ECE 445

SENIOR DESIGN LABORATORY

DESIGN DOCUMENT

ZJUI Clickers for Undergraduate Version 2

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

March 20, 2024

Contents

1 Introduction

1.1 Problem

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 I-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 I-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 I-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 I-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 Solution

In response to the aforementioned practical challenge, our project endeavors to augment the system's capacity to cater to a larger participant base of approximately 100-150 individuals. Furthermore, our objective encompasses the facilitation of diverse front-end devices, encompassing mobile, PC, and Clicker interfaces. A crucial aspect of this enhancement involves expanding the receiver's radius to encompass the spatial dimensions of a typical classroom setting.

To realize these objectives, our team has formulated a comprehensive plan involving five core components: front-end development, back-end implementation, Clicker design, receiver design, and shell design. Moreover, the system has been architected as a closed-source solution, utilizing an internal Local Area Network (LAN) for signal transmission. This strategic measure serves to safeguard the system's integrity and mitigate the potential risks associated with external interference.

1.3 Visual Aid

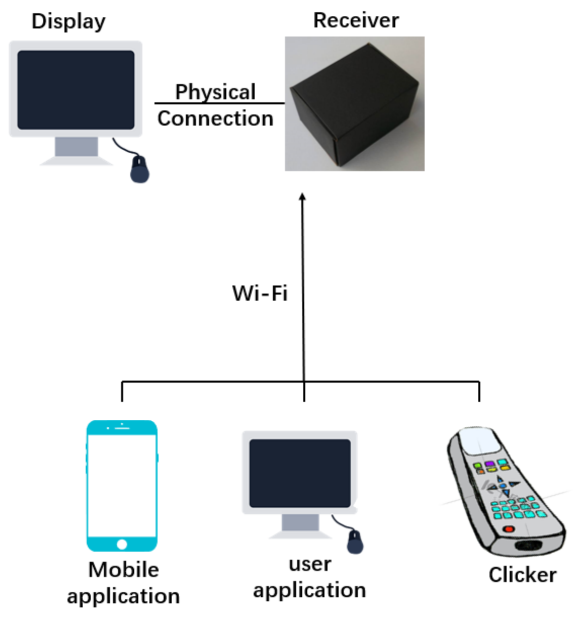


Figure 1: Visual Aid

1.4 High-level Requirement Lists

- The system has been designed to accommodate a substantial number of students, specifically supporting 100-150 individuals in a classroom environment to utilize the answer function efficiently. By considering the system's capacity to handle this volume of users, it aims to ensure smooth and uninterrupted functionality for all participants.
- The system has been optimized to ensure reliable signal transmission within a classroom environment of approximately 100 meters in size. Within this range, users can expect a strong and stable signal that enables them to effectively utilize the system's features, including the ability to answer questions. However, it's important to note that the system should not be designed to support long-distance signal transmission beyond the specified classroom size to avoid cheating.
- The delay from the user pressing a button on a mobile phone, web page or Clicker to the receiver receiving the signal does not exceed 2 s.

2 Design

2.1 Block Diagram

Our system architecture is delineated into three discrete subsystems: the power supply subsystem, the clicker subsystem, and the software subsystem. The power supply subsystem is tasked with furnishing power to the clicker subsystem, which serves as the interface for students to transmit signals via button presses. Meanwhile, the software subsystem encompasses essential components such as the back-end database and both the front-end classroom interface and student clicker interface. Additionally, external resources, including teachers' computers for software deployment and existing routers for local area network connectivity, are harnessed to augment system functionality. Through rigorous database optimization, we ensure that data processing latency remains within acceptable limits, not exceeding two seconds even under high volume conditions. Leveraging Wi-Fi technology, student check-ins are validated only within proximity to the teacher's device, ensuring accuracy and security. Our system architecture is designed to support simultaneous usage by up to 100 students, facilitated by local area network transmission and adherence to established protocols.

