**“An Integrated IoT and Web-Based Reading Engagement and Comprehension Tracking System for the Library Hub of Tambo, Lipa City”**

This study proposes a Smart Reading Engagement and Comprehension Tracking System designed specifically for the Library Hub of Tambo, Lipa City. The system integrates IoT-based QR scanning hardware with a centralized web platform for tracking reading engagement and comprehension among elementary students. Through this integration, the system enables real-time monitoring of reading activity, automated quiz assessments per book, and performance-based rewards, all managed through a librarian and teacher dashboard. The primary goal is to improve literacy development by encouraging consistent reading habits and providing actionable data for educators.

**General Objectives**

**1.** Develop a web-based platform to monitor and assess students' reading comprehension and engagement.

**2.** Integrate IoT-based QR technology to automate book tracking and quiz initiation.

**3.** Provide a teacher and librarian dashboard for real-time tracking of student reading progress and comprehension.

**4.** Implement a reward system based on reading milestones (e.g., 10, 25, 50 books read).

**5.** Enhance literacy outcomes through data-driven monitoring and student motivation.

## Integration Analysis

The system integrates hardware, software, and communication components into a unified solution. IoT hardware such as the ESP32-CAM captures book scan data, while the web application processes and stores information within a MySQL database. The integration ensures real-time synchronization between student activity and educator dashboards.

**Components:**

* **IoT Device (ESP32-CAM):** Scans book tags and sends data via Wi-Fi.
* **Web Application (PHP, HTML, CSS, JavaScript):** Hosts the quiz system and teacher dashboard.
* **Database (MySQL/XAMPP):** Stores student profiles, reading logs, and quiz results.
* **Local Server/Cloud Hosting:** Handles web requests and data synchronization.

**Integration Patterns:**

* **Book Scanning:** When a student scans a book’s QR code using the IoT device (ESP32-CAM), the system sends the *Book ID* and *Student ID* to the server via Wi-Fi. The web server validates this data and retrieves the appropriate quiz from the database.
* **Quiz Synchronization:** After the student finishes the quiz, their score and completion time are automatically recorded in the database. This ensures real-time data synchronization between the IoT device, web app, and educator dashboards.
* **Reward Tracking:** The system automatically updates the number of books read by each student. Once certain milestones are reached (e.g., 10, 25, 50 books), the reward module flags them as eligible for a corresponding incentive.
* **Dashboard Updates:** Teacher and librarian dashboards continuously fetch and display updated student performance data, allowing real-time monitoring of comprehension scores and reading activity.

**Data Flow:**

1. Student logs in → system authenticates user credentials via the database.
2. Student scans a book’s QR using the IoT device → the Book ID and Student ID are transmitted to the server via Wi-Fi.
3. Server receives the data → validates entries → retrieves the corresponding quiz from the database.
4. Student answers the quiz → submits responses through the web interface.
5. Web application computes the quiz score → updates the database with performance results and number of books read.
6. Dashboard automatically refreshes → teacher/librarian views updated reading logs and comprehension data in real time.
7. System checks total books read → triggers the reward module if milestones (10, 25, or 50 books) are reached.
8. Librarian can generate reports → summarize student engagement and reading comprehension progress.

The proposed system integrates IoT-based scanning, web-based quizzes, and automated reporting into one PHP–MySQL platform. By combining these components, it automates manual library logging, enhances student motivation, and provides teachers and librarians with real-time insights.

**Agile Methodology**

The development of the Smart Reading Engagement and Comprehension Tracking System follows the **Agile Scrum methodology**. This approach ensures iterative development, continuous feedback from the client (the Library Hub of Tambo, Lipa City), and adaptability to changing requirements during the development process.   
  
Requirements

* **Roles:**
  + **Product Owner:** Represents the Library Hub of Tambo, Lipa City, and provides requirements, insights, and feedback.
  + **Scrum Master / Team Lead:** Ensures the team follows Scrum practices and maintains proper coordination.
  + **Developers:** implement the system’s web and IoT integration using PHP, MySQL, HTML, CSS, JavaScript, and Arduino IDE.
  + **Tester/QA:** Validates system functionality, user experience, and IoT-to-web integration reliability.
* **Sprints (2 weeks each):**
  + **Sprint 1:** Requirements gathering, database design (ERD), initial UI mockups.
  + **Sprint 2:** IoT device configuration and communication with the web server.
  + **Sprint 3:** Student login module, QR book scanning, and quiz display functionality.
  + **Sprint 4:** Reward tracking module and teacher/librarian dashboard for real-time monitoring.
  + **Sprint 5:** Testing, bug fixing, documentation, and final integration presentation.
  + **Sprint 6:** Pre-order functionality, notifications, testing, and final deployment.

