ECE 445

SENIOR DESIGN LABORATORY

FINAL REPORT

Clickers For ZJUI Undergraduate MkII

Team #23

ZHENYU ZHANG
(zhenyuz5@illinois.edu)
BENLU WANG
(benluw2@illinois.edu)
LUOZHEN WANG
(luozhen2@illinois.edu)
SUHAO WANG (suhao2@illinois.edu)

Sponsor: Professor Fangwei Shao

May 10, 2024

Abstract

The incorporation of technology in classrooms has led to the widespread adoption of tools such as the I-clicker, which facilitates interactive learning experiences and streamlines administrative processes. However, the current version of the I-clicker system faces limitations in accommodating a large user base and lacks mobile application support. This paper aims to address these challenges by enhancing system capacity, extending device compatibility and integrating innovative functions in the software side to form a multi-front-end multi-function answer system. The project includes front-end and back-end development, Clicker and receiver design. The objective is to design a system that supports 100 to 150 simultaneous users in a classroom, with low latency and multiple fronts.

Keywords: technology in education, I-clicker, interactive learning, classroom technology, educational tools, system enhancement.

Contents

1 Introduction

1.1 Background

In the contemporary educational landscape, the incorporation of technology within classrooms has witnessed a widespread adoption, yielding significant impacts on teaching and learning practices. Among the various tools introduced to enhance classroom interactivity and streamline administrative processes, the Clicker has emerged as a pivotal instrument. This technological solution serves as an indispensable element in meeting the digital demands of the classroom environment. By enabling the efficient collection of attendance data and promoting interactive learning experiences through question-and-answer sessions, the Clicker empowers students to actively engage with academic material, fostering a deeper understanding of complex concepts and improving overall learning outcomes.

However, despite its evident advantages, the present iteration of the Clicker system is confronted with inherent limitations that impede its ability to accommodate a substantial user base. Issues such as limited capacity to handle higher user loads, significant signal delays, and signal loss hamper the system's efficacy. The preceding iteration of the Clicker, known as Version 1, remained unfinished and underwent limited testing. It exhibited a restricted capacity to accommodate a large user base, thereby rendering it inadequate for practical use. Furthermore, the absence of support for mobile applications as an alternate means of participation fails to cater to the preferences of students who favor mobile technology. To ensure a seamless and engaging educational experience for all students, it is imperative to address these challenges and enhance the functionality of the Clicker system.In reality, there is already a mature commercial Clicker available in the market, which enables functionalities such as mobile-based attendance and question-and-answer features[1]. However, due to its expensive price and the inability to use it in mainland China, we aspire to develop our own system with similar capabilities and innovative features that support multi-platform synchronization.

1.2 Objective

Our project aims to enhance the I-clicker system, increasing its capacity to accommodate 100-150 users simultaneously and ensuring compatibility with various front-end devices, including mobile phones, PCs, and physical clickers. By expanding the receiver's range to cover an entire classroom and employing a closed-source architecture over an internal LAN, we aim to boost system reliability and security.

To achieve these objectives, our team has developed a comprehensive strategy encompassing four main components: front-end development, back-end implementation, Clicker design, and receiver design. This approach includes leveraging external resources such as teachers' computers for software deployment and existing routers for local area network connectivity. Additionally, by optimizing the database, we ensure that data processing latency remains under two seconds, even during high-volume conditions.

1.3 High-level Requirement Lists

- The system is designed to support 100-150 students in a classroom environment, allowing them to use the answer function efficiently and ensuring smooth and uninterrupted functionality for all participants.
- Students can submit answers using mobile phones, web apps, or physical clickers. The system can accommodate multiple classes simultaneously without any interference.
- The delay from pressing a button on a mobile phone or web page to the receiver receiving the signal should not exceed 2 seconds.

1.4 Block Diagram

Our system architecture is divided into three main subsystems: the power supply subsystem, the clicker subsystem, and the software subsystem. The power supply subsystem provides continuous operation, the clicker subsystem enables student interaction through button presses, and the software subsystem manages data and user interfaces. Additionally, we utilize external resources like teachers' computers for software deployment and existing network infrastructure for enhanced functionality.

By optimizing the database, we ensure response times remain under two seconds, ensuring a seamless experience. Using Wi-Fi technology, we enable secure, proximity-based student check-ins, supporting large groups effectively. This comprehensive system redesign sets a new standard for educational technology integration, supporting up to 100 students simultaneously and ensuring reliable and efficient operation through local area network transmission and adherence to established protocols.

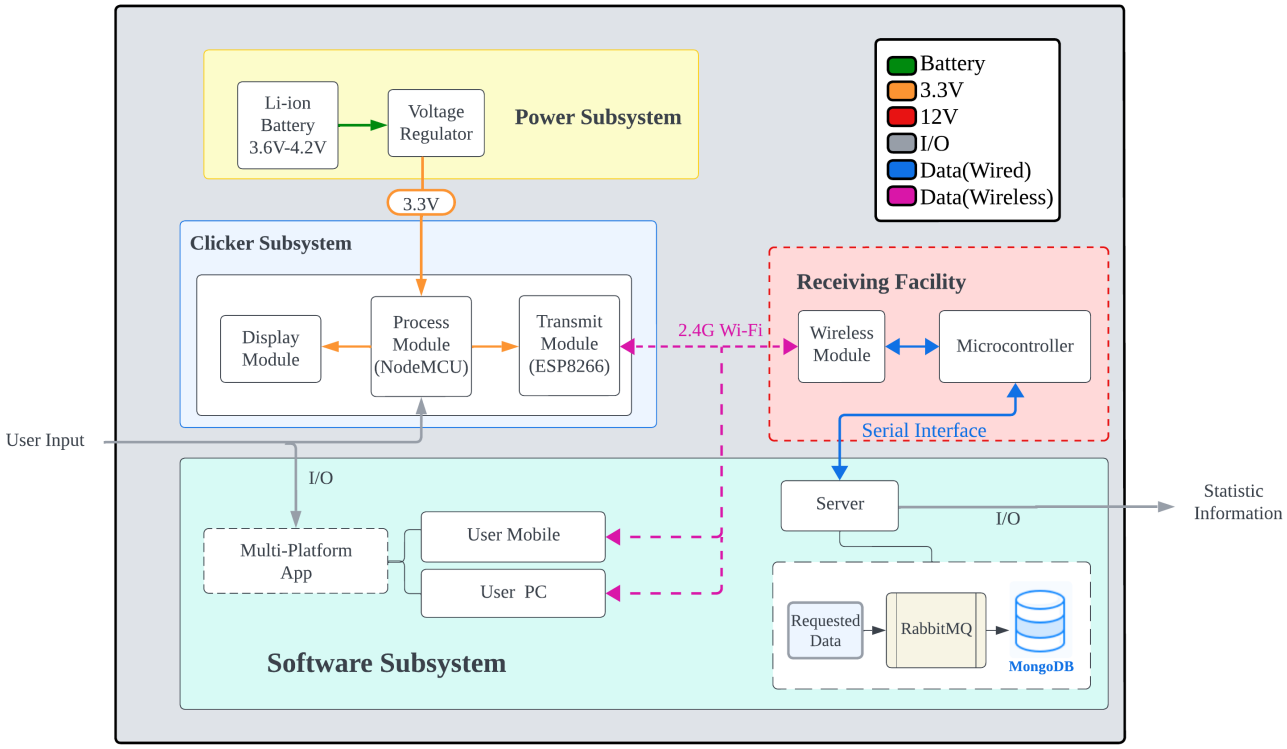


Figure 1: Block Diagram

2 Designs

2.1 Power Subsystem

This subsystem ensures that students and teachers can use the system for a long time without maintenance. This subsystem can power the Clicker subsystem.

Design Procedure & Alternatives
We need a DC power supply to power our microcontroller and OLED. Initially, our group considered dry batteries. Dry batteries typically have capacities between 1000 mAh and 3000 mAh, which, according to our calculations, meet the requirements of Clicker.

But dry batteries require bulkier packaging and a relatively large battery holder size, and once we use this design, our Clicker will be thicker than the thickness of the phone, which is a disadvantage.

