The Working Principles of ESP32 and Analytical Comparison of using Low-Cost Microcontroller Modules in Embedded Systems Design

Husam Kareem
Department of Automation and Applied Informatics
Budapest University of Technology and Economics
Budapest, Hungary
e-mail: hus_almagsoosi@edu.bme.hu

Dmitriy Dunaev
Department of Automation and Applied Informatics
Budapest University of Technology and Economics
Budapest, Hungary
e-mail: dunaev@aut.bme.hu

2021 4th International Conference on Circuits, Systems and Simulation (ICCSS) | 978-1-7281-6752-7/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICCSS51193.2021.9464217

***Abstract*—With the emergence of many low-cost development platforms, designing an embedded system has become more flexible using different microcontroller platforms. However, it is necessary to consider various factors before selecting the brain of embedded systems, the microcontroller. This paper compares two commonly used development boards in designing embedded systems with one of the latest boards manufactured for high-performance, low-cost embedded systems design called ESP32. The comparison is based on hardware specifications, as well as programming tools. A Wi-Fi analyzer system was designed and implemented as an example of using the ESP32 module in low-cost, low-energy embedded systems design. In addition to the ESP32, a Graphical LCD is used in the design as an output device. However, there is a lack of literature on establishing serial communication between the ESP32 module and the Graphical LCD. Therefore, this work presents a clear explanation of connecting the Graphical LCD to the ESP32 module using only three GPIO pins.**

Keywords-ESP32, Arduino, embedded systems, Wi-Fi, Graphical LCD, low-cost microcontrollers

I. Introduction

The recently growing demand for the control and automation of a wide variety of devices and gadgets has led to a rapid expansion in embedded systems markets. Accordingly, various microcontroller modules have been emerged along with many competitive advantages like portability, affordable cost, USB-based, readily available open-source programming tools.

Designing an embedded system can be described as the technique of choosing and constructing the compatible combination of software and hardware components to obtain system design goals such as efficiency, speed, reliability, and affordability. Each embedded system involves at least one microcontroller unit acting as a brain of the entire system, but in a way that it is hidden from the user [1]. There are various microcontroller units available on the market and vastly used in the design and development of embedded systems. However, Arduino is one of the most widely used low-cost open-source boards for designing and prototyping embedded systems.

The main motivation of this work is to focus on the working principles of the new powerful board from Espressif systems, ESP32 module [2]. It discusses an application example that represents a practical implementation of using ESP32 board in embedded systems design. Besides, it presents a detailed comparison between the ESP32 vs. the selected popular open-source Arduino development boards: Arduino UNO and Mega2560 in terms of hardware specifications and programming tools.

II. Related Work

Recently, several embedded systems and the Internet of Things (IoT) devices have been implemented based on contemporary low-cost microcontroller boards. Since this study's major concern is to make an analytical comparison for low-cost microcontroller modules that are commonly used in embedded system design, this section highlights the very related work that provides a comparative analysis using the ESP32 board. The authors of [3] discussed a comparative study of the ESP32 module in IoT devices' design. The study provides a detailed comparison between ESP32 against three different competitive boards that are used to design IoT devices: ESP8266, CC32, and Xbee modules. The comparison shows that ESP32 outperforms competitive boards in the design of IoT devices and applications in terms of functionalities and cost [3]. This is mainly due to the dual-core structure of ESP32, which makes it a perfect option for IoT applications and embedded systems design.

III. Technical Specifications and Functions of Arduino and ESP32 boards (Hardware)

Several microcontroller boards for designing embedded systems have been presented from different manufacturers in the recent few years. On the one hand, some of these boards are intended for very advanced projects, and they are relatively costly, like zedboard [4]. On the other hand, several open-source microcontroller boards have been released at a relatively low price, like Arduino boards [5] and ESP32 [2]. Thus, the boards selected for the analytical comparison are Arduino UNO and Mega2560 since they are one of the most used modules in designing low–cost embedded systems. Besides, the ESP32, due to its competitive specifications that combine such advantages as a low-cost and high-speed core processor board, is discussed.

A. Arduino UNO

1) System and Memory: The UNO is a microcontroller board based on Atmega328P [6]. The ATmega 328P is a low-power 8-bit Reduced Instruction Set Computer (RISC) with one Atmel AVR core with 16-Mhz CPU Speed. Moreover, the embedded memories are a combination of three types of memories represented by 1-Kb EEPROM, 2-Kb SRAM, and 32-Kb flash memory.