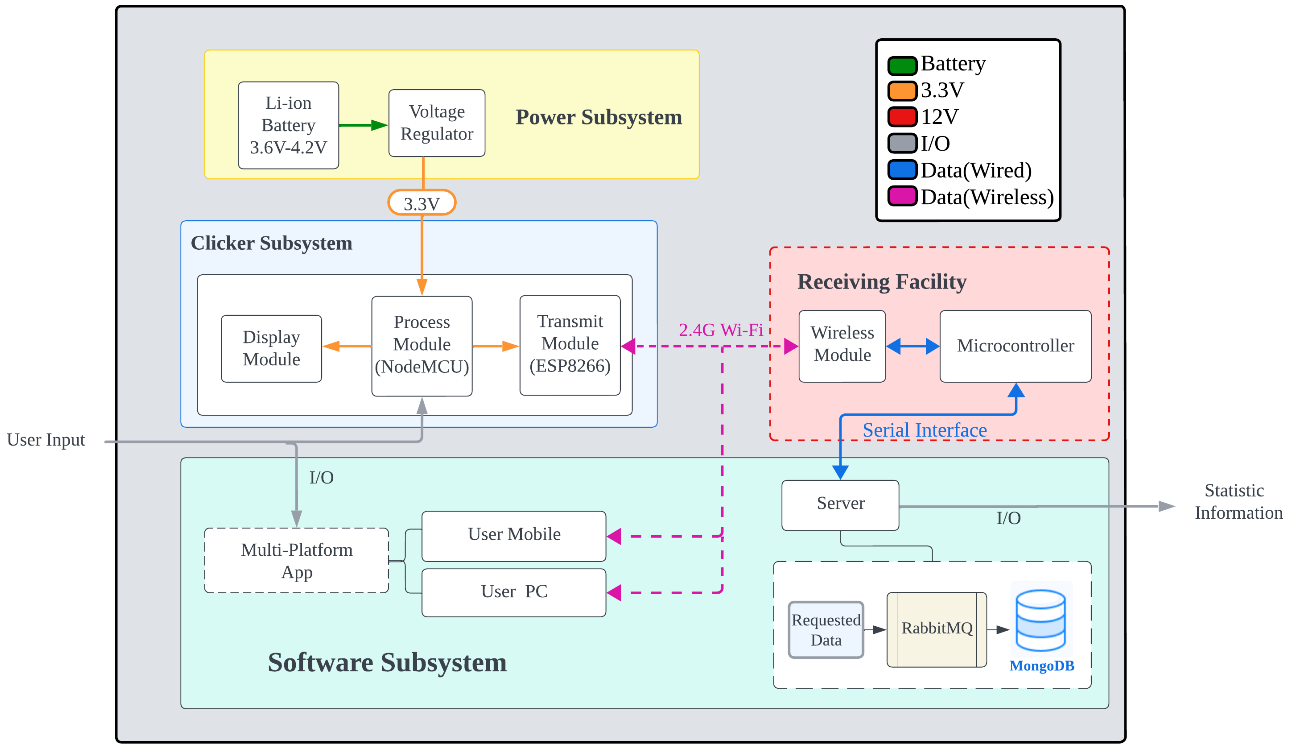


Figure 2: Block Diagram

2.2 Subsystem Overview

The power subsystem under consideration comprises several integral components, including a 220V DC voltage source, adapters, and a series of batteries with a voltage range of 3.3 V-5 V. These components play a crucial role in supplying power to the receiving device by employing a DC-to-DC Converter Module. The fundamental objective of this power supply system is to effectively support the clicker subsystem and the wireless receiving subsystem within the hardware segment.

The clicker subsystem encompasses a physical clicker equipped with Wi-Fi modules such as the ESP8266 and a micro-controller (MCU). Additionally, the physical clicker is equipped with auxiliary devices like display modules, display screens, and buttons. It can utilize the Wi-Fi module to transmit command information to the receiving subsystem through the local area network (LAN).

The software subsystem is aimed at developing a **cross-platform** classroom interaction application that supports Windows, Mac, iOS, and Android. It is designed to efficiently handle **high-concurrency** requests and includes a robust **identity verification** mechanism to prevent cheating practices such as proxy attendance.

The wireless receiving unit, consisting of a wireless module and a micro-controller, is basically outside the design scope of our project, and we will use existing equipment. Its primary function revolves around receiving signals from the wireless sending subsystem or the software subsystem and subsequently transferring the received data to the software subsystem via a physical connection.

2.3 Power Subsystem Requirement

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

We need a DC power supply to power our microcontroller and OLED. Considering the size of the microcontroller, OLED and energy consumption, we should choose a voltage of 3.3 V-5 V and a current of more than 100 mA. Excessive voltage can easily cause irreversible damage to the microcontroller and screen, including but not limited to burning.

Initially, dry batteries were considered as the power supply option. However, they did not meet the students' convenience requirements. As a result, rechargeable lithium batteries are being chosen instead.

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 [2].

With 50 button presses per day, each lasting 1 second, and a 200 mAh lithium battery, the battery can last for around 150 days after a full charge. Students will only need to recharge the battery once per semester.

2.3.1 Micro Lithium Battery

We chose to employ the use of a micro lithium battery 701224 (fluctuating from 3.7 V to 4.2 V) shown in Figure 3 as the power source, which will be supplied along with the clicker in.

Figure 3: battery

Requirement	Verification
• 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.

2.3.2 Regulator

In order to improve the utilization of battery energy consumption and prevent the chip from overheating, we will use the classic LM1117 regulator to control the voltage from 3.7 V-4.2 V to 3.3 V, at the maximum current of 200 mA.

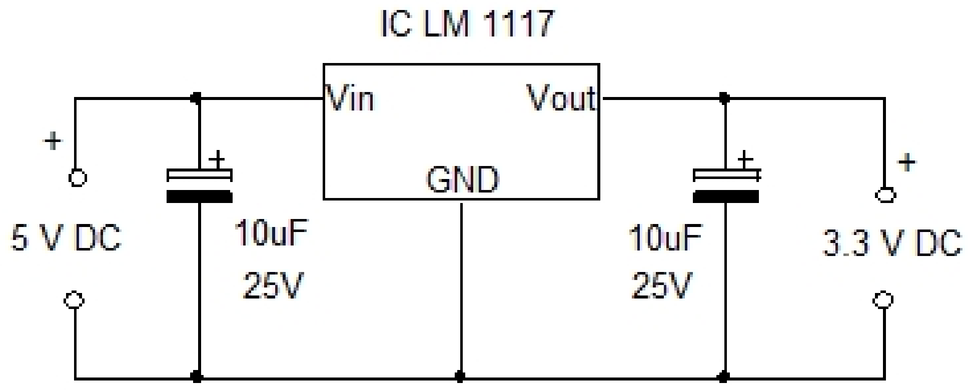


Figure 4: LM1117

Requirement	Verification
• The voltage can be stabilized to about 3.3 V, and the error is not more than 5%. • It can support 200 mA current.	• Get an adjustable power supply that can provide a voltage range of 3.7 V-4.2 V. • Connect the positive (+) of the power supply to the input pin of the voltage regulator and the negative (-) of the power supply to the ground (GND) pin of the voltage regulator. • Adjust the current from 0-200 mA to ensure that the output voltage is 3.3 V and the error of the regulator is within 5%.

2.4 Clicker Subsystem Requirement

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.

2.4.1 Transmit Module

On the clicker side, we 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.

