УДК 621.316 https://doi.org/10.36296/1819-8058.2023.4(75)29-38

IMPLEMENTATION OF A REAL-TIME FUZZY CONTROL SYSTEM FOR ELECTRIC VEHICLE CHARGING STATIONS BASED ON ARDUINO MEGA 2560 MICROCONTROLLER

Received Nov. 13. 2023; accepted Dec. 20. 2023
Available online Dec. 30. 2023

An. Bosak[1], Al. Bosak[2]

Author for correspondence: Andrii Bosak,
e-mail: avbosak@gmail.com

[1] PhD student
https://orcid.org/0000-0002-4667-9720

[2] Cand. of tech. Sciences., Assoc. Prof.
https://orcid.org/0000-0003-0545-9980

[1,2] National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute»;

Educational and scientific institute of energy saving and energy management

This study focuses on the problem of managing the energy efficiency of the fleet of electric vehicles of a motor transport enterprise operating in a large city, taking into account the requirements of sustainable electromobility and the constraints of the power system. As a tool for information support of the control process, a system of fuzzy control of charging stations of electric vehicles in real time based on a microcon-troller is considered. Planning for charging electric vehicles (EVs) in conditions The power limit of the charg-ing station is carried out using the Coef-ficient EV charging algorithm. The algorithm involves controlling the charging of electric vehicles by assigning a weight factor (WCC) to each vehicle connected to the charging station. Optimization of the electrical load of the charging station is carried out from the standpoint of min-imizing electricity costs and meeting the demand for charging electric vehicles without going beyond the network limitations. Computer simulation of the charging mode of electric vehicles and the load of the charging station was performed. The results of modeling the charging of electric vehicles using the pro-posed algorithm in the Matlab Simulink environment are compared with the results of modeling using cal-culations on the Arduino Mega 2560 board. The proposed implementation approach on the Atmega 2560 microcontroller provides a reduction in power consumption during peak hours of the power grid, as well as on-demand SOC charging of all connected electric vehicles.

Keywords: *electric vehicle fleet, fuzzy logic, state of charge, charging weight coeficient, microcontroller.*

РЕАЛІЗАЦІЯ СИСТЕМИ НЕЧІТКОГО КЕРУВАННЯ ЗАРЯДНИМИ СТАНЦІЯМИ ЕЛЕКТРОМОБІЛІВ У РЕАЛЬНОМУ ЧАСІ НА ОСНОВІ МІКРОКОНТРОЛЕРА ARDUINO MEGA 2560

Отримано 13 жов. 2023 р.; рекомендовано до публікації 20 груд. 2023 р.
Доступно онлайн 30 груд. 2023 р.

А. В. Босак[1], А. В. Босак[2]

Автор для кореспонденції: Андрій Босак,
e-mail: avbosak@gmail.com

[1] Аспірант.
https://orcid.org/0000-0002-4667-9720

[2] Канд. техн. наук, доцент.
https://orcid.org/0000-0003-0545-9980

[1,2] Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»;

Навчально-науковий інститут енергозбереження та енергоменеджменту

Це дослідження фокусується на проблемі управління енергоефективністю флоту електромобілів автотранспортного підприємства, що функціонує в умовах великого міста, з урахуванням вимог стійкої електромобільності та обмежень енергосистеми. Як інструмент інформаційного забезпечення процесу управління розглядається система нечіткого керування зарядними станціями електромобілів у реальному часі на основі мікроконтролера. Наведене планування заряджання електромобілів (EV) в умовах обмеження потужності зарядної станції виконано з використанням алгоритму заряджання на основі нечіткої логіки та оптимізації з урахунанням обмежень приєднання. Алгоритм передбачає контроль зарядки електромобілів шляхом присвоєння вагового індексу заряджання (CWI) кожному

транспортному засобу, підключеному до зарядної станції. Оптимізацію електричного навантаження зарядної станції виконано з позицій мінімізації витрат електроенергії та забезпечення попиту на зарядку електромобілів без виходу за обмеження мережі. Виконано комп'ютерне моделювання режиму заряджання електромобілів та навантаження зарядної станції. Результати моделювання зарядки електромобілів з використанням запропонованого алгоритму в середовищі Matlab Simulink порівнюються з результатами моделювання з використанням обчислень на платі Arduino Mega 2560. Запропонований підхід реалізації на мікроконтролері Atmega 2560 забезпечує зниження електроспоживання в години пікових навантажень електричної мережі, а також зарядку на вимогу SOC всіх підключених електромобілів.

Ключові слова: *парк електромобілів, нечітка логіка, рівень заряду, ваговий індекс заряджання, мікроконтролер.*

Introduction. Most of the solutions to the problem of charging a large number of EVs are based on various types of planning and the creation of their charging profile - charging scenarios with technical limitations. These restrictions depend on the situation at the charging station (the number of EVs, their charging levels, the possibility of simultaneous service, the declared charging time). A convenient charging period for the EV owner and the network load schedule may conflict [1]. The development of a real-time charging control algorithm should provide each EV with guaranteed charging in conditions of limited power consumption.

Assessment of the situation can be implemented by providing a weight index for each vehicle that is connected to the EV charging station. A simplified algorithm of the proposed system is shown in Fig. 1. The decision to charge or hold the EV depends on two stages of computing. At the first stage, a charge weight index (CWI) is calculated for all connected EVs, which determines their priority. At the second stage, optimization is carried out – network and price constraints are taken into account, the required final level of battery charge and the charging power delivered to electric vehicles with the highest CWI [2].

One of the primary benefits of this algorithm lies in its simplicity, rapid execution, and cost-effective implementation and upkeep. In contrast to other real-time algorithms designed to solve differential equations, which demand substantial computational resources, this algorithm can be effectively employed on a 16-bit microcontroller operating at a clock speed of only 2 MHz.

Development of fuzzy logic for the proposed charging system. The first step in the implementation of this approach is that Upon connecting a new electric vehicle (EV) to the charging station, the EV owner communicates key parameters, including the parking duration $t_{parking}$, the desired and initial levels of battery charge SOC_{req}, SOC_{init}, , to the charging station controller (CSC). This information is utilized to compute the energy required for battery charging W_{req} and the charging power S_{charge}. Notably, any uncertainty, as typically found in scheduling algorithms, is eliminated in this process [3].