**Definition of Done (DoD):** A feature is considered complete when it is fully functional, tested across IoT and web components, integrated into the main system, properly documented, and approved by the Product Owner (the client representative from the Library Hub of Tambo, Lipa City).

## User and System Requirements (Hardware and Software)

1. **Functional Requirements:** 
   * Allows teachers, librarians, and students to log in and manage their accounts.
   * The IoT device should be able to scan book QR codes and send data to the system.
   * The system must retrieve and display quizzes related to the scanned book.
   * The system should evaluate student answers and store the results in the database.
   * Teachers should be able to view reports and student progress from the dashboard.
2. **Non-Functional Requirements:** 
   * **Usability:** The interface must be simple, intuitive, and accessible for elementary-level users.
   * **Performance:** The system should respond within 3 seconds for quiz loading and data submission.
   * **Reliability:** Must maintain data integrity and handle multiple users simultaneously.
   * **Security:** Uses secure authentication and encryption for student and teacher data.
   * **Scalability:** System architecture allows for future expansion (more books, students, and quiz modules).

## System Model / Design Diagram

The system architecture follows a 3-tier structure composed of Presentation, Application, and Data layers. Below is a summary of the system model:

1. Presentation Layer: Web-based interface for students, teachers, and librarians.  
2. Application Layer: Business logic managing quiz generation, score computation, and reward system.  
3. Data Layer: Centralized database storing user information, book records, and performance analytics.

Refer to the attached System Architecture Diagram illustrating the integration between IoT hardware, server, and database components.

