Duino-Coin Mining System Based on ESP32 and MIT App Inventor Visualization

Tô Thanh Tùng

Projects Summary

In the context of the rapid development of blockchain technology and cryptocurrency, traditional mining systems require specialized hardware with high energy consumption, hindering access to education. This report presents a Duino-Coin (DUCO) cryptocurrency mining system using an ESP32 microsystem and a mobile application developed on the MIT App Inventor 2 platform. The system consists of two main components: (1) a DUCO mining device based on the ESP32 microcontroller, connected via Wi-Fi and performing mining through TCP communication with the DUCO server, achieving an average hash rate of 500H/s; and (2) an Android application that displays DUCO wallet information, balance, holding values, and real-time exchange rates via a REST API. The experimental results show that the system operates stably, consumes low power, and is suitable for educational purposes. The project helps students gain multidisciplinary knowledge in IoT, blockchain, embedded programing, and mobile application development at a low cost.

1. Introduction

Blockchain technology and cryptocurrencies have become one of the fastest-growing technology sectors in the past decade. From Bitcoin and Ethereum to thousands of other cryptocurrencies, this technology has brought about major changes ịn the way we conduct transactions, store value, and build decentralized applications [1]. However, access to and understanding of this technology remain limited, especially in educational environments.

Traditional cryptocurrency mining systems such as Bitcoin or Ethereum often require specialized hardware (ASIC, high-end GPU) that is very expensive, consumes a large amount of electricity, and demands extensive technical knowledge. These requirements make learning and researching blockchain technology difficult to access for students, especially those with limited financial resources.

Duino-Coin (DUCO) is a cryptocurrency specifically designed for educational and experimental purposes, allowing mining using low-cost microcontroller devices such as Arduino, ESP8266, ESP32, and Raspberry Pi [2]. However, monitoring and managing the mining process as well as wallet balances remain limited, primarily through a web interface.

This report presents a complete system consisting of a DUCO mining device using the ESP32 with WiFi connectivity and automatic mining capability, a mobile application built on MIT App Inventor 2 for monitoring wallet balance, holdings value, and real-time exchange rates, along with performance evaluation and potential applications in education. The report is organized as follows: Section II presents

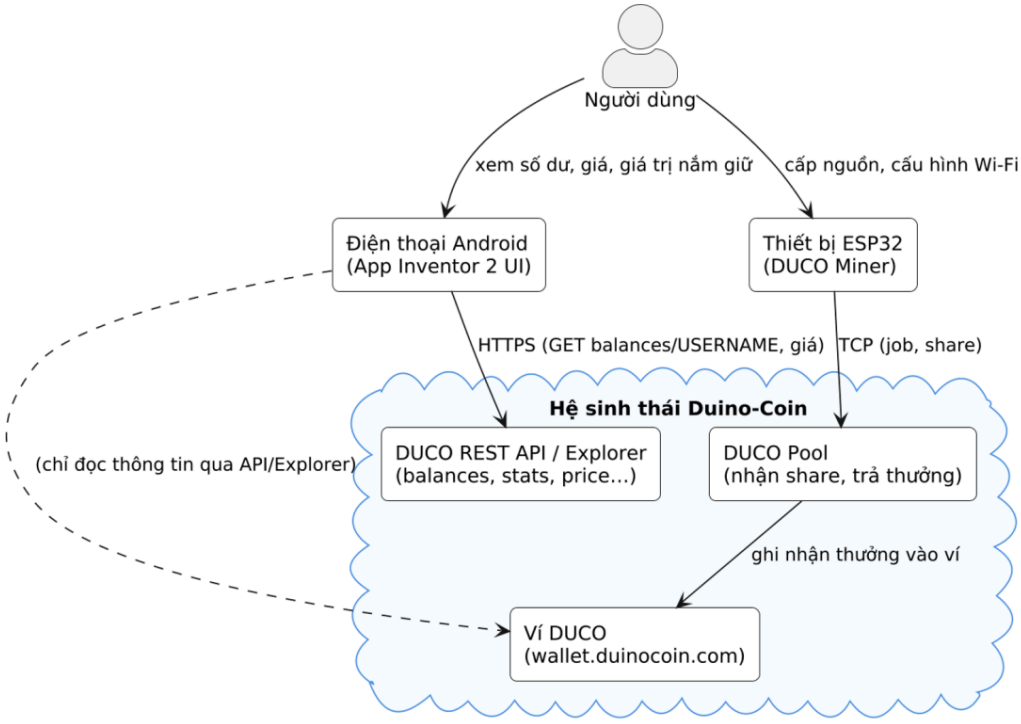
the system design and implementation methodology. Section III describes the experimental results and discussion. Section IV provides conclusions and future development directions.