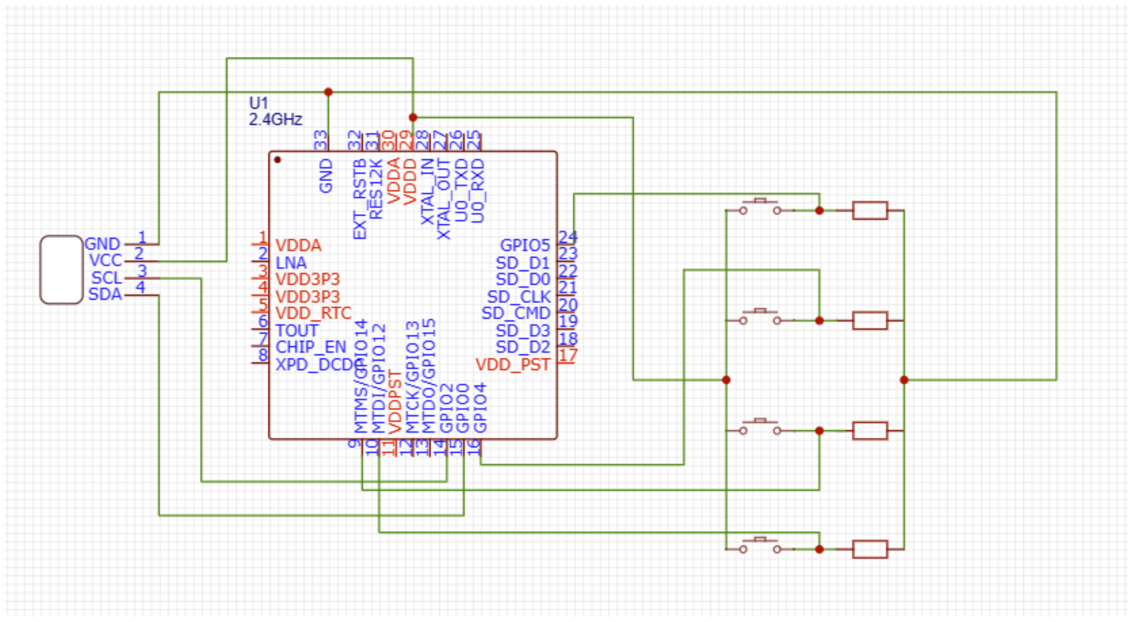


Figure 5: Schematic

Requirement	Verification
• Support for the IEEE 802.11b/g/n Wi-Fi protocol: The ESP8266 chip should be capable of connecting to wireless networks using the standard Wi-Fi protocols mentioned. • Wireless Communication and Data Transmission: The chip should enable wireless communication and data transmission through its built-in Wi-Fi module, allowing the microcontroller inside the clicker to connect with the teacher's receiver and exchange data. It should support transmission at 300 bps-4.5 Mbps. • The ESP8266 chip should operate efficiently under low-voltage and low-power conditions, making it suitable for battery-powered scenarios like the Clicker.	• Confirm Wi-Fi Protocol Support: Consult the ESP8266 chip's datasheet or documentation to ensure that it supports the IEEE 802.11b/g/n Wi-Fi protocol. • Test Wireless Connectivity: Use the ESP8266 chip in the clicker and attempt to connect it to a Wi-Fi network. Verify that it establishes a stable connection and is able to transmit data wirelessly at approximately 4.5 Mbps. • Measure Power Consumption: Monitor the power consumption of the ESP8266 chip while it is in operation. This can be done by measuring the current drawn by the chip with oscillators. We will combine this part with the battery verification.

2.4.2 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.

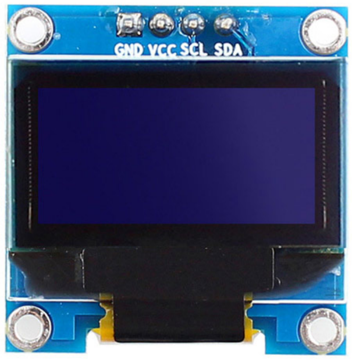


Figure 6: OLED

Requirement	**Verification**
• Communication Interface: The display module should support IIC (I2C) communication, as it is connected to the ESP8266 development board using this interface. • Power Consumption: The power consumption of the display module should be low, with a specified value of 0.04 W.	• To ascertain whether the OLED can be connected to the ESP8266, first consult the ESP8266's datasheet and the OLED module's specifications to confirm their supported communication interfaces and connection methods. Next, establish the correct connections and write test code to verify if the OLED module can successfully connect to the ESP8266. Adjust the connection method or code logic based on test results until a stable and reliable connection is confirmed. • Measure Power Consumption: Use a power meter or measure the current drawn by the display module while it is in operation. Verify that the power consumption matches the specified value of 0.04 W or is within an acceptable range.

2.4.3 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.

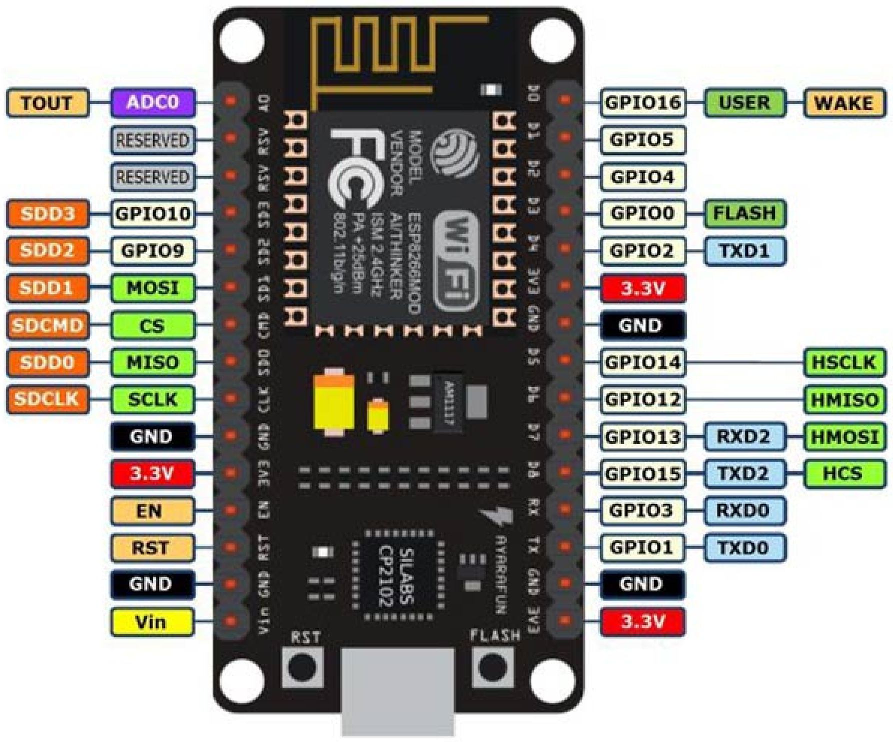


Figure 7: Nodemcu

Requirement	**Verification**
• Nodemcu must be able to receive input from the buttons. • It can both receive and transmit over UART at a speed of 4.5 Mbps.	• Button Input: Connect the button to the NodeMCU board according to its specifications. Verify that the board can detect the button press and process the corresponding signal. • UART communication: 1. Connect the NodeMCU board to a computer with USB-UART and open a serial terminal such as Putty. 2. Configure the serial terminal to use the appropriate COM port and set the baud rate to 4.5 Mbps. 3. Write a simple program to send data from the NodeMCU board to the serial terminal. Ensure all characters match those sent.