## Hardware and Software Specifications

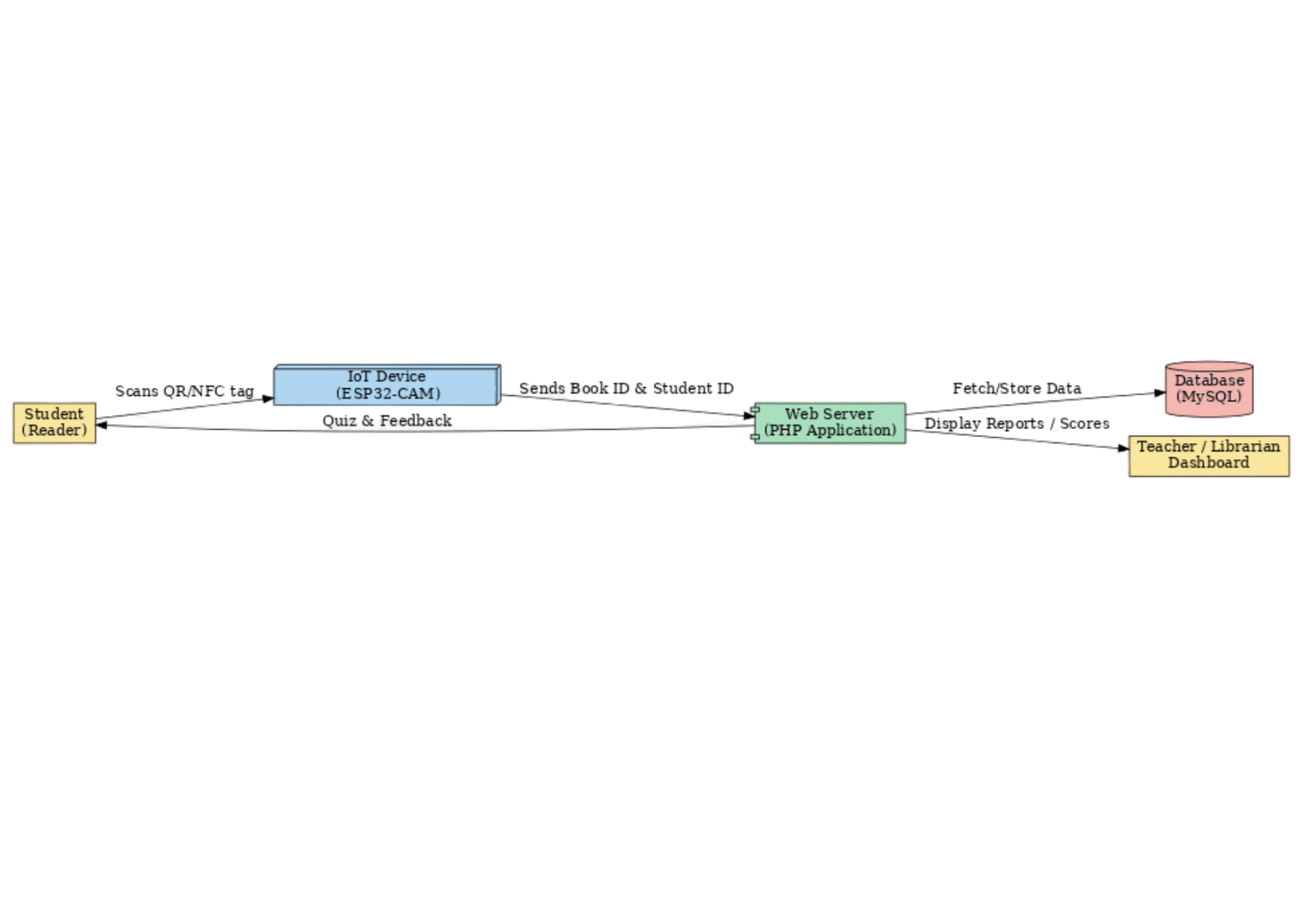
Table 1.1 Hardware Components

|  |  |
| --- | --- |
| Hardware Component | Specification / Description |
| **ESP32-CAM** | Embedded IoT camera module with Wi-Fi capability; used to scan QR codes and transmit data to the local server. |
| **FTDI Programmer (USB-to-TTL Programmer)** | Used for flashing/uploading firmware to the ESP32-CAM during setup. |
| **Jumper Wires (Male-Female)** | Connects ESP32-CAM to FTDI module and power source. |
| **5V Power Supply / USB Cable (Phone Charger or Power Bank)** | Provides stable 5V DC power to the ESP32-CAM; can use either a standard USB phone charger or a power bank. |
| **Wi-Fi Router (2.4GHz)** | Provides local network connectivity between IoT device and server. |
| **PC (Local XAMPP Server)** | Minimum 4GB RAM, 256GB storage; runs Apache, MySQL, and PHP environment for the web application. |
| **Breadboard** | Used during testing or prototyping for temporary connections. |
| **Protective Case/Enclosure** | Enclosure for ESP32-CAM; ensures camera stability and protection during use. |
| **USB Voltmeter / Ammeter** | Used to monitor voltage and current output to ensure stable and safe 5V power delivery to the IoT devices. |
| **QR Code Sticker** | Used to bridge the physical and digital worlds by connecting a physical object to digital information. |