And according to observations, in the case of the need to update the power supply every semester, users and students on Chinese campuses generally prefer to use charging ports. In summary, our team finally chose to use rechargeable lithium batteries.

In order to make the battery last longer and prevent the chip from getting too hot, we needed to add a voltage regulator module.

At first, we thought about using the LM317 module because it can control the output voltage. However, we soon realized that it has a high dropout voltage and is very complicated. Since we need to keep costs down for the Clicker project, we decided to go with the classic LM1117 regulator, which can keep the output voltage stable at 3.3 V while working with an input voltage range of 3.7 V to 4.2 V. And it can handle up to 200 mA of current, which is suitable for our Clicker.

Design Details

2.1.1 Micro Lithium Battery

We chose to use the miniature lithium battery 605060 with 2500 mAh (fluctuating from 3. 7V to 4.2 V) as the power supply as shown in Figure 3. Its size is more suitable for the final PCB size, and it can be completely hidden in the PCB, meeting the aesthetic requirements.

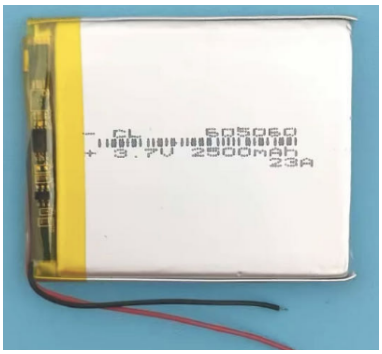


Figure 2: battery

2.1.2 Regulator

In this project, the voltage regulator module LM1117 is mainly used to reduce the voltage input from 5 V of lithium battery to 3.3 V. There is some adjustable voltage versions of LM1117, which can achieve the output voltage range from 1.25 V to 13.8 V through two external resistors, but our project only needs to use a fixed output of 3.3 V. The LM1117 datasheet shows that 3.3V is a preset output, so we can use the corresponding model directly[2]. The following diagram shows how this module is connected.

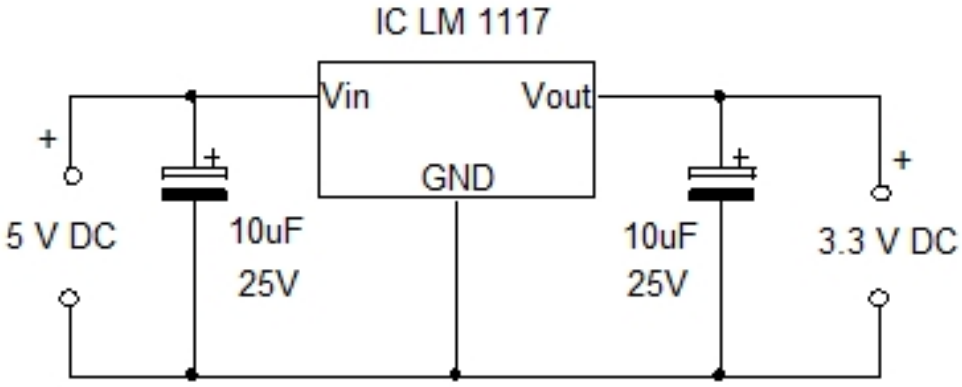


Figure 3: connect LM1117 to the circuit

2.2 Clicker Subsystem

This subsystem ensures that 100-150 students can send answers in a stable and efficient manner. This subsystem can send signals to the wireless receiving subsystem.

Design Procedure & Alternatives
The previous Clicker design incorporates the nRF24L01 wireless transmission module,

which utilizes Bluetooth technology to transmit signals. Notably, the nRF24L01 module exhibits low power consumption, offers multiple low-power modes, and enables data transmission over greater distances compared to WiFi technology. However, it should be noted that the data transmission capacity of nRF24L01 is lower compared to ESP8266, which could support transmission at 300 kbps-4.5 Mbps.

LCD screens offer certain advantages over OLED displays, such as the absence of strobe effects, high pixel density, and long lifespan. However, after thorough evaluation, the ultimate decision was made to opt for an OLED display due to its low power consumption of 0.04W.

We can place all the electronics on a breadboard or PCB. the thickness of the breadboard is usually between 8 mm-10 mm, and if we add the ESP8266 NodeMCU development board and the battery, the overall thickness may reach 30 mm, which is very uncomfortable for human hands to hold. Therefore, a breadboard is not a good vehicle for realising a physical Clicker. In contrast, PCB board is a better choice.

As for the shell material, While acrylic is an ideal material for the enclosure, its transparency allows the user to easily see the internal circuits and batteries, which may be uncomfortable for the user as the cluttered circuits are not aesthetically pleasing. In addition, the thickness of acrylic sheet is fixed, although we can cut the design precisely on a flat surface by laser cutter, but for some three-dimensional designs with certain thickness, acrylic sheet is not well realised. In the context of the project, a moderately strong, opaque 3D printing material is a more appropriate choice.

Design Details

2.2.1 Transmit Module

We finally use the ESP8266 as the functional module for data transmission to connect with the central processing module. The ESP8266 chip supports the standard IEEE 802.11b/g/n Wi-Fi protocol, and is able to connect with the wireless network. It realises wireless communication and data transmission through the built-in Wi-Fi module. The microcontroller inside the Clicker can connect to the teacher's receiver through this chip to achieve data exchange. It realises wireless communication and data transmission functions through the built-in Wi-Fi module.The microcontroller inside Clicker can be connected to the teacher's receiving end through the chip to realise data exchange. The ESP8266 chip is also characterised by its low power consumption, which enables it to operate stably under low-voltage and low-power conditions. This also makes it ideal for use in battery-powered scenarios, in line with Clicker's needs. In addition, advanced power management technology is integrated inside the chip, which enables intelligent sleep and wake-up functions to further reduce energy consumption and ensure the single battery life of the Clicker.

2.2.2 Display Module

The inclusion of a 0.96 inch OLED screen in the Clicker apparatus serves the purpose of furnishing users with requisite feedback. The chosen OLED screen is capable of accommodating a broad spectrum of power inputs, spanning from 3.3 V to 5 V. Noteworthy is its minimal power consumption rate, standing at a mere 0.04 W. Equipped with a resolution of 128 * 64 pixels, the screen ensures clarity of visual output. Furthermore, its wide viewing angle surpassing 160 degrees enhances user engagement. Communication between the OLED screen and the ESP8266 development board is facilitated through the utilization of the IIC protocol.

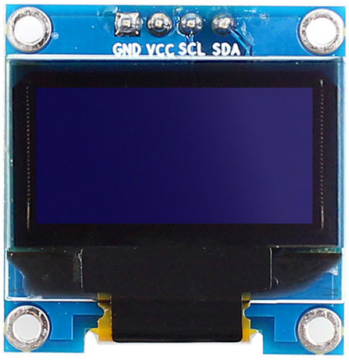
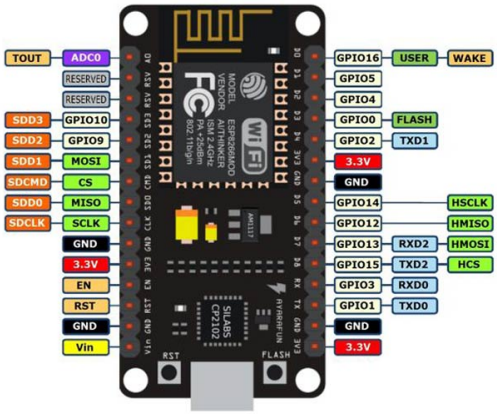


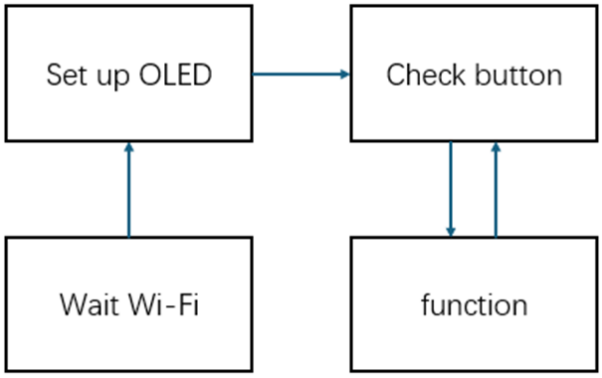
Figure 4: OLED

2.2.3 Hardware Process module

We use the NodeMCU development board (CP2102), which works with the ESP8266, as the signal processing module. It can send the information pressed by the user through the button to the receiver through the ESP8266 module. NodeMCU is an open source iot platform based on ESP8266. NodeMCU firmware can run on the ESP8266 Wi-Fi SoC (system-on-chip), which contains hardware and firmware specifically designed for the ESP-12 module. The Clicker logic diagram is shown below.