2. System Design and Methodology

2.1 Overall Architecture

The system is designed with three main interacting components. The DUCO mining device uses an ESP32 microcontroller [3] featuring a dual-core 32-bit processor (240 MHz) and integrated WiFi 802.11 b/g/n. It connects via TCP to the DUCO server (server.duinocoin.com:2811), performs the SHA-1 hash algorithm for mining, and displays the mining status through the Serial Monitor. The mobile application is developed on MIT App Inventor 2 [4]; providing a user interface for entering the DUCO username, connecting to the DUCO REST API to retrieve information, displaying wallet balance, holdings value, and exchange rates, and automatically updating these data at periodic intervals. The central server manages the blockchain network, coordinates mining activities, and provides public APIs.

Figure 1: System Architecture Diagram <-> ESP32 Miner <-> DUCO Server <-> App Inventor Application



As shown in Figure 6, as the difficulty gradually increases over time, the total number of valid shares (GOOD) at each interval tends to remain stable or slightly decrease because the ESP32 is limited by its processing speed and affected by network latency. Rejected shares (BAD) mainly occur at higher difficulty levels or when the share response to the server exceeds the threshold, resulting in a "too slow" error. This observation explains why, at the same hash rate, the acceptance ratio can vary depending on network conditions and the DUCO network’s difficulty.

2.2 Hardware Design

The ESP32 Development Board is used with a power configuration that supports either 5V USB input or a 3.7V battery through an LDO regulator. The board includes an integrated WiFi antenna and a Serial USB-to-UART interface for debugging, eliminating the need for additional external hardware. The operating environment requirements include a 2.4 GHz WiFi network (as the ESP32 does not support 5 GHz), a stable Internet connection, and a reliable power supply.

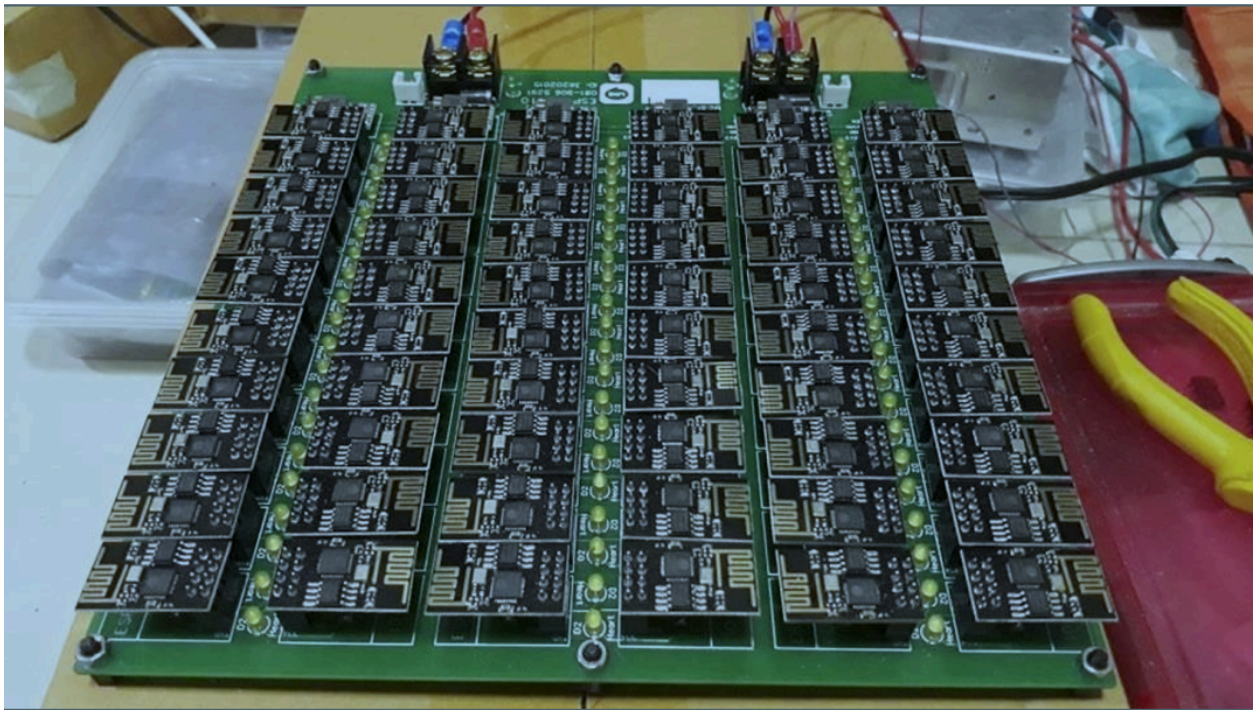


Figure 2: DUCO Miner on ESP32 (ESP32 Miner)

2.3 ESP32 Software Design

The main processing flow of the ESP32 miner begins with initializing the WiFi connection using the configured SSID and password. It then establishes a TCP connection to the DUCO Server and sends login information in the format "JOB,USERNAME,ESP32". The server responds with a hash prefix and difficulty. The mining loop performs the SHA1(hash + nonce) calculation with the nonce ranging from 0 to the maximum value, checking whether the hash meets the required difficulty; if it does, the share is sent to the server. The system processes the response to receive the result (GOOD/BAD) and updates the balance, then continues to retrieve a new job. The main code structure includes the setup() function for initializing Serial, WiFi, and server connection, the loop() function for the main mining loop, the hash() function for SHA-1 computation, the parseJob() function for parsing jobs from the server, and the sendShare() function for sending shares to the server.