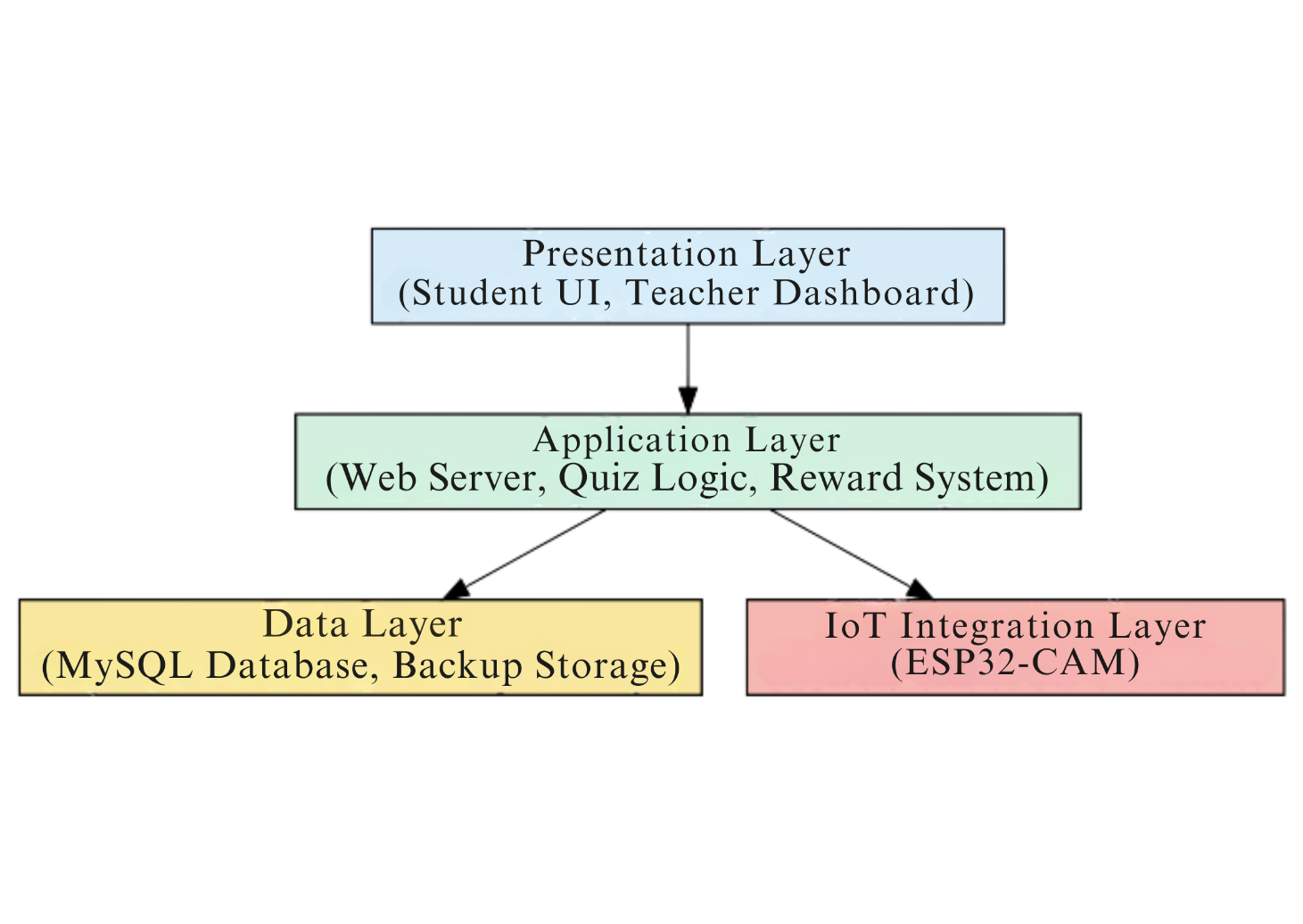
Table 1.2 Software Components

|  |  |
| --- | --- |
| **Software Component** | **Version / Platform** |
| Windows OS | Any recent version (Windows 10 / 11) |
| XAMPP (Apache, PHP, MySQL) | Latest stable version (3.3.0+) for local server hosting |
| Arduino IDE | Version 2.0 or later; used to program the ESP32-CAM IoT device |
| PHP / JavaScript / HTML / CSS/ Python | Web development stack for the system interface and logic |
| MySQL Database | Latest stable release for data storage and retrieval |
| VS Code | Version 1.8+; used as IDE for web system coding |
| GitHub | Online repository platform for version control and collaboration |
| Web Browser (Chrome / Edge) | For testing and accessing the web application interface |

**System model and diagram** The System Flow Diagram below illustrates how students, IoT devices, and the web application interact within the system. It shows the main process flows from scanning a book to recording comprehension scores and updating dashboards.

