting, unnecessary memorization and over fitting problems during training.

- Dense: It is the Fully Connected layer.

The visual representation of the created model is shown in Figure 5.

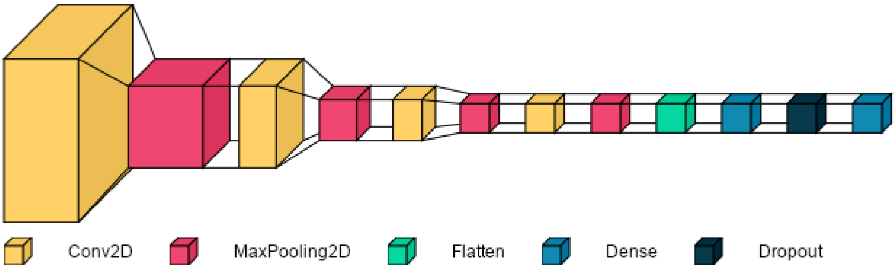


Figure 5. Visual representation of the model

The training of the model was done using the Balanced part of the EMNIST dataset. The model was trained 4 times according to different epochs and batch_size values. 10% of the training set is reserved for validation. The epochs and batch_size values used in the training of the model are as follows;

- 5 epochs and 512 batch_size

- 5 epochs and 1024 batch_size

- 10 epochs and 512 batch_size

- 10 epochs and 1024 batch_size

Pre-processing

In this study, a character-based handwriting recognition system has been developed. For this reason, some operations were performed to separate the documents in the data set created for the test into their characters. An application has been developed using the OpenCV Library for pre-processing. Pre-processing steps are explained in order. These steps are:

1. The texts on the paper were saved in the computer environment in a ".png" format after being scanned with the help of a scanner or camera.

2. The original image was first converted to a greyscale image.

3. The greyscale image is converted to a binary image, that is, to a black-and-white image, by applying the threshold function.

4. Lines were determined by applying morphological transformations on the black-and-white image.

5. The document is divided into words by applying the morphological transformation process on the lines again.

6. The normalization process was applied to the data set created with the existing characters.

At this stage, the application of the steps explained above was developed using the Python programming language. This source code was run with the help of Algorithm 1 given in Table 3; All documents in the test dataset are separated into characters.

Table 3. Pre-processing algorithm pseudocode

```
Algorithm 1: (Pre-processing)

Input: N images in the data set
Output: Dataset containing black and white toned character images

Step 1. Get Started
Step 2. Loop (N images):
    1. Upload the image from the folder.
    2. Convert the image to a grey scale.
    3. Convert the grey scale image to a black and white image.
    4. Determine the rows by applying a morphological operation.
Step 3. Loop (As many as detected rows):
    1. Separate the image into words by applying the dilation process on the image that is divided into lines.
Step 4. Loop (As many as detected words):
    1. Identify characters by applying morphological operations and contour subtraction in the image divided into words.
    2. Perform ROI operation on the characters that have been located
    3. Apply normalization to character images.
Step 5. Finish
```

The block diagram of the preprocessing step of the study is given in Figure 6.

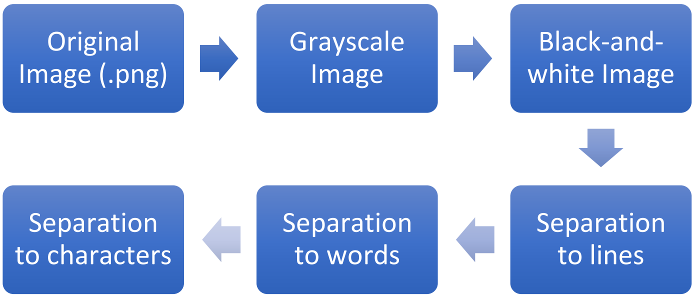


Figure 6. Block diagram of pre-processing

At this stage, classification was performed on the pre-processed characters. The classification application was developed using the Python language. In practice, the documents in the test dataset are separated into characters by applying the pre-processing step. Then, the prediction processes of the models were made using these characters. The block diagram of these steps is shown in Figure 7. Using the model's "predict" function, predictions were made on the characters.

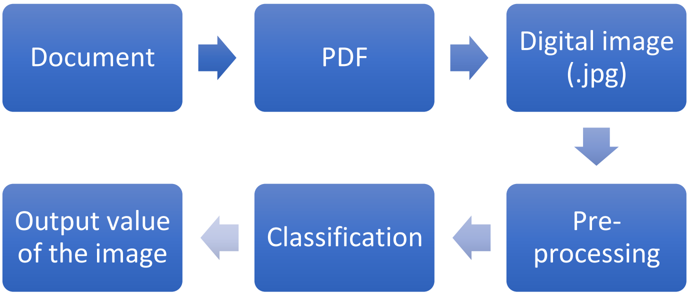


Figure 7. Block diagram of classification

Results

Epoch and 512 Batch_size

The first training of the model was performed using the EMNIST dataset and 5 epochs and 512 batch_size values. The desired success in training has not been achieved.

Epoch and 1024 Batch_size

The model was trained a second time using the EMNIST dataset with 5 epochs and 1024 batch_size values. The success (accuracy) and loss (loss) graph of the training is given in Figure 8 and Figure 9.

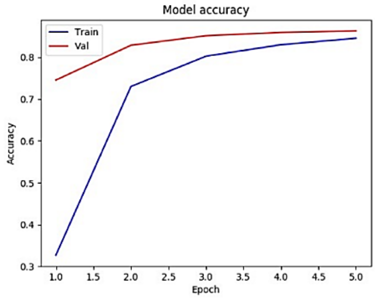


Figure 8. The accuracy graph of the model after training with 5 epochs and 1024 batch_size

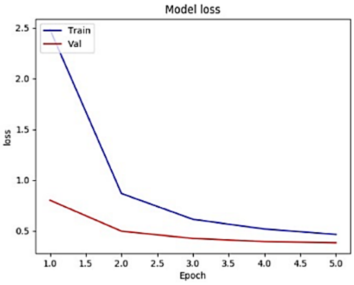


Figure 9. The loss graph of the model after training with 5 epochs and 1024 batch_size

Epoch and 512 Batch_size

The graphs of success and loss rates according to the 10 epoch and 512 batch_size values of the model are given in Figure 10 and Figure 11.

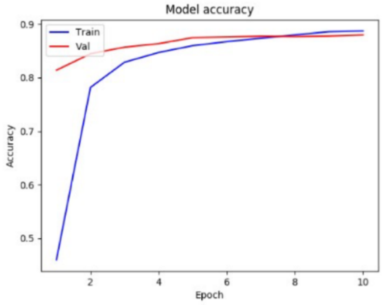


Figure 10. The accuracy graph of the model after training with 10 epochs and 512 batch_size

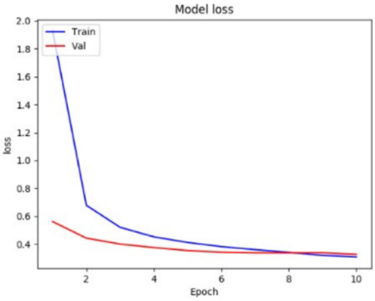


Figure 11. The loss graph of the model after training with 10 epochs and 512 batch_size

Epoch and 1024 Batch_size

The graphs of success and loss rates according to the 10 epoch and 1024 batch_size values of the model are given in Figure 12 and Figure 13.

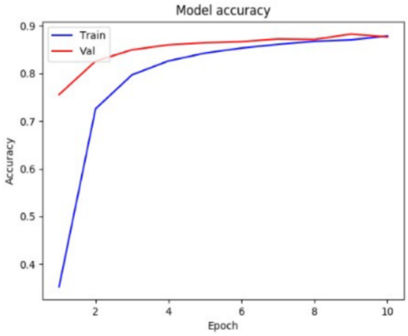


Figure 12. The accuracy graph of the model after training with 10 epochs and 1024 batch_size

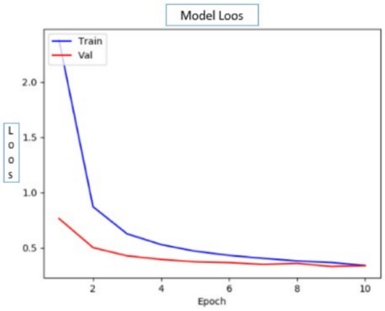


Figure 13. The loss graph of the model after training with 10 epochs and 1024 batch_size

Conclusion and Recommendations

Conclusion

In the study, a character recognition application was carried out by using a deep learning algorithm on handwritten character images. Since Azerbaijani has an infinite dimensional vocabulary, a character-based recognition process was carried out in the study. In the study, instead of only the border regions of the characters, a 2-dimensional classification process was made as a whole. Convolutional Neural Network algorithm, which is a deep learning architecture, is used for character recognition.

The study was carried out in two stages. The first stage is the creation and training of a model for the classification of characters. The second step is that the trained model performs the recognition on the handwriting. The operations carried out in these two stages are given in detail in section 2.

In the study, the model was created according to the Convolutional Neural Network architecture and was made using the Balanced part of the EMNIST dataset for its training.

The model, which was created according to the Convolution Neural Network architecture, was trained on the EMNIST dataset according to 4 different values. The learning results of the model are given in Table 4.

Table 4. The results according to the four different trainings of the model

Epochs/ batch_size	Acc	Loss	Val_acc	Val_loss	Test_acc	Test_loss
5/512	84.64%	37%	85.30%	35%	86%	36%
5/2014	84.54%	46%	86.29%	38.40%	86.48%	35.81%
10/512	88.72%	30.78%	87.98%	32.68%	87.50%	33.42%
10/1024	87.81%	34%	87.61%	34%	87%	35%

By training the model according to different epoch and batch_size values, the most successful model for character classification has been tried to be created. When Table 4 is examined, the increase in the epoch value in the training made according to different epoch values with the same batch_size values also increased the success of the model. However, the increase in the batch_size value in the training made according to different batch_size values with the same epoch values caused the success rate of the model to decrease.

Looking at the chart, the most successful training was done according to 10 epoch and 512 batch_size values with a rate of 88.72%.

After the training of the model, testing was done on handwritten documents on paper. At this stage, the handwritings on the paper were separated into their characters by going through the pre-processing described in section 2. A recognition process was performed on the separated characters according to the most successful training of the model.