2) Inputs and Outputs: The UNO board includes 16-I/O pins (6 of which can be used for PWM output). Also, it contains 6-analog input pins A0-to-A5 [7]. It is worth mentioning that the UNO board does not have an analog output pin. Therefore, when an analog output is required from the Arduino UNO based project, the PWM pins are used to compensate for this shortage. A top view of the essential components available on the Arduino Uno board is shown in Fig. 1.

3) Communication Principles: There are three different serial communication protocols: UART, SPI, and I2C. The Universal Asynchronous Receiver/Transmitter UART simple communication protocol needs two pins to establish its interface. Arduino boards must depend on the UART protocol to communicate with computers using the onboard USB-to-Serial convertor. The UNO pins for UART communications are pin 0 (RX) and pin 1 (TX) for data receiving and transmitting, respectively [6]. The SPI requires 4-pins to establish serial communication over Arduino UNO using the SPI library. These pins are as follows: pin 10 for slave-select (SS), pin 11 master-out/slave-in (MOSI), pin 12 for master-in/slave-out (MISO), and pin 13 for serial clock (SCK) [1]. Moreover, the UNO board provides a serial interface using the I2C protocol. It requires using the Wire library and two pins: the data line (SDA) on analog pin A4 and the clock line (SCL) on analog pin A5.

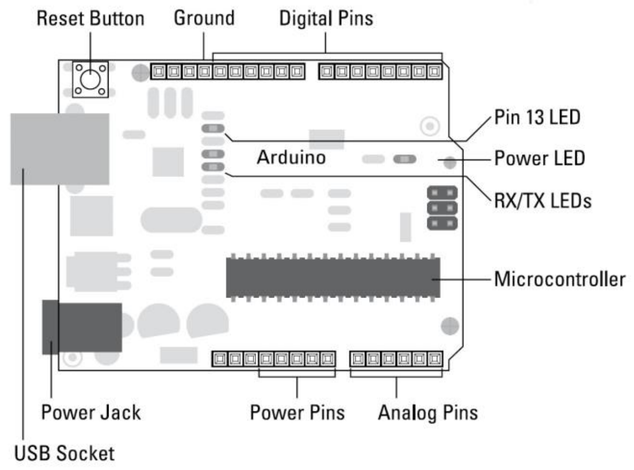


Figure 1. Arduino UNO with all essential components labeled.

B. Arduino MEGA2560

1) System and Memory: The MEGA2560 board is manufactured using the Atmega2560 microcontroller [8]. Atmega2560 microcontroller is an ultra-low-power 8-bit AVR enhanced-RISC architecture with 16-MHz CPU Speed. Its enhanced RISC- architecture allows executing powerful instructions within a single clock cycle. Consequently, the MEGA2560 provides throughputs approaching 1 MIPS per MHz, leading the embedded systems designer to enhance power consumption versus processing speed. Three types of memory constitute the embedded memory system of the MEGA2560 microcontroller: 4-Kb EEPROM, 8-Kb SRAM, and 256-Kb flash memory [8].

2) Inputs and Outputs: There are 54 digital I/O pins on the Arduino MEGA2560, 15 of which can generate PWM output. Furthermore, it has 16-analog input pins A0-A15 [8]. This board does not include an analog output pin. Like most Arduino boards, which do not contain any analog output, it uses PWM functions whenever an analog output is required.

3) Communication Principles: The communication protocols available on the UNO board are also used on the MEGA2560 board. However, they are different in two aspects: the pin location for each protocol and the number of pins used for the UART protocol.

Regarding the UART serial communication protocol, Arduino Mega2560 can communicate with four devices simultaneously using serial: Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX). As mentioned above, the SPI requires 4-pins to establish serial communication over Arduino boards. These pins, on the Mega2560 board, are as follows: pin 53 for (SS), pin 52 for (SCK), pin 51 for (MOSI), and pin 50 for (MISO) [7]. The I2C protocol also requires the Wire library and two pins: the data line (SDA) on pin 20 and the clock line (SCL) on pin 21. Fig. 2 shows a front view of the Arduino Mega2560.