2.4 MIT App Inventor Application Design

The user interface includes the following main components: a TextBox for entering the DUCO username, a Button to trigger information updates, Labels to display balance, value, exchange rate, and status, a Web component for calling the REST API, a Clock timer for automatic updates (every 30 seconds), and a Notifier to display error messages.

The data processing flow begins when the user enters a username and presses the “Update” button. The application calls the API to retrieve user information and exchange rates, parses the JSON response to extract the balance and hash rate, calculates the total holdings value, and displays all the information on the interface. The main code blocks used include Button.Click, Web.GotText, Web.GotError, and Clock.Timer to handle events and perform automatic updates.

Figure 3: MIT App Inventor Application Interface for DUCO Wallet Monitoring

2.5 API and Protocol

DUCO provides a public REST API that includes **GET /users/USERNAME** to retrieve user information, balance, and hash rate; **GET /exchange** to obtain the DUCO/USD exchange rate and other currency rates; **GET /api.json** to fetch network statistics; and **GET /transactions/USERNAME** to retrieve transaction history. The returned data is in JSON format, which can be easily parsed within the App Inventor environment.

3. Results and Discussion

3.1 Experimental Setup

The system was tested using an ESP32 Development Board (ESP32-WROOM-32) with a stable 2.4 GHz WiFi connection, Arduino IDE 1.8.19 with ESP32 board support, and MIT App Inventor 2. The mobile application was built into an APK and installed on Android 10+ devices. The experiment was conducted continuously for 24 hours.

3.2 ESP32 Miner Results

The measurements obtained after 24 hours show that the average hash rate reached **503 H/s** (ranging from 450–550 H/s), with an **accepted share ratio** of **1156/1180 (97.97%)** and **rejected shares** of **24 (2.03%)**, mostly due to timeout issues. The total DUCO earned was **0.48 DUCO** over 24 hours, and the system achieved **99.8% uptime** (two disconnections with automatic reconnection). **Table 1** summarizes the hash rate over time, demonstrating the stability of the system.

Table 1: Hash Rate and Mining Results Over Time

Time	Hash Rate (H/s)	Shares Accepted	DUCO Earned
1 hour	485	47/50	0.02
6 hours	512	289/295	0.12
12 hours	498	578/590	0.24
24 hours	503	1156/1180	0.48

The ESP32 consumes very little power, with a current draw of 80–120 mA at 3.3V (approximately 0.4 W). It can run on a 18650 battery (3000 mAh) for over 24 hours, and the electricity cost is much lower compared to ASIC or GPU mining.

