Internet-of-Things Device Set Configuration for Connection to Wireless Local Area Network

Yaroslav Krainyk[1[0000-0002-7924-3878]], Andrii Razzhyvin[1], Olena Bondarenko[1] and Irina Simakova[1]

[1] Petro Mohyla Black Sea National University10. 68 Desantnykiv str., Mykolaiv, Ukraine
yaroslav.krainyk@chmnu.edu.ua, greatwwgg@gmail.com,
elena.igorevna2205@gmail.com, darkling6667@gmail.com

Abstract. In this paper, new model of interaction in IoT-infrastructure for the set of devices is investigated. Integration into wireless network based on WiFi-technology, software identification, data exchange, and software upgrade problems have been considered for the end-user product that consists of group of devices. The established solution utilizes assignment of the server and client role to devices in the set. The server deploys temporary network so the user device can access it and enter necessary information about network to connect to. Communication mechanisms within customer's network for device set have been also considered. Using proposed models, customer can control device set, send and receive data (particular case for audio data is considered), update firmware and resources of devices directly from his/her own device via specially designed software. We have used set of ESP8266 boards and NodeMCU firmware to implement and test the developed model. The developed prototype system is able to receive configuration of the user network, receive audio data from client device and play audio. This proof-of-concept system can be further extended for the new functionality.

Keywords: device, device set, Internet-of-Things, configuration.

1 Introduction

In the era of Internet-of-Things (IoT) smart devices are emerging to provide different services to customers. They are built up to execute set of functions clients are interested in. Usually, the device is delivered to the customer as single ready-to-use gadget. Its configuration can be performed as any other device in the local network. However, it is not an obligatory and if the consumer gets set of devices rather than single device, individual configuration of every device in the set could be time-consuming. Moreover, if the set of devices are to be moved into another network, the manual configuration task should be performed repeatedly. Thus, apparatus for automatic configuration for the set of devices is necessary to satisfy best user experience practices.

ESP8266 WiFi-module [1] has been recognized as one of the most useful devices for IoT-applications. It has tremendous performance (runs on frequency up to 160

MHz) as for embedded device and combines it with low price. Due to the availability of digital interfaces (UART, SPI, I2C), it is a perfect candidate to perform sensor readings and provide network connectivity. Low-power consumption modes are available and, as a result, it can work even with battery power supply. Numerous novel investigations in IoT are based on usage of ESP8266. Hence, in this paper, we selected ESP8266 device as a sample WiFi-module for further description of the process and its implementation. As ESP8266 relies on the WiFi-technology, the proposed approach can be applied to modules of different producers.

After the devices from the set are configured, they can perform necessary programmed functions. In spite of that, usually, IoT-system development supposes that device constraints should be taken into consideration. As a consequence, it limits list of functions that can be implemented. However, ESP8266 modules can handle even simple multimedia data processing such as audio networking or graphics functions. In this investigation ESP8266 modules act as receivers of audio data. As they have been configured, their main function is to play audio data sent from server device.

Therefore, the main goal of the current paper is development of configuration methods for integration of IoT-devices into customer's wireless network and design of the communication scheme for further interactions between devices within the network where each device from the set is considered as an equal part.

2 Review of contemporary scientific sources

Despite the fact that at the first sight device set configuration seems a mere problem, we found out that few researches paid thorough attention to it. The known technical solutions for wireless devices are described by the patent documents [2, 3]. The authors of [3] offer the idea of usage of master or server device that identifies existing peripheral devices. The master is able to replicate configuration to other devices. However, the main presumption for this method is that all devices are already connected to the network. In opposite, we consider the case when device set is connected to the network by default. Moreover, the apparatus from [3] assumes that dedicated server node is always present while in the imposed challenge devices in the set are treated as clients to avoid usage of additional device. [2] concentrates on the communication scheme for wireless clients software update procedure. Nevertheless, it does not go further and does not formalize procedure of data exchange on the application level.

The results of [4] explain communication scheme for specific protocol and specific hardware type from single producer. Therefore, it cannot be considered as a universal method of configuration and communication for wireless IoT-device set.