Recommendations

In this study, character-based handwriting recognition was performed using the Convolutional Neural Network algorithm. A more successful recognition process can be done by using different deep learning algorithms and parameters on the data set used.

In future studies, a different recognition process will be performed on the same data set by using deep learning architectures in another way. And in order to increase the accuracy of classification, using databases containing combinations of 2 or 3 letters, the letters will be verified, the characters will be combined and the document will be transferred to the computer environment as a whole.

References

Aizenberg, I. N., Aizenberg, N. N., & Vandewalle, J. (2000). Multiple-Valued threshold logic and multi-valued neurons Multi-Valued and Universal Binary Neurons (pp. 25-80): Springer.

Akhtyamova, L., Ignatov, A., & Cardiff, J. (2017). A Large-scale CNN ensemble for medication safety analysis. Paper presented at the International Conference on Applications of Natural Language to Information Systems.

Arica, N., & Yarman-Vural, F. T. (2001). An overview of character recognition focused on off-line handwriting. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 31(2), 216-233.

Bektaş, B., Babur, S., Turhal, U., & Köse, E. (2016). Makine öğrenmesi yardimiyla optik karakter tanima sistemi optical character recognition system via machine learning. 5th International Printing Techologies Symposium İstanbul Üniversitesi - İstanbul - 04-05 Kasim / November.

Bilgin Taşdemir, E. F. (2018). A large vocabulary online handwriting recognition system for Turkish.

Cho, M., Ha, J., Park, C., & Park, S. (2020). Combinatorial feature embedding based on CNN and LSTM for biomedical named entity recognition. Journal of Biomedical Informatics, 103, 103381.

Ciresan, D. C., Meier, U., Gambardella, L. M., & Schmidhuber, J. (2011). Convolutional neural network committees for handwritten character classification. Paper presented at the 2011 International Conference on Document Analysis and Recognition.

Deng, L., & Yu, D. (2014). Deep learning: methods and applications. Foundations and trends in signal processing, 7(3–4), 197-387.

Fukushima, K., Miyake, S., & Ito, T. (1983). Neocognitron: A neural network model for a mechanism of visual pattern recognition. IEEE transactions on systems, man, and cybernetics(5), 826-834.

Hébert, J.-F., Parizeau, M., & Ghazzali, N. (1999). Learning to segment cursive words using isolated characters. Paper presented at the Proc. of the Vision Interface Conference.

Hennig, A., Sherkat, N., & Whitrow, R. (1996). Zone estimation for multiple lines of handwriting using approximating spline functions. Paper presented at the Proceedings of the Fifth International Workshop on Frontiers in Handwriting Recognition IWFHR-5.

Hennig, A., Sherkat, N., & Whitrow, R. J. (1997). Recognising letters in on-line handwriting using hierarchical fuzzy inference. Paper presented at the Document Analysis and Recognition, 1997., Proceedings of the Fourth International Conference on.

Hinton, G. E. (2007). Learning multiple layers of representation. Trends in cognitive sciences, 11(10), 428-434.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580.

Ivakhnenko, A. G. e., & Lapa, V. G. (1966). Cybernetic predicting devices: PURDUE UNIV LAFAYETTE IND SCHOOL OF ELECTRICAL ENGINEERING.

Le Cun, Y., Jackel, L. D., Boser, B., Denker, J. S., Graf, H. P., Guyon, I., Hubbard, W. (1989). Handwritten digit recognition: Applications of neural network chips and automatic learning. IEEE Communications Magazine, 27(11), 41-46.

Le Cun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Handwritten digit recognition with a back-propagation network. Paper presented at the Proceedings of the 2nd International Conference

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. Neural computation, 1(4), 541-551.

Momeni, A., Thibault, M., & Gevaert, O. (2018). Dropout-enabled ensemble learning for multi-scale biomedical data. Paper presented at the International MICCAI Brainlesion Workshop.

Mushtaq, Z., Su, S.-F., & Tran, Q.-V. (2020). Spectral images based environmental sound classification using CNN with meaningful data augmentation. Applied Acoustics, 172, 107581.

Plamondon, R., & Srihari, S. N. (2000). Online and off-line handwriting recognition: a comprehensive survey. IEEE Transactions on pattern analysis and machine intelligence, 22(1), 63-84.

Şeker, A., Diri, B., & Balık, H. H. (2017). Derin öğrenme yöntemleri ve uygulamaları hakkında bir inceleme. Gazi Mühendislik Bilimleri Dergisi (GMBD), 3(3), 47- 64.

Şekerci, M. (2007). Birleşik ve eğik Türkçe el yazısı tanıma sistemi.

Srihari, S. N., Cha, S.-H., Arora, H., & Lee, S. (2001). Handwriting identification: Research to study validity of individuality of handwriting and develop computer-assisted procedures for comparing handwriting. Technical Report CEDAR-TR-01-1.

Steinherz, T., Rivlin, E., & Intrator, N. (1999). Offline cursive script word recognition– a survey. International Journal on Document Analysis and Recognition, 2(2-3), 90-110.

Su, Y., Zhang, K., Wang, J., & Madani, K. (2019). Environment sound classification using a two-stream CNN based on decision-level fusion. Sensors, 19(7), 1733.

Sun, B., Yang, L., Zhang, W., Dong, P., Young, C., Dong, J., & Lin, M. (2019). Demonstration of Applications in Computer Vision and NLP on Ultra Power- Efficient CNN Domain Specific Accelerator with 9.3 TOPS/Watt. Paper presented at the 2019 IEEE International Conference on Multimedia & Expo Workshops (ICMEW).

Zouhal, L. M., & Denoeux, T. (1998). An evidence-theoretic k-NN rule with parameter optimization. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 28(2), 263-271.

Some types of Solar Cells and their characters

Babak Emdadi
Faculty of Engineering and Applied Science,
Khazar University
emdadi.babak2021@khazar.org

Abstract

A solar cell is a type of electronic device that directly converts solar energy from light into electricity. A current and a voltage are produced by the sun's light shining on the solar cell to produce electrical vigor. This process requires a material that, when exposed to light, elevates an electron's energy level in the first stage. Additionally, in the second stage, this stronger electron is transferred from the solar cell to an external circuit. As a result, the electron loses energy in the outside circuit before returning to the solar cell. The requirements for photovoltaic energy conversion can potentially be met by a variety of materials and techniques. Although not strictly speaking, nearly all photovoltaic energy conversion uses semiconductor materials in the form of a p-n link. Taking into account the increase in tolerable vigor. Solar energy is the best example to use to show this. We will examine the many types of solar cells and their characteristics in this essay.

Keywords: Solar cells, dye-sensitized solar cells, Quantum Dot solar cell, multi-junction solar cells, Hybrid solar cells

Introduction

The amount of solar energy that the Earth receives is astounding. It delivers more energy in a day than the present population would require in 27 years. According to this statement, "The amount of sun glow considering the earth over a three-day period is comparable to the energy stored in all fossil energy provenances." Despite being a free, endless resource, harnessing solar energy is a relatively new idea. We have gone a long way considering that "the first practical solar cells were made less than 30 years ago." There is no longer a justification not to consider solar vigor for your home due to the increase in solar handicraft companies creating unique and specific sun power systems for individual homes.

The rate of sunshine and the cost of the material are the only two drawbacks of using solar electricity. The amount of sunshine that a place receives "variously depends on its location, the time of day, the season, and the presence or absence of clouds. In this research, we will examine different types of solar cells in light of the relevance of solar cell usage and the astonishing application of solar energy.

A solar cell, also known as a photovoltaic cell, is an electrical system that converts light energy directly into electricity through a phenomena known as photovoltaic efficacy. A system whose electrical characteristics, such as current, voltage, or resistance, change when exposed to light is a type of photoelectric cell. The foundation of photovoltaic modules, also referred to as solar panels, are solar cells. Whether the source is artificial light or sunshine, solar cells are classified as photovoltaic. They can be utilized as a photodetector (for infrared detectors, for instance), to monitor light or other electromagnetic glows close to the visible spectrum or to gauge the intensity of light.

Three fundamental qualities are required for a photovoltaic (PV) cell to function:

1. The attraction of light, producing either electron-hole pairs or exactions.

2. The segregation of charge carriers of contrary types.

3. The apart exploitation of those carriers from an external circuit.

Meanwhile, we can see in the below photos brief history of the solar energy process, and the development of solar tech, which demonstrate to us how solar cell could become the world's cheapest source of energy:

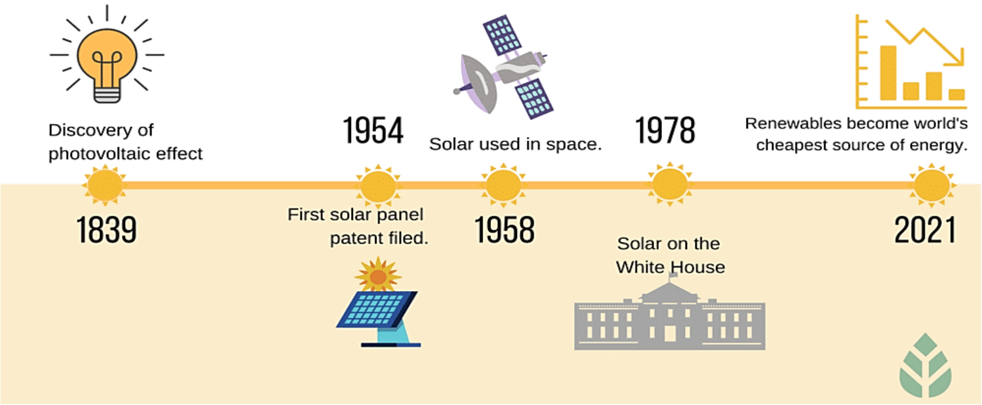

Discovery of photovoltaic effect
1839
1954
First solar panel patent filed.
Solar used in space.
1958
1978
Solar on the White House
Renewables become world's cheapest source of energy.
2021

