Journal of Electrical Engineering, Electronics, Control and Computer Science – JEEECCS, Volume 10, Issue 35, pages 1-12, 2024

Chat System Using Transmission Control Protocol/Internet Protocol And C Programming Language

1st POPA Diana-Ioana
Faculty of Electronics, Communications and Computers
National University of Science and Technology POLITEHNICA Bucharest
Pitești, Romania
popa.diana.ioana11@gmail.com

2nd CIOROBEA Adriana-Andrada
Faculty of Electronics, Communications and Computers
National University of Science and Technology POLITEHNICA Bucharest
Pitești, Romania
adrianaciorobea20@gmail.com

3rd BEȘLIU-GHERGHESCU Andrei-Alexandru
Faculty of Electronics, Communications and Computers
National University of Science and Technology POLITEHNICA Bucharest
Pitești, Romania
besliu_andrei_alex@yahoo.com

4th BIZON Nicu
Department of Electronics, Computer and Electrical Engineering
National University of Science and Technology POLITEHNICA Bucharest
Pitești, Romania
nicubizon@yahoo.com

5th DRĂGUȘIN Sebastian-Alexandru
Department of Electronics, Computer and Electrical Engineering
National University of Science and Technology POLITEHNICA Bucharest
Pitești, Romania
dragusin.sebi@yahoo.com

***Abstract*–In this paper, a chat system between two Wi-Fi microcontrollers was developed using the TCP/IP (Transmission Control Protocol/Internet Protocol) set of communication protocols and a simplified set of the C/C++ programming language in Arduino IDE (Arduino Integrated Development Environment). The purpose of the system is to allow users to send and receive text messages between connected devices. The main functionalities of the system include communication between two ESP8266 microcontrollers by implementing a robust and efficient communication protocol between server and clients, encryption and decryption of transferred data to ensure communication security and an accessible user interface for the chat system.**

Keywords–chat system; TCP/IP; concurrent C programming language; cryptography; Wi-Fi microcontroller; Arduino

I. INTRODUCTION

A. The Problem Analyzed And The Objectives Of The Paper

The paper addresses the implementation of essential functionalities of a chat system, including sending and receiving text messages. It focuses on developing a robust and efficient communication protocol between clients and the server using TCP/IP, as well as encrypting data transferred between them to ensure secure communication. Concurrent programming is utilized as an approach where multiple threads or processes run simultaneously, sharing resources and cooperating to solve problems.

B. Relevant Scientific Context

The chat system implemented between two Wi-Fi microcontrollers within the Arduino IDE development environment represents an interesting combination of IoT (Internet of Things), wireless communication and embedded programming. Here are some relevant aspects from a scientific perspective:

- **ESP8266 microcontrollers:** ESP8266 is a microcontroller with integrated Wi-Fi connectivity, used in IoT projects and smart devices. It offers a compact and affordable solution for connecting to Wi-Fi networks and transmitting data.
- **TCP/IP:** TCP/IP is a set of communication protocols used in computer networks. Implementing a chat system between microcontrollers involves using it to transmit messages between devices.
- **Arduino IDE:** Arduino IDE is an integrated development environment used for programming compatible microcontrollers. It provides a user-friendly interface for writing, compiling, and uploading code to hardware devices.
- **Communication between microcontrollers:** The chat system involves bidirectional communication between two ESP8266 microcontrollers. This includes managing the Wi-Fi connection, transmitting, and receiving messages.
- **I2C (Inter-Integrated Circuit):** I2C is a serial communication protocol used for connecting peripheral devices to a microcontroller. In this case, using two I2C lines for displaying messages can be an efficient solution for interfacing with display devices.
- **Security and efficiency:** In IoT projects, emphasis is placed on communication security and energy efficiency. Implementing a chat system must consider these aspects, including data encryption and resource management.
- **Applications and implications:** Chat systems between embedded devices can be used in various contexts, such as smart homes, environmental monitoring, or industrial automation. Studies in this field focus on optimizing protocols, communication security and scalability.

II. LITERATURE REVIEW

In the document [1], the authors describe the technologies and programming languages used to develop a web-based and mobile chat application. The server-side is built using Node.js, a runtime environment that enables the creation of scalable server-side applications. For real-time communication between the client and server, they implement Socket.IO, a JavaScript library that facilitates WebSocket-based interactions. The application's database is managed with MongoDB, a document-oriented database originally written in C++. On the front end, they use JavaScript and the Express framework to handle server functionality and ensure smooth data exchange between the client and server.

In the document [2], the authors propose a method for designing a network chat system using Socket technology and cloud computing. Socket is a key component of TCP/IP networking, enabling communication between processes over a network. The system uses Java Socket, a tailored solution for developing network applications. The integration of cloud computing enhances data storage and security by providing a distributed and scalable infrastructure. The system incorporates multithreading for concurrent chat operations and uses JDBC for database connectivity. It supports group and private chats with cloud computing offering additional efficiency and security. The Java Swing library is employed to create a user-friendly interface.

In the document [3], the authors present a LAN-based chat program designed to improve internal communication within organizations using the TCP/IP protocol. This system ensures secure and cost-free communication by allowing real-time messaging and file transfer between users connected to the same local network. The software is implemented in Java and was tested on the LANs of a university's laboratory.

In the document [4], the authors focus on designing and implementing communication applications (chat software and an email client) tailored for embedded network terminals. These terminals, based on limited-resource systems, face challenges when running conventional communication software designed for personal computers. The proposed solution specifically addresses these challenges by creating lightweight software that can run efficiently on embedded systems.