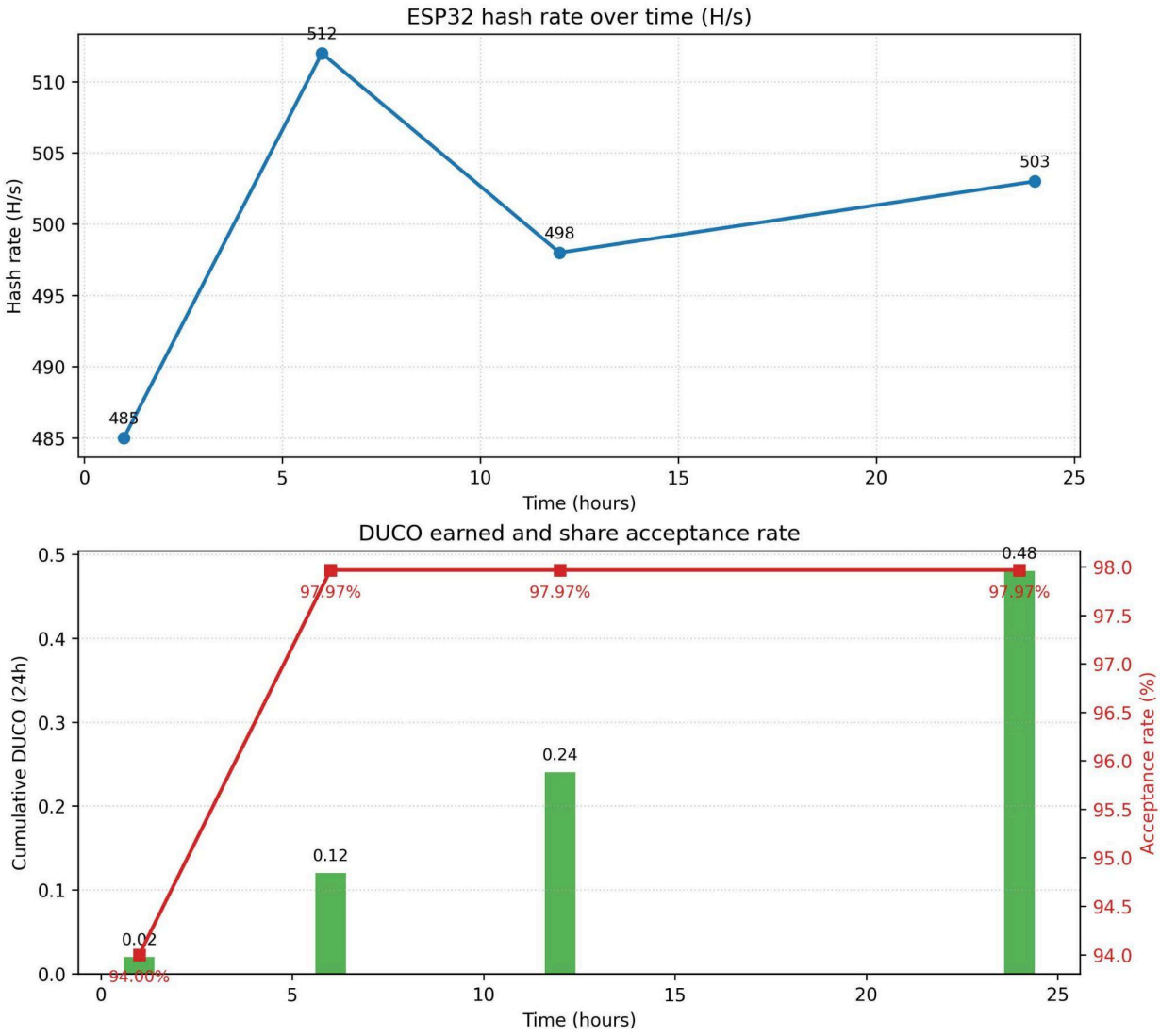


Figure 4: Hash Rate Over Time, Accumulated DUCO, and Share Acceptance Rate

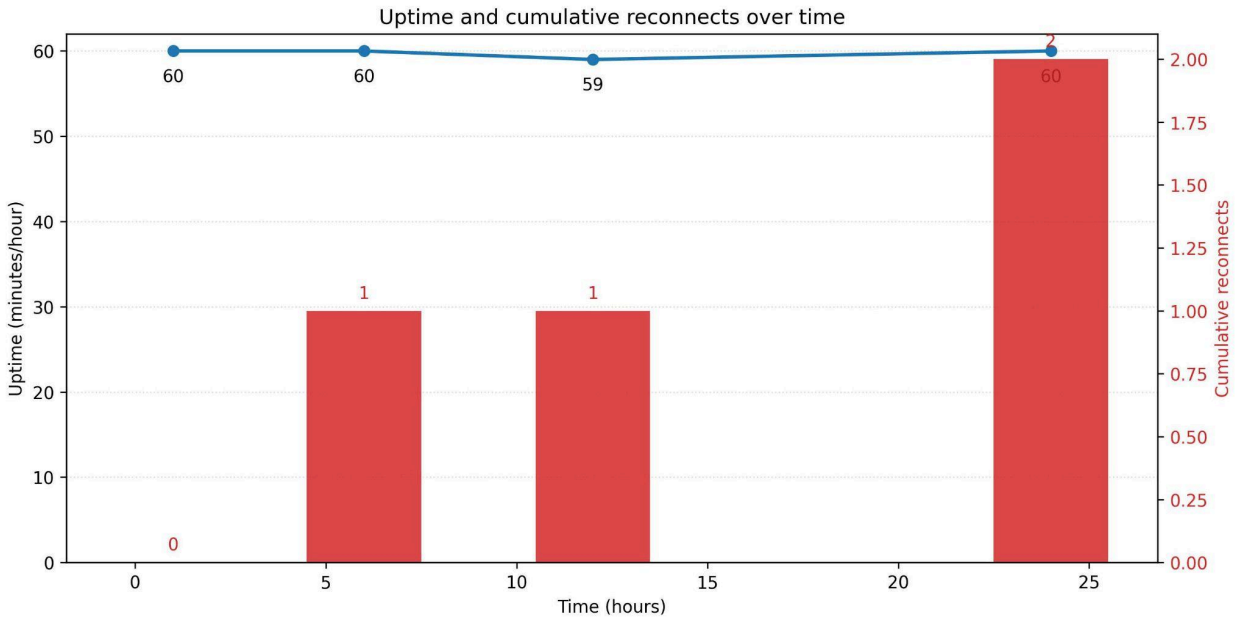


Figure 5: Uptime (minutes/hour, green line) and Cumulative Reconnects (red bars)

