Dynamic User-Guided Evolutionary System for Generative Music

Ievgen Fedorchenko[1,*,†], Andrii Oliinyk[1,†], Tetiana Fedoronchak[1,†] and Maksym Chornobuk[1,†]

[1] *National University Zaporizhzhia Polytechnic, Zaporizhzhia 69011, Ukraine*

Abstract

This article presents a music generation system based on a modified interactive genetic algorithm (IGA) with dynamic user engagement tracking. The developed approach allows the creation of monophonic MIDI compositions based on user feedback. The proposed model automatically adjusts the probabilities of mutations and injections in the population depending on user engagement level, there-by reducing user fatigue and accelerating convergence toward acceptable results. Conducted experiments with five volunteers showed that the system is able to generate compositions rated by users as attractive in an average of 4.6 iterations, while the generation time of a new generation is less than 1 ms. The system has a simple interface based on .NET MAUI and allows exporting results in MIDI format. The proposed solution combines high speed with the possibility of fine-tuning the generation parameters, which makes it promising for creating interactive music products in real time.

Keywords

genetic algorithm, music generation, MIDI

1. Introduction

Dynamic music generation has been known and used in various digital products since the end of the 20th century. Typical examples of such use are computer games Spore, No Man's Sky, Mini Metro. The use of generated music reduces costs otherwise spent on hiring a composer. Such music also enhances the user experience by introducing novelty during product use [1].

In recent years, music generation technologies based on artificial neural networks have gained particularly great popularity. Dozens of models allow users to generate high-quality instrumental and vocal music based on the user's description. For example, MuseNet from OpenAI [2] is widely used, MusicLM from Google [3]. Such tracks are almost indistinguishable from professional recordings made in the studio. However, such technologies still require significant amounts of computing resources, including RAM and CPU time. They are usually provided on a paid basis or require large computing resources and cannot run in real time on end-user computers. The use of such technologies is currently limited to generating music in advance, and real-time generation requires other solutions [4].

One solution that does not require significant amounts of computation is genetic algorithms, in particular interactive genetic algorithms (IGAs). Such methods are a modification of the standard genetic algorithm in cases where the fitness function cannot be defined formally. User input is then used instead of a formally defined fitness function, driving the evolutionary process. These methods

Information Technology and Implementation (IT&I-2025), November 20-21, 2025, Kyiv, Ukraine

* Corresponding author.

† These authors contributed equally.

evg.fedorchenko@gmail.com (I. Fedorchenko); olejnikaa@gmail.com (A. Oliinyk); t.fedoronchak@gmail.com (T. Fedoronchak); chornobuk.maksym@gmail.com (M. Chornobuk)

0000-0003-1605-8066 (I. Fedorchenko); 0000-0002-6740-6078 (A. Oliinyk); 0000-0001-6238-1177 (T. Fedoronchak); 0000-0003-3200-7306 (M. Chornobuk)

are widely used in the field of music generation, where the subjective attractiveness of the same track depends on the aesthetic preferences of a particular user or group of users [5].

Several existing systems demonstrate the potential of IGAs for music generation. For example, GenJam [6] uses user evaluation for generating jazz solos; Darwin-Tunes [7] uses similar evolutionary approaches for music generation. These systems typically encode musical properties (e.g. pitch, rhythm, harmony) as genomes, apply selection and mutation, and rely on user feedback to evolve increasingly attractive compositions. However, such models often suffer from limitations such as high user fatigue, slow convergence, or fixed mutation strategies that do not adapt to listener engagement.

This paper examines an improved adaptive interactive genetic algorithm designed to generate music in the MIDI format. The developed model is characterized by simplicity and high speed but differs from a number of other models. In particular, the developed system monitors user feedback in different generations and evaluates user engagement based on statistical indicators calculated from the flow of user ratings. The calculated indicator dynamically adjusts the probability of mutations and injections in the population in order to reduce user fatigue, accelerate convergence, and maintain interest throughout the entire process of using the system.

2. Overview of existing systems

The article [6] describes GenJam, one of the early systems for generating music using an interactive genetic algorithm. GenJam was created for generating jazz solos. The system can work in one of three modes:

- Learning mode. Random compositions are generated; the user evaluates them. The evolutionary process has not started.
- Demo mode. The best of the previously generated compositions is played.
- Evolution mode. Genetic operations are used for the dynamic generation of a new population of musical compositions.