In the document [5], the authors discuss the development of an intelligent chat service that utilizes AIML (Artificial Intelligence Markup Language) for creating chatbots. The primary focus is on a hosting platform that allows users to deploy multiple chatbots using pre-existing AIML templates. These bots can be customized to resemble specific personalities or serve as customer support for businesses and institutions. Additionally, the platform offers APIs for retrieving weather data, news updates, dictionary definitions, and more, which can enhance the functionality of the hosted bots.

III. THEORETICAL FOUNDATION

A. Communication Protocols: The TCP/IP Model

It is essential for any devices within a network to speak the same language or protocol to ensure that data packets can reach their destination from a source. Therefore, a communication protocol represents a set of rules that establishes the format and method of data transmission, managing aspects related to:

- The physical construction of the network.
- The way computers connect within the network.
- The format and transmission of data.
- Error resolution.

These rules are developed and maintained by various committees or organizations such as the IEEE (Institute of Electrical and Electronic Engineers), the ANSI (American National Standards Institute), and the ISO (International Organization for Standardization).

In the online environment, the TCP/IP model (Fig. 1.) is considered the technical standard. It was created as an open standard, meaning it can be used freely. The TCP/IP model is structured into four hierarchical levels:

- **Application level:** It ensures the correct representation of data. This level includes the Application and Presentation layers of the OSI (Open System Interconnection) model. At this level, there are several protocols: SMTP (Simple Mail Transfer Protocol) and POP (Post Office Protocol) used for sending and receiving messages, HTTP (Hypertext Transfer Protocol) or HTTPS (Hypertext Transfer Protocol Secure) for the web, FTP (File Transfer Protocol) for file transfer, DNS (Domain Name System) for domain name resolution, etc. Each of these protocols has a specific number of ports for communication.
- **Transport level:** It facilitates transport between the source and destination points. Additionally, it guarantees data flow control, error correction, and quality of service. At this level, there are two important protocols: TCP (Transmission Control Protocol) and UDP (User Datagram Protocol). TCP is a connection-oriented protocol that ensures information reaches its destination exactly as transmitted, providing stable and error-free communication. In contrast, the UDP protocol does not require establishing a connection with the recipient, being a connectionless protocol.
- **Internet level:** Its role is to guide data packets to their destination, identifying the optimal route and managing their transfer. At this level, the central protocol is the IP (Internet Protocol).
- **Network access level:** Its role is to manage the connection with the network's physical medium, regardless of the technology used. This level includes the Data Link and Physical layers, according to the OSI model.

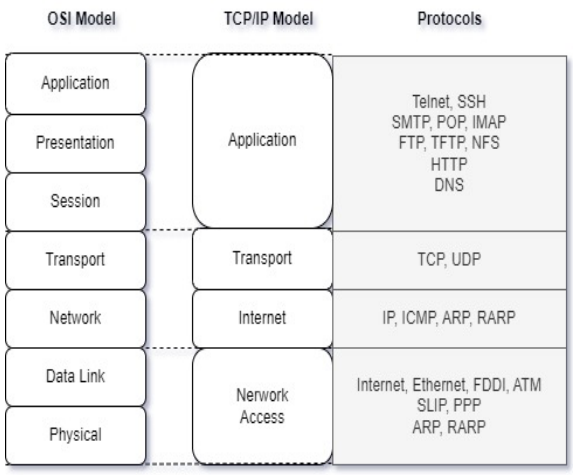

Fig. 1. Communication protocols. The TCP/IP model

TCP has many functionalities when connecting a client and a server. For example, it is suitable for applications that require high reliability and have a relatively lower transmission time. Other protocols like HTTP, HTTPS, FTP, SMTP, and Telnet use it. TCP rearranges data packets in the established order. There is an absolute guarantee that the data sent via email remains intact and arrives in the same order it was sent. Before user data is transmitted, TCP controls the flow and requires three packets to establish a socket connection. Reliability and congestion control are the responsibilities of TCP. Additionally, it performs error checking and recovery. Incorrect packets are retransmitted from the source to the destination.

The entire process can be divided into the following stages (Fig. 2.):

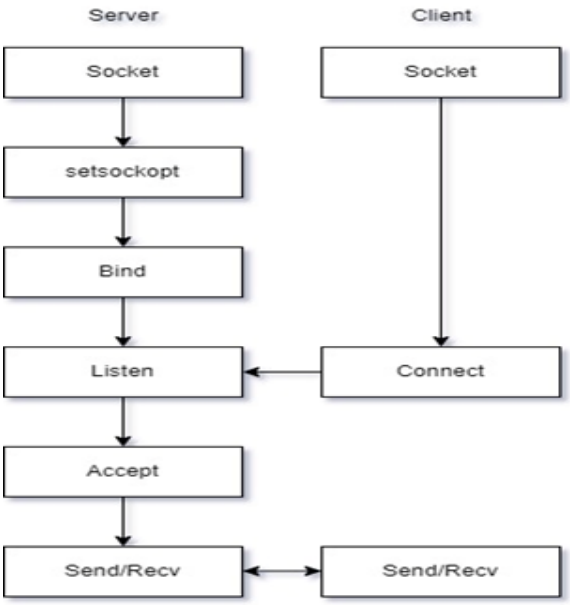

Fig. 2. Server and client for the TCP/IP protocol

Instructions for configuring a TCP Server:

- **create():** This function is used to create the TCP socket.
- **bind():** The socket is assigned to the server's address.
- **listen():** The server socket is configured to passive mode, waiting for a client to initiate a connection.
- **accept():** At this point, the connection is established between the client and server, and both are ready to transfer data.
- Repeat from step 3 to continue the process.