Following the principles of fuzzy logic, the procedure for calculating CWI using the fuzzy system comprises the following components, as outlined in reference [4]:

- Fuzzification: this step involves converting conventional numerical data into membership functions (MF).
- Decision Block: here, MFs are combined with control rules to derive fuzzy results.
- Defuzzification: the computation of each output obtained from the decision block leads to the creation of a lookup table, from which a numerical output is selected.

The structure of the fuzzy logic blocks described above is illustrated in Fig. 2.

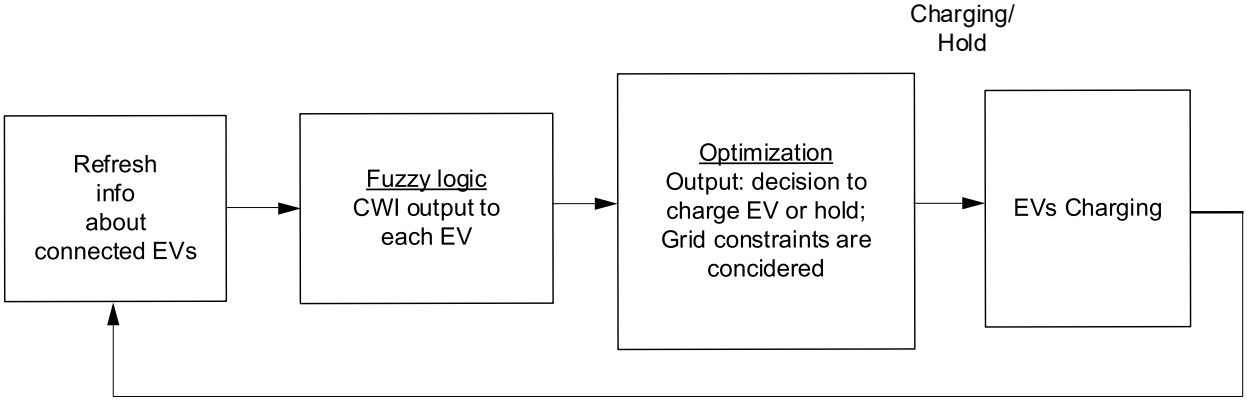

Fig. 1. Simplified representation of the proposed electric vehicle charging algorithm

The calculated index is then used in the second optimization stage of the algorithm.

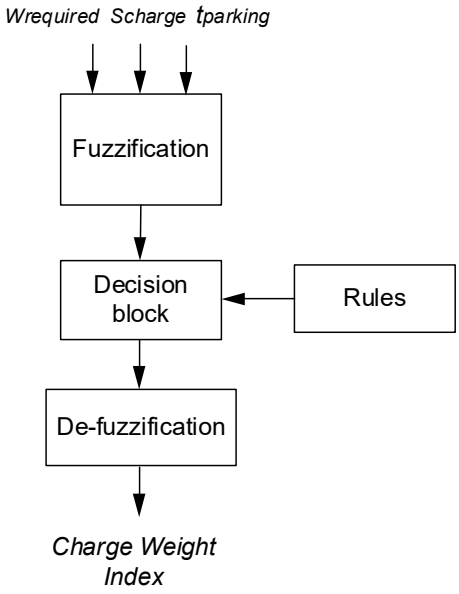

Fig. 2. Structure of fuzzy logic

CWI is determined as a function of the energy required for charging, the charging power, and the parking duration [5] . The CWI membership function is formed according to the functions of input parameters:

$$f: W_{\text{req}_i}, S_{\text{charge}_i}, t_{\text{parking}\kappa_i} \rightarrow \text{CWI}_i$$

$$\mu(W_{\text{req}_i}), \mu(S_{\text{charge}_i}), \mu(t_{\text{parking}_i}) \rightarrow \mu(\text{CWI}_i), \quad (1)$$

here W_{req} is the required energy to charge the battery, S_{charge} is the charging power, t_{parking} the time of EV being connected to the station, respectively, and μ is the membership function.

The energy needed for the charging process is determined by multiplying the discrepancy between the battery's specified initial charge level, as set by the EV owner, and the desired charge level by the battery's capacity:

$$W_{\text{req}} = \frac{(SOC_{\text{req}} - SOC_{\text{init}}) \cdot E_{\text{bat}}}{100}, \quad (2)$$

with SOC_{req} and SOC_{init} indicating required and initial charge levels of battery, and E_{bat} the capacity of the battery.

Conversely, the charging power at the battery end is defined by a relationship involving the charge level, voltage, and charging current:

$$P_{\text{charge}} = f_{\text{bat}}(SOC, V_{\text{charge}}, I_{\text{charge}}), \quad (3)$$

Hence, the charging power on the grid side:

$$S_{\text{charge}} = \frac{P_{\text{charge}}}{\eta}, \quad (4)$$

with η – the efficiency of the charger.

The characteristics of the MF (graphic representation is shown in Fig. 3), such as input and output variables, their corresponding forms of functions and terms, with their corresponding ranges, are taken by assessing the conditions of connection and operation of charging stations in Ukraine and are described in the previous paper [2].