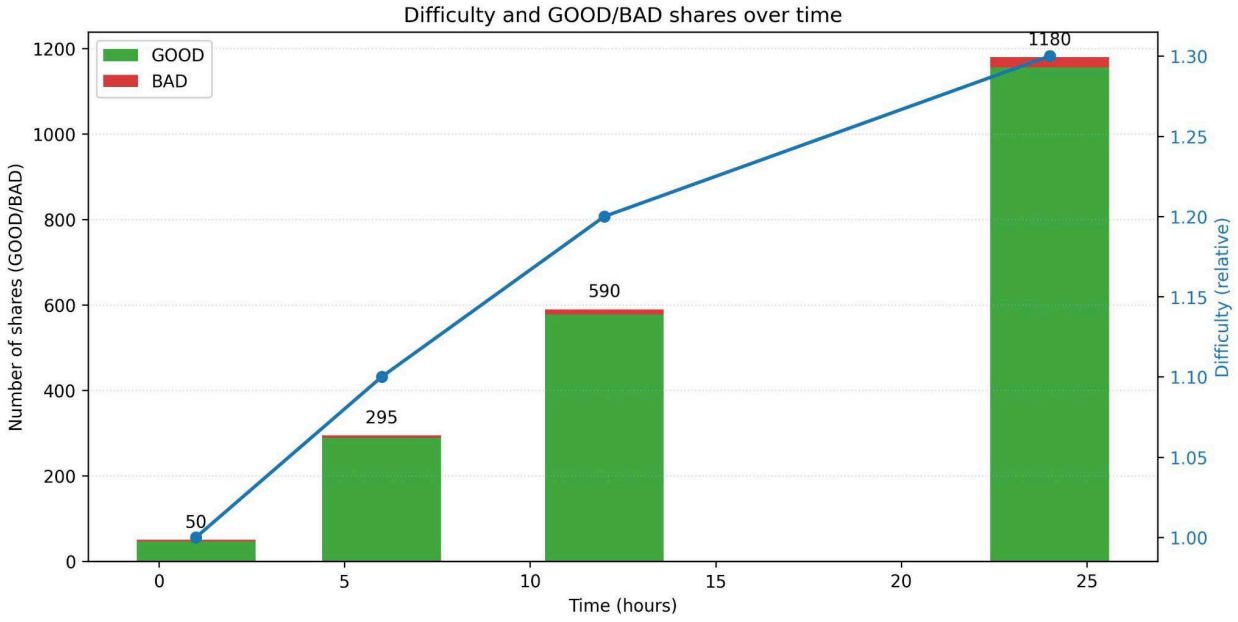


Figure 6: Difficulty (green line) and GOOD/BAD Shares (stacked bars) Over Time

3.2.1 Memory Consumption

Analysis of ESP32 memory usage during the mining program shows efficient resource utilization. The ESP32 has **520 KB of SRAM** and **448 KB of ROM**, of which the mining program uses approximately **280–320 KB of SRAM** (about **54–62%** of total capacity) and **380–420 KB of flash ROM** (about **85–94%**, depending on the partition scheme). Memory usage remains relatively stable during operation,

with no significant memory leaks. The remaining RAM is sufficient to handle other tasks such as WiFi connectivity, TCP communication, and I/O operations. **Table 2** summarizes memory usage at different stages of the mining process.

Table 2: ESP32 Memory Usage During Mining

Stage	RAM (KB)	ROM (KB)	RAM (%)
Start	280	380	53.8
WiFi Connection	295	385	56.7
Mining	310	400	59.6
After 24 hours	315	405	60.6