The idea described in [5] supposes division of area on separate zones each controlled by separate master controller. Each of master controllers connects to main controller in the network that supposes at least three levels of hierarchy in the device network. This architecture is specifically oriented on HVAC application and can be used for Wireless Sensors Network construction but it is hard to apply it multimedia data transmission. Also the system supposes that system is deployed only once and

the configuration is stable. Lots of additional manipulations are necessary to configure system to work in new environment.

Authors of [6] claimed the development of system for home appliances that occupies technique based on ID-mechanism. The appliances are identified by dedicated IDs sent to controller of the network. This approach supposes that every device have pre-assigned ID. However, it can not work with appliances that have no ID.

In [7] method and apparatus for client identification were proposed. They are oriented on network devices that has no non-volatile memory to store configuration. Configuration is also ID-based. However, the device can receive ID during configuration stage. Nevertheless, it demands numerous configuration steps to be performed that is not applicable for lightweight device configuration.

As has been stated above, ESP8266 modules are the state-of-the-art modules in IoT field. They have been used for home automation, sensor readings, etc.

The update mechanism for ESP8266 has been proposed in [8]. Over the Air (OTA) update procedure is preferable in the end-user device because it does not require additional devices. The problem of upgrading software is quite important due to possibility to change firmware in case bugs have been detected or new features have been implemented.

Security is one of the main concerns in IoT. Security of ESP8266 has been considered in [9]. It is a problem of the special importance in conjunction with software updating. Therefore, this paper also touches the point of reliable software update with security concerns.

The paper [10] considers the problem of video data transmission from the multiple sources in the dynamic networking infrastructure. It uses multi-agent based approach to coordinate actions of devices in the set. However, the main shortcomings in context of problem imposed in this paper is an assumption about pre-configured network structure with fixed connection points.

Authors of the [11], the problem of IoT integration into existing solutions is investigated. The proposed solution is based on the open standards and offers integration of various modules. However, the paper [11] considers the situation in general by developing a universal approach with scalable components responsible for all actions within a network. Configuration of IoT device set does not require presence of such complex system.

The contribution of the paper is the developed model of configuration for IoT-device set, proposed communications schemes for device identification and client software upgrade and data transmission. In this paper, we proposed model of device set connection into consumer wireless local area network and method for integrating device set into the network. The devices in the set are able to communicate with other user devices in the network according to the developed approach for their identification. In the considered examples, the devices can process audio data and can be upgraded OTA.

3 Configuration approach and communications organization for IoT-device set

3.1 Integration of device set into existing network

In order to start consuming device in the network, the customer have to connect it into the network. In the proposed model, we distinguish two types of devices in set and we denote them as following:

- Server device (deploys service network and sends data to client nodes).
- Client device (connects to the server).

According to the proposed approach, the connection of the device set is performed with the following sequence of steps:

1. One of the devices (selected as server) deploys its own network, starts inner web-server and waits for the incoming connection. All client devices from the set connect to the server.
2. The customer connects to the network and can access web-page to write credentials to the device.
3. As customer has sent credentials information to the server device, it stores it into internal memory and is able to connect to the customer's network.
4. The server device sends data to the other devices from the set.

The steps described above are performed only under circumstance that the devices cannot connect to the previously selected network. Information about last connection is stored in the devices' memory. Device set reconfiguration is only necessary in case when we move it to other network. It is also worth to mention, that when devices are connected to the client's network, they all behave as clients (no server device among them).

3.2 Identifying server device

When the set is connected to the customer's network, it can interact with application software installed on other customer's devices (smartphones, laptops, PCs, etc.). However, it knows nothing about where the software is deployed. To identify device with the software we suppose usage of the next procedure in our model:

- Clients start listening for incoming message on dedicated port.
- When the software is running it acts like a server and sends multicast request to the specified port (UDP-server). In addition, we need to accept connection from the client so TCP-server is also started on the customer's device.
- As clients receive message, they can instantly identify IP-address to connect.