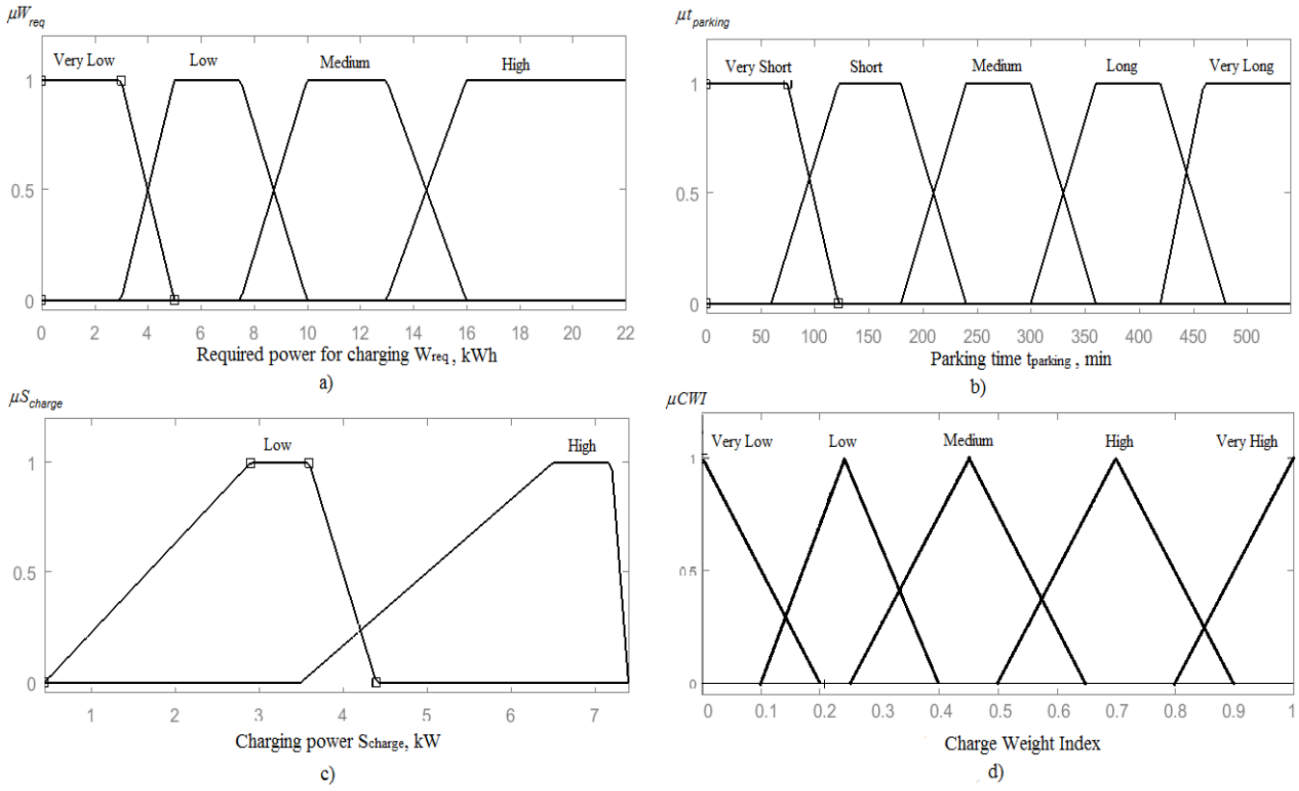

Fig. 3. Membership functions for the required energy for charging W_{req} (a), parking time $t_{parking}$ (b), charging power S_{charge} (c) and the output charging weight index of the CWI (d)

Fuzzy control rules are established based on the designer's expertise and data acquired through testing. Typically, these rules are structured as if-then statements, outlining the logical associations between input and output variables [6].

The proposed fuzzy system comprises 40 rules. Each solution phase includes a unique coefficient (index) used to scale the initial term value.

The decision-making block conveys the outcome of the applied rule to membership functions. A fuzzy implication is employed, resulting in an "if, then" expression. In the realm of fuzzy logic, this operation involves identifying the minimum of the initial membership function values [7]:

$$\mu(\mathrm{CWI}) = \mu(W_{\mathrm{req}}) \cap \mu(S_{\mathrm{charge}}) \cap \mu(t_{\mathrm{parking}}) = min[\,\mu(W_{\mathrm{req}}), \mu(S_{\mathrm{charge}}), \mu(t_{\mathrm{parking}})]_{EVi}\,, \quad (5)$$

The assessment and evaluation of the created fuzzy system were performed by employing a three-dimensional function graph (Fig. 4). Adjustments to the membership functions and rules were made to ensure that the figure displayed a descending pattern or a flat surface in its coordinates. This was done in a manner such that neighboring values had a minimal impact on the process of determining the initial variable. The choice of the descent direction was as follows: when moving closer to zero along the axis representing the energy required for charging, the graph descended, while in the case of parking time, it exhibited an opposite, ascending trend.

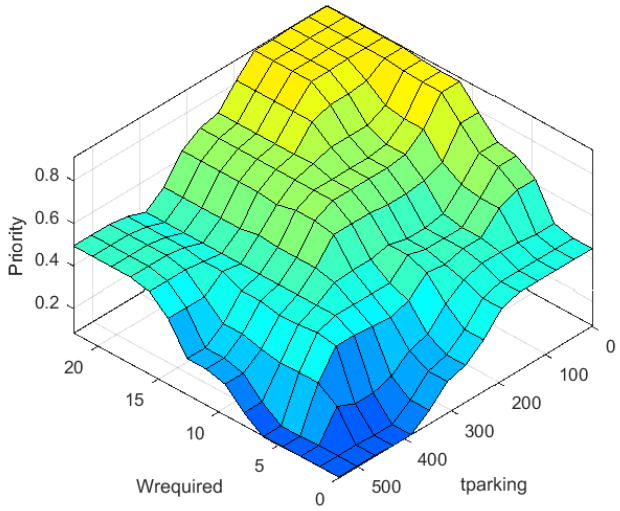

Fig. 4. Three-dimensional mapping of membership functions and rules

During the defuzzification phase, the CWI vector for each electric vehicle connected to the charging station is determined using the widely adopted gravitational method [4]. The calculation involves identifying the arithmetic mean of the elements derived from the membership function values with the highest degrees of membership:

$$\mathrm{CWI} = \frac{\sum_{EV} EVi\mu(\mathrm{EV}i)}{\sum_{EV} \mu(\mathrm{EV}i)}\,, \quad (6)$$

EV means all connected EVs, $\mu(\mathrm{EV}i)$ are the membership functions after the solution block.

Optimization of the objective function taking into account network constraints. The aim of the optimization process is to choose a maximum number of electric vehicles with the highest CWI priority indices. Equation (7) represents the objective function, which incorporates a decision vector concerning whether to allow charging for electric vehicles.

$$max\, F_{(t)} = \sum_{Spot_{(i)}} \mathrm{CWI}_{(Spot_{(i)},t)} \times \mathrm{Decision}_{(\mathrm{Spot}_{(i)},t)} \quad (7)$$

with $F_{(t)}$ is the objective function,

$\mathrm{Spot}_{(i)}$ - spot of the charging station,

$\mathrm{Decision}_{(Spot_{(i)},t)}$ - The decision to charge (1) or not (0) the charging spot for the calculation period of the method t, derived from conditions (10). A constraint is imposed on the objective function.

