TEMA T2.1: INOVACIJE U MODERNOM OBRAZOVANJU (Inovativne metode učenja, Tehnologija u službi znanja, Softverske platforme i alati u obrazovanju...)

Paper No.T2.1-1
01433

AN EDUCATIONAL APPROACH TO AN ENCRYPTED DATA TRANSFER IN AN EMBEDDED AND FRONTEND DEVELOPMENT ENVIRONMENT

Ivan Gutai[1], Platon Sovilj[2], Marina Subotin[3], Nemanja Gazivoda[4], Bojan Vujičić[5], Zdravko Gotovac[6], Milan Šaš[7]
[1,2,3,4,5,6,7]*University of Novi Sad, Faculty of Technical Sciences, Novi Sad, Serbia*
[1]gutai@uns.ac.rs, [2]platon@uns.ac.rs, [3]marina.bulat@uns.ac.rs, [4]nemanjagazivoda@uns.ac.rs, [5]bojanvuj@uns.ac.rs, [6]zdravko.gotovac@uns.ac.rs, [7]milansas@uns.ac.rs

Abstract: *Prior to IIoT (The Industrial Internet of Things), embedded programming and frontend development didn't even found themselves in the same sentence. Hardware that enables seamless integration of these two vast areas is the Espressif ESP32* MCU (MicroController Unit). *Once numerous devices were connected to the internet, the security of data became the focal point. ESP-NOW technology is Espressif's proprietary solution for wireless data transfer. It can be encrypted and it enables secure communication between multiple ESP32's. HTTPS (Hypertext Transfer Protocol Secure) is a protocol that increases security on the internet.*

This article provides a guide for configuring a development environment for ESP32, ESP-NOW example, HTTPS server example, and programming practice intended for error handling. In addition, it offers a responsive example of a graphical user interface of the IIoT device. At the moment, there are numerous possible paths in embedded and frontend programming. This paper follows the path using ESP32 as the hardware, and C++ for the firmware. *JavaScript, HTML5, and CSS3 are unavoidable parts of modern industrial devices. Hence, the article provides an example of using JavaScript Highcharts library. This combination of hardware and software is the solution that costs less than 10$, which makes it acceptable in countries under development. Highcharts library is proprietary, and for educational purposes, it is used under Creative Commons (CC) Attribution-Non-Commercial license.*

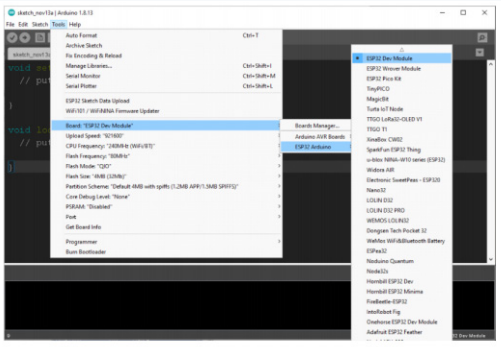
Key Words: IIoT, embedded programming, frontend development, ESP32, ESP-NOW, HTTPS, responsive design, SPIFFS, C++, JavaScript, HTML5, CSS3, Highcharts.

1. INTRO

Smart devices became an integral part of our work and hobbies. There is a rapid increase in the number of those devices and also in their functionalities. In that environment, we must be aware of information security, and we shouldn't forget good programming and engineering practices. Today we all have an equal chance to develop a prototype of an IIoT device. Also, it can be part of our home Wi-Fi network, and we can control it from the browser. This paper represents a guide for pointing in the right direction, with the intention to kick-start readers career as an embedded developer or as a frontend programmer. There is a practical example of the creation of one IIoT device. The hardest part always was choosing the right hardware or the right technology stack. Authors choose ESP32[1], C++, JavaScript, HTML5, and CSS3. ESP-NOW [2] technology enables interconnecting a large number of ESP32 devices, which communicate over Wi-Fi. Plain text is vulnerable to alteration in web applications, as is unencrypted data traveling on Wi-Fi. Always when we get a chance to encrypt something, we must embrace it. ESP-NOW data can be encrypted using LMK (Local Master Key), which must match on senders and receivers. Adding sensors or relays on this hardware is another lengthy topic and readers can choose between various open-hardware solutions or a proprietary alternative like Mikroelektronika. After completion of the hardware part, SPIFFS (Serial Peripheral Interface Flash File System) memory is used for deployment of the web application. SPIFFS contains a web application that is created with modern and free tools. The only difference is in hosting and this application isn't on the typical webserver but is deployed on ESP32.

