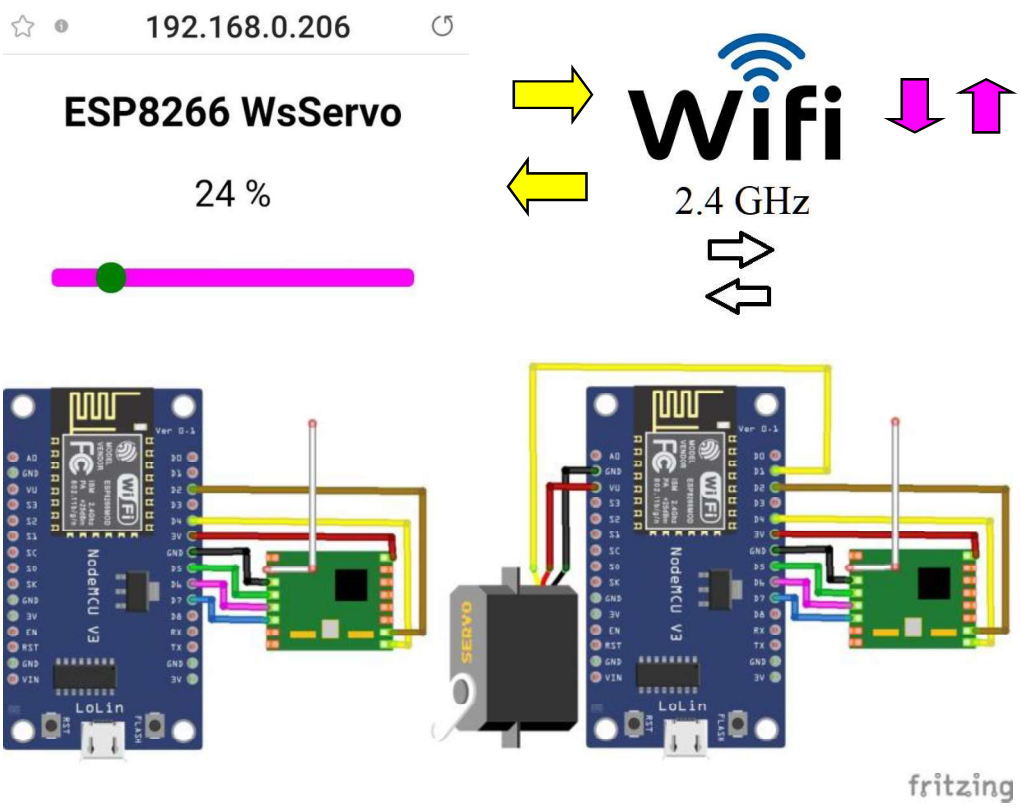
MIHAI TODICA

DATA AND REMOTE CONTROL WITH ESP 8266 AND ARDUINO



PRESA UNIVERSITARĂ CLUJEANĂ

Mihai Todica

Data and Remote Control with ESP 8266 and ARDUINO

PRESA UNIVERSITARĂ CLUJEANĂ

2026

Referenți științifici:

Prof. univ. dr. Radu Fechete

Lect. univ. dr. Anton Mircea

ISBN 978-606-37-3191-4

© 2026 AUTORUL VOLUMULUI.
Toate drepturile rezervate. Reproducerea integrală sau parțială a textului, prin orice mijloace, fără acordul autorului, este interzisă și se pedepsește conform legii.

Universitatea Babeș-Bolyai
Presa Universitară Clujeană
Director: Codruța Săcelean
Str. B.P. Hasdeu nr. 51
400371 Cluj-Napoca, România
Tel.: (+40) 744 687 884
E-mail: editura@ubbcluj.ro
editura.ubbcluj.ro | libraria.ubbcluj.ro