Power limitations. The energy demand in each time period is taken into account - the power limit looks like this:

$$S_{\mathrm{charge}_{\mathrm{EV}}(t)} + S_{\mathrm{load}} \le S_j^{max}\,, \quad (8)$$

$S_{\mathrm{charge}_{\mathrm{EV}}(t)}$ is the total load capacity of electric vehicles charged in a specific period of time, kVA; S_{load} - load capacity of other consumers connected to the grid, kVA, S_j^{max} - maximum network capacity.

Other are current and voltage limitations:

$$\left|I_{(t)}\right| \le I_{MAX(t)};$$
$$V_{min} < V(i,t) < V_{max}\,, \quad (9)$$

$I_{(t)}$ is the load current in a given period of time, is the maximum current;
$V_{min,max}$ – the minimum and maximum voltage on the bus to which the charging station is connected.

Status restrictions. The auxiliary variable *Decision* is a solution (0/1) that, prior to fuzzy logic calculations, allows the EV to be held or charged. Given the conditions where the EV reaches the desired SOC charge level or the EV is not connected to the charging spot, the following provisions are suggested:

$$\mathrm{Decision}_{(\mathrm{Spot}_{(i)},t)} = 0,$$
$$\text{if} \quad SOC_{(\mathrm{Spot}_{(i)},t)} \ge SOC_{\mathrm{req}(\mathrm{Spot}_{(i)},t)}$$
$$\text{AND} \quad (10)$$
$$\mathrm{Decision}_{(\mathrm{Spot}_{(i)},t)} = 0, \text{ if } \mathrm{Occupied}_{(\mathrm{Spot}_{(i)},t)} = 0,$$

Here $\mathrm{Occupied}_{(\mathrm{Spot}_{(i)},t)}$ is the availability of the charging spot: free (1) or occupied (0) by EV.

The real-time approach manages the charging procedure in a manner that enables swift responses to fluctuations in power consumption conditions during the connection.

Additionally, the calculation frequency can be adjusted, either upward or downward, to accommodate specific calculation requirements. Initially, the system refreshes all details related to electric vehicles and computes the necessary energy to reach the desired charge level. After determining the charging workload index in the phase-system, it proceeds to calculate the current SOC charge level, energy consumed by each vehicle, and the aggregate electricity demand at the conclusion of the 10-minute interval.

Following the completion of the fuzzy logic computation phase, the system chooses the maximum quantity of electric vehicles with the highest CWI, ensuring compliance with all limitations set by both the energy supply company and the charging station owner. Each authorized electric vehicle is charged at the maximum charging rate. Once the battery reaches the desired SOC charge level, the workload index for that particular vehicle becomes zero.

Review and comparison of existing microcontrollers. Approach of selecting the best microcontroller to implement fuzzy logic and integrate it into an existing charge control system depends on various factors, including the specific requirements of the system, available resources, and analysis of different microcontroller platforms. Here are some popular microcontroller options that are often used to implement fuzzy logic:

1. Arduino: Arduino boards, such as the Arduino Uno, Duo, and Mega, are widely used for prototyping and integration into various systems. They have a large community and numerous libraries that can simplify the implementation of fuzzy logic [8].
2. Raspberry Pi: While the Raspberry Pi is not a microcontroller in the traditional sense, it is a popular single-board computer with more processing power and memory than typical microcontrollers. It can handle complex fuzzy logic tasks and integrates seamlessly into existing systems [9].
3. STM32 Series: STMicroelectronics' STM32 microcontrollers offer a wide range of choices, from energy-efficient to high-performance devices. They are well-suited for real-time computing and have development tools available [10].
4. PIC microcontrollers: Microchip's microcontrollers from the PIC family are known for their reliability and ease of use. They have different sizes and capabilities, which allows you to find an option for implementation [11].
5. ESP8266/ESP32: These microcontroller modules are popular for the Internet of Things (IoT). They have Wi-Fi and Bluetooth capabilities, which can be useful for remote monitoring and control of fuzzy logic systems [12].
6. NXP/Freescale microcontrollers: NXP microcontrollers such as the HC12 are among the most popular for implementing fuzzy logic due to their prevalence and cost [13].

When choosing a microcontroller, it is important to take into account the following factors, which will be compared in the next paragraph [14]:

- Computing power – microcontroller should have enough computing power to work efficiently with fuzzy logic algorithms.
- Memory: Sufficient RAM and Flash memory is required to store fuzzy sets, rules, and other data.
- The I/O interfaces are necessary to obtain variables and issue a solution for charging or holding an electric vehicle.
- Development Tools: It is important to have development tools, development environments, and libraries available for the selected microcontroller.
- Cost: since there is a goal of efficient use of funds, it is required having an optimal controller in terms of price/cost.

Power and memory of the microcontroller. The computing power of microcontrollers can vary significantly depending on the model, series, and configuration. The comparisons below will be general, while a detailed overview of the board parameters is given in Table 1.

Usually, the microcontrollers mentioned above are divided into several classes depending on their power:

Arduino Uno, Duo, Mega (AVR-based):

- This is a classic representative of microcontrollers with low computing power.
- Low clock speed (16 MHz).
- Limited amount of memory (a few kilobytes in RAM and Flash).
- Uno is suitable for simple fuzzy logic tasks (drive control, fan control, etc.), while Mega is for computationally demanding fuzzy logic operations of the charging system.

Raspberry Pi 4:

- The Raspberry Pi is a single-board computer, and it has significantly higher processing power compared to typical microcontrollers.
- It has an ARM processor with a much higher clock rate and more memory (gigabytes of RAM).
- Suitable for complex fuzzy logic problems and other computationally intensive operations.

STM32 series (e.g. STM32F4xx):

- The STM32 has different models with different levels of computing power.
- The most powerful models in this series have a Cortex-M4 core with fast clock speeds and hardware support for numerous operations.
- They can usually handle complex fuzzy logic operations without much difficulty.