(a) NodeMCU



(b) Flow Chart Diagram

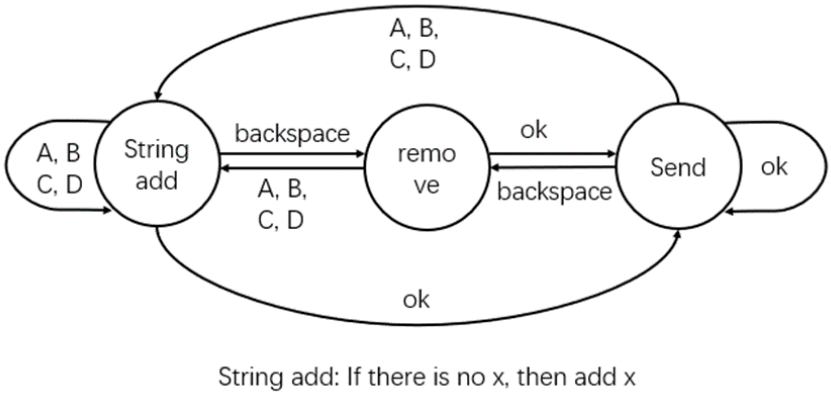


Figure 6: State Diagram

2.2.4 Circuit & PCB Design

Button functionality was a key consideration in our design. We designed the circuit with a total of eight buttons: ABCD, Back, Send, Shutdown (deep sleep mode), and Restart. The Restart button can be connected to a separate pin (RST), so the other buttons require a total of seven additional pins. NodeMCU offers nine digital ports (D0-D8) that can be manually configured for signal input and output. According to our design, we set D1 and D2 as the serial clock line (SCL) and serial data line (SDA) for the OLED display, respectively, and D0, D3-D8 as the input ports for the button signals.

Considering that the user may change his/her mind for further thinking when he/she enters the answer using the physical Clicker, we need to design a delete button so that the user can delete the entered options. We also need a button to confirm the choice on behalf of the user, when the button is pressed, the physical Clicker will send the answer to the server.

Based on the analysis of the functional requirements, we designated ports D0, D3, D4, and

D5 to represent the four options A, B, C, and D in multiple-choice questions, respectively. D6 was designated as the port to send the reply, D7 as the port to delete an answer, and D8 as the port to enter deep sleep mode.

Between 3.3 V and GND, we place the buttons and resistors in series, and we choose resistors with a size of 10 kΩ to ensure that the ports can accurately recognise high and low potentials. Meanwhile, in the design, the circuits of D0, D3-D7 use pull-down circuits, i.e., when the button is pressed, the port uses a low level as the trigger condition; while D8 uses a pull-up circuit, i.e., when the button is pressed, the port uses a high level as the trigger condition.

In addition, we have also considered the jitter problem that may be brought about when the button is pressed, so a capacitor is connected in parallel with the button switch to eliminate the mechanical jitter of the button. Generally, the jitter frequency of mechanical buttons is about 100 HZ, and the mechanical jitter time is about 10 ms. According to the capacitor charging and discharging time formula $t = 0.7\sqrt{RC}$, we choose R = 10 kΩ, C = 0.1 μF, at this time, the capacitor's charging and discharging time is about 20 ms, which is greater than the jitter time, so as to effectively eliminate the impact of jitter.