Figure 1. A succinct history of solar cells

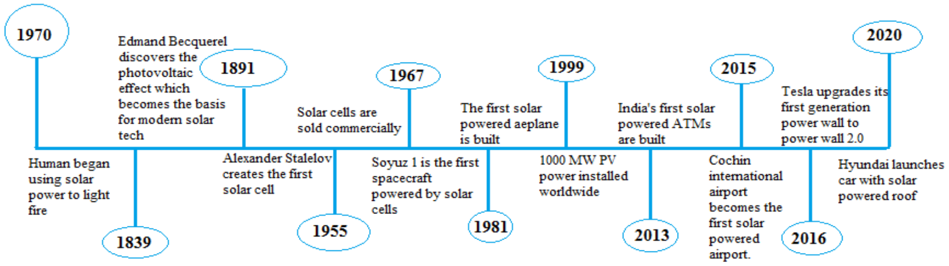

1970
Human began using solar power to light fire
1839
Edmand Becquerel discovers the photovoltaic effect which becomes the basis for modern solar tech
1891
Alexander Stalelov creates the first solar cell
1955
Solar cells are sold commercially
1967
Soyuz 1 is the first spacecraft powered by solar cells
1981
The first solar powered aeplane is built
1999
1000 MW PV power installed worldwide
India's first solar powered ATMs are built
2013
Cochin international airport becomes the first solar powered airport.
2015
Tesla upgrades its first generation power wall to power wall 2.0
2016
Hyundai launches car with solar powered roof
2020

Figure 2. Development of solar tech during the years

Materials and methods

Common names for solar cells are derived from the semiconducting materials used to make them. To draw sunlight, these materials need to possess certain qualities. While some cells are suited for use in the region, others are designed to control and alter sunlight that reaches the Earth's surface. In order to take use of numerous absorptions and charge segregation processes, solar cells can either use multiple physical configurations (multi-junction) or simply one layer of light-absorbing materials (single-junction). To take advantage of various attractions and charge segregation processes, solar cells can be constructed using a single substrate of a light-engrossing substance (single-junction) or using a variety of physical forms (multi-junction). There are three generations of solar cells: first, second, and third.

The first generation of cells, also known as conventional, traditional, or wafer-based cells, are made of crystalline silicon type, the most popular PV technology on the market. Slim film solar cells are the second generation of cells. Many thin-film applied sciences, also known as emerging photovoltaics, are present in the third generation of solar cells.

There has been a lot of research invested in these technologies despite the fact that their efficiencies had been at a low rate and the consistency of the absorber material was frequently too stunted for joinery applications. This is because they promise to achieve the goal of producing low-cost, high-efficiency solar cells. Silicon solar cells are found in "first generation" panels. They are either carved out of a block of silicon that contains many different crystals or constructed from a single silicon crystal (mono-crystalline) (multi-crystalline - shown at right).

Due to their decreased material requirements, "second race" thin-film solar cells are more affordable to produce than conventional silicon solar cells. The thin-film PV cells are a physically thin technology that has been used in photovoltaics, as the name suggests. Although they are only marginally less stunning than other types, they do need a larger surface area to provide the same level of power. The various solar cell types include the following:

Dye-Sensitized Solar Cell (DSSC)

Dye Sensitized solar cells (DSSC), also somewhat referenced to as dye-sensitized cells (DSC), are a third-generation photovoltaic (solar) cell that changes any observable glory into electrical energy. This new class of advanced solar cells can be likened to feigned photosynthesis due to the way in which it emulators nature's attraction of light vigor. DSSC is a factious technology that can be used to produce electricity in a broad span of light conditions, indoors and outdoors, enabling the user to turn both sophisticated and normal glory into energy to power a wide span of electronic systems. A dye-sensitized solar cell, also known as a thin film solar cell or DYSC (Roy et al., 2011) solar cell, is a low-cost type of solar cell.

The DSSC has a number of appealing qualities, including being simple to create using standard roll-printing procedures, being semi-pliable and semi-translucent, which implies a variety of uses not possible with glass-based systems, and the fact that the majority of the materials used are affordable.

An affordable solar cell from the group of slim film solar cells (Bose et al., 2015) is a dye-sensitized solar cell (DSSC, DSC, or DYSC; (Roy et al., 2011). It is based on a photoelectrochemical semiconductor that is formed between an electrolyte and a photo-sensitized material. The DSSC has some attractive features; it is uncomplicated to make utilizing customary roll-printing procedures, is semi-pliable and semi-clear which suggests a diversity of uses not suitable for glass-based orders, and most of the materials used are inexpensive.

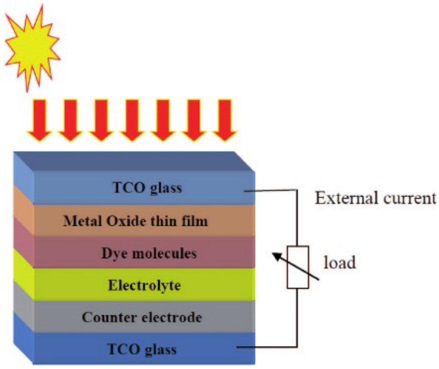


Figure 3. Dye-sensitized solar cell device schematic.

Practical experience has shown that it is difficult to remove some expensive materials, particularly platinum and ruthenium, and the fluid electrolyte makes it impossible to create a cell that is suitable for use in all weather conditions. Even though its conversion efficiency is lower than that of the best thin-film cells, from a practical standpoint, its value to implementation ratio must be high and sufficient to enable them to compete with fossil fuel electricity production by achieving grid parity. Commercial applications that were delayed because to issues with chemical consistency (Tributsch., 2004).

Hybrid Solar Cell

In hybrid solar cells, the advantages of both organic and mineral semiconductors are combined. Conjugated polymers that draw glory as of the giver and redeploy cavities are related to or produced from living matter in hybrid photovoltaics. In hybrid cells, minerals are used as the electron transmitter and acceptor in the structure (Milliron et al., 2005). The roll-to-roll hybrid photovoltaic systems have the potential to be not only affordable but also capable of scalable solar energy conversion.

In hybrid solar cells, the natural substance is mixed with a great electron carrier material to figure the photoactive layer (Shaeen et al., 2005). The two substances are mixed simultaneously in a heterojunction-sort photoactive substrate, which can have a better vigor alteration yield into a single material. One of the substances does as the photon absorber and exaction giver (Saunders et al., 2008).

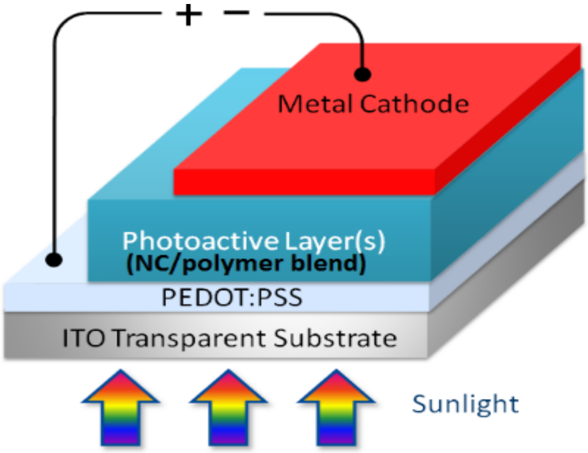


Figure 4. Schematic of Hybrid Solar cell.

Multi-junction Solar Cell (MJ)

Multi-junction (MJ) solar cells are solar cells with several p–n connections built of different semiconductor substances. Each material's p-n connection will produce an electric current in reply to several wavelengths of light. The utilization of numerous semiconducting materials permits the absorbance of a wider span of wavelengths, progressing the cell's daylight to electrical vigor transformation output. Common one-junction cells have a most visionary output of 34%. In a theory way, an unlimited quantity of junctions would have a limiting efficiency of 86.8% under extremely focused daylight.

Nowadays, the excellent laboratory instances of customary crystalline silicon solar cells have efficiencies of 20%-25%, while lab instances of multi-connection cells have indicated a performance of over 43% (Bagher et al., 2015).

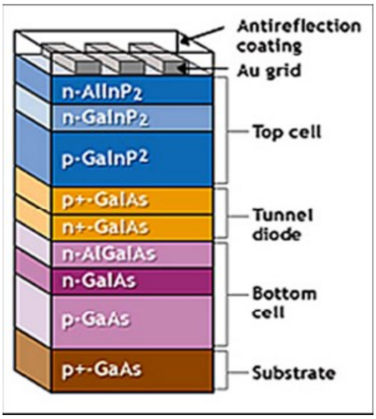


Figure 5. Multi-junction Cell.

Quantum Dot Solar Cell

A solar cell design that uses quantum dots as an appealing photovoltaic material is known as a quantum dot solar cell. It seeks to rebuild massive materials like silicon, CIGS (copper, indium, gallium selenide), or CdTe. Quantum dots have band gaps that are harmonic among a broad span of vigor steps by altering the dots' measure. In mass materials, the band gap is stable by the selection of substances.

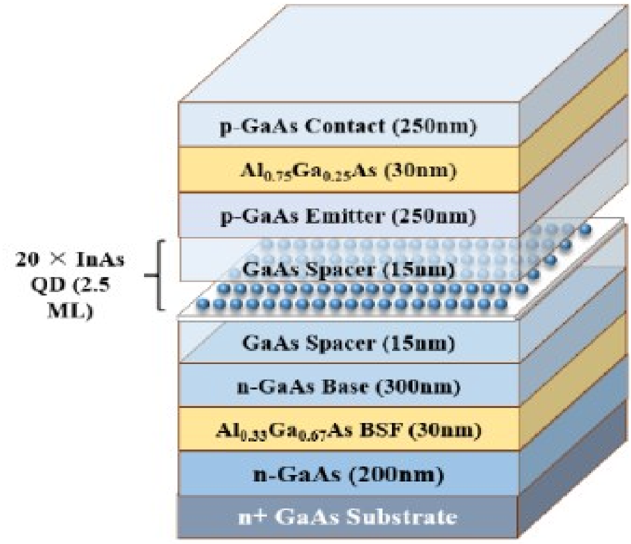


Figure 6. Schematic of Quantum-dot solar cell.

This trait creates quantum dots noteworthy for multi-connection solar cells, where a diversity of substances are used to ameliorate yield by collecting manifold shares of the solar vision. Quantum dots have been referred to as "artificial sophisticated atoms".

These vigor ranks are harmonic by altering their scope, which in turn describes the band gap. The dots can be extended over a scope of the measure, letting them describe a type of band gaps in the absence of altering the underlying substance or making procedures (Brabec et al., 2004).