GenJam uses a two-level genetic coding scheme to represent compositions. Musical content is hierarchically structured into phrases and measures, each encoded as binary chromosomes of fixed length. This presentation combines rhythm and pitch. Each bar corresponds to a 32-bit chromosome representing eight consecutive eighth-note positions in a 4/4-time signature. Each position is encoded by a 4-bit event: Rest, Hold, New Note.

The system is quite limited: the compositions do not differ in speed (BPM value), they use predetermined sequences of chords.

The GP-Music system described in the article [8] is more modern. This system has several distinctive features. The leaves of the trees represent individual musical notes, pseudo-chords or pauses. Internal nodes represent musical transformations or operations applied to sequences. Thus, each node of the tree returns a sequence of notes, and the final composition is returned by the root of the tree. Such a structure has its advantages: it is possible to carry out mutation and crossover operations conveniently. Another important feature is the use of a simple neural network that learns during the phase of feedback from the user and then is able to give its own evaluations to compositions, reducing the burden on the user. This method is called the Surrogate Fitness Function and has great potential, although in real conditions it faces the problem of not having enough resources for a complex model capable of qualitatively simulating user evaluations. Nevertheless, the system is quite limited and allows the creation of only compact, monophonic musical compositions. A big drawback is the fact that all the notes in the generated compositions are of the same length: this limits the system significantly, preventing the creation of complex compositions similar to real music.

Another approach is described in the article [7]. The authors developed the DarwinTunes system based on an interactive genetic algorithm that did not use a surrogate fitness function, but aggregated feedback from more than 6,000 users, using their aggregated ratings as a fitness function for small pieces of music. The proposed system used a population of tree-like digital genomes, each of which described a program that generated a small, looping piece of music 8 seconds long.

3. Problem statement

Musical Digital Interface (MIDI) is a standard developed in 1983 as a result of a collaboration between leading manufacturers of electronic instruments. The standard provides interoperability between equipment from different manufacturers and pro-vides a digital abstraction layer over analog and digital sound generation processes.

The Standard MIDI File (SMF) format allows compact storage of data about musical compositions. Unlike classic audio files that store the data about sound directly, MIDI files store sequences of discrete musical events that are then converted into music using software or hardware solutions. Each MIDI file consists of a header and one or more tracks. The header defines global parameters such as file type, number of tracks, and tempo. Each track consists of a sequence of time-ordered events, each of which has a timestamp relative to the previous event. This allows for precise placement of notes and other musical events throughout the track.

The most common are the Note On and Note Off event types, which together determine when a note starts and ends. Each note playback event is characterized by a MIDI channel (0–15), note pitch (0–127), note velocity (0–127). These parameters describe both the horizontal (time-based) and vertical (pitch-based) structure of a musical composition [9].

The simplicity and widespread nature of the MIDI format justify its choice as the basis for the system under development. Considering the need to create simple monophonic compositions, as well as the absence of the need to change the velocity of notes, it is possible to formally describe a musical composition as follows:

$$s = (BPM, \{e_1, e_2, \dots, e_N\}), \tag{1}$$

$$e_i = (p_i, d_i\ )\ or\ e_i = (\emptyset, d_i), \tag{2}$$

$$p_i \in P, d_i \in D, \tag{3}$$

where P is the set of allowed pitch values; D is the set of allowed values of the length of the note.

Then, formally, the task of developing a music generation system is reduced to finding such a function M:

$$S_{(n+1)} = M(S_n, Y_n), \tag{4}$$

$$Y_n = \{y_1, y_2, \dots, y_x\}, \tag{5}$$

$$S_n = \{s_1, s_2, \dots, s_x\}, \tag{6}$$

$$s \in T, \tag{7}$$

where T is the set of all possible musical compositions, considering the limitations of the system.

The most difficult challenge during system development is finding such a function M, which is able to quickly generate compositions that will receive a high rating from the user.

4. Development of a modified interactive genetic algorithm

A system was developed based on a modified interactive genetic algorithm capable of generating and playing small musical compositions using the MIDI technology described above. After the generation of the next generation S_n the system offers the user to rate each of the compositions on a 10-point scale. Obtained values Y_n are used as a fitness function of compositions.