2.4.4 Shell Design

The exterior design of the clicker will employ 3D printing technology, leveraging the available resources in the school laboratory to expedite bulk production of the casing.

The design will be tailored to accommodate the form factors of internal components such as the microcontroller, battery, and button layout. Incorporating buckles within the shell will secure these components, safeguarding the internal circuitry from potential damage due to environmental factors, thereby ensuring operational resilience and longevity of the device.

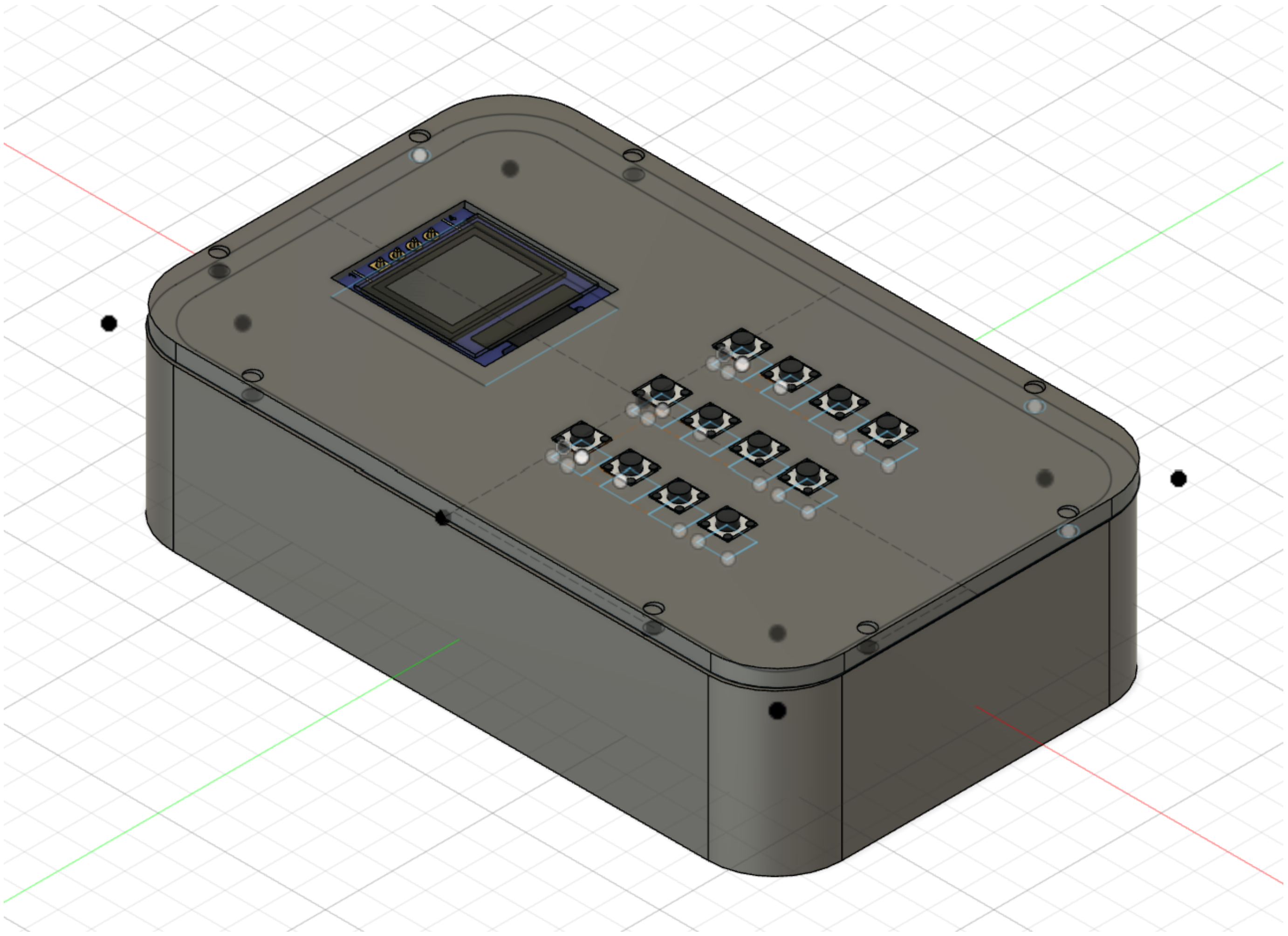


Figure 8: shell design

Requirement	Verification
• The design of the shell needs to be able to wrap the internal circuitry, protect the internal circuitry from changes in the external environment, and ensure the normal use of functions	• To test the iClicker's ability to withstand impact, drop the iClicker from a height of 60cm and observe whether it continues to function properly.