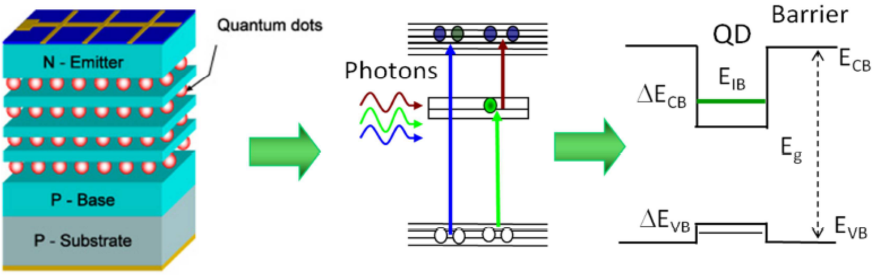


Figure 7. Schematic of operating Quantum dot solar cell.

Thin Film Solar Cell (TFSC)

The construction of a thin-film solar cell (TFSC), a second-generation solar cell, involves layering one or more thin substrates or thin films of photovoltaic material. Thin-film solar cells, which include amorphous and other thin-film silicon, cadmium telluride (CdTe), copper indium gallium diselenide (CIGS), and others, are used in a variety of technological processes (a-Si, TF-Si).

The film's width is adjustable between a few nanometers (nm) and tens of micrometers (m), making it significantly thinner than thin-films rival technology, such as the conventional, and first-generation crystalline silicon solar cell. This enables flexible, lighter-weight, and low-drag thin film cells. It serves as a semi-clear photovoltaic polishing agent that can be applied to windows and is used to create integrated photovoltaics.

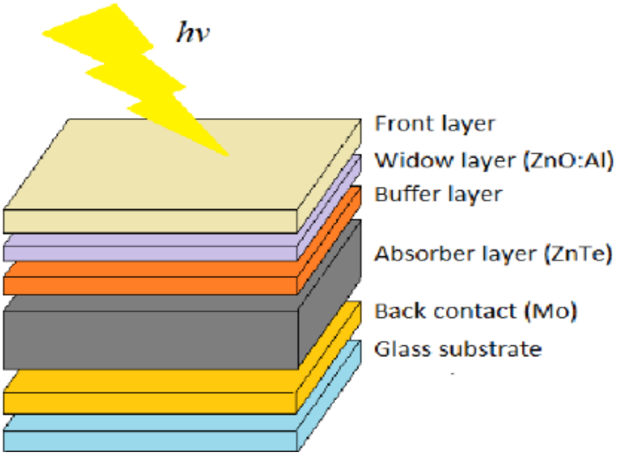


Figure 8. Schematic of this film solar cells structure.

Conclusion

Solar vigor which is a composition of glory and warmth is built by the sun. This vigor change position from the sun and arrives at the ground where people gather via solar gatherers and change it into any favorable shape of vigor. Considering a hypothesis this capable of being a renewed origin of vigor is strong adequate to exchange the demand of electric energy that we achieve from 650 tanks of oil yearly. Some of the properties of solar vigor:

1. Powerhouses: In customary vigor producing centers not able to be renewed vigor origins are utilized to boil water and generate vapor so that a machine for

producing continuous power in which a wheel or rotor can rotate and water to generate electricity. Albeit with the usage of solar vigor warmth of the sun is able to boil that water to generate vapor and spin turbines.

2. Dwellings: The utilization of solar vigor is boosting houses nicely. Territorial instruments are able to easily use electric vigor make via solar vigor. In addition, this solar vigor is operating the solar stove to provide warm water in dwellings.
3. Commercial benefits: On the ceilings of several constructions we are able to detect glass PV modules or other styles of solar panels. These boards are utilized there to reserve electric vigor to diverse companies or other sections of the construction in a faithful method.
4. Ventilation technique: In copious sites solar vigor is utilized for the provision of fresh air to a room, and building targets. It aids in running bath fans, floor fans, and ceiling fans at homes.
5. Vigor pump: Solar vigor does not just assist in enhancing the provision of the new air procedure in your dwellings of course with this it is able to even aid in distributing water in every home.
6. Swimming places: Pools are excellent entertainment for children and grown-ups during the year. While among the winter seasons it is hard to maintain the water warm in these swimming zones with the lowest energy use. Solar vigor is able to support copious in this concern very well.
7. Solar lighting: These kinds of lights are also famous for the light of the day, and task with aid of solar energy. These glares reserve the normal vigor of the sun during the day and afterward change this vigor into electric energy to the beam of the lamp during the night.
8. Solar Automobiles: It is an electric automobile that is refreshed by solar vigor or the light of the sun. Solar boards are utilized on this automobile that engrosses sunshine and so alters it into an electric system vigor.
9. Distant usages: Distant homes are taking the advantage of solar vigor on a wide range. Long-range academies, gathering places, and hospitals are able to use solar boards and batteries in them any place to generate and utilize electric energy.

Subsequent-races solar cells are able to be unlimitedly more practical thanks to a novel discovered nanotube design able to transform electrical charges 100 million times more elevated in comparison to formerly counted. The majority of the solar cells presently utilized silicon to engross light, albeit, vain in the substance has directed investigators to improve carbon nanotubes that are able to be installed to boost the light attraction abilities of current cells.

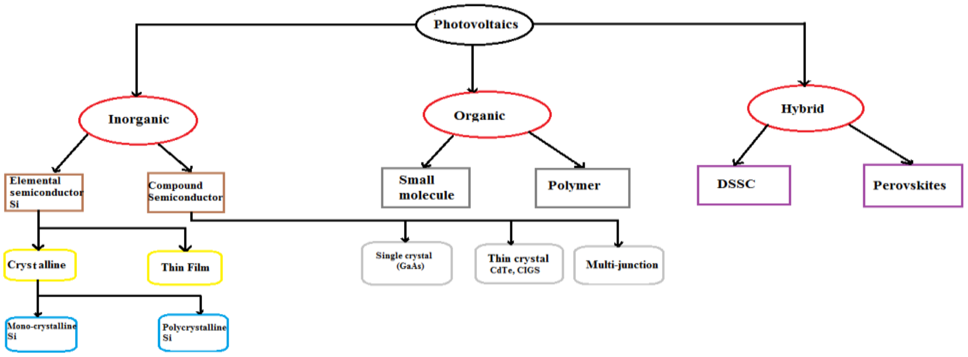


Figure 9. The diagram of improvement of photovoltaic solar cells

References

Bagher, A. M., Vahid, M. M. A., & Mohsen, M. (2015). Types of solar cells and application. American Journal of optics and Photonics, 3(5), 94-113.

Bose, S., Soni, V., & Genwa, K. R. (2015). Recent advances and future prospects for dye sensitized solar cells: A review. International Journal of Scientific and Research Publications, 5(4), 1-8.

Brabec, C. J., Neugebauer, H., Hummelen, J. C., & Sariciftci, N. S. (2004). Sol. Energy Mater. Sol. Cells solmat, 30(83), 273.

Milliron, D. J., Gur, I., & Alivisatos, A. P. (2005). Hybrid organic–nanocrystal solar cells. MRS bulletin, 30(1), 41-44.

Moss, S. J., & Ledwith, A. (1989). Chemistry of the Semiconductor Industry. Springer Science & Business Media.

Roy, M. S., Deol, Y. S., Kumar, M., Prasad, N., & Janu, Y. (2011, October). Dye-sensitized Solar Cells for Solar Energy Harvesting. In AIP Conference Proceedings (Vol. 1391, No. 1, pp. 46-49). American Institute of Physics

Shaheen, S. E., Ginley, D. S., & Jabbour, G. E. (2005). Organic-based photovoltaics: toward low-cost power generation. MRS bulletin, 30(1), 10-19.

Saunders, B. R., & Turner, M. L. (2008). Nanoparticle–polymer photovoltaic cells. Advances in colloid and interface science, 138(1), 1-23.

Tributsch, H. (2004). Dye sensitization solar cells: a critical assessment of the learning curve. Coordination Chemistry Reviews, 248(13-14), 1511-1530.

Volume 6 №2 2022, 66-75
DOI: 10.5782/2520-613.202.6.2.66

Mitigating Resource Management and Continuous Integration Obstacles in Heavy Traffic Systems Using Containerization and Orchestration Tools

Tural Ahmadov, Leyla Muradkhanli

Graduate School of Science, Art and Technology,
Khazar University, Azerbaijan
Corresponding Author. Email: ahmadov.tural@khazar.org

Abstract

In this paper I covered virtualization technologies, including Virtual Machines and Containerization engines analyzing their advantages and drawbacks. After reviewing possible container application areas, I introduce Kubernetes, an orchestration tool that manages a cluster of containers on autopilot given correct configurations. After that, I give more details on how an orchestration framework functions, communicates with containers, and keeps the containers up. Moreover, I propose CI/CD tools and compare their efficiency through a designed experiment built using a cluster of nodes provided by the accessible version of Google Cloud Platform and deployed containers via Google Kubernetes Engine. At the end of the experiment, I give a comprehensive analysis of the results. To summarize, both containers and virtual machine technologies allow users to describe and develop their software environments before running them on top of multiple resources in a portable, repeatable manner. With containers, it is possible to construct scalable architecture composed of a large number of services (microservices). Also, the integration of new features can be done more efficiently if deployed in a continuous manner using the proposed CI/CD tools. Nevertheless, there are open questions to be researched, such as how the various tools respond when something goes wrong in the pipelines and the best policies for reverting to previous versions to ensure high availability.

Keywords: VM; containerization; orchestration; CI; CD; pipeline.

Introduction

For high traffic systems any optimization means that the project may use less resources which means lowering the costs. I am going to look through and compare existing containerization and orchestration solutions which can assist with solving the issue.

Moreover, for real-time 24-hour active systems integration of new features can be done only in continuous way as down time has to be as close to zero as possible. CI/CD tools with Kubernetes are my primary target of research for the challenge. Given the wide selection of alternatives available, it can be very difficult for an organization that wants to adopt Kubernetes to decide on which CI/CD tools to adopt. This thesis will investigate the advantages and disadvantages of some of the most popular tools, as well as different types of CI/CD pipelines in Kubernetes, using a combination of literature studies and experiments.

This subsection has two subsections

The aims and objectives for the project are:

➢ A1 Research containerization technologies

▪ O1 Compare container-based state-of-the-art virtualization engines

➢ A2 Review orchestration tools

▪ O2 Contrast advantages and drawbacks of the existing container orchestration frameworks

➢ A3 Create an infrastructure to test the different CI / CD tools.