4.1. Compositions encoding

The system encodes each composition using a genome consisting of discrete, quantitative, and sequential genes. Discrete genes encode the scale and root note. Quantitative genes encode the number of beats per minute. Sequential genes encode harmonic and rhythmic sequences and arpeggio types in bars. Such data is enough to encode compositions that are not limited to one genre of music, as in the GenJam system. The system is flexible and customizable, because it allows the configuration of limit values for quantitative genes and possible values for discrete genes.

4.2. Workflow

The system has the following operating cycle:

1. Generate the initial population S_1 random musical compositions (individuals). Value x (generation size) is established in 5 empirically.
2. Show the user the current population. Get the rating value y_i for each individual s_i from the user.
3. Assess user engagement δ.
4. Generate a new population of x individuals. Each individual in the population S_n will be the offspring of two individuals from the previous population $S_{(n-1)}$ with probability 1-α or the result of injection with probability α.
5. Carry out mutations in the new population S_n. The probability of mutation of each gene is β.
6. Return to step 2.

During reproduction, the offspring receives a random combination of the genes of its parents. Each of the genes is independently generated on the basis of parental genes. Genes encoding quantitative parameters are chosen randomly between parental values. Genes encoding discrete parameters randomly take one of the parental values. Genes encoding sequence parameters are generated based on the random combination of the sequences of the parameters of the parents. Parents are chosen randomly as follows:

$$P(i) = \frac{y_i}{\sum_{j=0}^{n-1} y_j}, \tag{8}$$

where P(i) – the probability of choosing the i-th individual as a parent, and y_i is the i-th rating received from the user, $y_i \in [0,1]$.

A feature of the developed modified system is a change in the probability of injections and mutations based on the parameter user interest. This technique is important for a system performing a task such as music generation, where formal evaluation of the quality of the generated population is not possible. Using this parameter allows dynamic change of the rate of mutations and injections in the population of music compositions, decreasing the diversity proportionally to user interest.

Parameter δ takes a value from 0 to 1 and is calculated as follows:

$$\delta = \frac{\sigma}{0.5} * r_{max}, \tag{9}$$

$$r_{max} = max\,(r_1, r_2, \dots r_N), \tag{10}$$

where r_i – the i-th rating received from the user, in the range 0–1, σ is the mean square deviation of the values r.

Probability of injection α is calculated as follows:

$$\alpha = \alpha_{min} + (\alpha_{max} - \alpha_{min}) * (1 - \delta), \tag{11}$$

where α_{max} – the maximum, α_{min} – the minimum injection probabilities specified as system parameters.

Probability of mutation β is calculated as follows:

$$\beta = \beta_{min} + (\beta_{max} - \beta_{min}) * (1 - \delta), \tag{12}$$

where β_{max} – the maximum, β_{min} – the minimum injection probabilities specified as system parameters.

Crossover is implemented as follows. For discrete genes:

$$g_k^{(child)} = \begin{cases} g_k^{(parent1)} \\ g_k^{(parent2)} \end{cases} \tag{13}$$

For quantitative genes:

$$g_k^{(child)} = g_k^{(parent1)} + \left(g_k^{(parent2)} - g_k^{(parent1)}\right) * U(0,1) \tag{14}$$

For sequential genes:

$$L_{child} = U(min(L_1, L_2), max(L_1, L_2)), \tag{15}$$

$$g_{k,i}^{(child)} = \begin{cases} g_{k,i}^{(parent1)}, if\ i \le L_1 \text{ i } (U(0,1) < 0.5\ or\ i > L_2 \\ g_{k,i}^{(parent2)}, if\ i \le L_2 \text{ i } (U(0,1) < 0.5\ or\ i > L_1 \end{cases}, \tag{16}$$

$g_k^{\square}$ – value of gene k for generated individual. $g_{k,i}^{(parent1)}$ and $g_{k,i}^{(parent2)}$ – values of gene k for the first and second parent individuals respectively, U(x,y) is a random variable with a uniform distribution on the interval [x, y]. L_{child} – is the length of the generated sequenced gene, L_1, L_2 – lengths of this gene for the first and second parent, respectively. $g_{k,i}^{\square}$ is the i-th element of the k-th sequential gene.

This crossover implementation correctly handles different genes according to their nature. For example, the BPM (speed of composition) value for the generated individual will lie between the values of this gene in the parent individuals. And the chord sequence will be a combination of the chords used in the parent individuals.