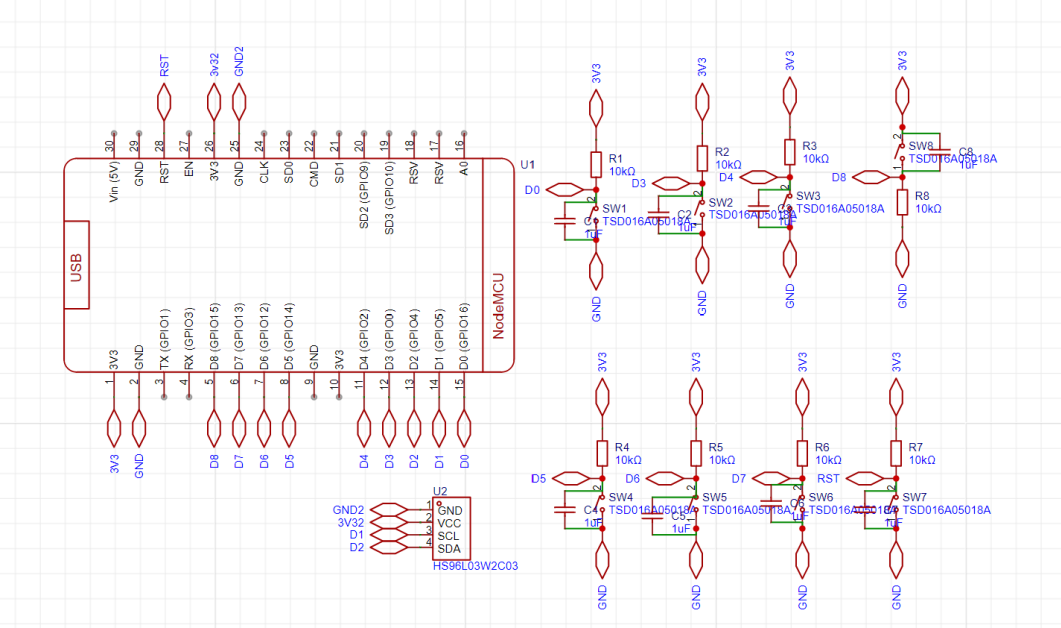


Figure 7: Schematic Diagram

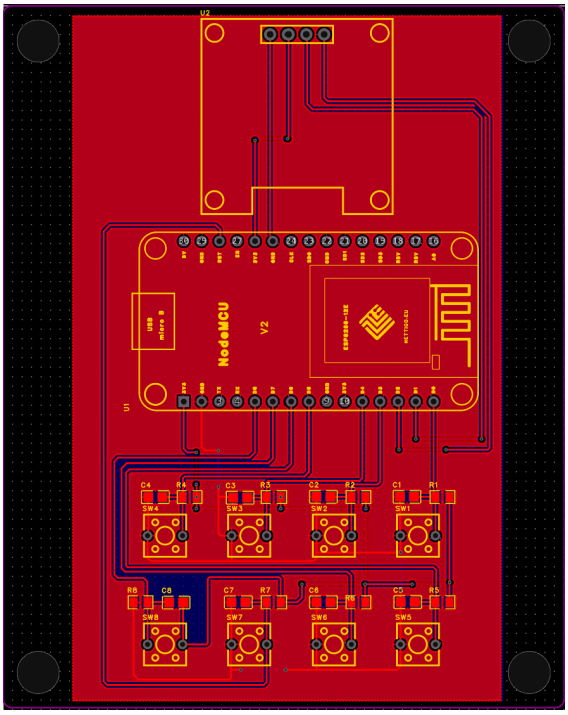


Figure 8: PCB Diagram

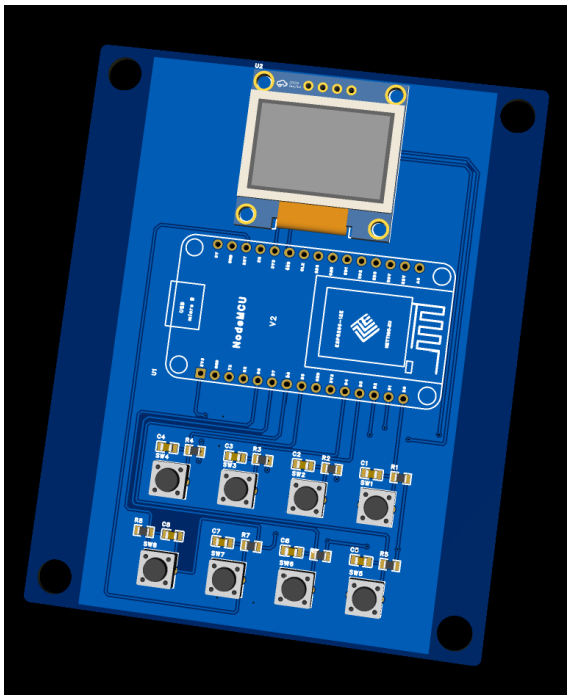
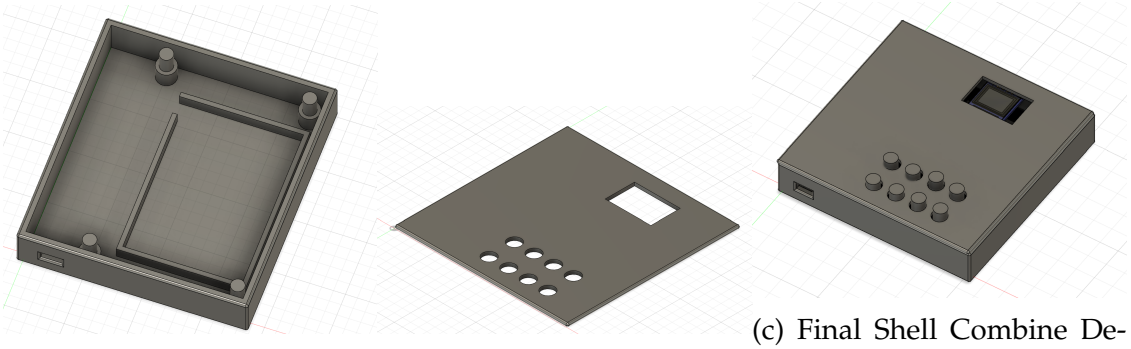


Figure 9: PCB 3D Diagram

Figure 10: Real PCB Diagram

2.2.5 Shell Design

The battery and PCB need to be placed in the shell. In order to reduce the size of the shell, we chose to arrange the battery with the PCB in a top-down manner, placing the battery in the lower part of the PCB. On the PCB board, we pre-designed four holes. In the design of the shell, we designed four pillars whose sizes match these four holes, which can have the effect of fixing the PCB board in the shell, as well as putting down our battery in the space between the PCB board and the shell. At the same time, in order to prevent the battery from moving randomly inside the housing, we designed an enclosure matching the size of the battery to hold it in place. In addition, we designed an opening at the lower end of the case whose size matches the size of the female micro-usb charging port of the battery, through which users can conveniently charge the physical Clicker using the micro-usb charging cable. In the print setup of the shell, we used a 30% infill density, and in the end we were able to reduce the weight of the shell to 60 g, which makes it very lightweight.



(a) Shell Button Design (b) Up Shell Design (c) Final Shell Combine Design

Figure 11: Shell Designs

2.3 Software Subsystem

This subsystem ensures that the signal range requirements in the high level requirement can be met, and because of the optimized architecture used in the backend, we expect to make the system more responsive. This subsystem, on the one hand, acts as the client that sends the signal, on the other hand, acts as the management side that processes the database information and finally displays the statistical results. It can receive data from the wireless receiving subsystem.

2.3.1 Frontend Design

- **Student Android App**

Our Android front-end technology stack now centers around the Flutter framework, utilizing the Dart programming language, selected for its excellent safety features, conciseness, and high interoperability with native code. Dart supports sound null safety, which

helps prevent common programming errors like null pointer exceptions, thereby enhancing code safety and boosting developer productivity. As Dart is officially supported by Google, it ensures that our application can continuously integrate the latest technological advancements and best practices, securing its sustainable development in the future.

Furthermore, development is carried out in the Android Studio integrated development environment (IDE), leveraging its array of efficient tools and features such as code autocompletion, performance analysis, and a visual layout editor to accelerate the development process. By incorporating Flutter widgets and the extensive libraries available in Flutter, building responsive UIs becomes effortless, while the use of packages like Sqflite for database operations simplifies data management, ensuring high application performance and smooth user experience. This choice of technology stack not only optimizes the development process but also lays a solid foundation for the app's long-term maintenance and upgrade, ensuring it can evolve to meet future user needs and expectations.

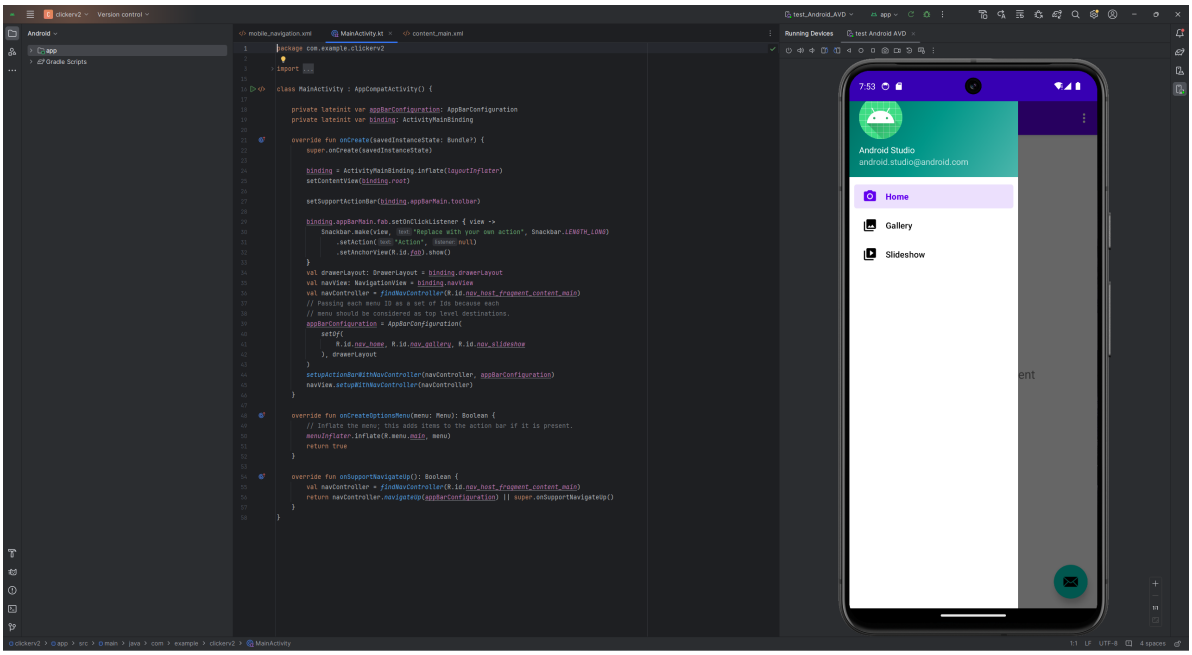


Figure 12: Android Design IDE

- **Student iOS App**

In our iOS frontend tech stack, we have made the strategic decision to utilize Flutter as the primary development framework. This choice is motivated by Flutter's exceptional capability for building high-performance, visually appealing applications across multiple platforms, including mobile, web, and desktop, from a single codebase. Flutter distinguishes itself through rapid development cycles facilitated by its hot reload feature. This allows developers to instantly see the effects of their changes in real-time, maintaining

the application state, which significantly speeds up the development process by enabling quick experimentation with UI designs and bug fixes. The framework employs Dart as its programming language, which merges the best aspects of dynamic languages with the efficiency of ahead-of-time compilation to native code, making it particularly suitable for front-end development. Dart, in combination with Flutter's comprehensive widget library and reactive framework, enables developers to craft complex, custom interfaces with smooth animations and transitions that enhance user experiences.

For development within this tech stack, Visual Studio Code has been chosen as the integrated development environment (IDE). Visual Studio Code supports Flutter development by offering a lightweight, yet powerful environment with rich features such as advanced code editing, debugging, and extension support. This IDE choice complements Flutter's flexibility and efficiency, providing an excellent toolset that boosts productivity for developers accustomed to different environments. Although Visual Studio Code is now our primary IDE, Flutter projects can still be integrated with Xcode for specific iOS-related tasks, such as app icon configuration, provisioning profile management, and other necessary adjustments for iOS deployment. This approach ensures that we can still leverage essential native tools and settings while benefiting from Flutter's cross-platform development capabilities. Adopting Flutter, paired with Visual Studio Code, empowers our development team to produce a consistent and high-quality user experience across all platforms, streamlining the development process, enhancing our ability to quickly adapt to market changes, and simplifying long-term app maintenance and iteration.