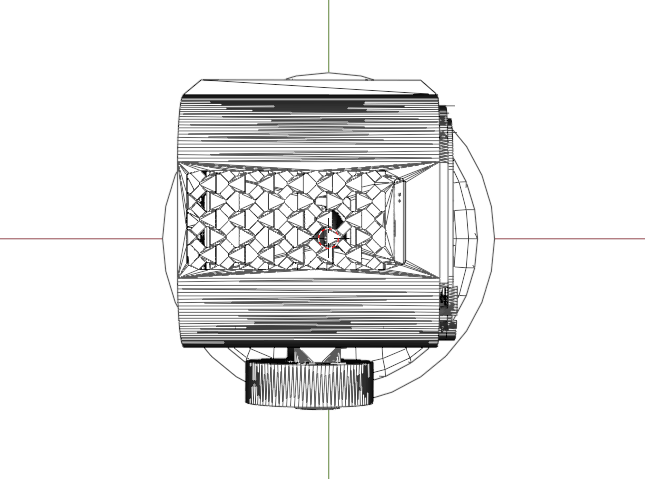
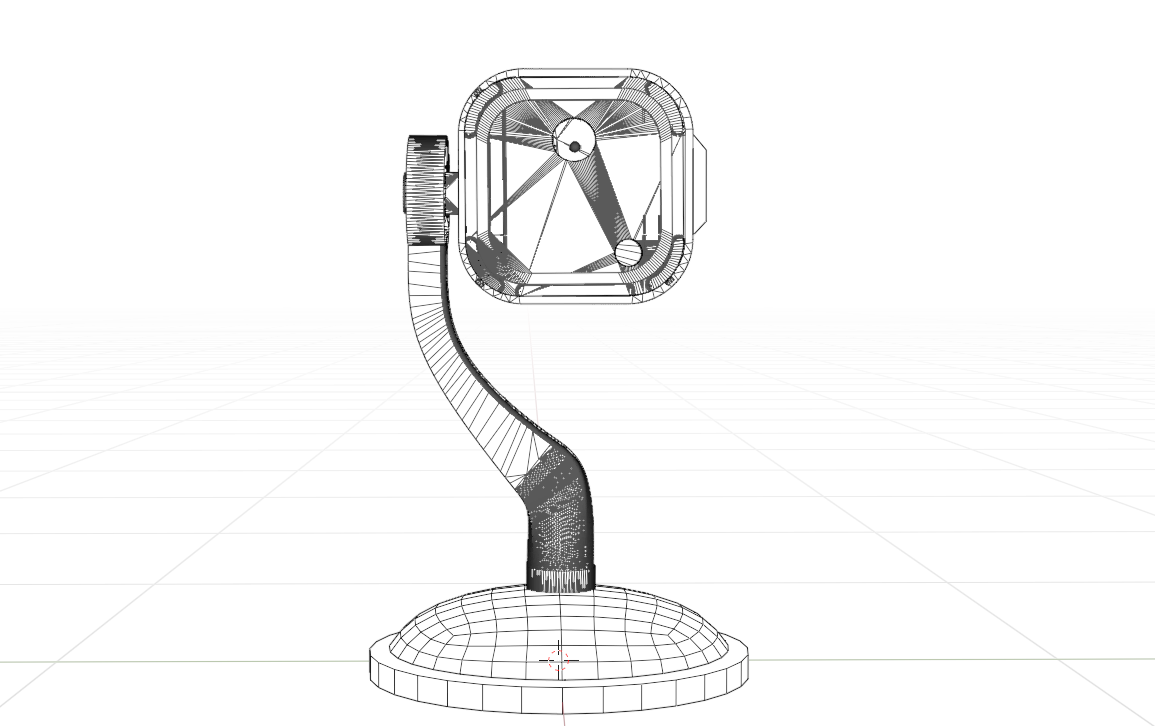


**Architectural Design**



• Presentation Layer: Manages the user interface for students and teachers.  
• Application Layer: Handles business logic, quiz generation, and reward computation.  
• Data Layer: Manages the storage of student data, book information, and reading logs.  
• IoT Integration Layer: Facilitates the connection between hardware devices and the web server.