Instructions for Configuring a TCP Client:

- **Create the TCP socket:** A new socket is initialized for communication using the TCP protocol.
- **Connect the newly created client socket to the server:** A connection is established between the client socket and the desired server [6].

B. The C Programming Language

The template is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You

may note peculiarities. For example, the head margin in this template measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

In the 1970s, Dennis Ritchie developed the C programming language, which is both an imperative and procedural language. Although it is a high-level language, it also offers low-level control over system resources. Its ability to create system software, portability, and efficient performance are all attributes of C.

Key features and aspects of the C programming language:

- **Simple syntax:** The easily understandable syntax of C makes it suitable for both beginners and experienced developers.
- **Portability:** Because C provides a lower level of abstraction and is not heavily dependent on platform-specific features, programs written in C are often compatible with multiple operating systems and hardware architectures.
- **Close resource control:** Programmers can accurately control system resources, such as memory and processors, making C suitable for developing operating systems and other types of system software.
- **Efficient performance:** C is often used to create system-level software or applications that require significant resources due to its renowned efficiency and high performance.
- **Standard libraries:** The rich standard library of C offers predefined functions and procedures for various operations, such as file manipulation, memory management, and working with strings.
- **Pointers:** Since C supports pointers, programmers can work directly with memory addresses and perform efficient operations on data.
- **Procedural programming:** The program is structured into functions that can call other functions to accomplish specific tasks, as C is a procedural language. This programming style organizes the code and makes it easier to understand.
- **Extensibility:** C++ and C# are examples of other programming languages derived from C. Additionally, C has played a significant role in shaping modern languages [7].

Concurrent programming in C:

Concurrent programming is an approach where multiple threads or processes run simultaneously, sharing resources and cooperating to solve problems. In the context of the C programming language, concurrent programming can be achieved using libraries such as pthreads (POSIX threads) or OpenMP.

Concurrent programming in C enables the development of applications that can benefit from increased performance and efficient use of hardware resources. Here are some important aspects:

- **Thread:** A thread represents a sequence of instructions that runs independently within a process. The pthreads library offers functionalities for creating, synchronizing, and managing threads in C.
- **Synchronization and access to shared resources:** Concurrent programming involves managing access to shared resources, such as global variables or shared memory areas. Mechanisms like semaphores, mutexes (MUTual EXclusion), and condition variables are used to avoid concurrency issues, such as deadlocks or race conditions.
- **OpenMP directive:** OpenMP is a simple and portable approach for concurrent programming in C. By adding special pragmas in the source code, loops, functions, or other critical sections of the program can be parallelized.
- **Benefits of concurrent programming:** Increased performance: Applications can run faster on multi-core processors. Efficient resource utilization: Hardware resources are used more efficiently, reducing waiting time [8].

C. Symmetric Encryption Algorithms: The Polybius Cipher

In the category of symmetric encryption systems, the Polybius cipher, also known as the Polybius square, is a simple encryption technique that transforms letters into numbers. To perform this conversion, a matrix or a table is used [9].

The plaintext consists of letters from the alphabet, and the data structure in which the alphabet letters are arranged is the secret key.

The fundamental encryption and decryption principles are as follows:

- A square of covering dimension contains the letters of the Latin alphabet.
- In the Latin alphabet, the letters I and J are used together to form a single character, as the final character (between I and J) can easily be chosen based on the context of the message.
- The pair of numbers (the intersection of the row and column) corresponding to the position of the character in a square is selected to encrypt it.

Observation 1: Rearranging the letters in a 5x5 square (Fig. 3.) allows for changing the code.

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I/J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

	1	2	3	4	5
1	C	R	I/J	P	T
2	O	G	A	F	E
3	B	D	H	K	L
4	M	N	Q	S	U
5	V	W	X	Y	Z

Fig. 3. 5x5 Polybius square

Observation 2: The square can be of size 6x6 (Fig. 4.), and in this case, the digits from 0 to 9 can also be added. The Polybius cipher, using a special matrix, offers fast and efficient encryption [10].

	1	2	3	4	5	6
1	A	B	C	D	E	F
2	G	H	I	J	K	L
3	M	N	O	P	Q	R
4	S	T	U	V	W	X
5	Y	Z	0	1	2	3
6	4	5	6	7	8	9

Fig. 4. 6x6 Polybius square

D. Arduino IDE

Arduino IDE is an open-source Integrated Development Environment used for programming and developing projects on Arduino boards. Its key features include:

- **Simple and intuitive interface:** It includes a modern editor (code writing interface with auto-completion and navigation), Serial Monitor window, library manager, and debugging tools (live debugger allowing real-time code analysis).
- **Programming language:** Arduino IDE uses a simplified dialect of the C/C++ language. Programmers can write code for Arduino boards using predefined functions and libraries.
- **Uploading code to the board:** After writing the code, it can be uploaded to the Arduino board via a USB cable. The Arduino board is selected from the "Tools" menu, and the code is uploaded subsequently.
- **Libraries and examples:** It comes with a set of predefined libraries to facilitate programming various functionalities. Additionally, code examples for different modules, components, and sensors can be accessed [11].

E. Wi-Fi Microcontrollers: ESP8266 Model

Microcontrollers play a crucial role in connecting and communicating with other IoT systems and devices. They gather data from sensors, process it, and then transmit it to other devices or systems using wired or wireless communication protocols. Additionally, microcontrollers can receive and interpret commands and control signals from other devices, which they can subsequently use to manage the activities and behavior of the embedded device. Due to their small size, minimal energy consumption, and affordability, microcontrollers are well-suited for IoT applications. They can be programmed to perform a variety of functions and are particularly suitable for battery-powered electronic devices. Various microcontrollers are available on the market, each with its unique features and capabilities [12].