2.5 Software Subsystem Requirement

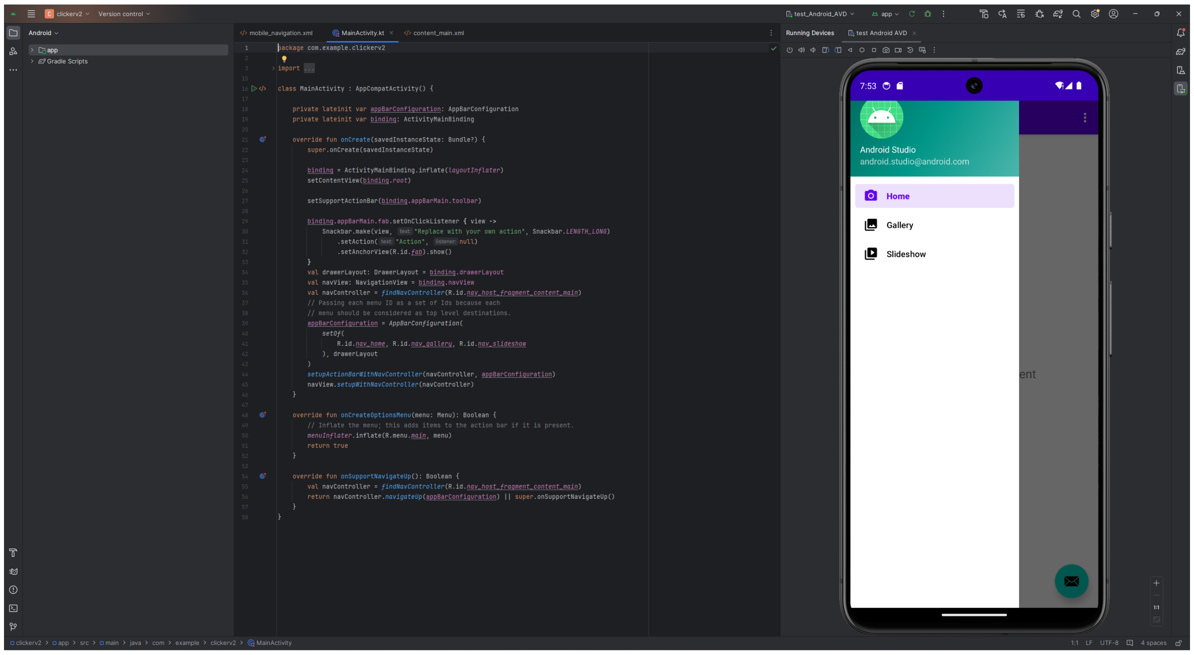
This subsystem ensures that the signal range requirements in the high level requirement can be met, and because of the optimized architecture used in the backend, we expect to make the system more responsive. This subsystem, on the one hand, acts as the client that sends the signal, on the other hand, 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.

2.5.1 Frontend Design

- **Student Android App**

Our Android front-end technology stack centers around the Kotlin programming language, chosen primarily for its excellent safety features, conciseness, and high interoperability with Java. Kotlin significantly reduces common programming errors, such as null pointer exceptions, through its null safety design. This not only enhances code safety but also greatly boosts developer productivity. Supported officially by Google, Kotlin ensures 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 like code autocompletion, performance analysis, and a visual layout editor to accelerate the development process. By incorporating the Jetpack library suite, including LiveData and ViewModel components, building responsive UIs becomes effortless, while Room database simplifies data operations, 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.



```kotlin
package com.example.clickerv2

import ...

class MainActivity : AppCompatActivity() {

    private lateinit var appBarConfiguration: AppBarConfiguration
    private lateinit var binding: ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        setSupportActionBar(binding.appBarMain.toolbar)

        binding.appBarMain.fab.setOnClickListener { view ->
            Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                .setAction("Action", null)
                .setAnchorView(R.id.fab).show()
        }
        val drawerLayout: DrawerLayout = binding.drawerLayout
        val navView: NavigationView = binding.navView
        val navController = findNavController(R.id.nav_host_fragment_content_main)
        // Passing each menu ID as a set of Ids because each
        // menu should be considered as top level destinations.
        appBarConfiguration = AppBarConfiguration(
            setOf(
                R.id.nav_home, R.id.nav_gallery, R.id.nav_slideshow
            ), drawerLayout
        )
        setupActionBarWithNavController(navController, appBarConfiguration)
        navView.setupWithNavController(navController)
    }

    override fun onCreateOptionsMenu(menu: Menu): Boolean {
        // Inflate the menu; this adds items to the action bar if it is present.
        menuInflater.inflate(R.menu.main, menu)
        return true
    }

    override fun onSupportNavigateUp(): Boolean {
        val navController = findNavController(R.id.nav_host_fragment_content_main)
        return navController.navigateUp(appBarConfiguration) || super.onSupportNavigateUp()
    }
}
```

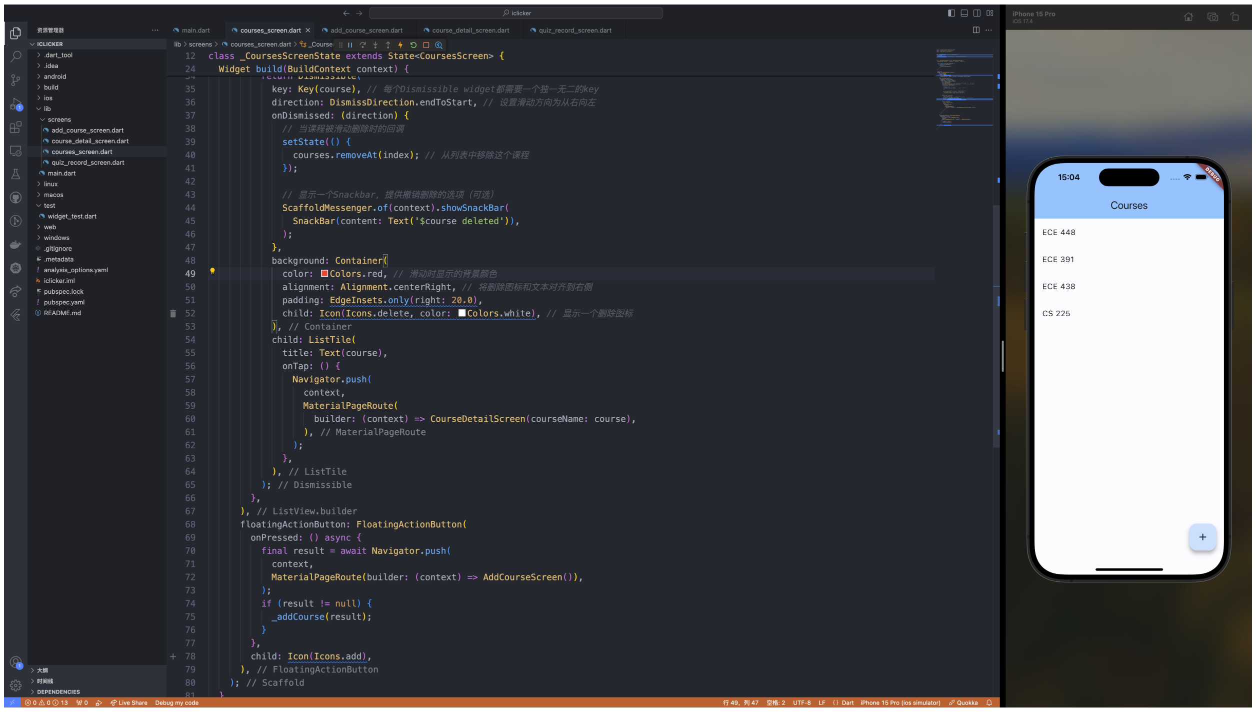
Figure 9: Android Design IDE

Requirement	**Verification**
• Develop an Android app using either the Flutter or Kotlin technology stack.	• The Android app must run smoothly on mainstream models and include a feature for answering questions during classes.