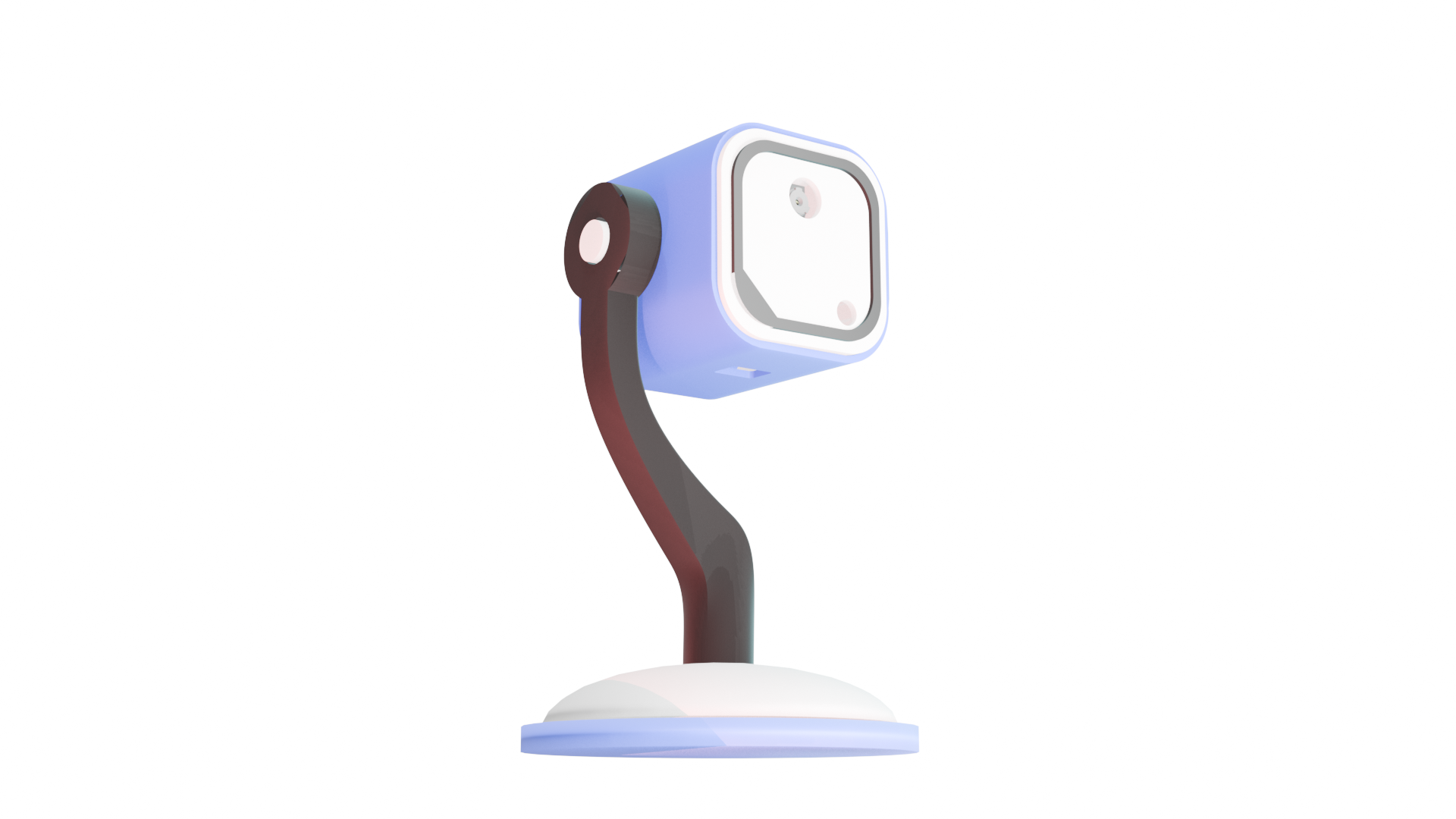
**2D Model**



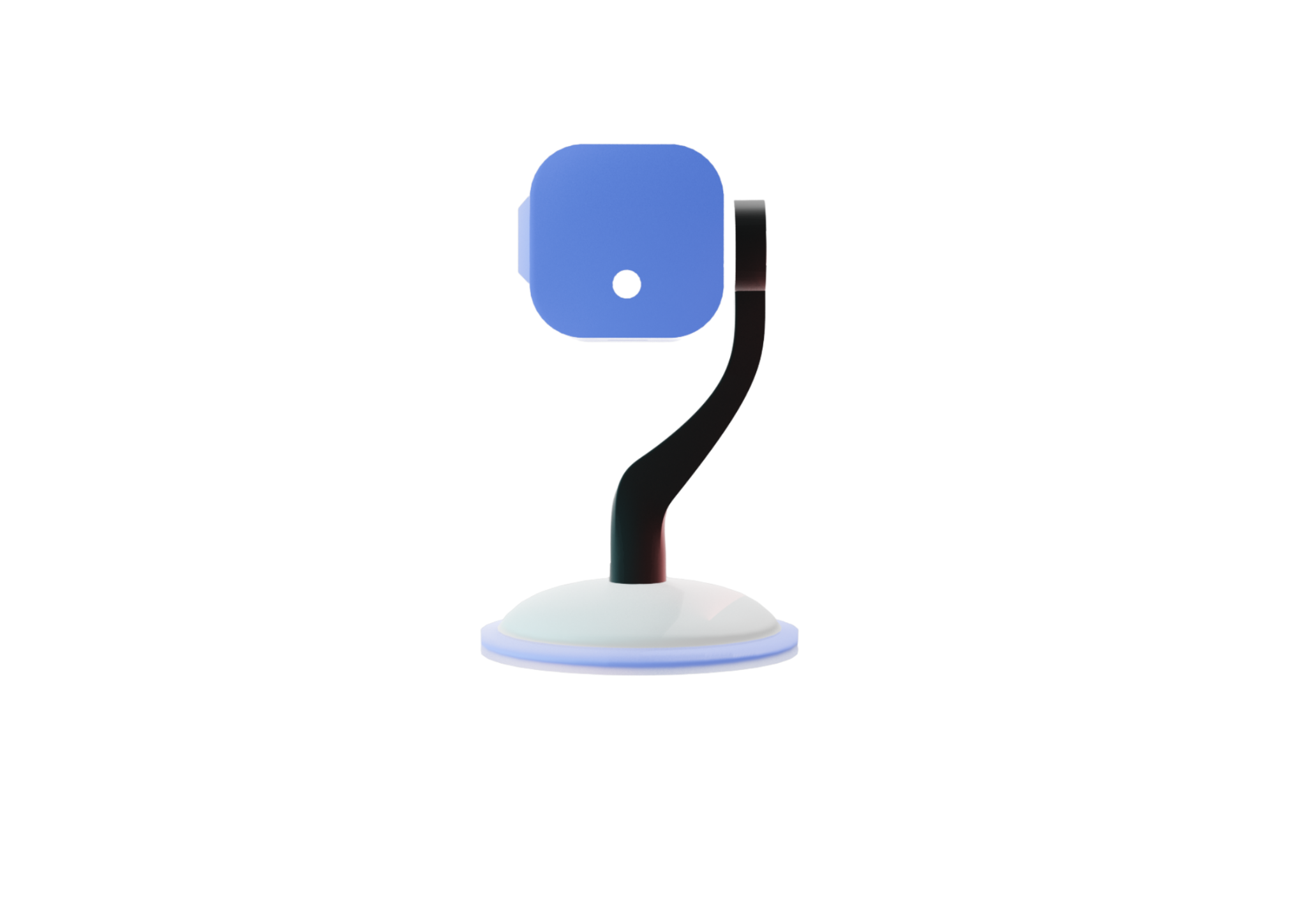