Thus, the software starts two servers to organize connection with clients. Both servers run on different ports to avoid collision.

3.3 Exchanging data with application

Constrained amount of resources (RAM and storage memory) should be taken into consideration to develop communication principles for the system. The end devices can not store and process large amount of data. Abuse of memory resources causes the device restart that can break all the communication states for the software and other clients and, furthermore, can cause crash of the whole system. This situation is not the first-class user experience and there is a demand for careful tracking of resource usage to avoid such situation.

As a consequence, of the previous statements, in case of implementation of multimedia functionality in the system, the server software is responsible for the processing of the multimedia data. In addition, the main dataflow for such features is supposed to go from the server to the clients.

Another point of interest concerned with software is the software architecture. Several devices (clients) connect to the device with server software. Therefore, it has to utilize multithreading concepts to control them simultaneously.

3.4 Updating client's software

In the proposed model, we suppose that end device functionality is defined by the scripts stored in the internal memory. Scripts are executed according to the external events (WiFi-connection, UDP data reception, etc.) and data received from server side. Software update in this case supposes updating script files or creation of the new ones. This operation could be performed without any wired connection. We denote this technique as "ad-hoc over-the-air software update" - ad-hoc OTA. Special caution should be taken to implement such feature because incorrect changes into the script could break the whole system. In the simplest case, the server initiates a new connection by sending command with specific token. It signals clients to create another connection to read scripts from the server. As a result, scripts stored on the file will be overwritten. To ensure that vital parts that is responsible for this action are always available, they should be placed into the separate script file with no manipulation allowed for it in network mode. With an additional requirement for security, the proposed mechanism should also utilize verification of the server identity. Thus, server has to confirm that it is not malicious device by providing necessary information. The procedure might be used only for software update process, however, it is also eligible for any interactions between server and clients.

3.5 Processing audio data

When the client devices are connected to the server, they can interact with server as well as with each other. As demonstration of this process, we selected transmission of audio data to the client devices. Despite the fact that audio processing at the end-user device is really straightforward, several peculiarities should be taken into account. First of them is memory constraint of the device. It should be noted, that several dozens of kilobytes of RAM are available to store data. Thus, it is necessary to free up the

memory as soon as data has been processed. The second one is the processing flow organization. At this point we assume that NodeMCU software is used in the module and class to play audio data via Pulse-Code Modulation (PCM) interface is responsible for this action.

The class supposes usage of callback mechanism to organize data processing. The data is received using network interface and is immediately appended into the list of data chunks to play. When the number of chunks is higher than lower threshold, data are sent into audio interface. Therefore, two processes run simultaneously:

- Reception of the data via the network interface and appending it into list of chunks.
- Transfer of data to PCM-interface.

Under these conditions, there is possible situation when the number of packets is close to the limit and addition and reception of one more new chunk into the list is not feasible (causes crush and system restarts). Thus, it is obligatory to control the upper threshold of the list. When the list length matches selected value it stops communication over the network interface and waits for the time when the list length is equal to lower threshold.

3.6 Implementation of the system

The system actions performed during configuration stage can be described in UML-notation. Visual Paradigm 12.1 software [12] has been used to create the appropriate diagrams for the developed system. In Fig. 1, sequence diagram for the process of connection to the server is depicted.