Table 1. Main parameters of microcontrollers

Microcontroller/ Platform	Frequency	Flash Memory	RAM	Specialized Functions/Interfaces
Arduino Uno	16 MHz (ATmega328P)	32 KB	2 KB	14 digital, 6 analog UART, SPI, I2C, PWM, USB
Arduino Mega 2560	16 MHz (ATmega2560)	256 KB	8 KB	54 digital, 16 analog UART, SPI, I2C, PWM, USB
Raspberry Pi 4 (4GB)	1.5 GHz (Quad-core ARM Cortex-A72)	4 GB	4 GB	GPIO, HDMI, Ethernet, USB, I2C, SPI, UART
STM32F407VG	168 MHz (Cortex-M4)	1 MB	192 KB	100 GPIO pins, UART, SPI, I2C, PWM, CAN, USB, Ethernet
ESP32-WROOM-32	240 MHz (Dual-core Tensilica LX6)	4 MB	520 KB	UART, SPI, I2C, PWM, Wi-Fi, Bluetooth
NXP Kinetis K64F (MK64FN1M0VLL12)	120 MHz (Cortex-M4)	1 MB	256 KB	82 GPIO pins, UART, SPI, I2C, PWM, CAN, USB, Ethernet
NXP LPC1768	100 MHz (Cortex-M3)	512 KB	64 KB	70 GPIO pins, UART, SPI, I2C, PWM, CAN, USB, Ethernet

ESP8266/ESP32 (ESP32):

- The ESP8266/ESP32 has little processing power compared to some other microcontrollers (are more powerful comparing to Arduino Uno, but less powerful than the Raspberry Pi).
- They are suitable for most fuzzy logic tasks, but may have limitations in the amount of available memory.

NXP/Freescale Microcontrollers (Kinetis, LPC):

- All of these series have different models with different computing power characteristics.
- The most powerful models can handle complex fuzzy logic operations.

The cost of the mentioned above boards will not be given in this article, as it can fluctuate significantly depending on the place of purchase, region and other factors. To justify the choice of the board after the analysis of prices, it is determined that it is directly proportional to the power of the processor and the amount of memory.

Experimental setup and results. Arduino Mega 2560 microcontroller. The Arduino Mega is a microcontroller board that belongs to the Arduino line and features greater resources compared to larger amounts of memory and more GPIO pins compared to the Arduino Uno. This board is well suited to the proposed system, including the implementation of fuzzy logic [14]. The appearance of the board is shown in Fig. 5. Since, in addition to fuzzy logic calculations, it is necessary to carry out the stage of optimization of the objective function, this microcontroller was chosen.

Fig. 5. Appearance of the Arduino Mega 2560 board

Detailed description of the characteristics of the Arduino Mega [8]:

Microcontroller: ATmega2560. Frequency: 16 MHz

Memory: Flash memory: 256 KB (for program code),
RAM: 8 KB (for variable and data storage)
EEPROM: 4 KB (for data storage)

Analog and digital GPIO pins:

Digital Pins: 54 digital GPIO pins, 15 of which can work as PWM outputs.
Analog pins: 16 analog inputs for reading analog signals from 0 to 5 V.

Serial interfaces:

- UART (Universal Asynchronous Receiver/Transmitter): There are many UART ports for communicating with your computer or other devices.
- SPI (Serial Peripheral Interface): There is hardware support for SPI to communicate with devices such as sensors and displays.
- I2C (Inter-Integrated Circuit): There is hardware support for I2C to communicate with devices via I2C.

External interface:

- USB: USB port for programming and communication with a computer.
- DC connector: To power the microcontroller.

Adjustable supply voltage: 7-12 V (7-12 V recommended). Operating voltage: 5 V (usually powered by voltage through a regulator on the board).

Connectors: The Arduino Mega has numerous connectors for connecting additional modules and sensors, including Arduino shields, which can be simply inserted onto the top of the board.

The procedure the microcontroller integration into the system. The electric charging station, which is the object of

research, is a combination of charging spots into one network. This charging spot and its connection diagram are shown in Fig. 6.

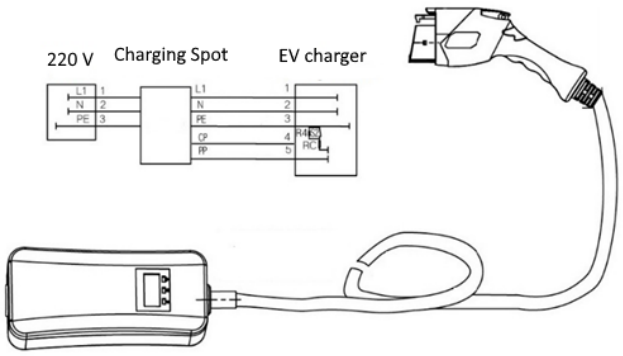

Fig. 6. Charging spot and wiring diagram

The main elements of the charging station, the functions of the connector and theType 2 J1772 connector, the process of connecting the EV to the charging station are described below. The switching, protection and control devices of the charging unit are shown in Fig. 7. Fig. 8 shows the pin location of the charging spot connector.

The controller manages the power supply process of the electric vehicle during charging [15].

Contactors are used for load control (powering EVs), external ventilation, and J1772 connector lockout. The LED indication indicates the current status of the system. Charging current can also be controlled during the charging process. The setting point of the charging current is set via the RS485 interface.

One of the biggest advantages of this algorithm is its simplicity, speed and low cost of implementation and maintenance. Compared to other real-time algorithms that solve differential equations and require high computing power, the presented algorithm can be implemented on a 16-bit microcontroller with an integer arithmetic logic device and a clock frequency of only 2 MHz. The diagram of connecting the controller to the existing EV charging station spots is shown in Fig.9.

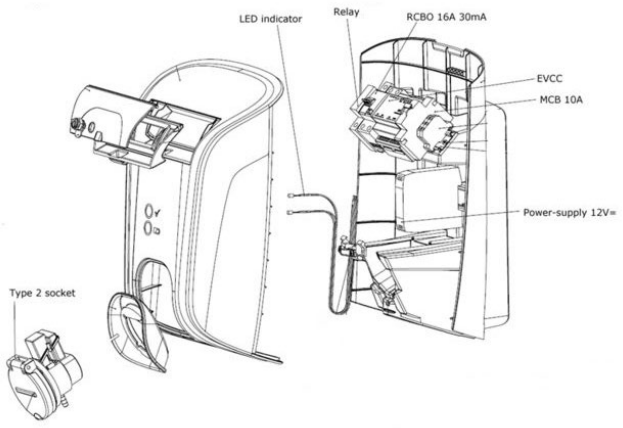

Fig. 7. Main devices and components of the charging station

After calculating the fuzzy logic and optimizing the target function, the microcontroller sends a signal to the control key E2 (Control Switch) – if charging during a given period of time, the allowed contact will be closed. If, according to the algorithm's decision, no charging is allowed during this period of time, contact E2 will remain open.