5. Developed software

A simple and concise user interface based on the popular cross-platform .NET MAUI library was developed for the system. This allows using the developed system on Windows and MacOS platforms [10].

The system allows users to listen and evaluate each of the compositions generated in the current generation many times in an arbitrary order. Each of the compositions can be exported as a file in the ".mid" file format. The system displays the current calculated value of the user's interest in the form of a graphic indicator at the bottom of the window. Additional functionality is also provided in the form of statistics ex-port containing user ratings for each of the compositions in each generation.

6. Experiments

The developed system was tested using the help of five volunteers. The volunteers were instructed to use the system until one of the generated short compositions was subjectively pleasing to the volunteer.

Figure 1 shows an example of a composition generated by the system in the first generation, which the user rated with the minimum possible rating. The composition sounds like a random set of sounds and is musically unappealing. Fig. 2 shows the composition that was generated by the system at generation 5 and evaluated for the maximum number of points. The first composition is characterized by a sharp transition to the use of short sixteenth notes and pauses, multiple repetition of individual notes in the middle of the composition, lack of harmonic movement, but the second composition does not have such problems. In the second composition, there is a harmonious movement that alternates ups and downs several times, there is no sharp and unnatural use of short notes and pauses. There is no repetition of the same notes several times in a row, instead simple but effective techniques of creating tension for the listener and relieving it are used. In general, the second composition is perceived as more aesthetically pleasing due to greater musicality and structural coherence.

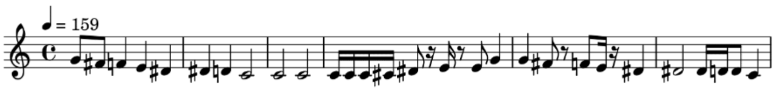

Figure 1: An example of a subjectively unattractive composition.

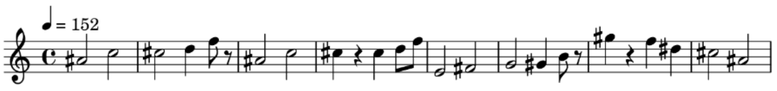

Figure 2: An example of a subjectively more attractive composition.

The results of the interaction of 5 volunteers with the system are shown in the graph in Figure 3. The graph shows the value of the maximum rating for compositions from each generation. The test results indicate the following properties of the system. Its disadvantages are the unpredictability of the results of the system. But the system also demonstrated the ability to generate an acceptable result quite quickly: the highest speed of achieving an acceptable result demonstrated by 5 volunteers is 9 iterations of the system. The average value is 4.6 iterations.

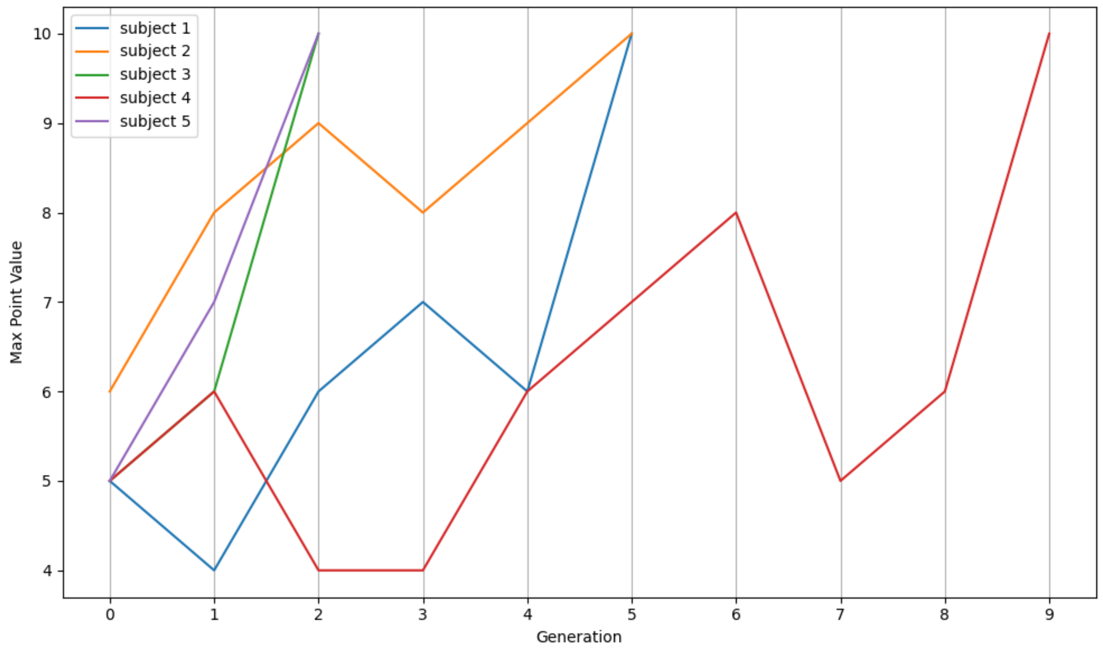

Figure 3: Results of interaction of 5 users with the system.