Wi-Fi represents one of the most widely adopted protocols for wireless connectivity among local area devices. It is commonly used to provide internet access to these devices. Wi-Fi is built upon the IEEE 802.11 family of standards and operates within the 2.4 GHz and 5 GHz frequency bands (with Wi-Fi 6E also utilizing the 6 GHz band) [13].

The ESP8266 (Fig. 5.) is an affordable Wi-Fi microchip equipped with a complete TCP/IP stack and microcontroller capabilities. Developed by Espressif Systems, it operates seamlessly with the Arduino IDE compiler. This versatile device combines features from a standard Arduino microchip and has the added capability to connect to the internet via its built-in Wi-Fi module [12].

Key features of the ESP8266 microcontroller include:

- **Compact size:** The ESP8266 is small and easily integrable into electronic projects.
- **Processing power:** It boasts powerful processing speeds onboard, with a central processing unit of either 80 MHz or 160 MHz.
- **Storage space:** It has ample storage capacity, allowing integration with other devices such as sensors.
- **Compatibility:** To make this module compatible with other development boards, external voltage leveling must be performed. The ESP8266 does not have a built-in voltage regulator.
- **Supported interfaces:** SPI (Serial Peripheral Interface), I2C (Interface for communication between integrated circuits), I2S (Interface for digital audio data transfer) and UART (Supported on certain pins).
- **ADC (Analog-to-Digital Conversion):** It features a 10-bit resolution ADC converter. The digital signal is used to turn on the LED.
- **TCP/IP protocol:** Protocol stack for network communication.

Configuration and programming of the ESP8266 can be done using Arduino IDE as follows:

- Open Arduino IDE and access the preferences window.
- In the preferences window, add the URLs for additional board managers.
- Install the necessary libraries and drivers to have the ESP8266 recognized by Arduino IDE.
- After configuration, the ESP8266 can be programmed using Arduino IDE [14].

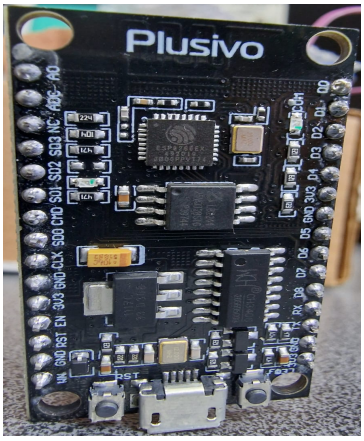

Fig. 5. ESP8266

F. I2C

The serial communication bus called I2C (Fig. 6.), also known as IIC, was developed by Philips Semiconductors (now NXP Semiconductors) in 1982. It is often used to connect low-speed peripheral integrated circuits to processors and microcontrollers for short-distance communications on the same board. The SDA (Serial Data Line) and the SCL (Serial Clock Line) are the only signals used by the IoT.

Features:

- Combines the best features of SPI and UART protocols.
- Multiple peripheral devices (slaves) can be connected to a single controlling device (master), like SPI. This is useful when using more than one microcontroller recording data to a single memory card or displaying text on a single LCD.
- Uses only two wires to transmit data between devices, like UART communication: SDA (Serial Data) is the line for transmitting and receiving data between master and slave and SCL (Serial Clock) is the line that carries the clock signal.

Operations:

- Data is transferred in messages in I2C.
- Messages are divided into data frames.
- Each message contains: address frame, data frame(s), start and stop conditions, read/write bit, ACK (acknowledge) or NACK (no-acknowledge) bit. Address frame contains the binary address of the slave, identifying the device when the master wants to communicate with it. Data frame contains data transmitted. Start and stop conditions are the signals for the beginning and end of the message. Read/write bit specifies whether the master is sending data to the slave (low voltage level) or requesting data from it (high voltage level). During communication over an I2C bus, the slave device responds with an ACK/NACK bit after each data frame.
- I2C uses addresses to communicate with devices since it does not have slave select lines like SPI: The master sends the slave address to all connected devices. Each slave compares the received address with its own address and responds with an ACK bit if the address matches [15].

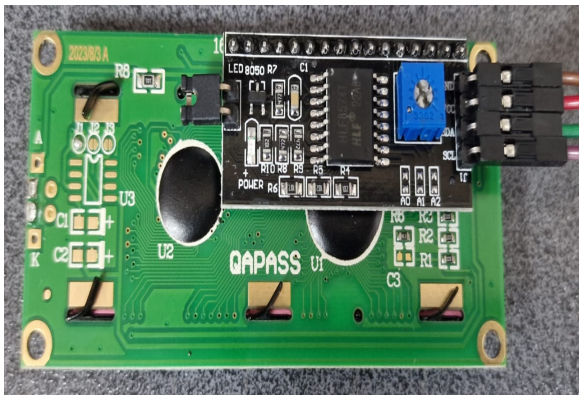

Fig. 6. I2C

G. Debouncing

Debouncing is the process of eliminating the possibility of producing oscillating signals after a single key press, ensuring a more consistent movement.

The purpose of debouncing is to provide stability and reliability to the keyboard by reducing or eliminating the "key bounce" or "chatter" phenomenon.

Methods of debouncing for keyboards in Arduino:

- **Software debouncing:** This method involves adding delays in the code to eliminate bounce. Programmers can add delays using the delay() function to force the controller to wait for a certain. The disadvantage of this method is that it can cause abrupt pauses in the program and increase processing time.
- **Hardware debouncing:** In case of using a pull-up or pull-down resistor, a 10 kΩ resistor is connected between the keyboard's input pin and the power supply pin (VCC or GND). This will establish a stable level for the input pin, reducing fluctuations. In case of using a capacitor, a capacitor is connected between the keyboard's input pin and the power supply pin. The capacitor will filter electrical noise and help stabilize the signal.
- **Example of debouncing in Arduino:** An example of debouncing for a button can be found in the Built-in Examples section of the official

Arduino website. This example uses the millis() function to track the time elapsed since the button was pressed [16].