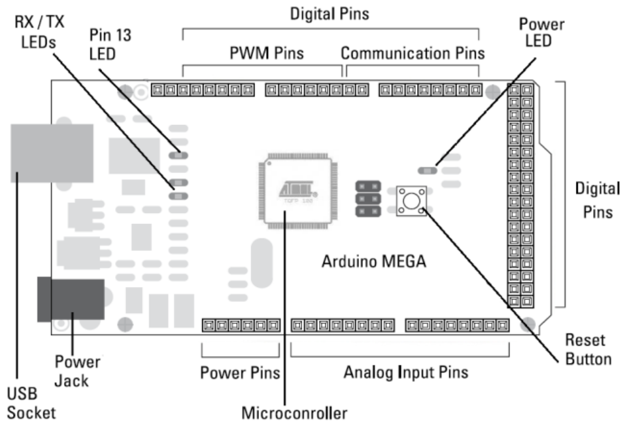


Figure 2. Arduino Mega2560 with all essential components labeled.

C. ESP32

1) System and Memory: A dual-core, 32-bit, Harvard architecture, Tensilica LX6 microcontroller with adjustable CPU speed from 160 to 240 MHz is combined in the ESP32 prototyping board [3]. Names of the two CPUs are "PRO_CPU" for protocols and "APP_CPU" for applications.

However, those CPUs are interchangeable for most purposes [2]. Therefore, the ESP32 board can be suitable for complex systems requiring very high processing and power efficiency. The low-power ESP32 board involves extra features like power scaling and power modes [9].

Furthermore, this board includes on-chip memory (internal memory) and onboard memory (external memory). SRAM and ROM provide the internal memory; both are used for specific functions: 448 KB of ROM is used for booting and core functions, and 520 KB of SRAM is used for data and instructions. The ESP32 board supports up to 16-MB of ISP flash RAM [3].

2) Inputs and Outputs: The ESP32 development board contains 36 GPIO pins [9], 18 of which are representing the ADC, while 2 of the pins are dedicated to DAC functionality. Therefore, this board has real analog output pins, while this feature is not available in most competitive boards (e.g., in the majority of Arduino boards). Besides, all pins that can operate as output can generate PWM output except (GPIOs 34 to 39). Furthermore, some other pins are dedicated to extra functionalities, which will be explained later.

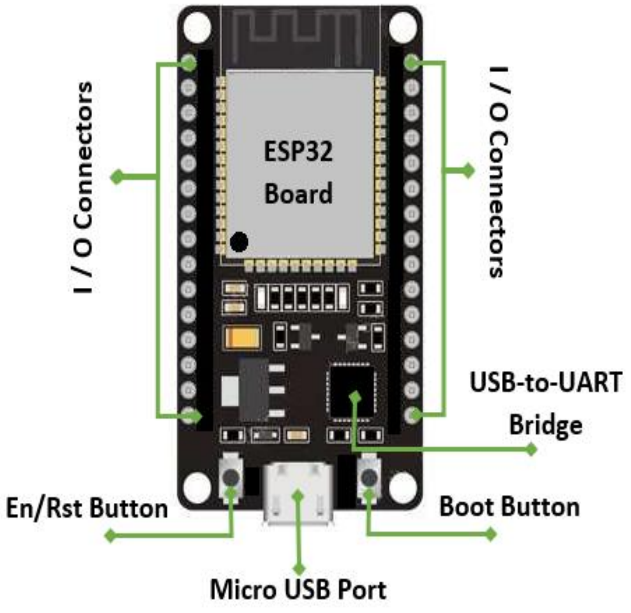


Figure 3. Essential components of the ESP32 board

3) Communication Principles: The ESP32 development board can communicate with the outside world using two main approaches: wired and wireless communication.

Wireless communication is accomplished based on two principles: Wi-Fi and Bluetooth. The ESP32 includes full 802.11 b/g/n/e/i WLAN MAC. It also provides a P2P group operation complaint using the ultimate P2P protocol [3]. Therefore, the ESP32 can function as a station and get connected to the Internet or server and access point to provide an interface that allows the user to control an embedded system by a mobile application. Regarding Bluetooth, the microcontroller board provides v4.2 BR/EDR and Bluetooth Low-Energy (BLE) with operation speed up to 4.0 Mbps.

The ESP32 supports the same wired communication protocols as Arduino boards (ISP, UART, and I2C). However, the ESP32 board offers additional options such as the control area network bus (CAN) and I2S serial bus interface [10]. Another feature that gives extra advantages for the ESP32 board is that it already includes different sensors: temperature sensor, touch sensor, and hall sensor. Table 1 provides a detailed comparison between the ESP32, Arduino MEGA2560, and Arduino UNO boards. Fig. 3 shows the pinout configuration of the ESP32 board.