Testing of the developed software system was carried out on a computer with the Windows 11 operating system equipped with an Intel I7-12650H central processor. Table 1 and Figure 4 show the results of testing over 10 generations. Based on the results of the tests, it was established that the average time for the computing of the next generation on this hardware is less than 1 millisecond, which indicates the system's high speed and low resource consumption. The time spent on generating the next generation is imperceptible to real users of the system.

Table 1
System performance test results

Generation	Time spent, milliseconds
1	2,3868
2	0,3068
3	0,1194
4	0,119
5	0,0923
6	0,0747
7	0,0646
8	0,1081
9	0,0931
10	0,1471

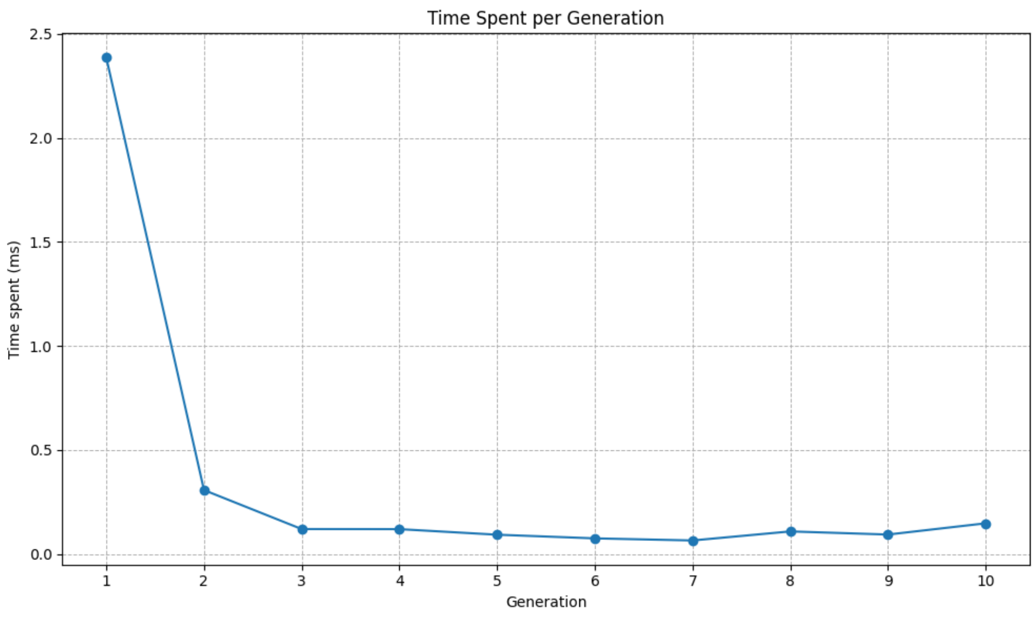

Figure 4: System performance test results.

7. Discussion of results

Table 2 shows a comparison of the developed system with the analogues discussed above.

Table 2
Comparison of the developed system with analogues

System	Type of music	Compositions encoding	Surrogate fitness function	User engagement tracking	Monophonic only
The proposed system	Arbitrary short	Complex genome, genes of three types	No	Yes	Yes
GenJam [6]	Jazz solos	Two-level hierarchy, limited number of 4-bit events per clock	No	No	Yes
GP-Music [8]	Short, same note lengths	Tree-like multi-level structure	Yes	No	Yes
DarwinTunes [7]	Short looped	Tree-like multi-level structure	No	No	Yes

While the developed system, like most models based on genetic encoding, is still restricted to generating only monophonic compositions, this limitation should be viewed in the broader context of its performance characteristics. For many practical applications: background music for games, simple generative soundscapes, educational tools, or interactive art installations, monophonic sound can be sufficient. More importantly, the system demonstrates several notable advantages that distinguish it from analogues.