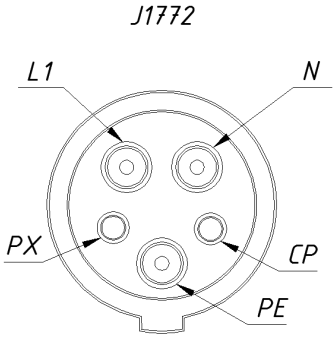

Fig. 8. Charging Spot Connector Contacts

The purpose of this work is to develop a real-time charging system of EVs. The choice was made on the combination of the MATLAB/Simulink environment and the Arduino board. MATLAB/Simulink provides the "Simulink Support Package for Arduino Hardware" to develop control algorithms in

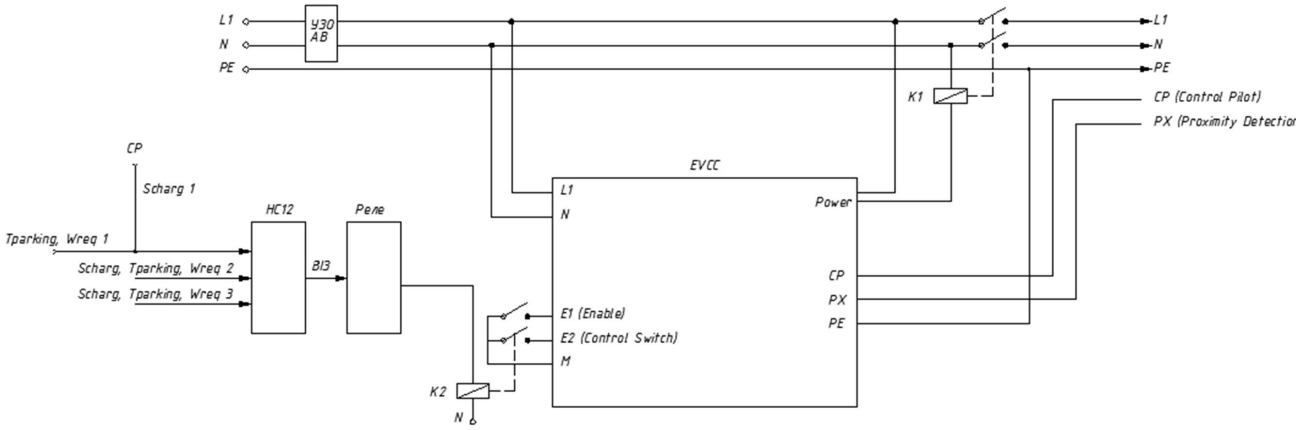

Fig. 9. Connection diagram of the controller into charging station circle

Simulink and load it directly into the hardware platform (Arduino board). The "Simulink Support Pack for Arduino Hardware" allows to create and run Simulink models in a variety of areas and uses. It offers a library of units for configuring and accessing Arduino sensors, actuators, and communication interfaces via a USB cable [16].

Fig. 10 shows a general view of the Matlab Simulink model and the connection of the Arduino board to the charging station. The power of the charging station is 24 kW. The transformer and network equipment are limited to a capacity of 120 kW. A simulation of the Simulink model was carried out and the SOC change graphs for each connected EV were evaluated after each iteration of the change of the fuzzy rules based coefficient.

The Arduino package allows to perform tasks such as:

- Retrieving analog and digital sensor data from an Arduino board.
- Control of devices with digital and PWM outputs.
- Communication with the Arduino board via USB cable.
- Access to peripherals and sensors connected via Integrated Circuit (I2C) or Serial Peripheral Interface (SPI)

The developed charging control system developed in Matlab was uploaded onto an Arduino Mega 2560 via USB interface. The MATLAB Coder solution converted the code of the proposed system, stored in an m-file, into code written in C++, which is "understandable" for Arduino. Also, with the help of Matlab, the transmission of collected the data from an electric vehicle was simulated, in paticullar S_{charge},$W_{req,}$ $t_{parking}$. Sending the microcontroller's decision to charge or hold the EV was also simulated via Arduino blocks.

The transfer of input data of the EVs' and network's parameters from the Matlab model to the board is carried out using the Serial Transmit block from the Simulink Support Package for Arduino Hardware library [17]. The output of the matrix with the decision to charge or hold connected electric vehicles is implemented using the Serial Receive block (Fig. 11).

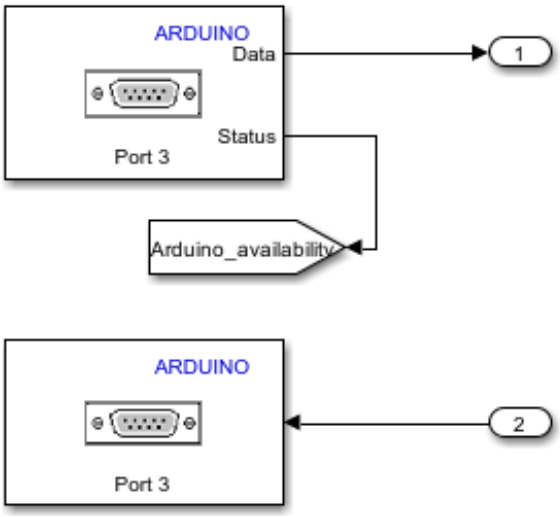

Fig. 11. Used in the project Arduino Library Blocks

The fuzzy system developed in Matlab Fuzzy Logic Designer was saved in a file with a .fis extension that cannot be operated by Arduino. In order to calculate the result of fuzzy logic for each iteration of calculations, the fuzzy system was rewritten in C++ and uploaded into the memory of the microcontroller board.

The logic of optimization of the objective function (taking into account the constraints of the network in the model) is written in the language of Matlab, which is also converted to C++ and stored in Arduino memory.

Analysis of simulated data. In this section, the proposed charging system with calculations on the Arduino Mega 2560 is compared to the calculations performed on a PC.