Contents

Introduction **6**

1. Server with ESP8266 and Arduino **9**

1.1. Features and settings 9

1.2. Server and Client 10

1.3. The radio connection 11

1.4. GET, REQUEST and POST instructions 15

1.4.1. Continuous transmission of the data 15

1.4.2. Transmission of data upon request. LEDs control 18

1.4.3. Transmission of data upon request. Servomotor control 20

1.4.4. Combined transmission, on demand and continuous 22

References 24

2. Asynchronous and WebSocket servers **25**

2.1. Servomotor control 25

2.2. PWM control of one RGB led 28

2.3. ON/OFF control of more LEDs 28

2.4. LEDs control with confirmation 29

2.5. Server with MPU 6050 accelerometer 30

References 34

3. Servers with WebSerial library **35**

3.1. Unidirectional text transmission with WebSerial and Android 35

3.2. Bidirectional text and data transmissions with WebSerial and Android 37

3.3. RGB and servomotor control by text messages with WebSerial 38

3.4. Bidirectional connections and feedback with WebSerial 40

3.5. Tracking movements with a PS2 mouse 42

References 45

4. Peer to Peer (P2P) connections **46**

4.1. P2P via Wi-Fi 46

4.2. P2P connections with the RF transceiver 69 47

4.2.1. Modification of the RFM 69 library 49

4.2.2. Bilateral text transmissions with confirmation 50

4.2.3. Data transmissions. Control of one servomotor 50

4.3. P2P gateway connections with RF69 and Android 53

4.3.1. The LEDs control 54

4.3.2. Servomotor control 56

4.4. Gateway for sensor data through Station Mode server 59

4.5. Gateway connections in Access Point mode. Sensor data 62

4.6. Gateway with asynchronous server. The LED control 63

4.7. Gateway with WebSerial. Servomotor control by text messages 66

4.8. Gateway with WebSerial for data sensors....68
4.9. Gateway with RxTx 315/433 MHz modules....70
References....73
5. Gateway with ESP8266 and Arduino Uno Pro Mini....75
5.1. Connecting the RF69 transceiver and loading the codes....75
5.2. Servomotor control through gateway with Arduino Pro Mini....76
5.3. Gateway with Arduino Pro Mini for analog data....78
5.4. Gateway with Arduino Pro Mini for LEDs control....79
References....83
6. Transmission of data and orders trough the internet....84
6.1. Simple commands with Blynk....85
6.2. Digital and analog data with Blynk....87
6.3. Bilateral connections and feedback....89
6.3.1. Controls of LEDs and servomotors....89
6.3.2. DC motor control and feedback....90
6.4. Gateway with Blynk via radio and IR....96
6.4.1. Radio control of a lighting system....97
6.4.2. IR control of LEDs lighting strip....99
6.5. Sending texts with Blynk. Unilateral pager....102
6.6. The use of the TFT display....103
6.6.1. Pager with TFT and Blynk....104
6.6.2. Analog voltmeter and text pager....105
References....106
7. Equivalent ESP8266 modules....107
7.1. ESP8266-01 module....107
7.2. ESP8266 WeMos D1 R2 module....109
7.2.1. LEDs control by generated server....110
7.2.2. One-sided pager with WeMos D1 R2....111
7.2.3. Connection of the Oled I2C display....112
7.2.4. Pager and led control with feedback and Blynk....113
7.2.5. WeMos D1 R2 P2P connections with Joystick Shield and ESP8266....115
7.2.6. P2P connection and feedback with Joystick Shield....119
7.2.7. Simple P2P pager with LCD Shield and WebSerial....123
7.3. ESP8266-WeMos D1 Mini module....124
7.4. ESP 12-E module....126
7.4.1. RGB led control and light sensor....127
7.4.2. Text pager with Oled display....129
7.4.3. Control of DC motors and servomotors....130
7.4.4. Local server and web page with ESP12-E....133
References....134

Introduction

The use of relatively simple automatisms, such as remote control for TV set, lights, air conditioning system, etc., with special devices customized for each system, or with the telephone, seem absolutely trivial and natural things on ours times, and we use these facilities almost instinctively, without bothering to understand how they work. In many cases, these orders are transmitted trough the *Internet* with the help of the personal computer, tablet, or mobile phone, falling into the so-called *Internet of Things*, (*IOT*), category. It is enough to install a dedicated *application* on the phone, or entering a particular *IP* address into a web browser, and the control panel of the controlled system appears on its screen. But not everything was so simple few years ago. These performances were reached through the sustained effort of numerous specialists in the field of electronics and computer science, but also through the contribution of many amateurs passionate by automation. At the beginning the development of such systems was strictly the task of specialists in the field, because it involved the use of specialized and expensive infrastructure, computers, sensor and actuator systems, etc., which were not accessible to the amateurs. However, the last ones expected to have access to an economical, simple and easy-to-program system, compatible with a large category of sensors and actuators, in order to develop theirs own autonomous and intelligent systems. This request conducted to the apparition of the *Arduino* platform, an integrated hardware and software system that includes the above facilities. If at the beginning this platform offered relatively simple operations of collecting and processing the data, in a few years things have evolved, not only in terms of the capacity and speed of data management, but especially in terms of connectivity with different external peripherals, either directly or via the internet. These performances involved the use of new equipment including the *wireless* communication facility, (*Wi-Fi*). So the initial hardware version *Arduino Uno* evolved to more complex models, of which a very popular module is the *Node MCU ESP8266*. Economically, reliable, with high performance and easy to be programmed, this module allowed to the specialists and amateurs to create numerous projects published in specialized papers or freely available on the Internet, but including them into an informational context, showing in a gradual manner its use and possibilities, is less common. It is the purpose of this work, which includes some examples, to familiarize the passionate reader with this field. We insisted on *Wi-Fi* connectivity, a facility that involves the generation and the management of a *web server* and the communication with a partner, the *client*. Frequently, the latter one is a smart device, such as a *PC* or *Android* phone, which displays the information received from the server through a *web page*. There are several possibilities to generate these elements, the *server* and the *web page*, but

only a few of the most common ones have been presented here. We started with *html servers* based on the use of *get*, *request*, *post* instructions, to simplify the creation of the web pages. Relatively simple, these servers generally have modest performances.

Faster servers are the *asynchronous* ones, or those that use particular protocols such *WebSocket*. To exemplify the use of such servers, some projects were presented to control *actuators* such as LEDs, servomotors, direct current motors, but also to collect the data of temperature, humidity, pressure, mechanical displacement, acceleration, from appropriate sensors. Examples with more complex web pages, including customized graphics elements, were also presented.