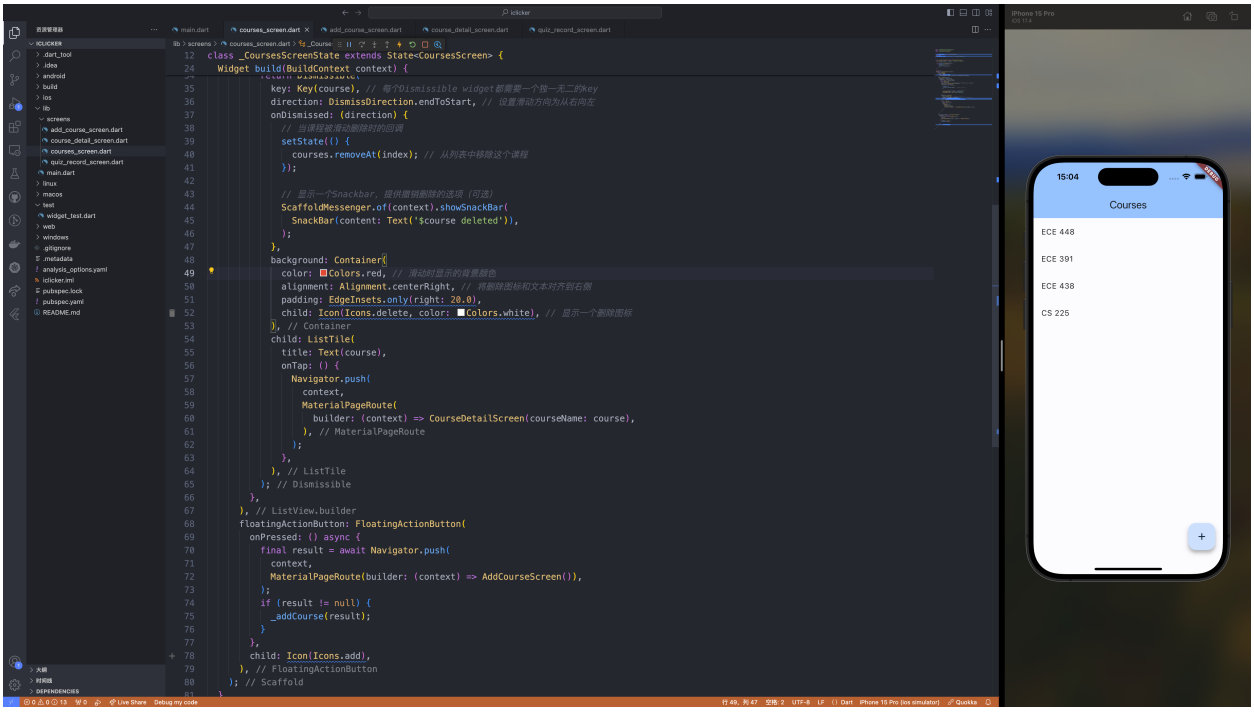


Figure 13: iOS Design IDE

- **Teacher Web Interface**

For our web interface, we've chosen React as the cornerstone of our frontend tech stack. React is a popular JavaScript library developed by Facebook, known for its efficiency in building interactive and complex user interfaces. Its component-based architecture allows for the development of reusable UI components, promoting code reusability and simplification of the development process. This architecture not only accelerates the development cycle but also ensures consistency across the application. React's virtual DOM (Document Object Model) is another key feature that enhances the performance of web applications by minimizing direct DOM manipulation, leading to faster rendering times and a smoother user experience.

To complement React, we integrate tools like Redux for state management, enabling us to maintain a predictable state across the entire application in a centralized store. This is particularly useful in complex applications with large amounts of data and interactions, as it simplifies state management and facilitates communication between components. For routing, we use React Router to manage navigation within our application, ensuring that users can seamlessly move between different parts of our web interface without page reloads, mimicking the feel of a native application.

2.3.2 Backend Design

Our software system's backend design is centered around the use of an SQL database, providing strong support for managing structured data such as user information and transaction records. The choice of an SQL database is crucial for ensuring data accuracy and reliability due to its strict data consistency, transaction management, and complex query capabilities. To enhance the system's ability to handle high concurrency and accommodate the diversity and scalability of database content, we have also integrated MongoDB, a NoSQL database. The incorporation of MongoDB allows our system to handle unstructured or semi-structured data more flexibly, such as log files and JSON data. Its schema-less nature facilitates rapid development and iteration, while its high performance and horizontal scaling capabilities enable the system to easily manage growing user bases and surges in data volume.

To manage communication and asynchronous processing, particularly under high load scenarios, the backend incorporates RabbitMQ, a message queue system. RabbitMQ serves as the backbone for handling communication between different services in the backend, ensuring that data processing remains efficient and reliable, even during peak usage times. This approach aids in maintaining the responsiveness and stability of the application, crucial for providing a seamless user experience. The combination of MongoDB and RabbitMQ in the backend is strategic, catering to the need for high performance, scalability, and reliability in handling concurrent operations and data management.

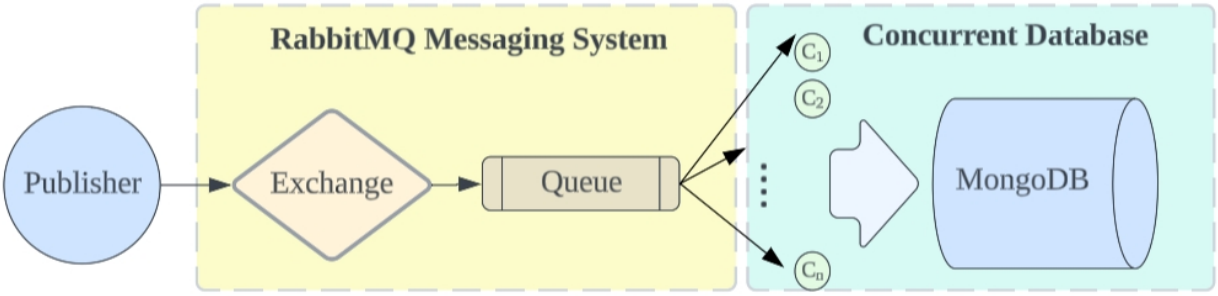


Figure 14: Backend System Diagram

2.3.3 Authentication module

Our application incorporates a robust identity verification mechanism that enhances security by utilizing unique device identifiers such as the Android ID. To facilitate the retrieval of the Android ID specifically for Android devices, we employ Flutter's channel feature, which allows for direct communication with the native platform—Android in this case—since accessing such device-specific information goes beyond the capabilities of Flutter's default SDK.

The process to obtain the Android ID involves defining a `MethodChannel` within the Flutter application with a unique channel name, such as `com.example/deviceInfo`. When the Android ID is needed, the Flutter side sends a method call request named `getAndroidId` through this channel. This request is transmitted to the native Android side via Flutter's platform channel.

On the Android side, within the `MainActivity` class, this specific channel is monitored. Upon receiving the `getAndroidId` method call from Flutter, the native code retrieves the Android ID using the method `Settings.Secure.getString(contentResolver, Settings.Secure.ANDROID_ID)`. Once obtained, the Android ID is sent back to the Flutter side through the same channel as a string. The Flutter application then utilizes this Android ID within its identity verification process.

The system automatically recognizes and collects the Android ID when a user first logs in on a device. During the device registration phase, the user's account information is associated with this Android ID and securely stored on the server. This ensures that each account can only be operated on its bound device, effectively preventing unauthorized access from other devices.

When users attempt to bind a new device or change a bound device, the system requires them to undergo secondary verification to confirm the legitimacy of the device binding or change request. Additionally, the system continuously monitors and records all login attempts and device change activities. Through this continuous monitoring, the system can promptly detect and prevent fraudulent activities, such as proxy attendance, maintaining the integrity and fairness of the system.