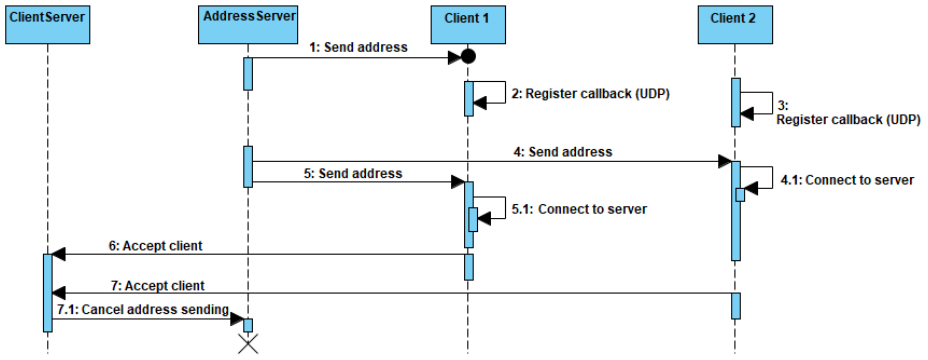

Fig. 1. Sequence diagram for the process of connection to the server

The software for the server side has been developed in JetBrains IDEA Community Edition 2017 development environment [13]. The main activity class contains communication-control nested classes that operate in ordinary client-server manner. Class diagram is shown in Fig. 2. Only the most important operations are illustrated on the diagram.

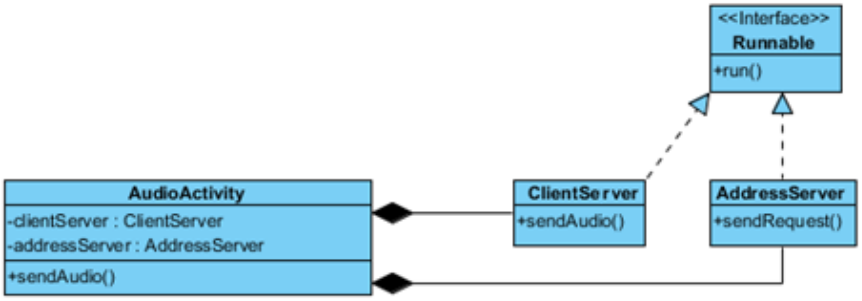

Fig. 2. Class diagram for the server software

To function properly, software components are to be deployed on server and client devices. While server device, e.g. smartphone, requires only application file to be installed, client devices store several script files. Deployment diagram for the system is shown in Fig. 3.

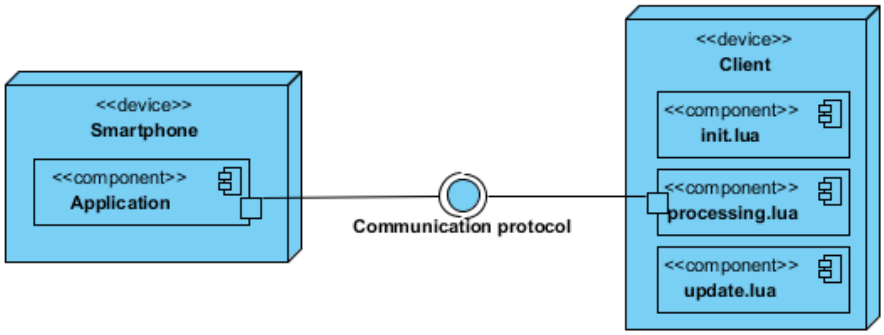

Fig. 3. Deployment diagram for the developed system

ESP8266 are Wi-Fi-modules that support different execution models. They could be pre-programmed using Arduino software. They also can run tiny versions of language interpreters such as Lua or MicroPython adopted for the requirements of the platform (small RAM available, constrained disk storage space, etc.). In the second case program code is distributed among the files (scripts) that are called depending on the user actions. The first way provides performance to the system while the second one is determined to be used for rapid prototyping and development of flexible solutions.

The system has been prototyped using NodeMCU v2 board that includes ESP8266 and simplifies power supply problem for the developer because it can be delivered directly via micro-USB interface. NodeMCU firmware for the client devices was obtained from cloud build service [14]. NodeMCU is a Lua-base firmware that includes interpreter of the Lua instructions. The instructions should be distributed among the .lua-files. NodeMCU programming model supposes intensive usage of

callbacks and this way it reduces resources consumption and simplifies program implementation.

In the implemented platform the security level matches security level of WiFi-protocol. It is enough to prototype and test the overall concept but it should be reviewed thoroughly to prevent possible attacks. For now, they can be performed almost on every stage of communication between device and server. Special attention should be paid to the upgrade stage due to its importance and possible negative consequences (up to fully non-functional device).