2. CONFIGURING INTEGRATED DEVELOPMENT ENVIRONMENT

It all starts with the installation of Arduino IDE *(Integrated Development Environment)* [3]. ESP32 needs to be added to the list of existing development boards: "Arduino IDE, File, Preferences, Additional Boards Manager URLs [4]." Expanding the list of existing development boards with ESP32: "Tools, Board, Boards Manager, esp32, and Install". After successful installation, on the list of available boards, "ESP32 Arduino" will appear, and plenty of options will become available, including "ESP32 Dev Module". Adding the "ESP32 Sketch Data Upload" option enables uploading files to SPIFFS. We can depict that as the deployment of frontend code to the ESP32 system. *For code deployment plugin [5] is needed and it needs to be placed under the Arduino IDE folder.* The plugin needs to be placed on the path: "C:\Program Files (x86)\Arduino\tools\ESP32FS\tool\esp32fs.jar". Dark theme for Arduino IDE is available at [6]. New theme files need to replace existing ones in the folder: "C:\Program Files (x86)\Arduino\lib\theme". The original theme might be preserved, as well. Files needed for the usage of ESP32 are on address [7]. Files needed to be extracted in the local Arduino folder: "C:\Users\Ivan Gutai(or any other User Name)\Documents\Arduino\hardware\espressif\esp32". It is worth mentioning that the esp32 folder is indeed just renamed the arduinoesp32-master folder. If the ESP32 development board isn't visible in Device Manager, the programmer driver needs to be installed. Most popular are CH340 [8] and CP210x [9]. After successful configuration, IDE looks like IDE shown in Picture 1.

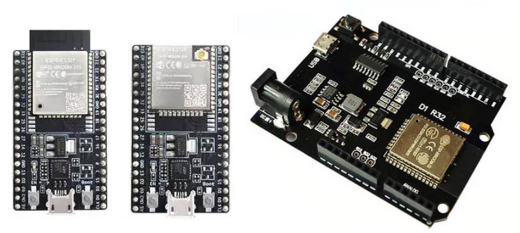


```
sketch_nov13a | Arduino 1.8.13
Board: "ESP32 Dev Module"
Upload Speed: "921600"
CPU Frequency: "240MHz (WiFi/BT)"
Flash Frequency: "80MHz"
Flash Mode: "QIO"
Flash Size: "4MB (32Mb)"
Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"
Core Debug Level: "None"
PSRAM: "Disabled"
```

Picture 1. Arduino IDE, configured for ESP32.

3. SELECTING THE RIGHT HARDWARE

ESP32 has multiple variations. For this project ESP32-DevKitC-32D [10] with an integrated antenna made of copper is used. If we needed to amplify the Wi-Fi signal ESP32-DevKitC-32U [11] would be an option. If someone is comfortable working with Arduino Uno, there is ESP32 which is very similar physically and it is Wemos D1 R32 [12]. Picture 2 represents the three mentioned types of ESP32 development kits.



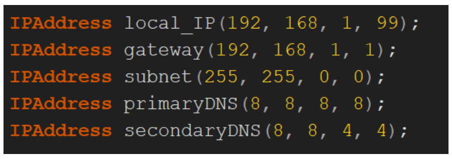
Picture 2 ESP32 development kits: ESP32-DevKitC-32D, ESP32-DevKitC-32U, and ESP32 Wemos D1 R32.

4. DELVING INTO ESP-NOW TECHNOLOGY

The entire tutorial is dedicated to configuring ESP-NOW, with multiple senders or multiple receivers [13]. ESP-NOW is used only if we have 2 or more connected ESP32 devices. It is worth mentioning that the data structure on the sending and the receiving end must be identical. If we decided to use encrypted communication, LMK must be identical on receivers and transmitters. Also, we can choose from channel 1 to channel 14 for Wi-Fi communication. Packets of 250 bytes of data are sent and received dozens of times every second, which makes them suitable for fast-changing data, e.g. data acquired from devices that are measuring spatial position or angular momentum.