▪ O3 Identify and setup example Kubernetes applications that will be used for testing.

▪ O4 Set up pipelines with tools that combine CI / CD, separate CI and CD tools,

Kubernetes-specific tools and generic tools.

➢ A4 Establish evaluation criteria and use the testing infrastructure to evaluate CI / CD

pipelines.

▪ O5 Identify the appropriate evaluation criteria.

▪ O6 Characterize each pipeline according to the criteria.

➢ A5 Evaluate which CI/CD technologies are most suitable for a microservices application

on Kubernetes.

- O7 Investigate Kubernetes-specific tools compared to generic CI/CD tools.
- O8 Investigate integrated compared to standalone CI/CD tools.

Background and Related work

This section gives a short-term clarification of the associated work studies in resource management, containerization, orchestration and CI/CD.

Researchers in this study (MinSu Chae, 2017) examined the effectiveness of KVM and Docker. According to them, three different techniques were employed to gauge performance: (a) comparing the host operating system's CPU and memory utilization, (b) measuring idle CPU and memory usage and IO performance via massive file copying, and (c) comparing Web server performance using JMeter. The measured findings revealed a 3.6–4.6 times difference in memory consumption. When you launch a virtual computer using KVM, the operating system must start. To execute the program in containers, Docker needs the absolute least resources. A performance comparison reveals that Docker utilizes CPU, HDD, and RAM more quickly and effectively than KVM. In fact, even when no action is carried out, KVM wastes extra resources for the operating system. Additionally, while using KVM, the process of creating a new VM is time-consuming. When building a distributed system, it takes some time to generate a new VM for load balancing if a VM suddenly suffers a load. To do more processing on the same PM, utilize the Docker's Container as opposed to a VM. KVM and Docker are only contrasted when set up on a single physical machine in the study. Based on the placement technique, a clustering environment influences the performance of a virtual machine and containers. As a result, further research is required to compare the effectiveness of KVM and Docker in a clustering setting.

Another research examined the outcomes of several Kubernetes resource management tools, including the Horizontal Pod Autoscaler and resource allocation through request and limit settings. Experiments demonstrate that identifying appropriate requests boosts cost-efficiency in contexts with few applications without significantly affecting other factors. This was confirmed for a Cassandra-based application, a made-up SaaS service, and workloads that were both seasonal and bursty. Regardless of the scaling technique chosen, scaling Cassandra in Kubernetes hurts performance rather than increases it because of an overhead added by running Cassandra on Kubernetes. Even when pods are co-located and the workload is seasonal, the HPA works effectively for an artificial SaaS application. Other strategies could be used for workloads that come in bursts. In conclusion, despite certain drawbacks, Kubernetes' scaling capabilities show considerable promise for preventing SLA breaches and improving resource cost-efficiency in settings focused on containers (Stef Verreydt, 2019).

The case study in this research addressed a variety of adoption issues, according to studies on continuous integration [3], with the following conclusions standing out: 1) A key element for successful CI implementation is mentality. In order to convert skeptics, one must take into account their resistance to the introduction of a new procedure. 2) In order to make the everyday activities involved in the CI process easier, testing tools and the infrastructure supporting it must be mature. To enable more frequent and effective integrations, continuous integration promotes the use of automated technologies. (3) Like Agile, the assumptions behind the CI concept could not hold true for all businesses, goods, or projects, particularly those with broader scopes. Some of the difficulties with the shift to continuous integration have been recognized, such as testing, infrastructure maturity, tools, and attitude. Software needs, however, were also mentioned in this survey as a barrier to CI adoption.

Knowing how to overcome the obstacles that an organization may have while implementing CI gives practitioners degree of understanding that they may not have had before. Companies who are going to implement CI might utilize these problems as a checklist.

Resource management using containerization

IBM (Zhang et al., 2018) presented containerization technology for the first time in 1979. Implemented in the UNIX operating system V7, with the addition of a chroot (Ltd, 2022) system call. This was the first step toward isolation, with segregated groups functioning on a single host. This separation relied on numerous underlying technologies included into the Linux kernel, including namespaces and cgroups (Chiang, 2022). Namespace support was introduced in Linux kernel version 2.4.19, whereas cgroups, often known as control group technology, was published in Linux kernel version 2.6.24.

The introduction of microservices architecture (Campeanu, 2018) based on containers technology, including Linux containers (LXC) (Zhang et al., 2018), OpenVZ (Openvz, 2022), Docker (Inc, 2022), Singularity (Sylabs 2022), and uDocker (2022), produced a change in the way we construct applications, from operations to programming. Container lifecycle management was utilized by several major orchestration systems, including Kubernetes (2019), Docker Swarm (2019), and Apache Mesos (2022). These orchestrators offered frameworks for container management inside a microservices architecture. Furthermore, these frameworks include capabilities ideal for scheduling containers with limited resources, fault tolerance, and auto-scaling. Kubernetes and Open-Shift are two orchestration platforms that are becoming more popular in computer systems, particularly those in the industrial and scientific areas. Following that, Rancher-compliant orchestration management solutions arose to manage orchestrators while maintaining efficiency characteristics that assure performance throughout the computing infrastructure.

Although cloud computing is the most common setting for application containerization (Pahl et al., 2017), containerization technique is also applicable to various application domains other than cloud services, such as scientific computing, big data processing, high performance computing, and development operation (devops).

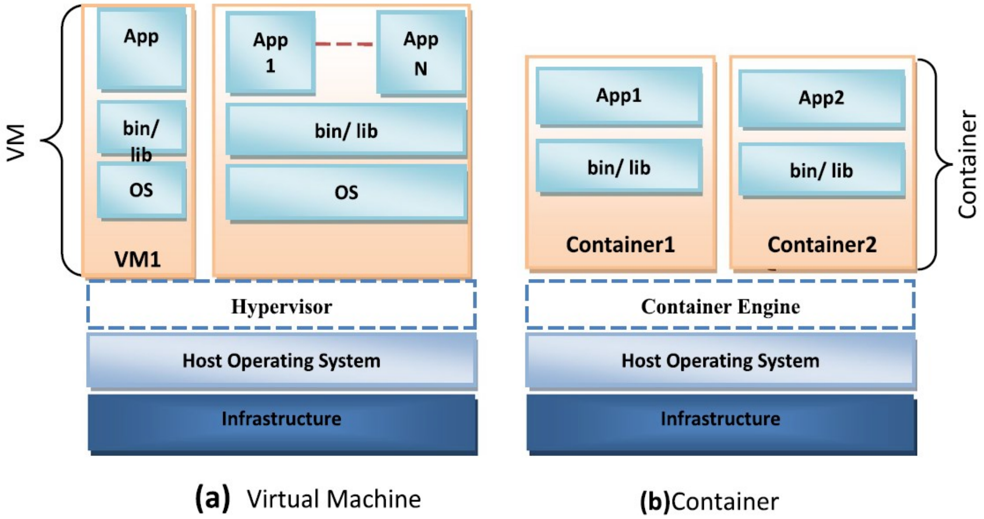


Figure 1. Comparison of system architecture-based virtualization.

The most significant advantage of virtualization is that it abstracts the hardware. It does, however, provide an isolated working environment for programs by aggregating logical resources such as CPU, memory, network, and storage. As demonstrated in Figure 1-a, the virtual machine (VM) instance's whole guest OS operates as a single process on the host. This results in high resource needs, which cause the VM to start slowly.

I/O routing is used in virtualization to coordinate requests between virtual devices and shared physical hardware. Instead of controlling resources, virtual machine migration between real computers created security risks. This vulnerability makes the system more insecure, and installing virtualized systems has become considerably more difficult. Operating system-level virtualization is the most popular form of virtualization, which allows for the use of isolation methods. The isolation method offers users with virtual environments that are comparable to those seen on dedicated servers. Container refers to the isolated virtual environment seen in Figure 1-b.

Container orchestration

Container orchestration enables cloud and application providers to describe how multi-container packaged applications in the cloud are selected, deployed, monitored, and dynamically configured. It is a framework that provides a collection of APIs for managing the container's whole life cycle (cf., Figure 2).

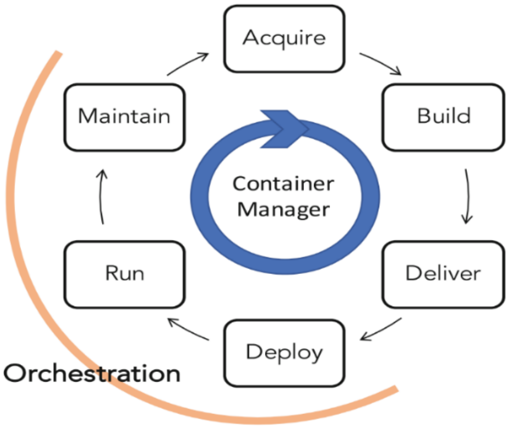


Figure 2. The life cycle of a container.

Container managers may be on-premise (to be deployed, configured, and maintained on private datacenters or in the cloud) or managed (offered by cloud providers as a service). Docker was created as a container management solution; however, the container management ecosystem is constantly evolving. Docker, for example, can handle both Windows Server containers and Hyper-V containers. rkt also provides APIs for simple application container management. Google Container Engine, Microsoft Azure Container Service, and Amazon ECS are three cloud platform managed container managers (usually they support Docker and LXC). LXD is the manager for LXC in terms of system containers. OpenVz also offers APIs for container management.

Material and method

Implementation

This section depicts the experiment's execution. Figure 3 depicts the implementation that will be used in this investigation.