IV. Programming Tools (Software)

There must be an integration between hardware components and a developing software tool and programming language to design any embedded system. Hence, for hardware programming, a development tool (software) is required compatible with the selected microcontroller [11]. Therefore, for each microcontroller available for embedded systems design, there should be at least one open-source programming tool freely available for design engineers.

A. Arduino Programming

Programming the Arduino boards is accomplished using a free, open-source, and easy-to-use integrated development environment (IDE). The Arduino IDE is a cross-platform software designed to run under different operating systems like Mac OS, Linux, and Windows [12]. A user can write, compile, and upload the programs (called sketches) to any Arduino board. The Arduino sketches can be written using the C++ programming language. Users with good knowledge of C/C++ programming might have an extra advantage in coding Arduino sketches.

B. ESP32 Programming

The official development framework for programming the ESP32 is called ESP-IDF: Espressif Systems Internet of Things development framework, available on a Git-Hub repository. The ESP-IDF was designed with Linux compatibility in mind. Although it can be run on Windows, it requires special software such as MSYS2 [3]. Furthermore, ESP-IDF-Template is necessary to start a project since it provides all essential files for a compilation process. The standard language for programming the ESP32 is C since most of its API libraries are offered in the C programming language. The ESP32 can also be programmed using the Arduino programming IDE. Since this microcontroller follows an open-source paradigm, many other options are also available for programming the ESP32, such as JavaScript and LUA.

TABLE I. DEVELOPMENT BOARDS COMMONLY USED IN LOW-COST EMBEDDED SYSTEMS DESIGN

Microcontroller board	***ESP32***	***Arduino MEGA2560***	***Arduino UNO***
Details			
Processor type	Tensilica Xtensa LX6 (32-bit Dual-Core)	ATmega2560 (8-bit single core)	ATmega328P (8-bit single core)
Processor speed	160/240 MHz	16 MHz	16 MHz
SRAM (KB)	520	8	2
ISP Flash RAM	2 MB (up to 16 MB)	256 KB	32 KB
Nonvolatile memory	448 KB of ROM	4 KB of EEPROM	1 KB of EEPROM
Operating voltage	2.3V. to 3.6V.	5V. (7V. – 12V. recommended)	5V. (7V. – 12V. recommended)
Operating current	80 mA average	500mA. Maximum	500mA. Maximum
Programming language (s)	ESP-IDF Linux based, Arduino IDE using C++, JavaScript, LUA	Arduino IDE using C++	Arduino IDE using C++
Open-source	Yes	Yes	Yes
Size (m.m)	Length 25.5 × width 18.0	Length 101.52 × width 53.3	Length 68.6 × width 53.4
Wireless communications			
WiFi	802.11 b/g/n/e/i	Not available	Not available
Bluetooth	4.2 BR/EDR + BLE	Not available	Not available
Inputs and outputs			
UART	✓	✓	✓
SPI	✓	✓	✓
I2C	✓	✓	✓
I2S	✓	Not available	Not available
CAN	✓	Not available	Not available
GPIO	36	54 Digital + 16 Analog input	14 Digital + 6 Analog input
PWM	All output pins except (GPIOs 34 to 39)	15	6
ADC	18	16	6
DAC	2	0	0

V. PRACTICAL EXAMPLE OF EMBEDDED SYSTEM DESIGN USING ESP32

Applications that can comprise the ESP32 board as a hardware core are limited to embedded systems design and include various IoT applications. Therefore, the presented example can be attributed to both the embedded systems design and IoT. This paper presents a portable Wi-Fi analyzing system that detects the surrounding Wi-Fi networks and shows various details about them, such as SSID, signal strength, MAC address, and encryption type.

VI. STATEMENT OF THE PROBLEM

The use of the Internet is indispensable for most people; thus, it has become an essential aspect of humans' daily lives. Nowadays, one can find dozens of Wi-Fi networks with Internet access in various places such as university campuses, restaurants, shopping malls, etc. However, cannot get connected to any arbitrary network as it can be either password protected or use encrypting techniques not supported by all device. Therefore, a portable Wi-Fi analyzer can help decide which Wi-Fi network is more suitable for connection in terms of open/protected or signal strength.