5. ASSIGNING FIXED LOCAL IP ADDRESS TO DEVICE

Every time that this device connects to the home or industrial Wi-Fi network, it gets a different local IP address. DHCP (Dynamic Host Configuration Protocol) is responsible for that, which is fine for developers, not for regular users. We need to assign a fixed local IP address, which fits the parameters of the WI-Fi network. Picture 3 represents the configuration which we need to apply.



```
IPAddress local_IP(192, 168, 1, 99);
IPAddress gateway(192, 168, 1, 1);
IPAddress subnet(255, 255, 0, 0);
IPAddress primaryDNS(8, 8, 8, 8);
IPAddress secondaryDNS(8, 8, 4, 4);
```

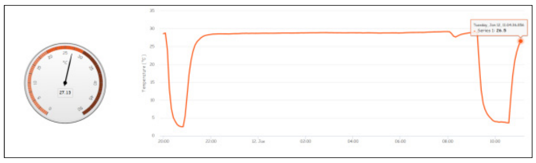
Picture 3 Local IP address is set, and Google's DNS (Domain Network Server) servers are used.

6. DELVING INTO SPIFFS

The library that enables the creation of home REST (Representational State Transfer) is available at Github [14]. In Arduino IDE, all additional libraries can be downloaded in this way: "Tools, Manage Libraries, search, and install". Keywords are: "esp32 HTTPS", and install : "ESP32_HTTPS_SERVER" library. From the stated library, an example is "REST-API". This library can generate a Self Signed certificate that is useful for development and enables usage of HTTPS. Files with extensions .html, .js, .css and other need to be placed at: "REST-API\data\public" folder. This can be deployed to SPIFFS with an option: "ESP32 Sketch Data Upload", under the Tools section.

7. **CREATING A WEB** INTERFACE OF IIOT DEVICE

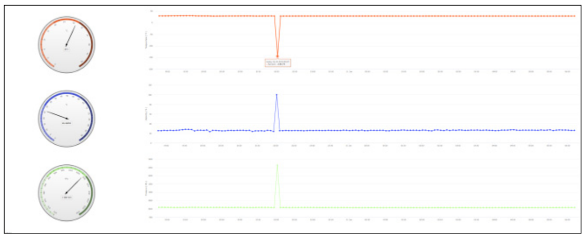
Picture 4 represents part of the responsive web interface with the Highcharts library [15], based on JavaScript.



Picture 4. Part of the responsive web interface consisting of an analog scale and a chart.

8. ERROR HANDLING AND PROBLEM AVOIDING

During the creation of complex systems, we need to take into account dozens of complex parts. Many people tend to forget basic principles that are as old as programming. To handle and ultimately avoid errors, we must prevent faulty or incomplete information. Picture 5 represents the web user interface which is feed with the faulty signal from the hardware unit.



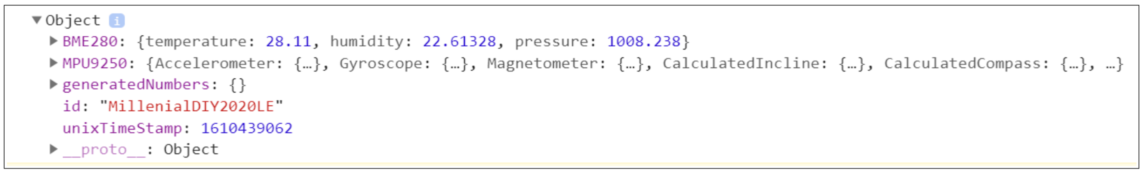
Picture 5. Web user interface which is feed with the faulty signal from the hardware unit.

Technically speaking this is not an error, but it is a number that came out of the range. This is a reminder that we must be aware of the range of numbers that are we expecting on every single component of a complex system. We must take into account that if we take values from sensor Bosch BME280 [16], for measuring environmental parameters, including atmospheric pressure, that range of values which we expect is from 300 hPa to 1100 hPa. Everything outside that range is the result of some kind of error, e.g. unnecessary overloading of the hardware side, which results in intermittent faulty readings of sensors. This kind of information can't reach the user, because there are faulty. Also, they shouldn't be hidden. User must be provided with precise instruction, what to do, without too many technical details. The frontend has a plethora of advanced options for providing users with information, e.g. toast messages or filling fields with specific messages. Developers have the creative freedom to choose from if..else if..else or try...catch...finally blocks. Doing good analysis prevents the showing of faulty information, like atmospheric pressure in a living room of 1264 hPa. To be sure that we are using up-to-date information it is recommended to use the timestamp, Unix like or another kind. Time can be read from the hardware component DS3231 [17], over NTP (Network Time Protocol) server [18], or in any arbitrary way. It is worth mentioning that timestamp is an essential part of the stream of data. This ensures that the chart is not updated if there is no fresh data. This ensures the prevention of spreading misinformation.