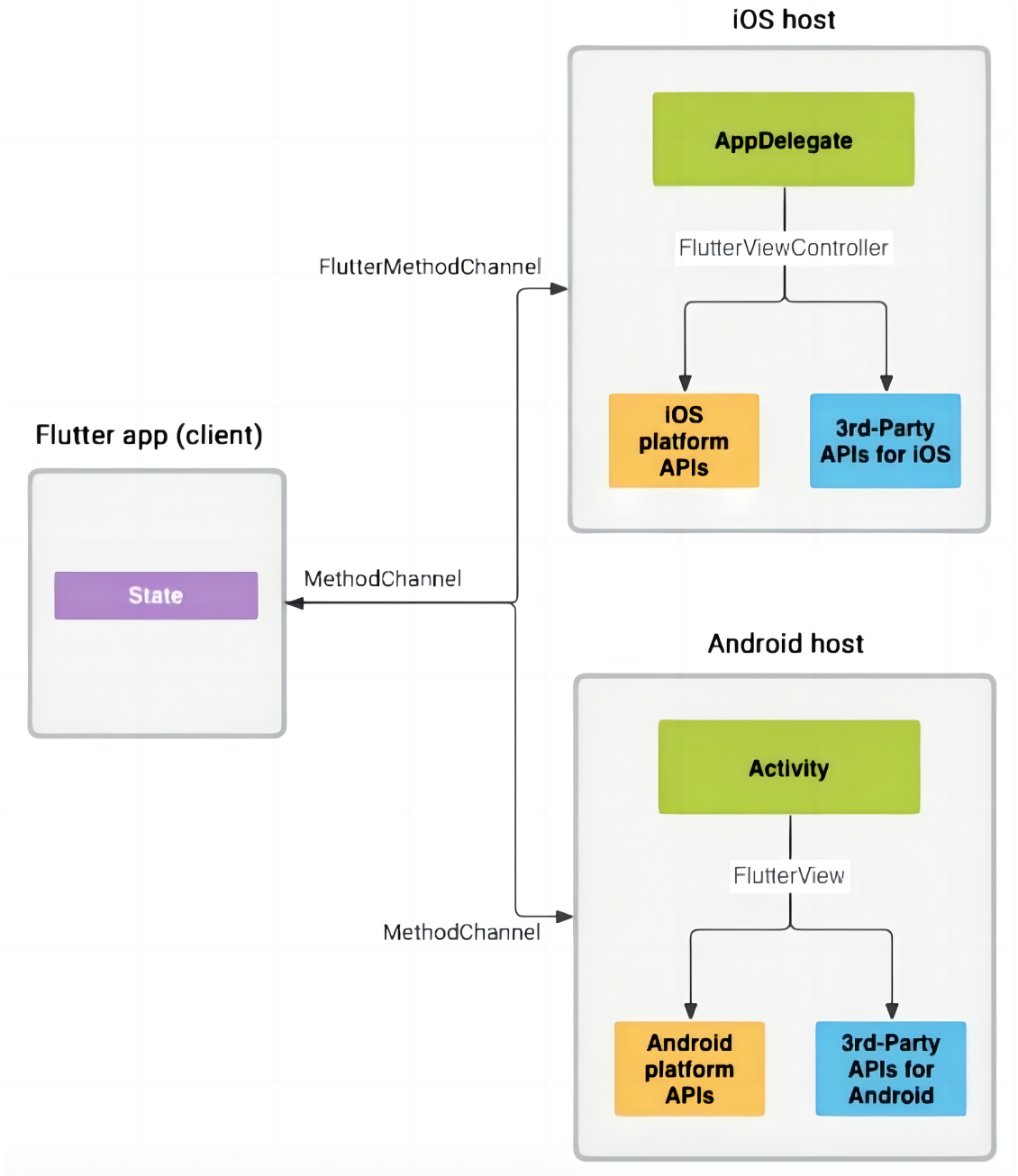


Figure 15: Flutter Channel

3 Verification

3.1 Power Supply Verification

To consider whether the capacity of the battery is appropriate, we need to measure and calculate the Clicker's power consumption current and power consumption, and compare it with the battery capacity.

The system involves pressing a button to wake up the ESP8266, sending a 1-second signal, and then entering the sleep state again. The signal transmission process consumes approximately 100 mA, while the deep sleep power consumption specified in the ESP8266 data manual is 20 µA [3].

Table 1: Clicker Current

Components	**Current**
D0, D3-D8	0.33 mA
OLED	20 mA
ESP8266	80 mA
Full Performance Current	110 mA
Wi-Fi Synchronizes Current	46 mA

Suppose that students turn on and connect to Wi-Fi for 1 hours a day and work for 100s at full performance (pressing 50 buttons, each lasting 2s). Then the total power consumption is about 49 mAh (46 * 1 + 110 * 0.028). Thus the 2500 mAh battery can last for around 50 days after a full charge. We assume that the students will put clicekr into deep sleep mode in time after class. According to the conservative estimate above, clicker can be used for about 2 months on a single charge, which means that it can meet the requirement of convenience and the capacitor of the battery is suitable.

3.2 Shell design Verification

Considering the strength analysis of the real case, we will test the shell for damage when it is dropped statically from a height of 60 cm, while we perform a stress analysis of the shell using fusion360. For the stress analysis of the shell, we set the weight of the physical i-clicker to be 300 g. It falls from 60 cm and the contact time with the ground is 0.05 seconds. According to the momentum impulse theorem $Ft = Mv, t = 0.05s, M = 300g, v = \sqrt{2gh} = \sqrt{2 * 9.8 * 0.6}$, the force is 20.575N. In fusion360, we assume that the side is fixed, and simulate the test force perpendicular to the fixed surface and parallel to the fixed surface, respectively.The simulation results show that when the force is perpendicular to the fixed surface, the maximum stress suffered is 2.422Mpa, and the safety factor is 8.26; when the force is parallel to the fixed surface, the maximum stress suffered is 2.244Mpa,

and the safety factor is 8.91.The surface of the simulation data that the The design of the shell is consistent with meeting our requirements.