IV. Proposed Solution: Communication between two ESP8266 microcontrollers

The steps for developing a chat system using embedded systems with ESP8266 microcontrollers are as follows:

A. Planning And Creating The Organizational Charts

It is important to start by having a plan in mind for creating the organizational charts.

Given the fact that the paper is about the communication between two ESP8266 microcontrollers, there are going to be two users, one of them serving as both a server and a client, and the other one serving only as a client.

Therefore, the organizational charts corresponding to the server (Fig. 7.) and the client (Fig. 8.) codes have been created.

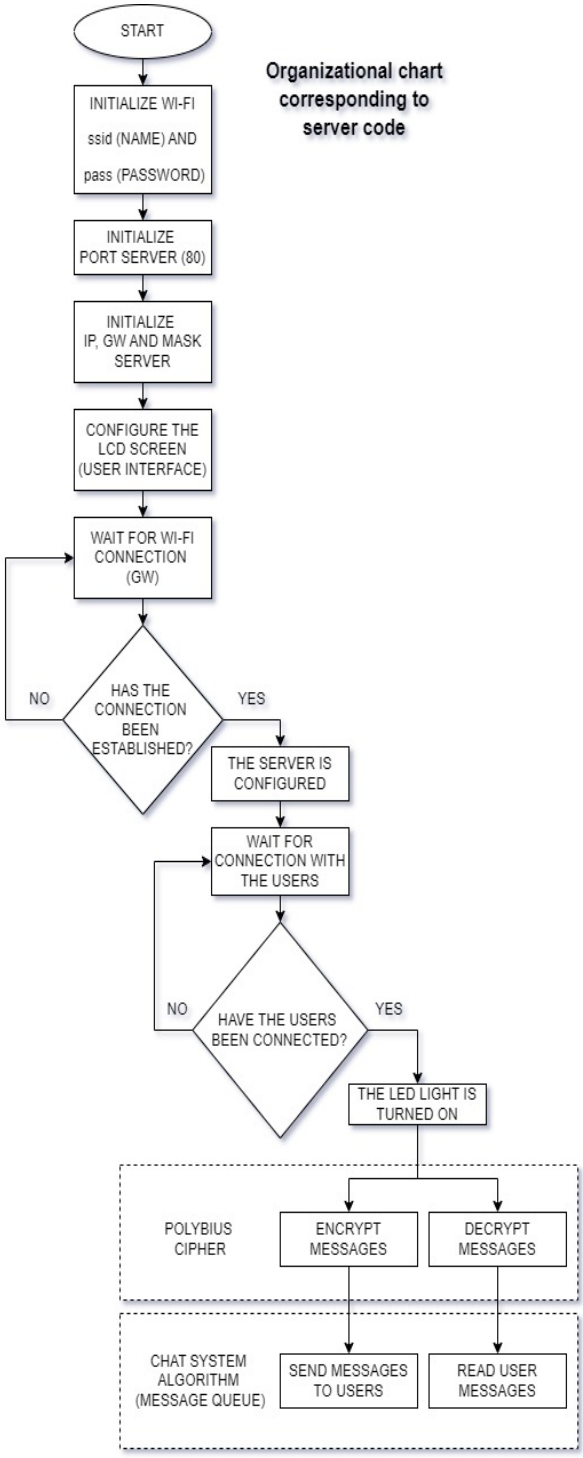

Fig. 7. Organizational chart corresponding to server code

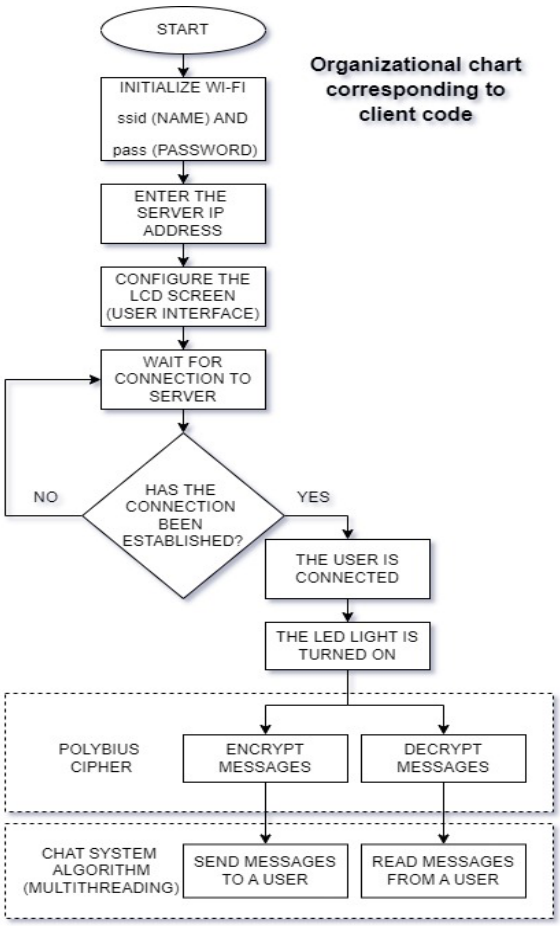

Fig. 8. Organizational chart corresponding to client code

B. Connecting The Physical Components Used

The physical components used (an ESP8266, an I2C, a 16x2 LCD, a resistor and a LED for each user) must be connected as shown in Fig. 9.