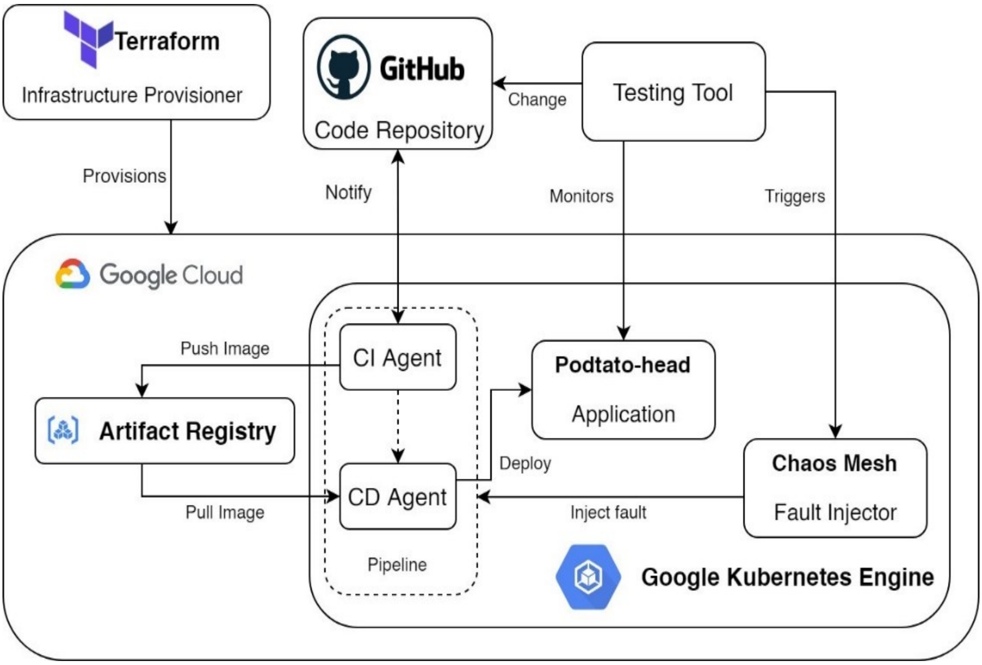


Figure 3. Implementation overview.

As seen from Figure 3, cluster was setup using Google Cloud environment. Terraform for infrastructure provisioning, custom testing tools, github as code repository.

Results and disscussions

According to the experiment findings:

- Kubernetes-specific tools have a quicker deployment time and can sustain pod defects with minor deployment time increases.

- Integrated tools take longer to deploy. However, this is most likely due to the tool's general nature. Because both the CI and CD tools must be installed in the application cluster, integrated tools are more difficult to set up in a pull fashion.

The standalone and Kubernetes-specific tools produced the greatest outcomes based on the metrics utilized in this research. However, integrated and generic tools may offer additional advantages, such as being quicker to set up or having plugins to facilitate the integration process.

Conclusion

Users may specify and create their software environments in containers and virtual machines and then execute them on top of multiple resources in a portable, repeatable manner. This article provided an in-depth examination of commonly used containerization technologies and their key characteristics. Moreover, I illustrated and discussed various aspects of application domains to define container architecture for computing systems. Furthermore, the research has proven that understanding the capabilities and methodologies available for a specific containers-based solution, as well as the characteristics of workloads, is critical for optimizing systems. The container technique is now at the core of contemporary computing infrastructure because it eliminates various issues associated with sophisticated execution environment requirements that are often in conflict with other components of application operations. Containers have been embraced by various efforts and are becoming a standard technology, such as Cloud Native and Dev/Ops. Containers allow for the creation of scalable architectures built of a large number of services (microservices). IT businesses such as Google, Microsoft, Netflix, and others already depend on container technology in their production environments.

There are no established comparison criteria to assess alternative CI / CD solutions in my cluster deployment since research on CI / CD tools and pipelines is few, and there are few big studies that compare different tools. Although the deployment time utilized in this experiment is useful, other considerations may be more significant in selecting which CI/CD systems to employ.

More research is required to determine which criteria and characteristics are most relevant to consider when selecting tools to employ. More studies are also necessary to evaluate pipeline design security and how firms should pick between different CI / CD methodologies, such as a push or pull. Organizations who support the pull-style and GitOps pipeline approaches say that there are security advantages, but this has to be verified and explored more.

Other tools may have additional characteristics that make them more appropriate for certain areas or jobs. More research on similar instruments and their merits and limitations might be conducted. This might include tool functionality that is not completely integrated with the CI/CD process. For example, to automatically prune obsolete application installations and monitor for changes in the running application.

Another area that may need more research is how the different tools react when anything goes wrong in the pipelines. For example, if a new software deployment has defects, there may be a simple way to return to prior versions of the application to guarantee high availability.

Acknowledgments

Thanks to Google Cloud Platform for providing 90 days free service trial. That helped me in setting up infrastructure for conducting the experiment.

References

Adam Debbiche, M. D. R. B. S. (2015). "Challenges When Adopting Continuous Integration: A Case Study. Product-Focused Software Process Improvement, 17-32.

Beltre, A. M., Saha, P., Govindaraju, M., Younge A., & Grant, R. E. (2019). Enabling HPC Workloads on Cloud Infrastructure Using Kubernetes Container Orchestration Mechanisms. 2019 IEEE/ACM International Workshop on Containers and New Orchestration Paradigms for Isolated Environments in HPC (CANOPIE-HPC), 11-20.

Campeanu, G. (2018). A mapping study on microservice architectures of Internet of Things and cloud computing solutions. 2018 7th Mediterranean Conference on Embedded Computing (MECO), 1-4.

Chiang, E. (2022). Containers from Scratch. Available:
https://ericchiang.github.io/post/containers-from-scratch/#container-file-system.

Inc, D. (2022). The Docker platform. Available: https://docs.docker.com/.

Ltd, C. (2022). Container and virtualization tools. Available: https://linuxcontainers.org/.

MinSu Chae, H. L. K. L. (2017). "A performance comparison of linux containers and virtual machines using Docker and KVM," *Cluster Computing,* p. 1765–1775.

Openvz. (2022). Open source container-based virtualization for Linux. Available: https://openvz.org/.

Pahl, C., Brogi, A., Soldani H., & Jamshidi, P. (2017). Cloud Container Technologies: A State-of-the-Art Review. IEEE Transactions on Cloud Computing, 677-692.

Stef Verreydt, E. H. B. (2019). "Leveraging Kubernetes for adaptive and cost-efficient resource management," *Association for Computing Machinery,* p. 37–42, 2019.

Sylabs. (2022). Singularity. Available: https://www.sylabs.io/docs/.

T. a. s. foundation. (2022). Apache Mesos. Available: https://mesos.apache.org/..

Tarek Menouer, P. D. (2019). Containers scheduling consolidation approach for cloud computing. Pervasive Systems, Algorithms and Networks, 178-192.

uDocker. (2022). uDocker. Available: https://github.com/indigo-dc/udocker.

Zhang, Q., Liu, L., Pu, C., Dou, Q., Wu, L., & Zhou, W. (2018). A comparative study of containers and virtual machines in big data environment. 2018 IEEE 11th International Conference on Cloud Computing, 178-185.

Volume 6 №2 2022, 75-84
DOI: 10.5782/2520-613.202.6.2.75

Security Analysis While Transitioning from Monolithic Applications to Microservices

Tural Sadigov

Graduate School of Science, Art and Technology,
Khazar University, Azerbaijan
Corresponding author: tural.sadiqov@khazar.org

Abstract

Microservice architectures have evolved as an enticing alternative to more typical monolithic software application approaches. Microservices give various benefits in terms of code base knowledge, deployment, testability, and scalability. As the information technology (IT) industry expands, it makes sense for IT behemoths to adopt the microservice, but new software solutions creates new security vulnerabilities, as the technology is young and the faults have not been adequately mapped out. Authentication and authorization are key components of any software with a significant number of users. However, owing to the lack of microservice research, which derives from their relatively young, there are no specified design standards for how authentication and authorization are best performed in a microservice.

This thesis analizes existing microservice in order to safeguard it using a security design pattern for authentication and authorization. Different security patterns were assessed and different degrees of security helped in identifying an acceptable security vs. performance trade-off.The objective was to strengthen the patterns' validity as known security patterns. Another purpose was to establish a security pattern that was suitable for the microservice.

Introduction

Usually in backend applications, often referred as monolithic applications, the code is developed and deployed as a whole, single project. But in microservices, this artifact is divided into multiple small applications or services that can be developed, tested and deployed independently from each other. Today most companies are trying to shift from monolithic to microservices because of its effective approach to development, but in the current microservices DevOps environment, there are new and evolving challenges for developers and teams to consider on top of the more traditional ones.

As it becomes much more difficult to maintain a microservices setup than a monolithic one, each microservice setup may evolve from a wide variety of frameworks and coding languages and this brings new challenges to the development environment and security is the top one to consider.

Authorization and authentication are the foundations of security for every application, monolithic or not. The MSA offers potential improvements at many stages of the software development process, but it also creates new challenges. Unsurprisingly, when multiple components of an MSA need to communicate with one another, securing requests to and from as well as within a microservice becomes a much more difficult task than it is for a monolithic application, where authentication and authorization can be done once when accessing the application.

The main goal of the research is look into the security patterns that may be used to organize authentication and authorization in a microservice to implement a security solution.

Background

The concept of microservices has been known since the early 2000s, however the name "microservices" only appeared in particular situations in the early 2010s (Richardson, 2019). However, others claim it was coined as late as 2014 (Zimmermann, 2017). Still, the concept of microservices is very new when compared to other software development methods and architectures. Microservices can be thought as different small applications independent of each other (in reality no services are fully independent though), so it makes the whole application more reliable, because if some of the services is not available, it does not affect the others. Since small services becomes large in number, Kubernetes can be used to orchestrate the whole application by running each service in individual virtual machines. Figure 1 describes MSA and communication flow of microservices.

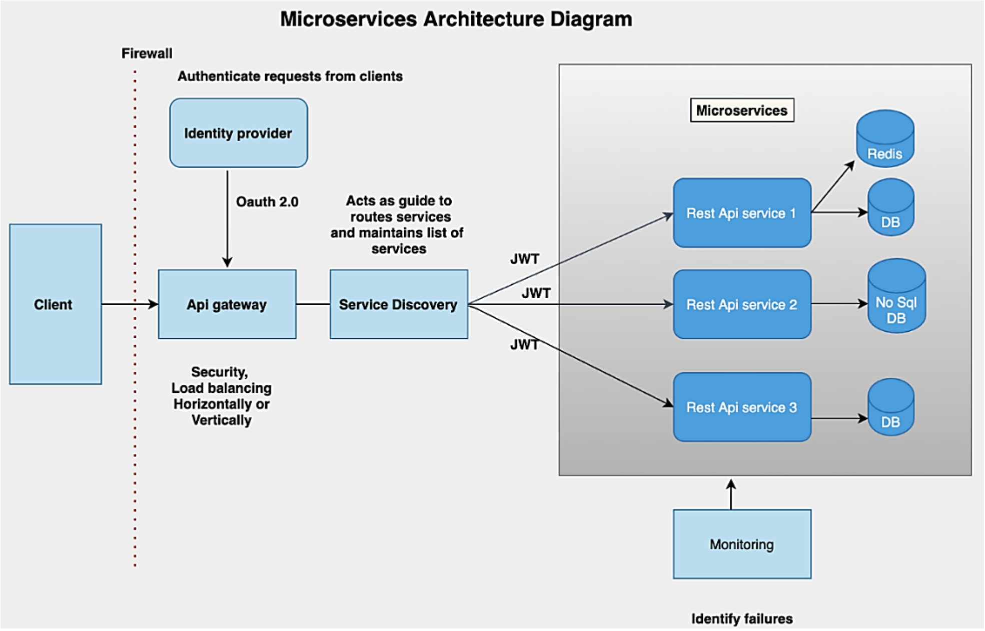


Figure 1. Graph displaying a microservice and communication flows

So, as seen in the figure, each service can communicate with each other through API gateway. API gateway is a middleman between services and external client requests.