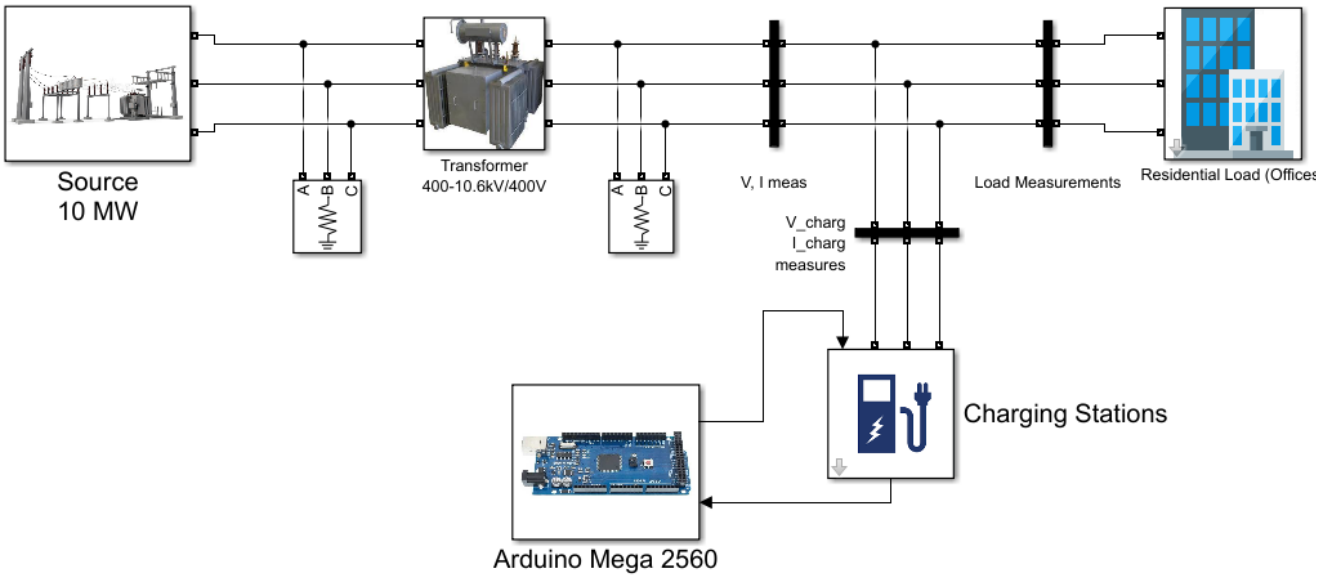

Fig. 10. General view of the entire model in Matlab Simulink

The demand for electricity during a 24-hour charging scenario for different computing scenarios is shown in Fig.12, Fig.13 and Fig.14.

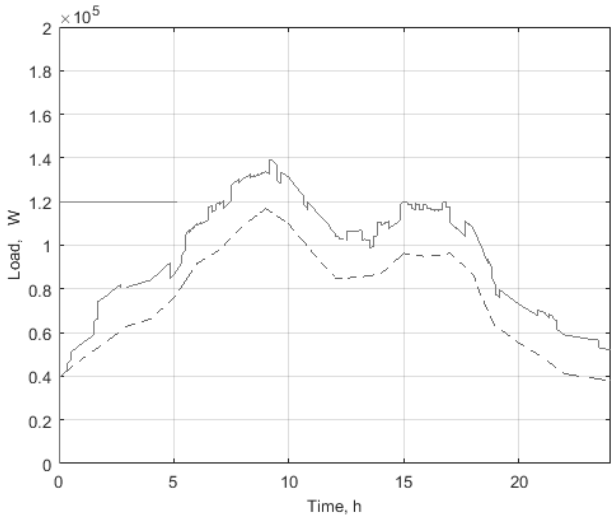

Fig. 12. Electricity load within 24 hours based on uncontrolled charging. (Solid line – total demand with electric vehicles, dashed line – other load)

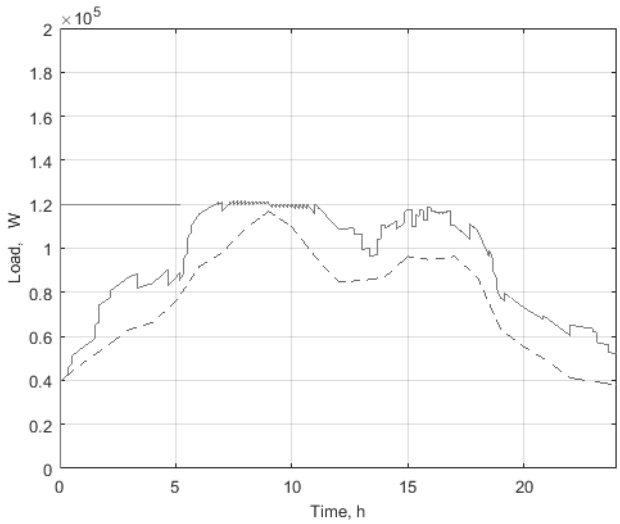

Fig. 13. Electricity load within 24 hours based on the proposed charging method is calculated on a PC in the Matlab Simulimk environment. (Solid – total demand with electric vehicles, dashed – other load)

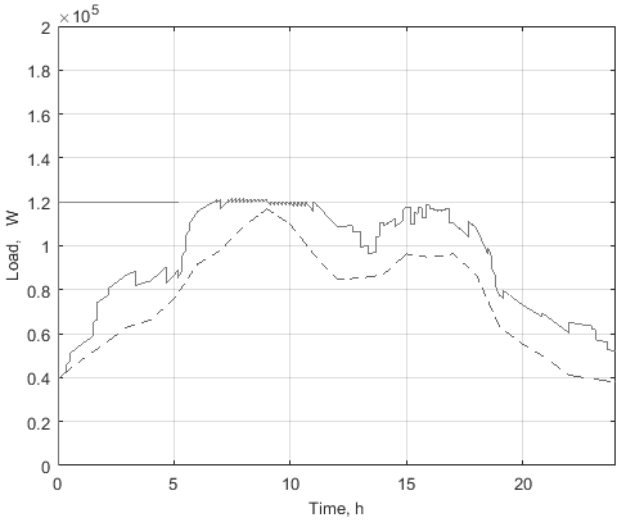

Fig. 14. Electricity load for 24 hours based on the proposed charging method when performing calculations on the Arduino Mega 2560. (Solid – total demand with electric vehicles, dashed – other load)

As can be seen from Fig. 12 network overload (load above 120 kW) up to 16% occurs during uncontrolled charging from 7 a.m. to 11 a.m

Table 2 shows a comparison of the energy consumed per day for different computing modes.