Communication between server and client can be done either in *Station Mode* through a *local web network*, but without requiring Internet access, or directly, in *Access Point* mode, when the partners communicate without being connected to any local network. This last version allows the connection of the partners in any spatial location, without the constraints related to existence of any local web router. The two situations are exemplified for each type of server.

Generating the web pages is substantially simplified when using specialized libraries such as *WebSerial*, which allows either the transmission of text messages or orders, but also the implementation of the *feedback* function for monitoring the achievements of the orders.

Establishing a direct connection between partners, without involving any *PC* or *Android* device as a *client*, can be done within the so-called *Peer to Peer*, (*P2P*), connection. It is a direct, fast and economical connection between two ESP8266 modules, that allows bilateral transmission of data and commands, but which is limited to the *Wi-Fi* range of the integrated radio modules.

An extension of the P2P communication range is obtained by inserting *radio* or *IR* transmission modules, the so-called *transceivers*, between the *server* and the *client*. In this way a communication *gateway* system is obtained between the Android or PC and the client, which can be use also for the control of commercial automation systems such as air conditioning, lighting, automatic gates, etc.

However the most interesting facilities offered by these modules is the ability to communicate via the Internet, trough stationary or mobile data networks. This is possible by using dedicated applications, generally freely accessible on the Internet or *Play Store*, of which *Blynk* was selected here. It is simple to implement in the Arduino platform, reliable and perfectly compatible with ESP8266 modules. Data, text messages or orders can be transmitted *unilaterally* or *bilaterally* using this application.

In the last part of the work were presented several ESP8266 equivalent modules, which differ from the original only by dimensions and the connecting facilities with external peripherals. Most of the examples presented in the first

chapters can be used with these modules, either directly or with small modifications of the codes.

The work was conceived in two parts, an *edited text* part, which includes the description of the projects, the electrical circuits used and images with the practical realization of some of them, as well as a *data part*, accessible on a *website*, which includes the *codes* and the *libraries* related to the examples presented. That allows the user their direct loading, without the need of the edited part.

https://github.com/napocavox/esp8266-arduino-data-remote-control

The projects included in the work represent the author's own contribution or adaptations of some examples, freely available on the Internet, specifying each time the source of information. The libraries used for coding, as well as certain some images and drawings were taken from freely accessible public sources. The electric circuits were generated with the *Fritzing* software, also freely available to the public.

All the codes and examples included in the work have been effectively tested and practically implemented by the author and are free to be modified and adapted by the reader.

With a medium degree of difficulty, the work is addressed to all robotics/informatics enthusiasts, requiring only minimal knowledge in the field.

Chapter 1
Server with ESP8266 and Arduino

The transmission of data, or the remote control of some electronic devices through a local *network* or the *Internet*, is no longer a special novelty, but the understanding of the basic principles and the use of some simple systems to achieve these goals, could be of interest for some amateurs or other readers. Generally such system contains a programmable *microcontroller* and some other hardware facilities, *sensors* and *actuators*, to acquire, process, transmit and receive wirelessly, (*Wi-Fi*), the data, between the partners. There are several devices that can perform these functions, but here we focused our interest on a relatively simple model, very popular at the beginning of the era of *Wi-Fi* connections, the *NodeMCU ESP8266*, (*Node Micro Controller Unit*), developed by *Espressif Systems*. Not expensive and easy to be programmed through the *Arduino* platform, with very reliable Wi-Fi facilities, this device can be included into the category of *IoT* (*Internet of Things*) systems.

1.1. Features and settings

Based on the *Tensilica Xtensa® 32-bit LX106 RISC* microprocessor, this module can operate at clock frequencies between 80 and 160 MHz, has a *RAM* memory of 128 KB, a *flash* memory of 4 MB, can handle characteristic data strings, *JSON/XML* web pages and *IoT* protocols. For Wi-Fi connections it contains one *802.11b/g/n HT40* type transceiver, allowing it to connect to a local web, or to generate its own network, [1]. It has an integrated 10-bit *ADC* converter connected to one analog input, *SPI and I2C* interfaces, micro *USB* connector, and several *GPIO* pins, (*General Purpose In and Out*), that can be configured for different tasks. Serial communication ensures a data transfer up to 4.5 Mbps. From a constructive point of view there are several variants, of which the most widespread are those equipped with the *CH340* and *CP2102* microcontrollers. In this work we will mostly use the first variant, for which the pin diagram is displayed in figure 1.1.a, but also the second variant, figure 1.1.b. The module is designed for working at 3.3V, but it can be supplied also with +5V through the *USB* port or through the *VIN* pin, because the module contains a voltage regulator. It can also provide an output voltage of +3.3V, available on the pins specified in the diagram. When using peripherals working at 5V voltages, they will be connected to the ESP8266 through a *logic level converter*. Programming is done with the Arduino *IDE*, (*Integrated Development Environment*), platform in C++ language, [2]. When using the *Windows* operating system, several stages must be followed.

1. Installation of Arduino platform, version 1.8.13 or higher;