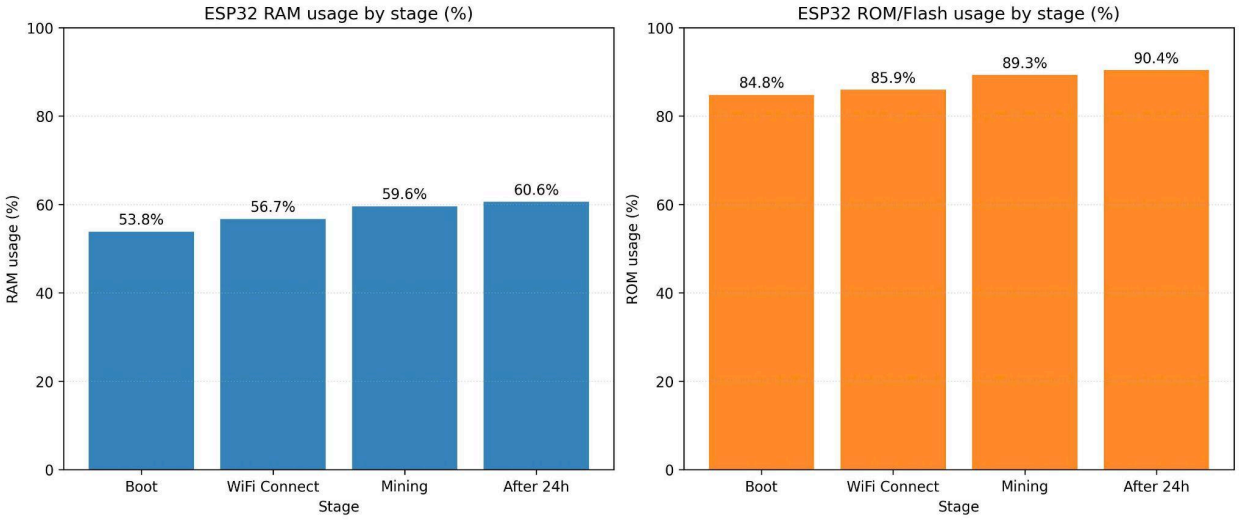


Figure 7: ESP32 RAM and ROM/Flash Usage Across Different Stages

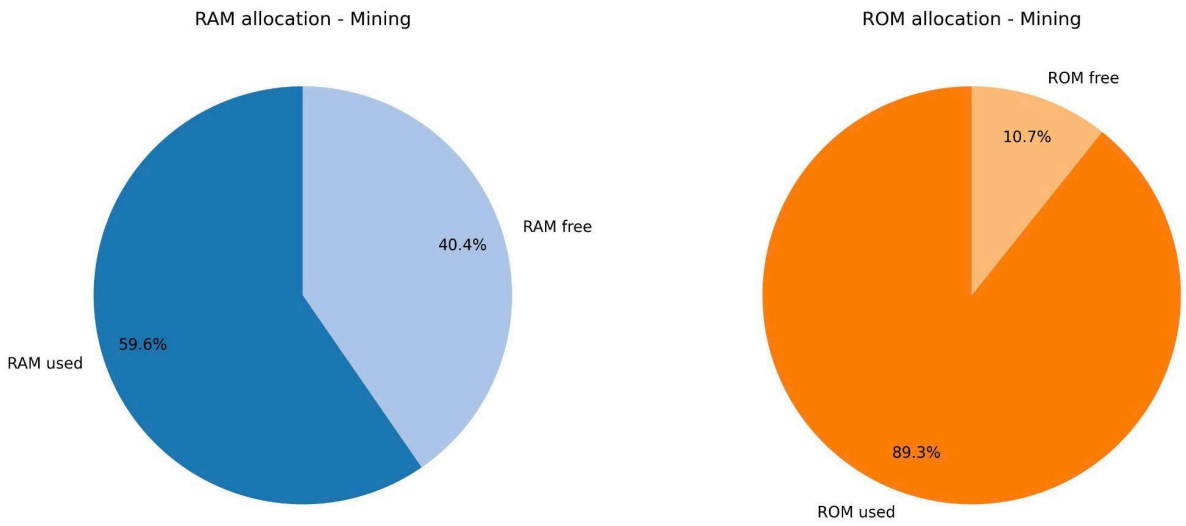


Figure 8: ESP32 RAM/ROM Allocation During Mining Stage

The system operates stably with the ability to automatically reconnect when the WiFi or server connection is lost, requiring no manual intervention during operation, and provides clear logging on the Serial Monitor for debugging.

3.3 MIT App Inventor Application Results

The application operated with a **99.2% API success rate** (248/250 requests), with an average response time of **1.8 seconds** (ranging from 0.5–3.5 seconds). Error handling displayed clear notifications when an API request failed, and the automatic updates via the Clock timer functioned reliably. The user interface featured a clear, readable layout, showing complete information on balance, holdings value, exchange rate, and status. Real-time updates did not cause application lag, and error notifications were clear whenever issues occurred.