4 Testing of the system

The first stage is to perform setup for user network, so devices can connect to it. First, it is required to connect to the device set access point and provide information about user network. In the Fig 4, connection screen for the setup network is shown.

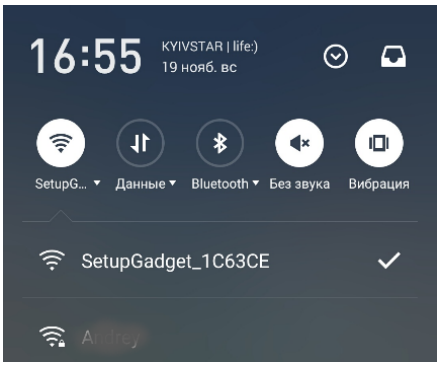

Fig. 4. Selection of the configuration network on smartphone device

The network to connect has name that follows the pattern “SetupGadget_xxxxxx” where the last part can vary according to the device. The setup network requires no credentials to enter.

Built-in functionality of NodeMCU library (enduser_setup) has been used to implement the setup stage. It allows to integrate setup functionality with single function call and takes responsibility for all the further internal processing. From under the hood, it provides a web-page deployed on the device and returns it in case of incoming request. User connects to the device using web-browser and can access configuration page shown in Fig. 5.

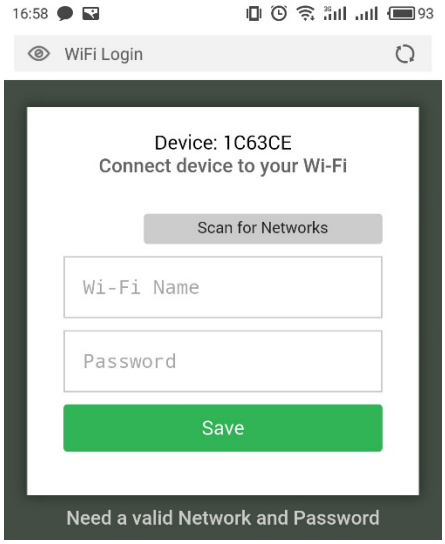

Fig. 5. Page for entering parameters of WiFi-network

As necessary information is filled into the fields, connection attempt can be performed. In case of successful configuration, the device updates page and shows the following message that is demonstrated on Fig. 6.

Fig. 6. Notification about successful connection to the user's network

As main device receives network parameters, it immediately sends them to the client devices in the set. When the devices are turned on next time, they will try to connect to the last configured network automatically. However, if they are supposed to work in different network (even if they have been connected previously), the user should repeat setup procedure.

The developed application has been tested on Meizu M2 smartphone in the network with TP-Link router and two ESP8266 devices where the first device performs the role of temporary server and the second one is a client-only device. The user interface of the application is shown on figure. It sends UDP-messages to help device set to identify network location of the running program. The program notifies user about connected devices showing up an information message. Then, it can transmit audio data to the clients through TCP-connection when user presses the button. The screenshot of the launched application with single connected device is shown in Fig. 7.

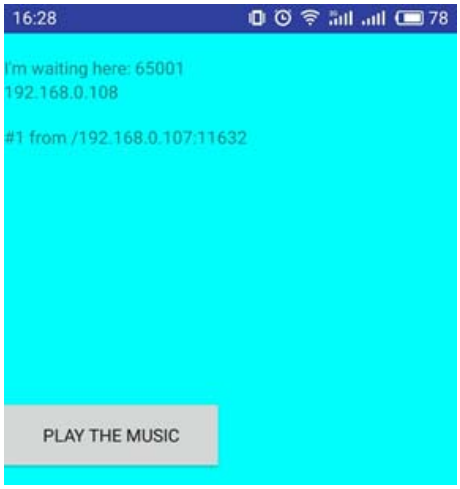

Fig. 7. User interface of the developed application for testing device connections

Fig. 7 demonstrates an application developed according to the previously mentioned class diagram (see Fig. 2). This is a multi-threading application that supports work with multiple clients. The click on the button initiates transmission of the prepared music data to all connected devices.