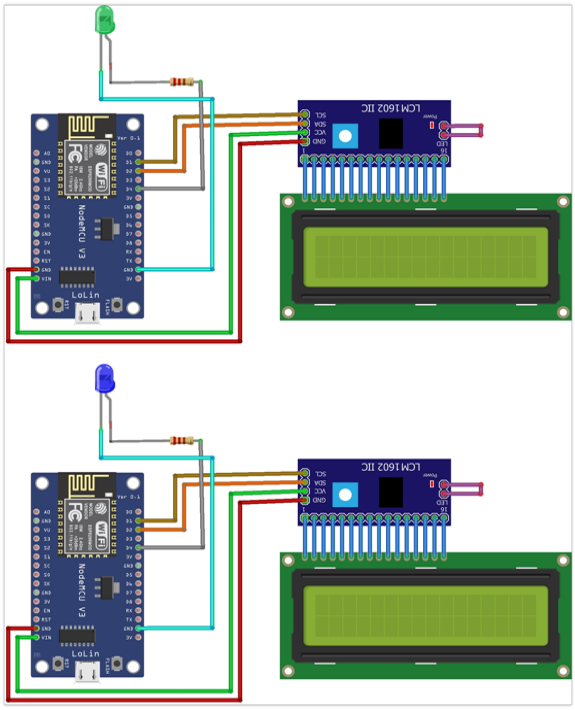

Fig. 9. Circuit diagram of the physical components used for two users

The second law of Kirchhoff is used to calculate the values of the resistors used as shown in equation (1).

$$V_{CC} = R \times I_{LED} + V_{LED} \Rightarrow R = \frac{V_{CC} - V_{LED}}{I_{LED}} \quad (1)$$

The value of the resistor of each user depends on the color of the LED. For example, the documentation indicates that the green LED operates at 2.2V, while the blue LED operates at 3.2V. Both LEDs have a forward voltage of 2.2V.

Also, the documentation for the ESP8266 specifies that this microcontroller operates at 3.3V.

Equation (2) is for the green LED used for the first user:

$$R = \frac{3{,}3V - 2{,}2V}{20mA} = \frac{1{,}1V}{20mA} = 55\,\Omega \quad (2)$$

The standard nominal value R = 56 Ω from the E24 series was chosen.

Equation (3) is for the blue LED used for the second user:

$$R = \frac{3{,}3V - 3{,}2V}{20mA} = \frac{0{,}1V}{20mA} = 5\,\Omega \quad (3)$$

The standard nominal value R = 6.8 Ω from the E24 series was chosen.

C. Setting Up Arduino IDE And Connecting ESP8266 To Wi-Fi Network

Open Arduino IDE, connect the ESP8266 board to the laptop, and select the correct board model in the development environment, as shown in Fig. 10.

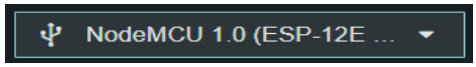

Fig. 10. NodeMCU, the development board based on the ESP8266 module

The **#include <ESP8266WiFi.h>** library is used to connect the ESP8266 board to the Wi-Fi network, and the **#include <LiquidCrystal_I2C.h>** library facilitates communication between the Arduino board and the I2C LCD display.

Additionally, the line of code **LiquidCrystal_I2C lcd(0x27, 16, 2);** creates an object named lcd that will be used to control the I2C LCD display with address 0x27, having 16 columns and 2 rows, as shown in Fig. 11.

```
#include <ESP8266WiFi.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27,16,2);
```

Fig. 11. Libraries and settings for the ESP8266 and the I2C in the Arduino IDE code

D. Wi-Fi network name and security key

Turn on the mobile hotspot, identify the SSID (Wi-Fi network name) and the security key (Wi-Fi password) to connect to the Wi-Fi network on the laptop (Fig. 12.) and use them in the Arduino IDE code, as shown in Fig. 13.

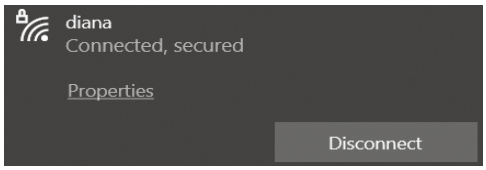

Fig. 12. Connecting to the Wi-Fi network

For the code in Arduino IDE, write the mobile hotspot's name and password as follows:

```
char ssid[] = "diana";
char pass[] = "12345678";
```

Fig. 13. SSID and Wi-Fi password in Arduino IDE

E. Command Prompt

Open Command Prompt: Press the Windows button on the bottom bar of the laptop, type "cmd," and press Enter.

Use the "ipconfig" command to display information about the IP address (192.168.203.122), subnet mask (255.255.255.0), and default gateway for all network adapters on the laptop (192.168.203.50), as shown in Fig. 14.

```
Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::1be4:698c:d47e:87a3%17
   IPv4 Address. . . . . . . . . . . : 192.168.203.122
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.203.50
```

Fig. 14. "ipconfig" command used in Command Prompt

Following the information obtained earlier from cmd, the architecture of the chat system communication is being planned as shown in Fig. 15.