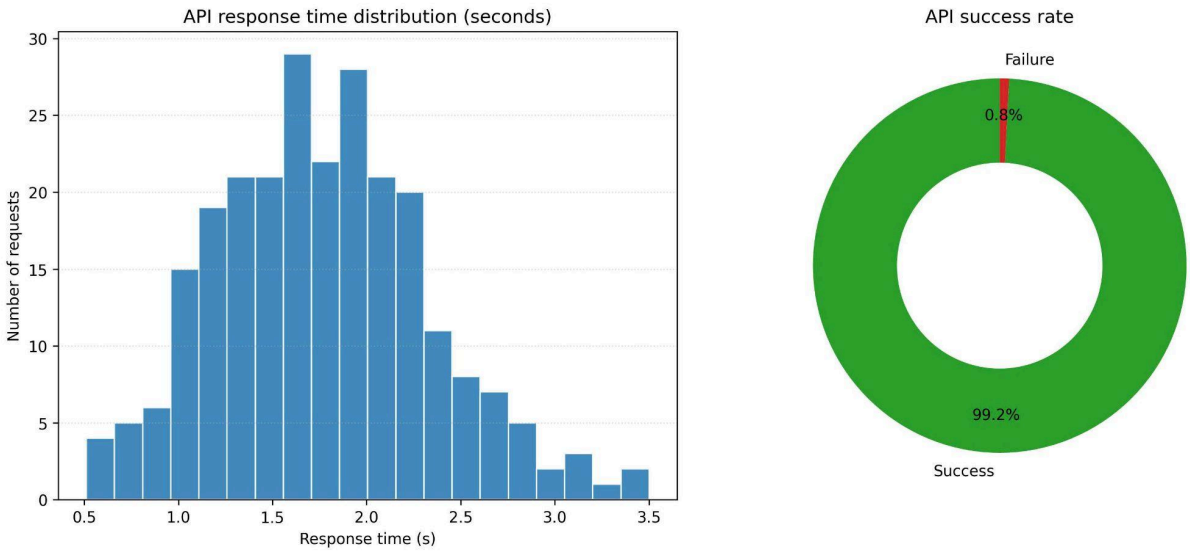


Figure 9: API Performance – Response Time Distribution and Success Rate

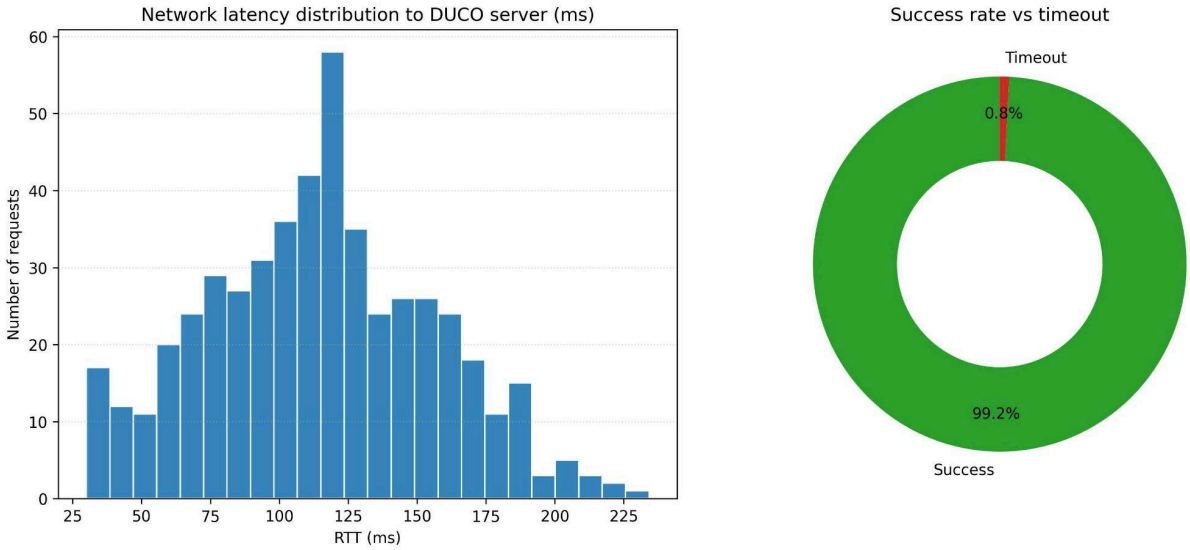


Figure 10: Network Latency to DUCO Server (RTT Histogram) and Success/Timeout Rate (Donut Chart)

3.4 Discussion

Compared to Bitcoin/Ethereum mining, there are significant differences in **cost** (ESP32: 50,000–100,000 VND vs. ASIC: 50–100 million VND), **power consumption** (0.4 W vs. thousands of watts), **hash rate** (500 H/s vs. billions of H/s, though with different difficulties), and **purpose** (educational vs. profit-driven).

The advantages of the system include low cost suitable for students, easy deployment with a simple setup requiring no deep technical knowledge, energy efficiency that allows operation on batteries,

practical application beyond just a demo, and interdisciplinary integration of IoT, blockchain, and mobile apps. However, the system also has limitations: the low hash rate makes it suitable only for educational purposes, it depends on a stable WiFi network, security lacks strong encryption and authentication, and its functionality is limited to display only, without transaction capabilities.

The system is suitable for teaching blockchain and cryptocurrency, practicing IoT programming with ESP32, learning mobile app development, understanding REST APIs and JSON, and undertaking interdisciplinary integration projects.