9. TEST DEVELOPMENT

Development and test ratio must be at least in 1:5 ratio. If the device would have an industrial application, there is even more testing involved. Extreme values can be useful during testing because if there is a slight chance of under 1 % for something to happen, it will happen eventually. As developers, we must be prepared to deal with it and have it under control. One of the advantages of a system based on ESP32 which has a web interface is the possibility to be used as a test bench. It is possible to connect a lot of sensors, assign IDs to them and monitor their behavior, e.g. reading sensor values every 5 minutes for 24 hours. The microcontroller has enough memory to monitor these parameters even without the web interface. We would take a web interface option because it enables us to monitor values from dozens of sensors and show results on big screen devices like smart TV in UHD (Ultra High Definition) resolution of 3840 px × 2160 px. Writing media queries for such a large resolution isn't more complex than writing media queries for mobile phones and tablets, and it begins from: "@media only screen and (max-width: 3840px) {}".

In the middle of writing firmware and creating the web application, it is useful to monitor "raw" data. Picture 6 represents logged data in the browser's console, which is received from the hardware.



```
▼Object
  ▶BME280: {temperature: 28.11, humidity: 22.61328, pressure: 1008.238}
  ▶MPU9250: {Accelerometer: {…}, Gyroscope: {…}, Magnetometer: {…}, CalculatedIncline: {…}, CalculatedCompass: {…}, …}
  ▶generatedNumbers: {}
   id: "MillenialDIY2020LE"
   unixTimeStamp: 1610439062
  ▶__proto__: Object
```

Picture 6. Example of data logging into the console of the web browser.

Another programming practice that is applied is the meaningful naming of variables and appropriate grouping of them. Timestamp 1610439062 represents how many seconds are passed from January 1st, 1970 and represents January 12th, 2021 at 08:11:02.

10. CONCLUSION

This paper is intended for all ambitious people, that are entering the world of device creation and programming. Also, intended for experienced developers and makers. This paper is proof of the concept that if you have enthusiasm, you can start building an industrial-grade device, for just under 10 $. It makes it suitable for emerging economies.

11. ACKNOWLEDGMENT

This paper is supported by Centre for Metrology at Faculty of Technical Sciences and by projects KALCEA "Knowledge Triangle for a Low Carbon Economy" 618109-EPP-1-2020-1-EL-EPPKA2-CBHE-JP and "Razvoj naučno-stručnih metoda u oblasti metrologije, industrijsko-tehničkih merenja u digitalnom konceptu Industrije 4.0, biomedicinskih mernih sistema i kognitivnih neuronauka primenom napredne metodologije i digitalne tehnologije".

12. REFERENCES

[1] Internet page https://docs.espressif.com/projects/esp-idf/en/latest/esp32/
[2] Internet page https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_now.html
[3] Internet page https://www.arduino.cc/en/software
[4] Internet page https://dl.espressif.com/dl/package_esp32_index.json
[5] Internet page https://github.com/me-no-dev/arduino-esp32fs-plugin/releases/
[6] Internet page https://github.com/jeffThompson/DarkArduinoTheme
[7] Internet page https://github.com/espressif/arduino-esp32
[8] Internet page https://learn.sparkfun.com/tutorials/how-to-install-ch340-drivers/all
[9] Internet page https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers
[10] Internet page https://eu.mouser.com/ProductDetail/Espressif-Systems/ESP32-DevKitC-32D?qs=%252BEew9%252B0nqrDsObWEpDx6YQ==
[11] Internet page https://eu.mouser.com/ProductDetail/Espressif-Systems/ESP32-DevKitC-32U/?qs=%252BEew9%252B0nqrCEVvpkdH%2FG5Q%3D%3D
[12] Internet page https://www.aliexpress.com/item/4001136108709.html?spm=a2g0s.9042311.0.0.7a9b4c4dMRyFWh
[13] Internet page https://randomnerdtutorials.com/esp-now-esp32-arduino-ide/
[14] Internet page https://github.com/fhessel/esp32_https_server
[15] Internet page https://www.highcharts.com/
[16] Internet page https://www.bosch-sensortec.com/products/environmental-sensors/humidity-sensors-bme280/
[17] Internet page https://www.maximintegrated.com/en/products/analog/real-time-clocks/DS3231.html
[18] Internet page https://www.pool.ntp.org/zone/rs