- **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.



```
class _CoursesScreenState extends State<CoursesScreen> {
  Widget build(BuildContext context) {
            key: Key(course), // 每个Dismissible widget都需要一个独一无二的key
            direction: DismissDirection.endToStart, // 设置滑动方向为从右向左
            onDismissed: (direction) {
              // 当课程被滑动删除时的回调
              setState(() {
                courses.removeAt(index); // 从列表中移除这个课程
              });

              // 显示一个Snackbar，提供撤销删除的选项（可选）
              ScaffoldMessenger.of(context).showSnackBar(
                SnackBar(content: Text('$course deleted')),
              );
            },
            background: Container(
              color: Colors.red, // 滑动时显示的背景颜色
              alignment: Alignment.centerRight, // 将删除图标和文本对齐到右侧
              padding: EdgeInsets.only(right: 20.0),
              child: Icon(Icons.delete, color: Colors.white), // 显示一个删除图标
            ), // Container
            child: ListTile(
              title: Text(course),
              onTap: () {
                Navigator.push(
                  context,
                  MaterialPageRoute(
                    builder: (context) => CourseDetailScreen(courseName: course),
                  ), // MaterialPageRoute
                );
              },
            ), // ListTile
          ); // Dismissible
        },
      ), // ListView.builder
      floatingActionButton: FloatingActionButton(
        onPressed: () async {
          final result = await Navigator.push(
            context,
            MaterialPageRoute(builder: (context) => AddCourseScreen()),
          );
          if (result != null) {
            _addCourse(result);
          }
        },
        child: Icon(Icons.add),
      ), // FloatingActionButton
    ); // Scaffold
  }
```

Figure 10: iOS Design IDE

Requirement	Verification
• Develop an iOS app using the Flutter technology stack.	• The iOS app must run smoothly on mainstream models and include a feature for answering questions during classes.

- **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.

Requirement	Verification
• Develop a web application using the React technology stack.	• Teachers can assign in-class questions through the web application and obtain statistics on students' responses.

2.5.2 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.

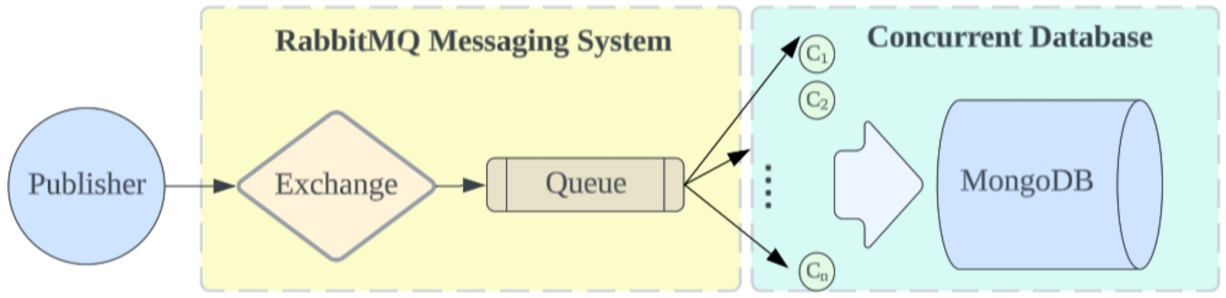


Figure 11: Backend System Diagram

Requirement	**Verification**
• Develop a backend system utilizing an SQL database for managing structured data, including user information and transaction records. Incorporate RabbitMQ for efficient and reliable communication and asynchronous processing under high load scenarios.	• The backend must reliably manage structured data using the SQL database and ensure efficient communication between services with RabbitMQ, maintaining system responsiveness and stability even during peak usage times.

2.5.3 Authentication module

The application includes a robust identity verification mechanism. When a user first logs in on a device, the system automatically recognizes and collects the device's unique identifiers, such as device ID, MAC address, or serial number. Then, during the device registration phase, the user's account information is associated with the unique identifier of the device used and is securely stored on the server. This means each student's account can only be bound to a specific device. When users attempt to log in, the system verifies whether the device identifier being used to access matches the one bound to the account. This mechanism ensures that each account can only be operated on its bound device, effectively preventing the possibility of logging in using someone else's device.

When binding a new device or changing a bound device, the system requires users to go through a secondary verification to confirm the legitimacy of the device binding or change request. Additionally, the system continuously monitors and records the process, including tracking every login attempt and device change activity. Through this continuous monitoring, the system can promptly detect and prevent fraudulent activities like proxy attendance, thereby maintaining the integrity and fairness of the system.

Requirement	**Verification**
• Implement an identity verification system that binds user accounts to their devices using unique identifiers (e.g., device ID, MAC address). Include a secondary verification process for device changes or new bindings.	• Ensure the system only allows account access from the bound device, with successful secondary verification for any device change. The system must track and monitor all login attempts and device changes to prevent fraudulent activities.

2.6 External facility

2.6.1 Wireless Receiving supply

This part ensures that it can meet the high level requirement for the signal range. The Unit can receive the signal stably within a certain range and convert the signal into data for transmission to the software system for processing.

Our receiver unit features 8 antennas and utilizes MTK's MT7986A CPU, which is a quad-core processor operating at 2.0 GHz. It is manufactured using a 12 nm process, ensuring efficient heat control. The memory configuration consists of 512 MB of DDR4 RAM and 128 MB of flash storage.