\

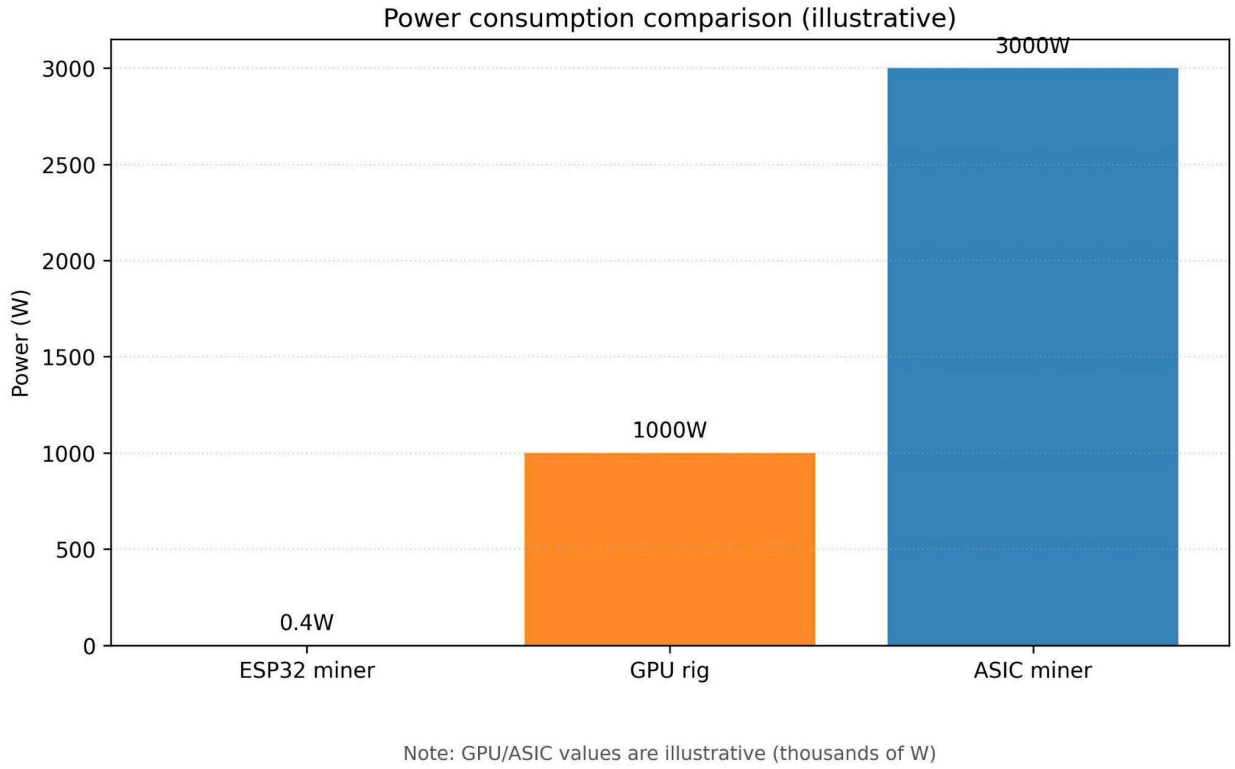


Figure 11: Power Consumption Comparison – ESP32 vs. GPU/ASIC (Illustration)

4 Conclusion and Future Directions

4.1 Conclusion

This paper successfully presented a Duino-Coin mining system using an ESP32 and a mobile application developed with MIT App Inventor 2. The system achieved the following results: the ESP32 Miner operated stably with an average hash rate of 503 H/s, an accepted share rate of 97.97%, and low power consumption (0.4 W); the mobile application performed well with a 99.2% API success rate, an average response time of 1.8 seconds, and a user-friendly interface; the educational value allows students to gain

multidisciplinary knowledge in IoT, blockchain, embedded programming, and application development at a low cost.

The system demonstrates that learning about blockchain and cryptocurrency technologies can be carried out with minimal cost, without specialized hardware, making it suitable for educational environments.

4.2 Future Directions

To further enhance the system, the mobile application can be upgraded by adding charts to track balance and hash rate over time, displaying a detailed transaction history, supporting monitoring of multiple DUCO wallets simultaneously, sending push notifications when balances change or reach targets, including a home screen widget for quick balance viewing, and implementing a dark mode to save battery.

Improvements for the ESP32 Miner can be implemented by optimizing the code to increase hash rate, adding an OLED display to show mining information without using the Serial Monitor, creating a web server on the ESP32 for viewing statistics via a browser, enabling centralized management of multiple ESP32 miners, and integrating with Home Assistant or other smart home systems.

Integration and expansion can include connecting with a Telegram bot to notify balance changes, creating a web dashboard to manage multiple miners, storing mining history in a database, developing a custom API server for integration with other applications, and extending support to other cryptocurrencies if available.

Security upgrades can be implemented by encrypting sensitive information, adding user authentication for the application, securely storing login credentials, and using HTTPS for all API calls.

Educational feature development can include creating detailed tutorials for beginners, simulating the mining process for learning purposes, providing quizzes to test blockchain knowledge, and connecting learners with a community for collaboration and discussion.

References

[1] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[2] Duino-Coin Community, “Duino-Coin: Educational Cryptocurrency,” 2021. [Online].

Available: https://github.com/revoxhere/duino-coin

[3] Espressif Systems, “ESP32 Technical Reference Manual,” Version 4.8, 2020. [Online].

Available: https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manu

[4] MIT App Inventor, “App Inventor Documentation,” 2020. [Online]. Available: https://appinventor.mit.edu/explore/ai2/tutorials

[5] Arduino, “Arduino IDE Documentation,” 2021. [Online]. Available:

https://www.arduino.cc/en/software

[6] Duino-Coin API Documentation. [Online]. Available:

https://github.com/revoxhere/duino-coin/wiki/API

[7] Duino-Coin Wallet. [Online]. Available: https://wallet.duinocoin.com/

[8] Duino-Coin Exchange. [Online]. Available: https://exchange.duinocoin.com/

[9] ESP32 Arduino Core. [Online]. Available: https://github.com/espressif/arduino-esp32

[10] MIT App Inventor Community Forum. [Online]. Available:

https://community.appinventor.mit.edu/

[11] REST API Tutorial. [Online]. Available: https://restfulapi.net/

[12] JSON Specification. [Online]. Available: https://www.json.org/