There is a lack of research examples of connecting the ESP32 board to a Graphical Liquid Crystal Display (GLCD). Therefore, the system design will implement and clearly illustrate the connection between the ESP32 and the GLCD using the minimal number of general-purpose input/output (GPIO) pins. This paper also provides a comprehensive explanation of the libraries and programming code used to establish communication between the GLCD and ESP32.

VII. SYSTEM DESIGN

The project's prototype design and implementation include the hardware and the software. The hardware constitutes the system's physical architecture, while the software comprises the rules and protocols that control the hardware [13].

A. Hardware

The hardware components used in the system design are the ESP32 board and a Graphical LCD (GLCD 128×64) with the ST7920 controller type. This design's novelty is that data is sent to the GLCD using only three GPIO pins of the ESP32 module using the serial data transfer technique. The GLCD has an advantage in receiving data in parallel or serial mode from the microcontroller. To select the desired mode, pin 15 (PSB) on the GLCD must be connected to VCC or

GND for setting parallel or serial data transfer mode, respectively.

The serial data communication technique used in the system is the SPI protocol. As mentioned before, the SPI protocol requires four pins to establish its communication (SCK, MISO, MOSI, and SS). However, the developed system needs only three pins since the data communication process in this design involves only data transmission from the master (ESP32 module) to the slave (GLCD module). Therefore, the pin used for data transmission from slave to master in SPI protocol (MISO pin) is not required and does not need to be connected. The three other SPI pins are connected as follows: the master-out slave-in (MOSI) pin is connected to GPIO23 on the ESP32 board and pin 5 (R/W) on the GLCD. The system clock (SCK) pin is connected to GPIO18 on the ESP32 board and pin 6 (E) on the GLCD. The slave select (SS) pin is connected to GPIO5 on the ESP32 board and pin 4 (RS) on the GLCD. The rest of the pins used on GLCD are connected either to the VCC pin or GND pin, as illustrated in Fig. 4.

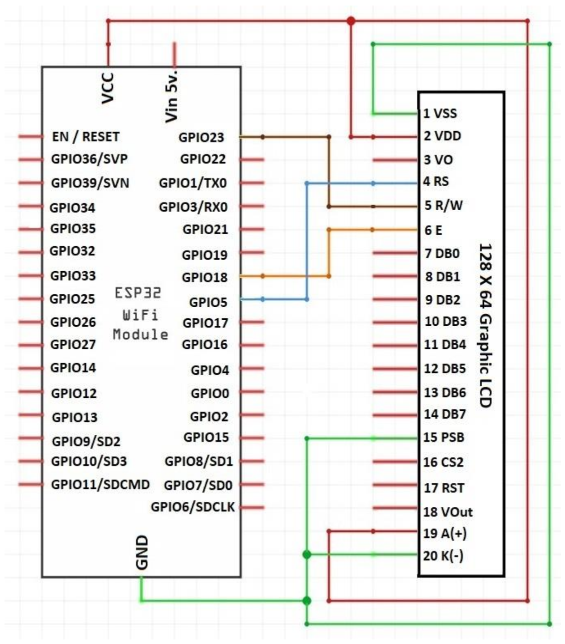


Figure 4. The schematic diagram for the system interconnection

B. Software

Programming the ESP32 can be divided into two main tasks. The first is scanning the in-range Wi-Fi networks, while the second is showing each network details (parameters) on a GLCD. To implement both, it requires a programming tool compatible with the ESP32 module. As it was mentioned, programming the ESP32 board can be done using several programming tools; Arduino IDE is selected for this job. Based on the design tasks, two libraries are required: (i) for Wi-Fi scanning < Wi-Fi .h>, which is available in Arduino IDE, and (ii) for passing and visualizing the results on the GLCD <U8g2lib>. The U8g2lib [14] is a key solution for establishing serial communication between various programming modules and different screens based on I2C or SPI protocols. The major challenge faced during the system design was that the U8g2lib does not support the ESP32. However, this problem could be successfully solved by adding the following additional piece of code:

U8G2_ST7920_128X64_1_SW_SPI u8g2 (U8G2_R0, 18, 23, 5).

Here ST7920 stands for the controller type of GLCD, 128×64 is the size of the GLCD screen, U8G2_R0 means that no rotation in the screen display is required, and finally 18, 23, 5 refers to GPIO pins of SPI protocol on the controller board (SCK, MOSI, and SS pins). During the research, various controller boards have successfully connected to the GLCD, even those not supported by the U8G2 library by default. This paper provides actual parameters in the line of code above for a specific design.