**3D Model Perspectives**

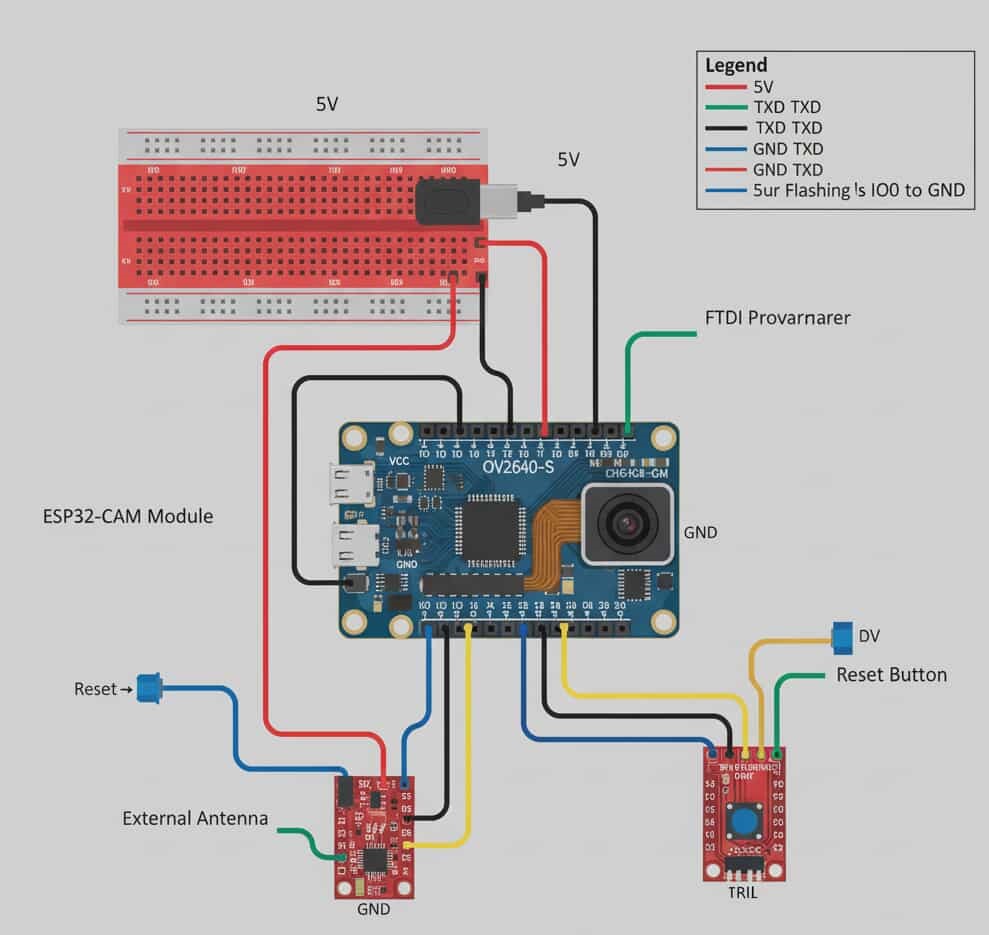
**Top View**

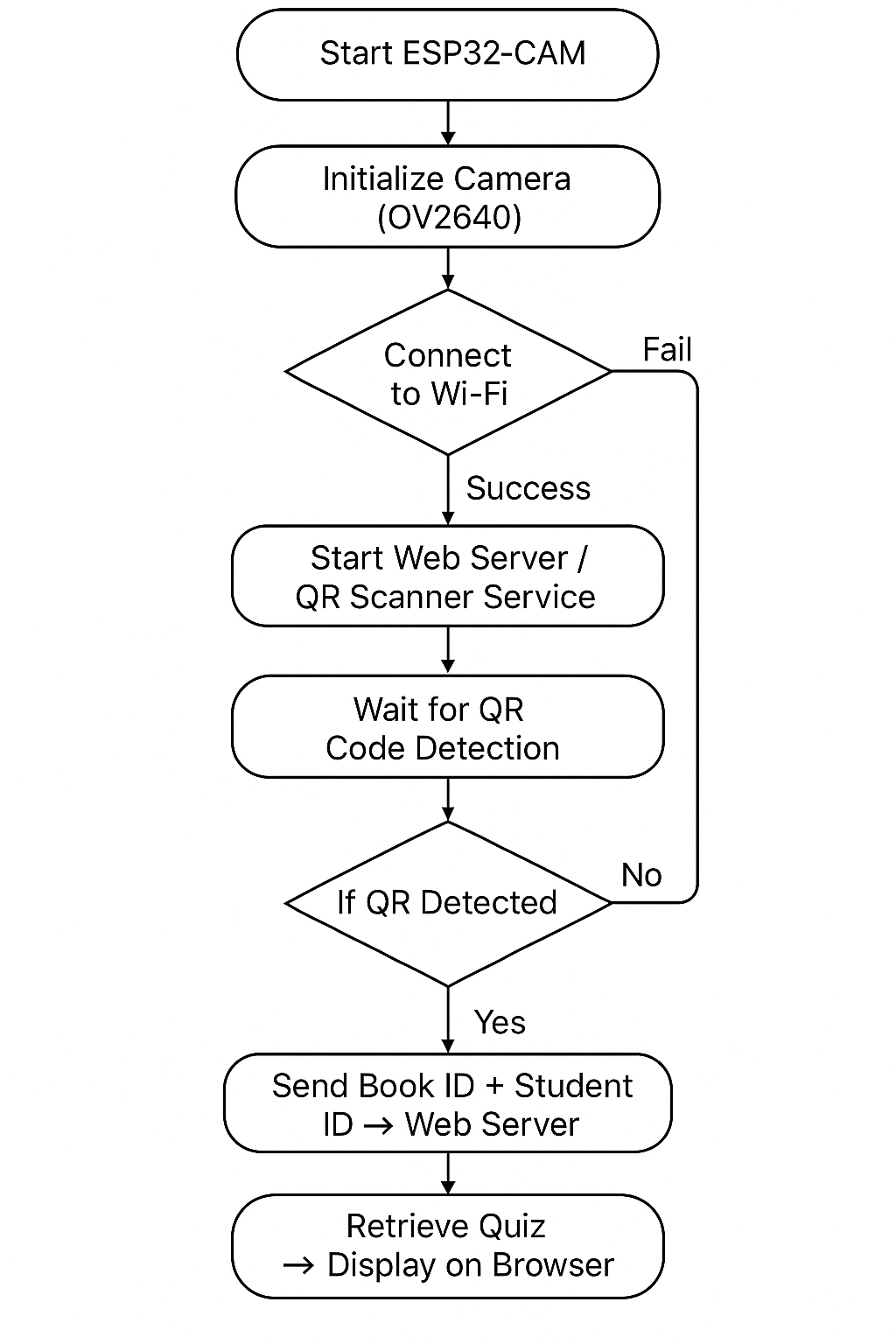


**Side View**



**Front View Bottom View Rear View**

**Wiring Diagram**

**Wiring Diagram (Flow)**