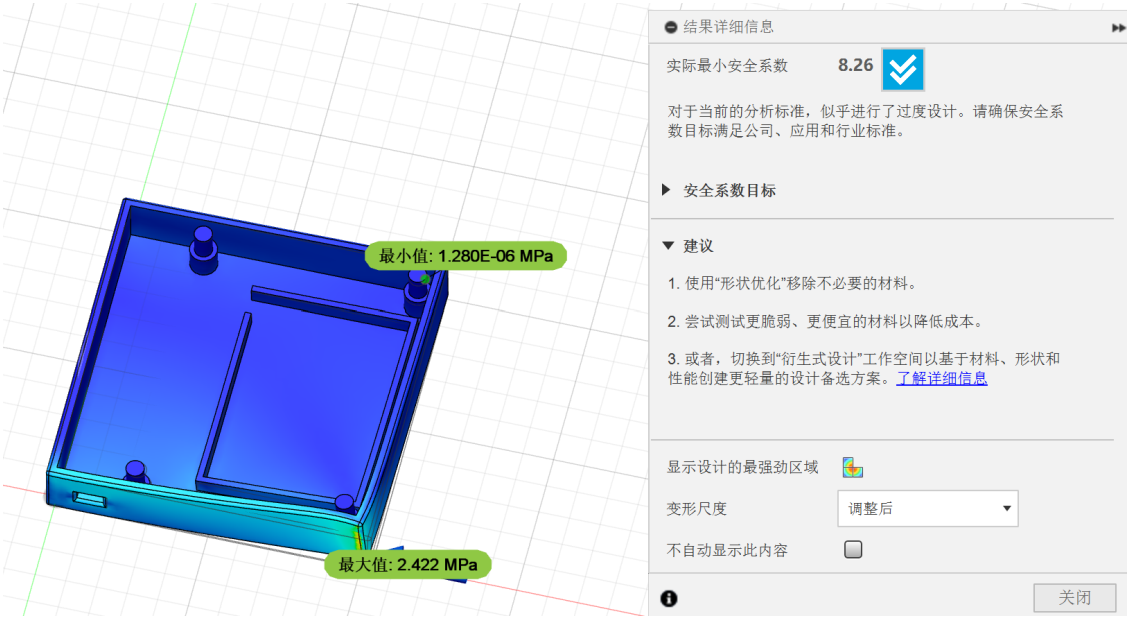


Figure 16: perpendicular force test

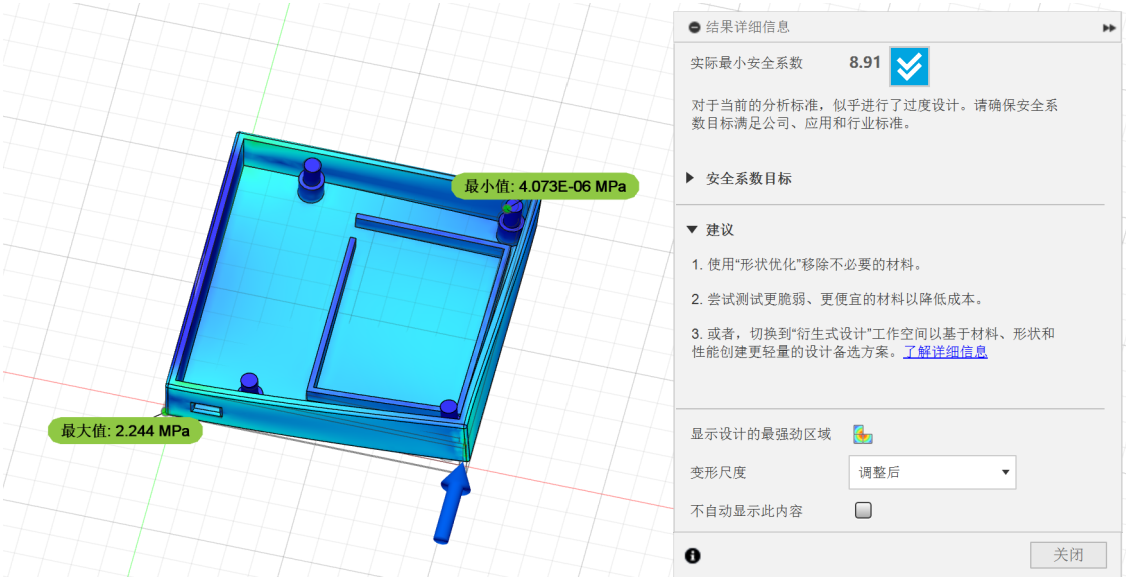


Figure 17: parallel force test

3.3 Software Subsystem Verification

Verifying our software system is crucial to ensure all functions operate smoothly and reliably, without encountering malicious bugs such as crashes or freezes. Additionally, it is essential to ensure both the frontend and backend systems can maintain stable data transmission and processing, even under high concurrency with over 500 simultaneous users.

3.3.1 High-Concurrency Testing

- Python scripts were tested with 500 simultaneous users.
- The system was field-tested in the MATH257 discussion class with over 60 students and TAs.

3.3.2 Software Development

- The application was tested on 40+ Android devices, iOS, multiple browsers, and computers.
- It is ready for distribution in app markets.

3.3.3 Software Security

- Executed malicious code tests to ensure robustness.
- Submitted specially formatted data for testing.
- Utilizes standardized APK signing for secure distribution.

3.3.4 Android & iOS Application

For our mobile app, the functionalities verified include:

- Account registration and login.
- Maintaining user identity information across different pages after login.
- Adding and displaying courses.
- Showing the list of available quizzes set by the teacher in a course.
- Displaying detailed quiz questions and options.
- Enabling multiple selections and successful submission of answers.
- Ensuring the app can only be used within Zhejiang University's intranet.

3.3.5 Teacher's Web Interface

For the teacher's web interface, the functionalities verified include:

- Teacher management account registration and login.
- Maintaining identity information across different pages after login.
- Displaying a list of courses and available quizzes in each course.
- Adding and removing quizzes, setting specific quiz questions, and correct answers.
- Displaying students' quiz responses.
- Ensuring the system can only be accessed within Zhejiang University's intranet.

4 Cost & Schedule

4.1 Cost

4.1.1 Labor Cost

According to a report on employment data released by Chinese Education Online[4],fresh graduates with a bachelor's degree in computer science earn about ¥6,800 a month,or ¥42.5 an hour. Fresh graduates of mechanical engineering earn around ¥6,000 a month, or ¥37.5 an hour. We have 8 weeks this semester. Assuming that each person spends 10 hours on the graduation project every week, we will spend a total of 8*10 = 80 hours on this project.

Table 2: Labor cost

Name	Major	Hourly Salary	Hours Needed	Total Cost
Zhenyu Zhang	ECE	42.5	80	3400
Benlu Wang	ECE	42.5	80	3400
Suhao Wang	ECE	42.5	80	3400
Luozhen Wang	ME	37.5	80	3000
Total				13200

4.1.2 Parts Cost

Table 3: Parts Cost (Part 1)

Description	Quantity	Manufacturer	Cost/Unit	Total Cost
TP-LINK TL-WDR5620 router	1	TP-LINK	109	109
ESP8266 NodeMcu microcontroller	1	ZEJIE	18.8	18.8
0.96″ OLED display screen	3	TELESKY	11.68	35.04
5V Boost Li-ion Battery 2500mAh capacity	2	ZON.CELL	25.91	51.82
A56 Cap 6*6 Keypad Cap for matrix keyboard	2	ZEJIE	2.6	5.2
0805 10K Chip Resistor with 1% Accuracy	100	zave	0.0248	2.48
0805 100NF Chip Capacitor with 10% Accuracy	50	zave	0.0576	2.88

Table 4: Parts Cost (Part 2)

Description	Quantity	Manufacturer	Cost/Unit	Total Cost
3.7v lithium battery 2500mAh	1	ANGJIE	20.56	20.56
Middle 2-leg 6*6*5mm micro-keypad switches	40	ZEJIE	0.1225	4.9
TP4056 charging power module board	1	TELESKY	4.07	4.07
RNT Matrix Keyboard module 1 row of 4 keys	2	RNT	1.71	3.42
			Total	**258.17**

4.2 Schedule

See the Appendix A.

5 Conclusion

5.1 Accomplishment

In conclusion, the software system developed for the enhanced I-clicker project has successfully met its design objectives and demonstrated its capability to support a large number of simultaneous users with minimal latency, thereby significantly improving the interactive learning experience in classrooms. The frontend design utilizing Flutter for both Android and iOS applications, combined with a robust backend architecture featuring SQL and MongoDB databases, ensures a seamless, efficient, and secure operation. The integration of these technologies facilitates an engaging and responsive user interface while maintaining high performance under varying load conditions.

The authentication module, designed with security as a priority, employs unique device identifiers to enhance system integrity and user trust. The successful verification processes of the software subsystem, including extensive testing of user interactions and data handling on both mobile applications and the teacher's web interface, have validated the reliability and stability of the system.

In a multi-front answer system, the physical Clicker plays a crucial role as an important component. It allows users to input options and send them to the server over Wi-Fi. The project utilizes ESP8266-based NodeMCU and Arduino programming to achieve this functionality. Additionally, OLED displays are incorporated into the PCB design to facilitate interaction. Furthermore, a housing design is created using 3D printing technology to protect the PCB and peripheral circuits, ensuring user comfort when using the physical i-clicker. The power supply subsystem ensures stable voltage for the physical Clicker,

with a large capacity battery and voltage regulator module included in the design. Overall, the physical Clicker effectively meets its planned goals, resulting in lower latency in classroom settings.

5.2 Ethics

Everything we do is in compliance with the IEEE Code of Ethics.

During the design and development of our products, we always adhere to the highest ethical standards and avoid any violations of the law. We ensure that the privacy of the users of our products is protected, regardless of the transmitting terminal. We guarantee to collect only the personal data we need, including but not limited to account information, device information, location information, locally stored bio metric information, etc. We will use 2.4 GHz, 5 GHz Wi-Fi as the data transmission medium, and we will strictly control the emission power of the RF module to prevent the radiation from harming the user. This is our adherence to Article 1 of the IEEE Code of Ethics[5].