IPC is the protocol used to communicate between services and the API gateway. Hypertext Transfer Protocol (HTTP) and, by extension, Hypertext Transfer Protocol Secure are two implementations of such a mechanism (HTTPS). Because the HTTP protocol is stateless, there is no built-in mechanism for a server to remember any interactions with a client. In order to safeguard resources, future HTTP requests must remember a previously authenticated and approved client in case they need to be reauthenticated and reauthorized. To preserve the verified status, a token comprising user information and permissions may be supplied with each subsequent request. The JSON Web Token is one such standard (JWT). It may be used to transfer information in the form of a JSON object, which can then be signed or encrypted to guarantee integrity or confidentiality. In their paper (Xu, 2019), Rongxu Xu, Wenquan Jin, and Dohyeun Kim propose how an MSA may be protected using JWT. It is anticipated in this technique that an API Gateway intercepts all requests so that an authorization server may give JWTs for future requests to sensitive data services.

Moreover, OAuth 2.0 is used to secure the microservices. OAuth is an open standard which minimizes the number of permission stages by requesting a user to give a service authorization to other services holding sensitive data (OAuth, 2022). Today, the OAuth protocol is regarded outdated since its successor version accomplishes the same function but has minimal technical similarities (Hardt, 2012).

Authentication and authorization in a Kubernetes microservice

This chapter dives into the technical aspects of the authentication and authorization components' implementation. The authentication and authorization service (abbreviated auth-service) were required, along with a Redis store, gateways, and some example services (which emulates the business logic of a microservice).

Kubernetes is used to implement the microservice. Because all traffic inside the cluster is inaccessible from the outside, an ingress controller allows communication into the cluster from the outside (i.e., the internet or the local network in which the cluster is implemented). The microservice is deployed in a Kubernetes cluster that employs an NGINX ingress controller variation. The service that is to be exposed (in this case, an edge level gateway) will be assigned an ingress object (which defines how the ingress controller should route traffic related to the service) that specifies a reachable URL if the requests come from a device connected to the internet or an internal network.

The auth-service may authenticate users by interacting with an LDAP server, which also replies with a user's roles. By providing a JWT to a logged-in user, this token may be simply utilized to both identify a user for authentication and locate the related roles of this user for authorization.

The services are implemented in Spring Boot and provide straightforward REST endpoints that may either return a value directly or trigger another call to another microservice to get further resources.

The implementation of the three distinct authentication and authorization security patterns—edge level (3.1), service group (3.2), and service level (3.3) gateway patterns—is covered in depth in this section

Edge level gateway pattern

The simplest of the three security patterns is the edge level gateway pattern. Despite offering the least level of protection, it was shown to be the most popular method of establishing authentication and authorization in a microservice. As a result, it may also be used as a benchmark against which to evaluate the other two designs (as both are more complex and provides a higher level of security). Without initially submitting a request to the edge API gateway, none of the services offered by the microservice are accessible to clients or servers outside of the microservice.

Service group gateway pattern

The service group gateway design extends the edge level gateway pattern by grouping together services that need the same degree of access to access. The auth-service also handles this authorization. An example would be a collection of services that all need the same role to access. This subset of services, just like the edge gateway, would be secured by an additional gateway that is likewise located behind the edge gateway. This adds an extra layer of security. Another feature that is comparable to the edge level gateway approach is that communication that does not need to transit through a gateway is not subject to authentication or authorization. This implies that services behind the same internal gateway may send requests without being authenticated or authorized

Service level gateway pattern

The service level gateway pattern is the third and final pattern examined in this thesis. This security architecture necessitates that each service has its own gateway that secures it through authentication and permission. Because it is not always possible to have one role linked with one service, some or all of the gateways might have the same role necessary to provide access to the protected service. While it may seem that role verification is unneeded when two service-gateway pairs interact and require the same role to access, it really offers a unique type of security. Because all connections inside the microservice between its services need a security check, services controlled by a bad actor cannot reach any other conceivable targets without first undergoing authentication and authorization

The testing framework

This chapter discusses the two types of tests that are conducted. Section 4.1 discusses the research methodology that influenced the testing technique. Section 4.2 covers the security testing procedure. Finally, section 4.3 goes into depth on the load testing that produced the most of the findings.

Research process for load testing

The load testing software (JMeter) was used to analyze the findings and provide numbers such as the median and average. Additional Python programs were used to analyze the data. The Python programs employed linear regression to display the data trend such that a forecast for how larger loads than what was tested may result could be made. The scripts were also used to generate visual representations of the data, such as scatter plots and box plots, in order to provide a broader variety of data.

Security tests

- The security tests examined four situations that may occur during normal use. These were:
- Using the microservice with valid credentials
- Omitting the authorization token
- Sending an invalid token
- Sending a valid token to a user who does not have the appropriate role to access the requested resource

Load testing

Using JMeter load testing, the three distinct security schemes' effects on response times were evaluated. The real loads and service should preferably be as similar to the tested service as feasible for accurate load testing, meaning that the load generator's delay should be as low as possible. As a result, a Spring Boot application specifically designed for testing was installed on the machine being tested, making all communication within the cluster local. Figure 2 shows a graphic depiction of the location of the JMeter-running service and the flow of requests to the microservice.

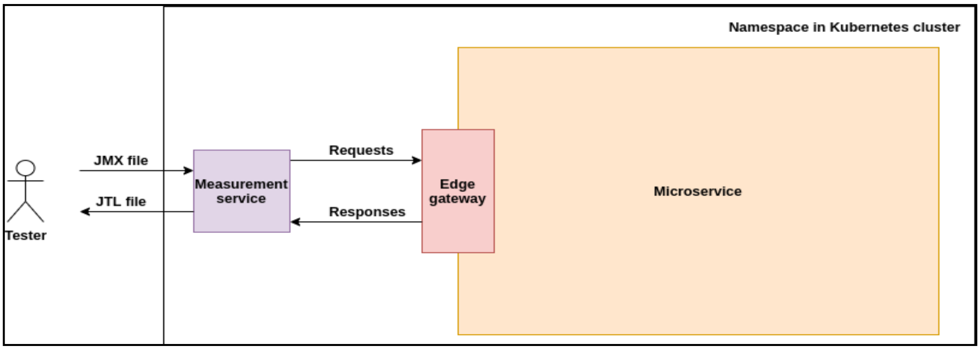


Figure 2. Graph displaying the tester and measurement service in relation to the microservice being tested

Results and disscussions

Three situations were evaluated to see whether the auth-service and gateways provide the necessary protection.

It was crucial to make sure that the security solution also rejected requests that were found invalid since the load testing mostly focused on valid requests. The situations listed in Section 4.2 were put to the test to confirm this. Since the token is legitimate and the corresponding user has all necessary responsibilities for permission. When a no or incorrect token is received, an error message with a cause is sent.

The findings of the testing are shown in this section using box plots and scatter charts. As the number of threads rises across all security types, they show a consistent rise in response times. What is particularly notable is that as the load grows, the gap in reaction times between the security patterns widens. It seems that the service level gateway pattern is more significantly impacted than the service group gateway design. Comparing the service level gateway pattern to the other two security patterns, this shows a quicker increase rate in response times.

For simpler comparison, Table 5.1 presents all median values. Response times for the edge level gateway pattern increased from 1225 milliseconds for a single thread to 2362 milliseconds for 2000 threads. This represents a growth of roughly 93%. The comparable increase for the service group gateway pattern was 1244 milliseconds to 3086 milliseconds, or a 148% increase. Last but not least, the increase for the service level gateway pattern was 1260ms to 4367ms, or a 247% percentage increase. There was a 31% rise from edge level to service group, an 85% increase from edge level to service level, and lastly a 42% increase from service group to service level when comparing the percentage increases of the 2000 threads load across the three security models. All percentages were rounded to the nearest integer, as you will see.

Table 1. Median response times for the security patterns

Threads	Edge level gateway response time (ms)	Service group gateway response time (ms)	Service level gateway response time (ms)
1	1225	1244	1260
100	1243	1284	1315
200	1248	1306	1332
300	1272	1323	1371
400	1370	1487	1473
500	1516	1719	2013
600	1555	1687	2049
700	1644	1815	2094
800	1697	1765	2536

900	1715	2007	2639
1000	1752	2115	2687
1100	1854	2274	2740
1200	1937	2201	3065
1300	2006	2124	3292
1400	2026	2526	3447
1500	2047	2595	3388
1600	2067	2625	3750
1700	2207	2374	3892
1800	2181	2901	4017
1900	2292	2925	4145
2000	2362	3086	4367

Conclusion

It is necessary to establish what is considered a favorable outcome before recommending a security pattern. The natural logic would be to seek a balance between strong security and fast reaction times. However, even if a security pattern has longer reaction times than the other patterns, it may still be considered to be doing well in comparison to a fair benchmark. When seeking to secure a system, it is frequently preferable to utilize technology that has been shown to be capable of delivering the desired security. This concept guided the selection of technologies for the security patterns, which resulted in the use of JWT, role-based authorization, and API gateways.

In terms of performance against security, the service group gateway pattern is the best option. There is no reason not to propose the security pattern with the highest level of protection, as the findings indicate that the impact on performance will be almost equivalent to that of the design with the lowest level of securityAs a result, the service level gateway design is the preferred security pattern for the project's microservice.