A comparison of these charging calculations shows that the calculations of the power transferred to electric vehicles, taking into account the constraints in the Matlab Simulink environment, do not differ from the calculations on the Arduino Mega 2560.

Discussion and Conclusions. The operation of the fuzzy controller based on the proposed methodology was tested on the Arduino Mega 2560 microcontroller. The simplicity of implementation of the real-time method is confirmed by the scheme of the fuzzy logic controller connection to the existing control system of electric charging stations, which will control the charging process.

The results show that the simulation graphs with calculations of the proposed system based on fuzzy logic on a PC in the Matlab environment and on an Arduino board do not differ. The fuzzy controller integration procedure into the

Table 2. On-grid energy consumption for different charging methods

	Network Load without EM (Matlab Simulink)	**Uncontrolled EV charging (Matlab Simulink)**	**Network Simulation – Matlab Simulink, ChargingManagement System Computing – Matlab**	**Network Simulation – Matlab Simulink, Charging Control System Computing – Arduino Mega 2560**
Total energy consumed in 24h W_{all} , kWh	1825,2	2267.1 (grid overload)	2292,1	2292
Consumed energy by EVs in 24h W_{EV}, kWh	1825,2	441,9	467,5	467

process is also described. Compared to other works, the greatest contribution of this lies in describing the elements for programming and implementation each of the fuzzy controller software and hardware aspects. This allows to estimate the optimal parameters, uncertainties and nonlinearities of the system without a mathematical model. Finally, simulation and implementation on the Arduino Mega 2560 microcontroller were shown to verify that the proposed methodology, as well as the theoretical and experimental results, are valid.

REFERENCES

1. P.-Y. Kong and G. K. Karagiannidis, “Charging Schemes for Plug-In Hybrid Electric Vehicles in Smart Grid: A Survey,” IEEE Access, vol. 4, pp. 6846–6875, 2016, doi: 10.1109/ACCESS.2016.2614689.
2. A. Yandulskii, O. Kurson, A. Bosak, S. Kondratiev, and A. Kuznietsov, “Improvement of Electric Charging Station Efficiency using situation-dependent Fuzzy Algorithms,” in 2018 IEEE International Conference on Electrical Systems for Aircraft, Railway, Ship Propulsion and Road Vehicles & International Transportation Electrification Conference (ESARS-ITEC), Nottingham: IEEE, Nov. 2018, pp. 1–6. doi: 10.1109/ESARS-ITEC.2018.8607378.
3. K. Qian, C. Zhou, M. Allan, and Y. Yuan, “Modeling of Load Demand Due to EV Battery Charging in Distribution Systems,” IEEE Trans. Power Syst., vol. 26, no. 2, pp. 802–810, May 2011, doi: 10.1109/TPWRS.2010.2057456.
4. Y. Bai, H. Zhuang, and D. Wang, Eds., Advanced Fuzzy Logic Technologies in Industrial Applications. in Advances in Industrial Control. London: Springer London, 2006. doi: 10.1007/978-1-84628-469-4.
5. E. Akhavan-Rezai, M. F. Shaaban, E. F. El-Saadany, and F. Karray, “Online Intelligent Demand Management of Plug-In Electric Vehicles in Future Smart Parking Lots,” IEEE Systems Journal, vol. 10, no. 2, pp. 483–494, Jun. 2016, doi: 10.1109/JSYST.2014.2349357.
6. C. C. Lee, “Fuzzy logic in control systems: fuzzy logic controller. I,” IEEE Trans. Syst., Man, Cybern., vol. 20, no. 2, pp. 404–418, Apr. 1990, doi: 10.1109/21.52551.
7. F. O. Karray and C. W. De Silva, Soft computing and intelligent systems design: theory, tools, and applications. Harlow, England ; New York: Pearson/Addison Wesley, 2004.
8. L. Louis, “Working Principle of Arduino and Using it as a Tool for Study and Research,” IJCACS, vol. 1, no. 2, pp. 21–29, Apr. 2016, doi: 10.5121/ijcacs.2016.1203.
9. S. Karthikeyan, R. A. Raj, M. V. Cruz, L. Chen, J. L. A. Vishal, and V. S. Rohith, “A Systematic Analysis on Raspberry Pi Prototyping: Uses, Challenges, Benefits, and Drawbacks,” IEEE Internet Things J., vol. 10, no. 16, pp. 14397–14417, Aug. 2023, doi: 10.1109/JIOT.2023.3262942.
10. B. Deng, Z. Bo, Y. Jia, Z. Gao, and Z. Liu, “Research on STM32 Development Board Based on ARM Cortex-M3,” in 2020 IEEE 2nd International Conference on Civil Aviation Safety and Information Technology (ICCASIT, Weihai, China: IEEE, Oct. 2020, pp. 266–272. doi: 10.1109/ICCASIT50869.2020.9368860.
11. The Quintessential PIC® Microcontroller. in Computer Communications and Networks. London: Springer-Verlag, 2005. doi: 10.1007/1-84628-202-0.
12. D. Hercog, T. Lerher, M. Truntič, and O. Težak, “Design and Implementation of ESP32-Based IoT Devices,” Sensors, vol. 23, no. 15, p. 6739, Jul. 2023, doi: 10.3390/s23156739.
13. D. H. Summerville, Embedded Systems Interfacing for Engineers using the Freescale HCS08 Microcontroller II: Digital and Analog Hardware Interfacing. in Synthesis Lectures on Digital Circuits & Systems. Cham: Springer International Publishing, 2009. doi: 10.1007/978-3-031-79803-0.
14. A. M. Ibrahim, Fuzzy logic for embedded systems applications. in Embedded technology series. Amsterdam ; Boston, Mass: Newnes, 2004.
15. Dana Plymouth Technology Center, “OpenECU-EVCC (OE-EVCC) User Guide.” 2022. [Online]. Available: https://www.scribd.com/document/589256904/OpenECU-EVCC-UserGuide
16. Mathworks, “Arduino Support from MATLAB and Simulink.” [Online]. Available: https://www.mathworks.com/hardware-support/arduino.html
17. MathWorks, “Simulink Support Package for Arduino Hardware.” [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/40312-simulink-support-package-for-arduino-hardware