Based on respect for human rights, our devices are open for use by everyone, regardless of race, religion, gender, disability, age, nationality, sexual orientation, gender identity or gender expression. This is our adherence to Article II of the IEEE Code of Ethics[5].

5.3 Uncertainties

The Clicker we designed has the following possible uncertainties.

Component differences: The electronic components used internally may vary depending on the manufacturer. In addition, resistors, capacitors and integrated circuits may be damaged and aged over time, all of which may prevent Clicker from meeting the high level requirement

Wi-Fi connection issues: Clicker does not use a closed source system, but chooses to use an external Wi-Fi facility, and the stable performance of the network will undoubtedly affect Clicker usage. Signal interference and poor router performance will undoubtedly cause Clicker to fail at times.

While we could theoretically support more loads, actual testing has not yet been done, and future tests may reveal other limitations to the transmitted signal.

Power supply stability problem: Due to the short development time of the project, we did not fully test how long it would last after a full charge, whether it would meet the initial requirement of two months, and whether Clicker would explode or cause damage to users in extreme circumstances.

User actions related issues: Some users may not understand the content of Clicker's display and perform rude actions, which may cause damage to the device. This problem may be solved by issuing instruction manuals.

5.4 Future Outlook

The broader impacts of our enhanced I-clicker project are distinctly evident in its seamless integration with China's burgeoning online interactive education ecosystem. This project has successfully implemented a multi-front-end answer system that supports unhindered interaction through web interfaces, mobile applications, and the physical Clicker connected to the server side. This integration ensures that users across different platforms can engage interactively, enhancing the accessibility and flexibility of educational resources.

Particularly in China, where digital education tools are rapidly evolving, our project not only aligns with existing technologies but also introduces significant innovations such as advanced error correction algorithms that enhance the reliability and accuracy of the response system. Despite the presence of similar products globally, our system stands out in the Chinese educational sector due to its robustness and adaptability, potentially influencing widespread adoption across various educational institutions.

Economically, the project promises to reduce operational costs for educational entities by minimizing reliance on traditional resources and streamlining administrative processes. Environmentally, it contributes to sustainability by reducing paper use and encouraging digital interaction. Societally, the project fosters educational inclusivity and equity by providing tools that accommodate diverse student needs and learning environments.

Looking ahead, there are opportunities for further enhancements in backend processing speed and efficiency, as well as in the ergonomic and functional design of the physical Clicker. We are optimistic about the project's potential to integrate more deeply with university education systems, aiming to deliver superior educational services and support the development of a more interconnected and technologically advanced educational landscape in China.

References

[1] M. L. UK. "Iclicker." (2024), [Online]. Available: https://www.macmillanlearning.com/ed/uk/digital/iclicker (visited on 03/26/2024).

[2] *LM1117 800mA Low-Dropout Linear Regulator Datasheet*, Texas Instruments, 2023. [Online]. Available: https://www.ti.com/lit/ds/symlink/lm1117.pdf.

[3] E. Systems. ""ESP8266 Technical Reference"." (2020), [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp8266-technical_reference_en.pdf (visited on 03/21/2024).

[4] M. Research. ""2023 Jobs Blue Book Released, Monthly Earnings of College Graduates Revealed"." (2023), [Online]. Available: https://news.eol.cn/yaowen/202306/t20230612_2435553.shtml (visited on 06/12/2023).

[5] IEEE. ""IEEE Code of Ethics"." (2016), [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html (visited on 02/08/2020).

Appendix A Schedule

Table 5: Benlu Wang's Schedule

Date	Task
3.18-3.24	Plan and prepare the project, determine its scope and objectives, and set up team communication and collaboration tools.
3.24-3.31	Develop basic UI components and layout, integrate Jetpack components such as LiveData and ViewModel, and develop Room database related functions.
4.1-4.7	Optimize and test the Android front-end, focusing on enhancing performance and user experience.
4.7-4.14	Develop the iOS front-end, implement state management and routing for Flutter, and integrate necessary iOS-related functions such as app icon configuration.
4.15-4.21	Optimize and test the iOS front-end, write unit tests and integration tests to address potential issues and vulnerabilities in iOS applications.
4.22-4.28	Develop the web front-end, understand the requirements and functions of web applications, develop basic UI components and layouts, achieve state management and route navigation, and integrate React Router for page navigation.
4.29-5.5	Integrate, test, and deploy the project, integrating Android, iOS, and web front-ends, conducting overall performance testing and user experience testing, and resolving cross-platform compatibility issues.
5.6-5.12	Complete project documentation and finalize the final report.
5.13-5.19	Prepare slides and complete any remaining tasks.

Table 6: Zhenyu Zhang's Schedule

Date	Task
3.24-3.31	Analyze and understand the requirements of the Android application, and configure the Android Studio development environment.
4.1-4.7	Conduct code reviews and refactoring, and write unit tests and integration tests for the Android front-end.
4.7-4.14	Analyze system requirements and design the SQL database model. Configure the SQL database server to ensure reliability and security. Integrate the MongoDB database to ensure compatibility with the SQL database. Configure the RabbitMQ message queue system for internal communication and asynchronous processing.
4.15-4.21	Design and implement the device identification and binding mechanism, collect unique identifiers of devices. Develop logic to associate user account information with device identifiers and store it on the server. Develop login authentication logic to ensure that users can only log in using bound devices. Implement a secondary verification mechanism for device replacement and binding new devices.
4.22-4.28	Configure the system log function to record user login and device replacement activities. Set up a monitoring system to monitor the health and security of the system in real-time. Conduct security audits to check for potential vulnerabilities and security risks in the system. Fix vulnerabilities and security issues found, and conduct system re-testing and validation.
4.29-5.5	Integrate, test, and deploy the project, integrating Android, iOS, and web front ends. Conduct overall performance testing and user experience testing to resolve cross-platform compatibility issues.
5.6-5.12	Complete project documentation and finalize the final report.
5.13-5.19	Prepare slides and complete any remaining tasks.

Table 7: Luozhen Wang's Schedule

Date	Task
3.18-3.24	Determine the labor requirements for the project.
3.24-3.31	Purchase the required equipment from online sources.
4.1-4.7	Design the Printed Circuit Board (PCB) for the project.
4.7-4.14	Install the PCBs and other components, and conduct testing.
4.15-4.21	Develop a detailed design for the housing of the project.
4.22-4.28	Optimize the housing design and debug the internal circuitry.
4.29-5.5	Perform practical tests of the project in a classroom environment.
5.6-5.12	Demonstrate and optimize solutions to any identified problems.
5.13-5.19	Prepare presentation slides and complete any remaining tasks.

Table 8: Suhao Wang's Schedule

Date	Task
3.18-3.24	Implement the TCP connection of ESP8266 module.
3.24-3.31	Utilize the ESP8266 client to send information in JSON format through the HTTP protocol.
4.1-4.7	Test the button display using a breadboard and integrate it into the sending function.
4.7-4.14	Add the power subsystem and test the rectifier.
4.15-4.21	Integrate the clicker subsystem and power subsystem, complete the PCB design, obtain finished products, and perform a preliminary software system test.
4.22-4.28	Optimize the clicker design, identify innovative points, and write the final paper.
4.29-5.5	Test the clicker subsystem in the classroom setting.
5.6-5.12	Prepare a PowerPoint presentation for the demonstration.
5.13-5.19	Complete all remaining tasks.

Appendix B R & V

Table 9: Shell Hardness Test Table

Test Items	First Test	Second Test	Third Test
Whether it's broken	not broken	not broken	not broken

Table 10: PCB Connectivity/Functionality Test Table

Test Items	First Test	Second Test	Third Test
Send Signal (A, B, C, D)	✓	✓	✓
Delete Signal (A, B, C, D)	✓	✓	✓
Store Signal (A, B, C, D)	✓	✓	✓
Sleep Mode	✓	✓	✓
Wake Up Mode	✓	✓	✓

Table 11: battery Requirements and Verification

Requirement	Verification	Qualification
• Make sure the battery has at least 50 mA of charge.	• Connect a fully charged (4.2 V) lithium battery to the VDD and GND with suitable resistors so that the current is approximately 200 mA. • Discharge the battery at 200 mA for 15 minutes, using a voltmeter to ensure that the voltage is maintained above 3.7 V.	• Y