Acknowledgments

The load tests in this thesis were inspired by the performance testing in Akhan Akbulut and Harry G. Perros's work (2019). Also The recommended solution in the paper was one of the key sources of inspiration for what would become the most basic layer of security: the edge level gateway design. Finally, great thanks to the Google, which provided free 3 month trial to use all services that Gcloud offers.

References

Akbulut, A., Harry G. (2019). Performance Analysis of Microservice Design Patterns, IEEE Internet Computing. Vol. 23, pp. 25, 69.

Hardt, D. (2012). The OAuth 2.0 Authorization Framework. Available: http://www.rfc-editor.org/rfc/ rfc6749.txt.

OAuth Core 1.0 Revision A. (2022). Available: https://oauth.net/core/1.0a/.

Richardson, S. (2019). Microservice Patterns: With Examples in Java. Manning Publications, pp. 2, 10, 12, 13, 18, 2019.

Xu, W. J. (2019). Microservice Security Agent Based On API Gateway in Edge Computing. Sensors, 19 (22)

Zimmermann, O. (2017). Microservices tenets. Computer Science-Research and Development. vol. 32, no. 3, p. 10

Energy Management in the Smart Home

Mammad Mammadzada

Faculty of Engineering and Applied Sciences,
Khazar University, Azerbaijan
Email: mammad.mammadzada@outlook.com

Abstract

The issues of energy supply of a smart home (building) are considered, taking into account alternative power sources. The components of the energy supply system and the life support subsystem of a smart building, the functions assigned to the energy management system are analyzed, the feasibility of developing such a system to save electricity and solar in the building is shown. The structure of the building automation system is proposed, taking into account the characteristics of alternative power sources and possible restrictions on their use. Approaches to energy management in a smart building using the theory of fuzzy sets and features of such intelligent control are shown.

Keywords: smart home, alternative power sources, solar power plant, fuzzy logic

Introduction

A smart home or a smart building is a complex of special and household equipment that ensures the vital activity of people living or working in this house. At the time, automating the life support processes of a smart home involves the development and implementation of subsystems for monitoring and controlling all equipment and security alarms, which naturally requires the presence of a power supply subsystem, the functions of which are to provide energy to the entire complex and optimize energy management in order to save energy.

Energy management in the smart home

There are a number of concepts for constructing an energy supply subsystem (Danilov et al., 2006), which differ in the use of energy from alternative power sources, mainly solar energy and wind energy. The use of the energy of thermal waters has a number of limitations due to the location of the object - of the smart home. Each of the concepts has the right to exist on the basis of economic, and in some cases special - strategic considerations.

Currently, due to the crisis in the fuel and energy sector, the increase in energy prices around the world, alternative power sources are increasingly being introduced (Stychinsky & Voropai 2016). Solar power plants for individual homes and individual buildings are widely used. In the latter, in addition to roofs, they also try to use facades, which increases the effective area of solar panels. However, with the development of alternative power sources and, in particular, solar energy, the problem of economical consumption of the collected energy, optimization of the operation of the entire smart building complex urgently arises. This problem has a number of particularities.

In individual separate smart houses, the problem is solved relatively easily, based on the peculiarities of using a smart house and on the preferences of its owner. Things are different in smart buildings, due to the nature of the work of different tenants, if the building is intended for business. Given these circumstances, the approach to developing a power supply subsystem with an optimal structure and advanced functions depends on the configuration of the solar power plant system and the devices and equipment used in it. It should be noted that for the purpose of universality or unification, energy management or energy management in such a smart building should cover all possible power sources connected to the power supply system: a solar power plant, a diesel generator, a wind power plant, backup batteries, as well as a solar collector that provides water heating, and finally, all the equipment and systems in the building.

Based on this, the structure of the smart home automation system can be represented as shown in Figure 1.

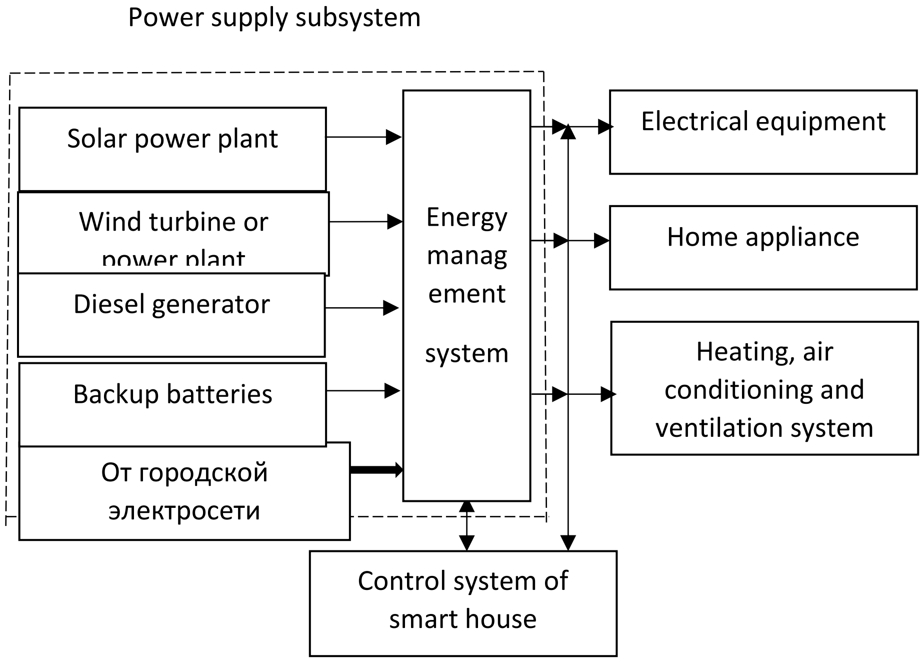

Power supply subsystem
Solar power plant
Wind turbine or power plant
Diesel generator
Backup batteries
От городской электросети
Energy manag ement system
Electrical equipment
Home appliance
Heating, air conditioning and ventilation system
Control system of smart house

Figure1. The structure of a smart home automation system

The energy management system, together with the smart building control system, ensures timely switching on and off of power supplies and equipment, regulation of the main energy-intensive processes in the building, software shutdown of loads and shutdown of high-power equipment if necessary and in emergency cases (Dyakonov, et al., 2008). In addition, this system controls the mode of transmitting of excess energy into the city network during the hours set for this mode, which increases the efficiency of the solar power plant and the smart building automation system itself.

As the experience of operating solar power plants on photovoltaic solar panels shows, in order to increase the efficiency of a solar power plant, it is necessary to solve the issues of choosing the solar panels themselves and the corresponding equipment, optimizing the power take-off from solar panels depending on the time of day and weather, optimizing charge and discharge control of batteries.

The principle of directing a battery of solar panels to the sun is given in (Arzhanov et al., 2014), which sets out the basic conditions and limitations for automatic control of a solar power plant. In (Dontsov et al., 2015), two algorithms for the solar battery controller are proposed, according to which the controller operates in the modes of charging and searching for extreme power. In (Zubova & Rudykh 2018), the problem of choosing the membership functions of fuzzy sets for optimizing the control systems of a wind turbine is considered.

An analysis of existing systems and control algorithms allows us to conclude that the most appropriate approach to managing the energy of a solar power plant as part of a smart building power supply system is the use of fuzzy logic, both in the control and management system of the station, and in smart building energy management.

Thus, for fuzzy energy management algorithms when using a solar electric system, linguistic variables to be controlled and managed and their corresponding membership functions should be selected. As such values, we choose the power of solar panels and load, the range of charge and discharge of the battery, the limits for changing the coordinates of the solar battery, the limits for changing the temperature of air and panels, the duration of operation of various equipment in a smart building, etc. For each of the selected values, fuzzy rules of the type "if ..., then ..." and the corresponding linguistic terms such as "very small", "small", "medium", "large", "very large" or "very low", "low", "very low", "low average", "average", "very average", "high", "high", very high", etc. should be develop.

Obviously, to use the developed rules for each of the processes, the well-known scheme of fuzzyfication, logical inference and knowledge base, defuzzyfication, is used, which is given, for example, in (Shtovba, 2003), which also describes the modeling of certain processes using the Simulink package of the MatLAB program (Dontsov et al., 2015). The regulation of the main process parameters is carried out on the basis of a PID controller with fuzzy settings. Modeling and solving problems of this type can also be performed in the Fuzzy Toolbox in MatLAB. As the practice of using fuzzy logic shows, in order to successfully solve the problem, it is necessary to correctly choose the membership functions of each value and the fuzzy model to identify the process under consideration.

Conclusions

The efficiency of a smart building automation system depends on many factors, one of which is the availability of high-quality electrical energy. This circumstance urgently requires the development and implementation of an energy supply subsystem, the operation of which, together with the control system and other subsystems of a smart building, will allow solving the issues of energy management and saving solar, electric and thermal energy at a higher level.

The widespread introduction of alternative and renewable power sources, especially solar panels for powering smart buildings and houses, also allows reducing hydrocarbon emissions in the long term, thereby improving the environment.

The use of fuzzy set theory and fuzzy logic to manage and control various processes in a smart building improves the quality of management of all processes, including the power supply subsystem, by developing an adequate model of processes that in most cases cannot be identified by deterministic models.

Reference

Arzhanov, K.V., & Arzhanova, A.V. (2014). Automatic control of a solar power plant. Materials of the XX All-Russian Scientific and Technical Conference "Energy: Efficiency, Reliability, Safety". - Tomsk: Publishing House of the Tomsk Polytechnic University, 312-314.

Danilov, N. I., Shchelokov, & Ya. M. (2006). Fundamentals of energy saving. Ekaterinburg, UPI, p. 564.

Dontsov, O. A., Ivanchura, V. I., Krasnobaev, Yu. V. (2015). Solar controller with extreme regulation based on fuzzy logic. Journal of Siberian Federal University. Engineering & Technologies. Issue 6, 786-794.

Dyakonov, V.P. (2008). Simulink: Tutorial. DMK, Press, p. 784.

Shtovba, S.D. (2003). Identification of non-linear dependencies using fuzzy inference in the MATLAB system. Mathematics in applications. No. 2 (2)/2003, pp. 9-15.

Stychinsky Z.A., & Voropai N.I. (2016). Renewable Energy Sources: Theoretical Foundations, Technologies, Specifications, Economics. Magdeburg, p.210.

Zubova N.V., & Rudykh V.D. (2018). Search for optimal fuzzy set membership functions for optimizing wind turbine control systems.