To summarize the results of the investigation we can state that proposed configuration procedure reduces time of configuration of IoT device set because they are configured simultaneously. Network mechanisms for identification of the server device in customer network have been proposed. The implemented software for both server and client side allowed us to prove correctness of the outlined developments.

5 Conclusions

In the presented paper, we have devised an approach for network configuration of the IoT device set. The approach is based on picking out server device within a set that serves as an access point at the initial stage. It receives information from the user about network to connect to and distributes it among other devices in the set. Thus, user does not need to configure each device individually that reduces time of configuration for full set of devices. Then, stage of communication with server device has been outlined. The server software is to await incoming connections and, in addition, has to send advertising broadcast message via UDP about server address in the network. With the reception of the message, devices can identify network node they need to connect to. The transmission of data for the software level relies on the TCP and stable connections between the server and clients. The case of audio data transmission has been considered in the work and implemented in the application for Android platform using Java programming language. The testing of proposed statements has been done using a pair of ESP8266 devices. All stages of connection procedure have been checked and shown the correctness of the apparatus presented in the paper.

References

1. ESP8266 Overview, https://www.espressif.com/en/products/hardware/esp8266ex/overview, last accessed 2019/03/27.
2. Bovell, M.C., Devlin, T.P., Goodner, A.S., Northway, T.N., Usery, P.H.: Wireless device configuration management. US Patent US8180860B2, 15 May 2012.
3. Lehotsky, D.A., Mannion, M., Nobutani, T.: System, apparatus and method for automated wireless device configuration. US Patent US7940744B2, 20 May 2011.
4. Lane, N.: Development and Integration of Honeywell's One-Wireless Network. Engineering Honours Thesis, Murdoch University (2018).
5. Estrin, D., Girod, L., Pottie, G., Srivastava, M.: Instrumenting the world with wireless sensor networks. In: International Conference on Acoustics, Speech, and Signal Processing, vol.4, IEEE (2001). doi: 10.1109/ICASSP.2001.940390
6. Cerpa, A., Elson, J., Estrin, D., Girod, L., Hamilton, M., Zhao, J.: Habitat monitoring: Application driver for wireless communications technology. In: ACM SIGCOMM Computer Communication, rev. 31(2), pp. 20-41, ACM (2001). doi: 10.1145/844193.844196
7. Ding, J., Cheung, S.Y., Tan, C.-W., Varaiya, P.: Signal processing of sensor node data for vehicle detection. In: The 7th International IEEE Conference on Intelligent Transportation Systems, Washington, USA, 3-6 October, 2004, IEEE (2004). doi: 10.1109/ITSC.2004.1398874
8. Gore, S., Kadam, S., Mallayanmath, S., Jadhav, S.: Review on Programming ESP8266 with Over the Air Programming Capability. International Journal of Engineering Science and Computing, 7(4), 6684-6686 (2017).
9. Naikoti, A., Kodali, R.K.: ECDH based security model for IoT using ESP8266. In: 2016 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), Kumaracoil, India, 16-17 December 2016, IEEE (2016). doi: 10.1109/ICCICCT.2016.7988026

10. Burlachenko, I., Zhuravska, I., Musiyenko, M.: Devising a method for the active coordination of video cameras in optical navigation based on the multi-agent approach. Eastern-European Journal of Enterprise Technologies, 1, 9(85), 17-25 (2017). doi: 10.15587/1729-4061.2017.90863
11. Dave, B., Buda, A., Nurminen, A., Framling, K.: A framework for integrating BIM and IoT through open standards. Automation in Construction, 95, 35-45 (2018). doi: 10.1016/j.autcon.2018.07.022
12. Ideal Modelling & Diagramming Tool for Agile Team Collaboration, https://www.visual-paradigm.com/, last accessed 2019/03/25.
13. IntelliJ IDEA: The Java IDE for Professional Developers by JetBrains, https://www.jetbrains.com/idea/, last accessed 2019/03/27.
14. NodeMCU Custom Builds, https://nodemcu-build.com/, last accessed 2019/03/27.