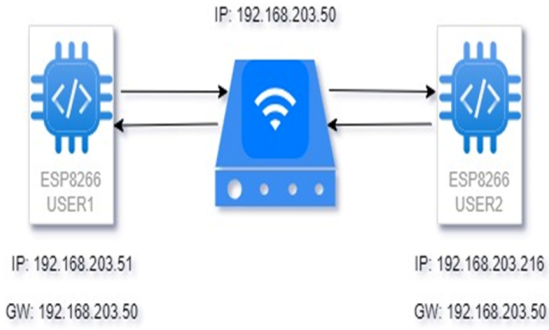

Fig. 15. Chat system communication architecture

For the code in Arduino IDE, write the information found from cmd as shown in Fig. 16.

```
IPAddress ip(192, 168, 203, 51);
IPAddress gateway(192, 168, 203, 50);
IPAddress subnet(255, 255, 255, 0);
```

Fig. 16. IP, GW and MASK in Arduino IDE

F. Programming The LED To Turn On When Connected

The digital pin D4 is used to control an LED as shown in Fig. 17.

```
int LED = D4;
```

Fig. 17. The digital pin D4 used to control an LED

Additionally, within the setup() function, the LED pin is set to output mode (OUTPUT), which means we send signals to this pin and, following the initiation of communication between the two microcontrollers, the logical level of the LED pin is set to HIGH (3.3V), resulting in the LED turning on upon connection.

The LED is defined as a digital output as shown in Fig. 18.

```
pinMode(LED, OUTPUT);
```

Fig. 18. LED defined as a digital output

The LED lights up when connected as shown in Fig. 19.

```
digitalWrite(LED, HIGH);
```

Fig. 19. LED lights up when connected

G. Implementation Of Encryption And Decryption Functions Using The Polybius Cipher

The secret key and the size of the 5x5 matrix are established as shown in Fig. 20.

```
String key = "PolybiusCipher";
const int tableSize = 5;
```

Fig. 20. LED lights up when connected

Subsequently, an encryption function and a decryption function are implemented using the Polybius cipher, the encryption being carried out as follows:

- **Initialization of the encrypted message:** An empty variable encryptedMessage is created to store the encrypted message.
- **Building the Polybius square:** A two-dimensional matrix polybiusTable is created to represent the Polybius square. This matrix is constructed using the given key.

- **Encrypting the message:** Each character in the original message is processed. If the character is 'j', it is replaced with 'i'. The coordinates (row, column) of the character are found in the Polybius square. If the character is found in the table, the encrypted coordinates (represented as digits) are added to the encryptedMessage. If the character is not found in the table, it is added unchanged to the encryptedMessage.
- **Returning the encrypted message:** The function returns the encrypted message.

V. Conducting Experiments

A. Typing And Sending Messages Between Two Users

Typing and sending messages between two users using Serial Monitor in Arduino IDE as shown in Fig. 21.

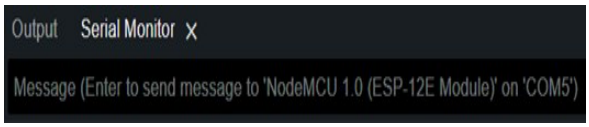

Fig. 21. Serial Monitor in Arduino IDE

B. Testing The Communication Between Two ESP8266 Microcontrollers

Testing the communication between two ESP8266 microcontrollers (Fig. 22, Fig. 23, Fig. 24.):

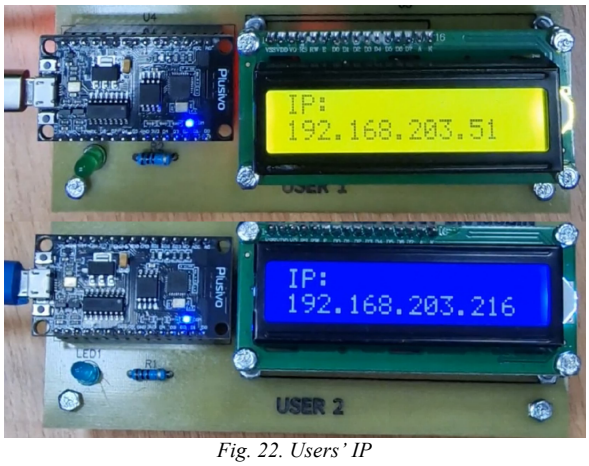

Fig. 22. Users' IP

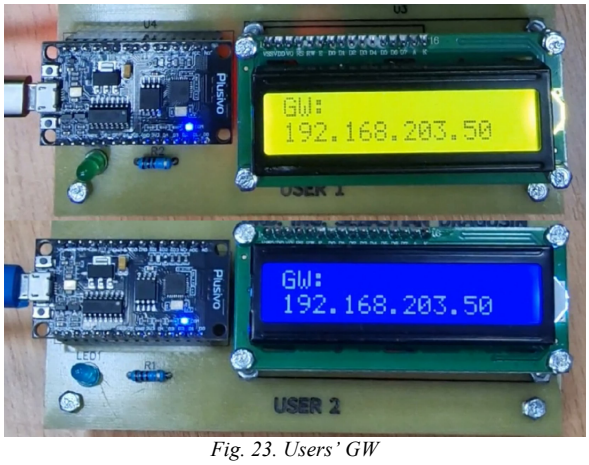

Fig. 23. Users' GW

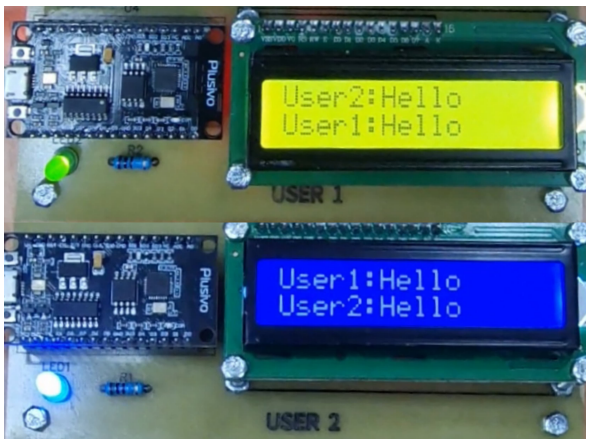

Fig. 24. Users' conversation

Conclusions

The development of a chat system between Wi-Fi microcontrollers highlights the practical implications of integrating technology, programming, and communication in IoT and smart devices.