For the 2.4 GHz frequency band, our receiver supports up to 8 OFDMA users. On the other hand, the 5GHz frequency band can accommodate up to 16 OFDMA users. The

RF chip utilized for the 5 GHz band is the MT7976AN, which supports 4x4 MIMO. It enables a maximum data rate of 4804 Mbps with a 160 MHz bandwidth and 1024-QAM modulation. The amplifier chip for the 5 GHz band is the RTC66568.

As for the 2.4 GHz band, we employ four external FEM chips, specifically the RTC66266 model. This configuration allows for a maximum data rate of 1147 Mbps. To meet our requirement of supporting 100 mobile devices, our receiver can handle a load of up to 248 units. This is achieved through openwrt system code override.

2.7 Tolerance Analysis

2.7.1 MongoDB Analysis

MongoDB's robust tolerance capabilities, including features like replication and sharding, are further exemplified by its performance in handling operations efficiently. For instance, in a scenario where 10,000 search operations are conducted, MongoDB's performance is markedly superior, taking only 0.55 milliseconds per operation, compared to 4.47 milliseconds in a traditional SQL database[3]. This efficiency, alongside its flexible write concern levels and schema-less design, highlights MongoDB's suitability for applications requiring rapid data retrieval and high availability. However, it's essential to balance these benefits with effective data governance, especially given the potential for inconsistencies in its schema-less nature[4]. This combination of features and performance makes MongoDB an attractive choice for applications where speed and fault tolerance are critical.

2.7.2 RabbitMQ Analysis

In a teacher-student interactive Q&A application, employing RabbitMQ as the message queue greatly enhances the efficiency and stability of the database system. RabbitMQ's asynchronous message processing accelerates application responsiveness and decouples various system components, such as the teacher and student interfaces, reducing their interdependencies. Additionally, RabbitMQ excels in load balancing, effectively preventing system overloads during peak periods, and offers remarkable fault tolerance by safely storing messages in case of processing failures[5].

However, the introduction of RabbitMQ is not without drawbacks. Firstly, it increases system complexity, necessitating additional maintenance and management, especially for applications aimed at being user-friendly. Secondly, RabbitMQ can become a performance bottleneck in scenarios of high load or heavy message traffic, particularly when message size increases or the number of clients surges. For instance, test data shows that when the number of clients increases from 1 to 20, the average message processing time in RabbitMQ increases from 2 milliseconds to 16 milliseconds. Furthermore, this processing time tends to escalate more rapidly as the number of clients continues to grow[5].

This aspect is particularly crucial in our designed teacher-student interactive Q&A application. Given the potential need to support over 100 student users simultaneously, the increased latency due to a higher number of clients could lead to unexpected delays, failing to meet the minimal response time requirement of 500 milliseconds and adversely affecting user experience. Therefore, ensuring the effective operation of RabbitMQ under high-load conditions through appropriate configuration optimizations and performance testing becomes critically important.

2.7.3 Wi-Fi Analysis

The IEEE802.11 standard operates within a freely available frequency band, such as 2.4GHz, which is shared by numerous wireless technologies, including Bluetooth, 3G, and cordless telephony. This shared frequency environment gives rise to the possibility of signal overlap and RF interference when wireless devices operating within the same band transmit signals.

CSMA/CA (Carrier-sense multiple access with collision avoidance) is a network multiple access method utilized in computer networks. It employs carrier perception to ensure that nodes initiate transmission only when they sense that the channel is free, thereby avoiding collisions. When transmission occurs, nodes transmit their packet data in its entirety, following the successful carrier sensing process[6].

In our project, the Wi-Fi receiving device, utilizing CSMA/CA collision detection, encountered competition from other routers within the classroom environment. It is worth noting that in areas such as schools, where routers are extensively deployed, only one device on the same channel can receive or send a signal simultaneously. However, in cases where the load capacity of the receiving device is insufficient, the distribution of multiple receiving devices becomes necessary.

In the context of CSMA/CA, the Wi-Fi signal operates on three non-overlapping channels. If the total number of routers exceeds three, it becomes imperative to consider the potential implications of interference and signal-to-noise ratio (SNR). Specifically, the utilization of CSMA/CA may lead to a lower overall throughput[7]. This slower response may pose challenges in meeting the high-level requirement of achieving 100-150 users.

In our project, we employed a research paper's model to estimate the receiver throughput[8]. The model, operating under the assumption of an ideal error-free channel, incorporates various aspects crucial to wireless communication systems. These include arbitrary packet arrival rates, channel access delays due to collisions and other station transmissions, resetting of transmission retry counters, combined media access methods, arbitrary packet length assignment, and the existence of beacon frames. To analyze the system, the interface queue at each site is modeled using an M/G/1 queue, which provides a suitable framework for inspecting our project. This model's comprehensive consideration of relevant parameters and its utilization of the M/G/1 queueing approach contribute to its applicability and effectiveness in estimating receiver throughput accurately.

The probability q, representing the likelihood of an empty message interface queue, can be approximated using the following expression:

$$q = 1 - \rho \tag{1}$$

Under the assumption that the model of the arriving signal follows a Poisson distribution with a rate of λ, we express the probability as follows:

$$p_i = e^{-\lambda T} \tag{2}$$

where the probability p_i is the probability that queue is empty after successful transmission and followed back-off. T is the average time spent by the system at the upper row of the Markov chain.

Additionally, the collision probability (p) in a CSMA/CA network can be approximated using the formula:

$$prob = 1 - (1 - p)^N \tag{3}$$

where p represents the probability of collision, and N denotes the number of contending nodes. These mathematical models enable the evaluation of CSMA/CA-based networks, considering factors such as collision avoidance and channel contention.

A refined formula, derived by combining the three aforementioned formulas and incorporating additional relevant equations, was obtained, omitting the need for explicitly including the transmission probability λ:

$$S = C\frac{(1 - p^l)p_s E[P]}{E[\Psi] + p_s T_s + (1 - p_s)T_c + p^l l(E[B] + T_c)} \tag{4}$$

$E[P]$ is the average IP packet length; C is the beacon coefficient; T_s is the average time that the channel is busy due to successful transmission, and T_c is the average time that the channel is busy due to collision; p^l is the probability that the packet will be dropped because the maximum number of retry attempts has been reached; $E[B]$ is the average number of slots required for the back counter to reach zero; $E[\psi]$ is the expectation of a random value; p_s is the probability that the system will successfully transmit[8].