VIII. System Operation

The Wi-Fi analyzer functionality is divided into two parts: the scanning initiates the operation, and the displaying finalizes the process and shows up the results.

A. Scanning Stage

The on-chip Wi-Fi module scans and detects network(s) in range using [Wi-Fi.scanNetworks()] function. If no network is detected, the system shows a message "No networks found." If Wi-Fi networks are detected, the system requests the detected networks' parameters, such as the SSID, encryption type, RSSI level, and MAC address.

B. Displaying Stage

This stage follows the scanning. The system shows the obtained parameters of the surrounding Wi-Fi networks, first on the Graphical LCD and then on the serial monitor available in Arduino IDE. Wi-Fi networks are listed in descending order based on their signal strength. Therefore, the network with the strongest signal (highest RSSI value) is on the top. However, all detected networks are shown on the serial monitor simultaneously. If a Wi-Fi network is not password protected, then the Encryption type is shown as "open". Fig. 5 shows an example of the scanned Wi-Fi networks on the GLCD, while Fig. 6 shows the same scan result on the serial monitor.

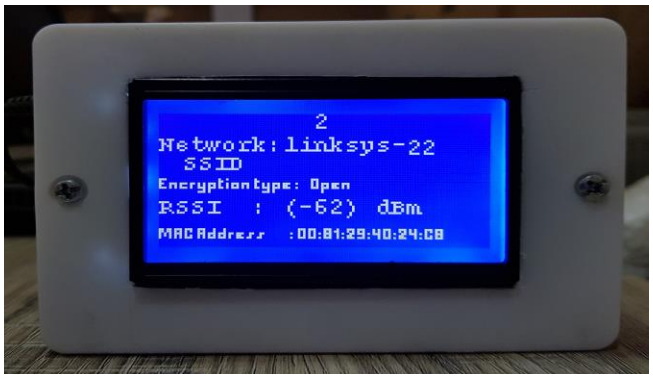
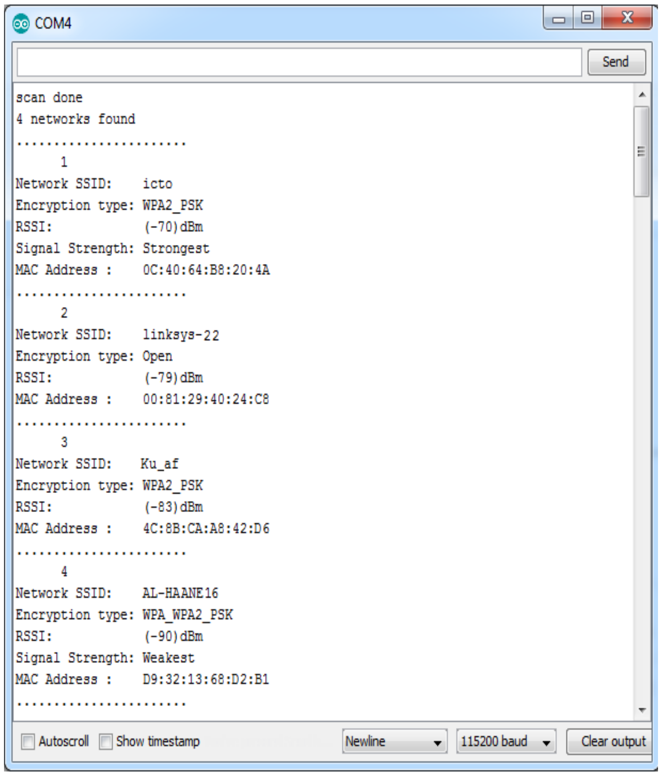


Figure 5. Displaying the Wi-Fi networks in sequence on the GLCD.



```
scan done
4 networks found
.......................
      1
Network SSID:    icto
Encryption type: WPA2_PSK
RSSI:            (-70)dBm
Signal Strength: Strongest
MAC Address :    0C:40:64:B8:20:4A
.......................
      2
Network SSID:    linksys-22
Encryption type: Open
RSSI:            (-79)dBm
MAC Address :    00:81:29:40:24:C8
.......................
      3
Network SSID:   Ku_af
Encryption type: WPA2_PSK
RSSI:            (-83)dBm
MAC Address :    4C:8B:CA:A8:42:D6
.......................
      4
Network SSID:    AL-HAANE16
Encryption type: WPA_WPA2_PSK
RSSI:            (-90)dBm
Signal Strength: Weakest
MAC Address :    D9:32:13:68:D2:B1
.......................
```