The chat system's efficiency and flexibility were enhanced using the C programming language, a foundational tool in system software development and hardware design. Concurrent programming in C required careful synchronization and resource management to achieve robust performance.

Communication between the two ESP8266 microcontrollers was established using the TCP/IP set of protocols. One microcontroller functioned as a server, while the other acted as a client, demonstrating the practical application of networking concepts. The ESP8266 microcontrollers, selected for their compatibility with IoT projects and Wi-Fi capabilities, were programmed using the Arduino IDE, an essential tool for developing embedded systems.

The implementation also incorporated the I2C protocol for short-distance communications, balancing its simplicity and efficiency with its known limitations of speed and distance. Additionally, keyboard debouncing was implemented to ensure stable and reliable operation of the system.

Cryptography played a key role in securing communication. The Polybius cipher, using a special matrix, offers fast and efficient encryption, ensuring data confidentiality within the system.

This paper underscores the importance of seamlessly combining hardware and software to create innovative IoT solutions and highlights the versatility of tools like the Arduino IDE and ESP8266 microcontrollers in advancing connected technologies.

The following research will focus on developing an ergonomic and user-friendly interface for the proposed chat system and implementing new encryption algorithms for the data transferred between clients and the server to ensure communication security.

REFERENCES

[1] D. Henriyan, Devie Pratama Subiyanti, R. Fauzian, D. Anggraini, M. Vicky Ghani Aziz, and Ary Setijadi Prihatmanto, "Design and implementation of web based real time chat interfacing server," pp. 83–87, Feb. 2017, doi: 10.1109/ICSENGT.2016.7849628.

[2] M. Cai, "The design method of network chat system based on socket and cloud computing," Proceedings - 2012 International Conference on Computer Science and Service System, CSSS 2012, pp. 610–613, 2012, doi: 10.1109/CSSS.2012.157.

[3] Mohammed A. Ahmed, "Design and Implement Chat Program Using TCP/IP," Iraqi Journal for Computers and Informatics, vol. 44, no. 1, Jun. 2018, doi: 10.25195/2017/4417.

[4] P. Wang, H. He, R. T. Cai, H. L. Jiang, and S. W. Xu, "The design and implementation of application communication based on embedded network terminal," Proceedings - 2010 1st ACIS International Symposium on Cryptography, and Network Security, Data Mining and Knowledge Discovery, E-Commerce and Its Applications, and Embedded Systems, CDEE 2010, pp. 278–282, 2010, doi: 10.1109/CDEE.2010.61.

[5] G. Saqib, K. Faizan, and N. Ghatte, "Intelligent Chatting Service Using AIML," Proceedings of the 2018 International Conference on Current Trends towards Converging Technologies, ICCTCT 2018, Nov. 2018, doi: 10.1109/ICCTCT.2018.8550989.

[6] H. Rili, "Research and application of TCP/IP protocol in embedded system," 2011 IEEE 3rd International Conference on Communication Software and Networks, ICCSN 2011, pp. 584–587, 2011, doi: 10.1109/ICCSN.2011.6014961.

[7] O. Gazi, "Modern C Programming," Modern C Programming, 2024, doi: 10.1007/978-3-031-45361-8.

[8] M. Sonnenschein, "An extension of the language C for concurrent programming," Parallel Comput, vol. 3, no. 1, pp. 59–71, Mar. 1986, doi: 10.1016/0167-8191(86)90007-4.

[9] S. A. Dragusin, N. Bizon, and R. N. Bostinaru, "A Brief Overview Of Current Encryption Techniques Used In Embedded Systems: Present And Future Technologies," 15th International Conference on Electronics, Computers and Artificial Intelligence, ECAI 2023 - Proceedings, 2023, doi: 10.1109/ECAI58194.2023.10194034.

[10] Petre. Anghelescu, Automate celulare : fundamente şi abordări practice cu aplicaţii în criptare. Bucuresti: Matrix Rom, 2012.

[11] F. Asadi, "Essentials of ArduinoTM Boards Programming," 2023, doi: 10.1007/978-1-4842-9600-4.

[12] K. M. Hosny, W. M. El-Hady, and F. M. Samy, "Technologies, Protocols, and applications of Internet of Things in greenhouse Farming: A survey of recent advances," Information Processing in Agriculture, Apr. 2024, doi: 10.1016/J.INPA.2024.04.002.

[13] "Insights - Internet of Things | ScienceDirect.com by Elsevier." Accessed: May 07, 2024. [Online]. Available: https://www.sciencedirect.com/journal/internet-of-things/about/insights

[14] N. Cameron, "Electronics Projects with the ESP8266 and ESP32: Building Web Pages, Applications, and WiFi Enabled Devices," Electronics Projects with the ESP8266 and ESP32: Building Web Pages, Applications, and WiFi Enabled Devices, pp. 1–697, Jan. 2020, doi: 10.1007/978-1-4842-6336-5/COVER.

[15] K. M. Lynch, N. Marchuk, and M. L. Elwin, "I2C Communication," Embedded Computing and Mechatronics with the PIC32, pp. 191–211, Jan. 2016, doi: 10.1016/B978-0-12-420165-1.00013-5.

[16] R. Toulson and T. Wilmshurst, "Interrupts, Timers, and Tasks," Fast and Effective Embedded Systems Design, pp. 199–233, Jan. 2017, doi: 10.1016/B978-0-08-100880-5.00009-8.