In the figure below, exponential time periods with a rate of lambda=25 were configured, accompanied by a fixed payload size of 1000 bytes for access. The channel's bit rates were set at 1148 Mbps, and the resulting throughput was examined for verification purposes. Through the calculations performed, it was determined that exceeding 100 clients would lead to sub-optimal optimization of the receiver's throughput. However, despite this limitation, the application requirements could still be met due to the relatively short length of the transmitted signal.

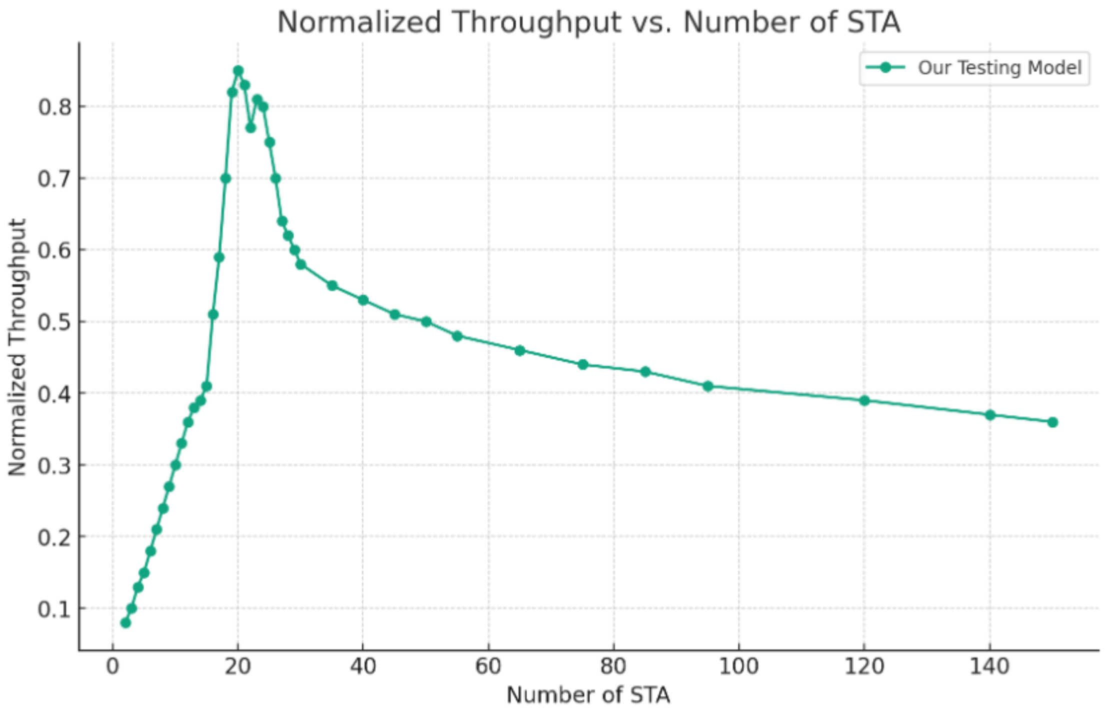


Figure 12: Throughput Vs Number of STA

3 Cost & Schedule

3.1 Cost

3.1.1 Labor Cost

According to a report on employment data released by Chinese Education Online[9],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 1: 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

3.1.2 Parts Cost

Table 2: Parts Cost

Description	Quantity	Manufacturer/Vendor	Cost/Unit	Total Cost
Mi AX6000 Router	1	Mi/ JD	449	449
ESP8266 NodeMcu	3	Xintai Microelectronics/Taobao	15	45
0.96 OLED screen	3	Xintai Microelectronics/Taobao	5	15
Breadboard 3x7cm	3	Jicheng Technology/Taobao	1.25	3.75
Dupont wire	100	Jicheng Technology/Taobao	0.05	5
3D Printing Consumables	1	Qipang Technology/Taobao	49	49
Membrane button 3*4	3	Chenyi Technology/Taobao	10	30
Total				596.75

3.2 Schedule

See the Appendix A.

4 Ethics & Safety

4.1 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[10].

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[10].

4.2 Safety

In terms of product safety, our central processing modules use 12 V-19 V DC. Although the voltage of all equipment is lower than the safe voltage for human beings, we always pay attention to the risk of short-circuits and leakage that may exist inside the electrical equipment, and comply with the safety rules in order to prevent the user from suffering from unintentional electrical injuries.

In order to prevent all accidents that may occur during the operation of the central processing module, as well as to prevent injuries to people, we will design the enclosure to isolate the module from the external environment, and in order to prevent the enclosure from harming people, we will avoid the appearance of sharp bends in order to maximise the protection of the user.

References

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

[2] 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).

[3] B. Dipina Damodaran, S. Salim, and S. M. Vargese, “Performance evaluation of mysql and mongodb databases,”

[4] S. H. Aboutorabi[a], M. Rezapour, M. Moradi, and N. Ghadiri, “Performance evaluation of sql and mongodb databases for big e-commerce data,” in *2015 international symposium on computer science and software engineering (CSSE)*, IEEE, 2015, pp. 1–7.

[5] V. M. Ionescu, “The analysis of the performance of rabbitmq and activemq,” in *2015 14th RoEduNet International Conference-Networking in Education and Research (RoEduNet NER)*, IEEE, 2015, pp. 132–137.

[6] W. contributors. “”CSMA/CA”.” (2024), [Online]. Available: https://en.wikipedia.org/wiki/Carrier-sense_multiple_access_with_collision_avoidance (visited on 03/07/2024).

[7] J. E. W. Vijay K. Garg, *Wireless and Personal Communications*. Prentice Hall, 1996.

[8] J. Sudarev, L. White, and S. Perreau, “Performance analysis of 802.11 csma/ca for infrastructure networks under finite load conditions,” in *2005 14th IEEE Workshop on Local & Metropolitan Area Networks*, 2005, 6 pp.–6. DOI: 10.1109/LANMAN.2005.1541535.

[9] 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).

[10] 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 3: 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 4: 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 5: 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 6: 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.