Figure 6. Displaying the available Wi-Fi networks on the serial monitor

IX. CONCLUSION

The main objective was to provide an example of ESP32 as a low-cost, low-energy embedded systems design. This paper provided a way of establishing serial communication between the ESP32 module and the Graphical LCD, even if the U8G2 library does not support the ESP32 by default. This article's novelty includes connecting the ESP32 board to Graphical LCD using a minimum number of GPIO pins.

In addition, a detailed comparison is provided to highlight the advantages of the ESP32 in designing embedded systems compared to similar boards such as Arduino UNO and Arduino Mega2560. A use-case of developing an embedded system on the ESP32 board with a built-in Wi-Fi module has been presented. The proposed device can scan the surrounding Wi-Fi networks and visualize several network parameters on a graphical LCD screen. The system's total cost is less than $25, including the ESP32 module, GLCD, and double-sided PCB. For future work, a wireless sensor network for agriculture monitoring can be developed using ESP32 and utilize routing algorithms for data collection in such networks.

ACKNOWLEDGMENT

Project no. FIEK_16-1-2016-0007 has been implemented with the support provided by the National Research, Development and Innovation Fund of Hungary, financed under the Centre for Higher Education and Industrial Cooperation - Research infrastructure development (FIEK_16) funding scheme.

REFERENCES

[1] Russell, David J. "Introduction to embedded systems: using ANSI C and the Arduino development environment." Synthesis Lectures on Digital Circuits and Systems, 2010, 5.1: 1-275.

[2] Espressif Systems. espressif.com [Online]. Available: https://espressif.com/sites/default/files/documentation/esp32 technical_reference_manual_en.pdf. [Accessed 07 Jan. 2020].

[3] Maier, Alexander, Andrew Sharp, and Yuriy Vagapov. "Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things." In Internet Technologies and Applications (ITA), 2017, pp. 143-148. IEEE, 2017.

[4] J. A. F. Calderon, J. S. Vargas and A. Pérez-Ruiz, "License plate recognition for Colombian private vehicles based on an embedded system using the ZedBoard," 2016 IEEE Colombian Conference on Robotics and Automation (CCRA), Bogota, 2016, pp. 1-6.

[5] Anand Nayyar and Vikram Puri, "A review of Arduino board's Lilypad's & Arduino shields", 3rd IEEE International Conference In Computing for Sustainable Global Development (INDIACom), pp. 1485-1492, 2016.

[6] Arduino, "ARDUINO UNO REV3." [Online]. Available: https://store.arduino.cc/usa/arduino-uno-rev3 [Accessed 07 Jan. 2020].

[7] Badamasi YA. The working principle of an Arduino. InElectronics, computer and computation (icecco), 2014 11th international conference on 2014 Sep 29 (pp. 1-4). IEEE.

[8] Arduino, "ARDUINO MEGA 2560 REV3." [Online]. Available: https://store.arduino.cc/usa/arduino-mega-2560-rev3 [Accessed 08 Jan. 2020].

[9] Sarjerao, Borade Samar, and Amara Prakasarao. "A Low Cost Smart Pollution Measurement System Using REST API and ESP32." In 2018 3rd International Conference for Convergence in Technology (I2CT), pp. 1-5. IEEE, 2018.

[10] Birau, Eduard Valentino. Development of a CAN-Wifi converter based on a ESP32. MS thesis. Universitat Politècnica de Catalunya, 2018.

[11] Jameel, H. and Kareem, H., 2016. Low-Cost Energy-Efficient Smart Monitoring System Using Open-Source Microcontrollers. International Review of Automatic Control (IREACO), 9(6), pp.423-428.

[12] Mellis, D., Banzi, M., Cuartielles, D. and Igoe, T., 2007, April. Arduino: An open electronic prototyping platform. In Proc. Chi(Vol. 2007).

[13] Kareem, Husam. "Embedded real-time system for detecting leakage of the gas used in iraqi kitchens." Indonesian Journal of Electrical Engineering and Computer Science 14.3 (2019): 1171-1176.

[14] https://github.com/olikraus/u8g2 [Accessed 11 May. 2020].