First, its dynamic tracking of user engagement directly addresses one of the key drawbacks of interactive genetic algorithms: user fatigue. By continuously adapting mutation and injection probabilities, the system reduces repetitive or unproductive iterations, enabling users to reach satisfying results in fewer cycles. This approach can be seen as a lightweight alternative to surrogate fitness functions, which are computationally demanding and often difficult to generalize across users.

Second, the computational efficiency of the model makes it practical for real-time applications. Tests confirm that new generations can be produced in less than one millisecond, even on low-end hardware. Thus, the system can be used with software where performance is critical without requiring cloud-based processing or expensive hardware. For example, video games or live interactive performances.

Third, the system allows users to configure stylistic parameters such as tempo, scale, and chord sets, thereby ensuring that the generated music aligns more closely with the intended genre or style. This adaptability makes the system more adaptive than solutions like GenJam or DarwinTunes, which are narrowly specialized in genre or structural constraints.

In summary, while the lack of polyphonic capability represents a structural limitation of the chosen genome structure, the system's high performance, adaptability, and innovative approach to reducing user fatigue make it a promising and practical tool. These strengths collectively suggest that the proposed solution may serve not only as a research prototype but also as a basis for a real-world interactive music generation system.

8. Conclusions

Methods and existing systems of music generation based on genetic methods were reviewed. In particular, systems that use interactive genetic algorithms and also use surrogate fitness functions in the process of evolution are considered.

A system is proposed that generates music using a modified interactive genetic algorithm and also uses dynamic user engagement tracking. The proposed system also differs from analogues in the ability to generate music in different styles, as well as more subtle settings available to users.
In the future, it is possible to improve the system by integrating a surrogate fitness function similar to the one used in the GP-Music system [8].

Acknowledgements

The article was prepared as part of the research work "Information Technologies of Intelligent Computing".

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] K. Collins, An introduction to procedural music in video games, Contemp. Music Rev. 28.1 (2009) 5–15. doi:10.1080/07494460802663983.
[2] MuseNet, 2019. URL: https://openai.com/index/musenet/.
[3] MusicLM - AI model for music generation, 2023. URL: https://musiclm.com.
[4] S. Ji, J. Luo, X. Yang, A comprehensive survey on deep music generation: multi-level representations, algorithms, evaluations, and future directions, Preprint, 2020. arxiv. doi:10.48550/arXiv.2011.06801.
[5] H. Takagi, Interactive evolutionary computation: fusion of the capabilities of EC optimization and human evaluation, Proc. IEEE 89.9 (2001) 1275–1296. doi:10.1109/5.949485.
[6] J. A. Biles, Life with GenJam: interacting with a musical IGA, in: IEEE SMC'99 conference proceedings. 1999 IEEE international conference on systems, man, and cybernetics, IEEE. doi:10.1109/icsmc.1999.823290.
[7] R. M. MacCallum, M. Mauch, A. Burt, A. M. Leroi, Evolution of music by public choice, Proc. Natl. Acad. Sci. 109.30 (2012) 12081–12086. doi:10.1073/pnas.1203182109.
[8] GP-Music: an interactive genetic programming system for music generation with automated fitness raters, in: Genetic programming 1998: proceedings of the third annual conference, 1998, pp. 181–186. doi:10.1109/TEVC.1999.771172.
[9] Midi. URL: https://midi.org.
[10] .NET Multi-platform App UI documentation - .NET MAUI. URL: https://learn.microsoft.com/en-us/dotnet/maui.
[11] I. Fedorchenko, A. Oliinyk, A. Stepanenko, T. Zaiko, S. Shylo, A. Svyrydenko, Development of the modified methods to train a neural network to solve the task on recognition of road users, Eastern-European J. Enterp. Technol. 2.9 (98) (2019) 46–55. doi:10.15587/1729-4061.2019.164789.
[12] N. A. Afifie, A. W. Y. Khang, A. S. Bin Ja'afar, A. F. B. M. Amin, J. A. J. Alsayaydehahmad, W. A. Indra, S. G. Herawan, A. B. Ramli, Evaluation Method of Mesh Protocol over ESP32 and ESP8266, Baghdad Sci. J. 18.4(Suppl.) (2021) 1397. doi:10.21123/bsj.2021.18.4(suppl.).1397.