

Development and Experimental Evaluation of a Smartphone-Based Mixed Reality Gun Simulator

Pengembangan dan Evaluasi Eksperimental Simulator Menembak berbasis Mixed Reality Ponsel

¹Victor Sinaga*)

¹Simulation System Engineer, PT. AVS, Bandung 40131, Indonesia

*) corresponding email: victor.sinaga021@gmail.com, victor@victorsinaga.com

Abstract

Mixed Reality (MR) has gained increasing attention in simulation and training because it enables immersive interaction between virtual and physical environments. However, many existing MR systems rely on expensive head-mounted displays and dedicated tracking hardware, limiting their accessibility for educational and proof-of-concept research. This study aims to develop and experimentally evaluate a low-cost smartphone-based MR gun simulator operating in Virtual Reality (VR) and Augmented Reality (AR) modes. Its novelty lies in integrating consumer-grade hardware into a unified MR platform that supports immersive interaction at an affordable cost. The prototype integrates a smartphone, a custom handheld controller, embedded sensing, computer vision, and a real-time simulation environment. Experimental evaluation examined crosshair mapping accuracy, response latency, system frame rate performance, and AR object detection reliability. Results show that the prototype achieved acceptable interaction accuracy, responsive real-time performance, stable rendering in both VR and AR modes, and reliable object detection throughout the evaluation. These findings demonstrate that affordable smartphone-based MR systems can provide sufficient performance for proof-of-concept research and educational applications, establishing a practical foundation for developing immersive training and human-computer interaction systems.

Keywords: *mixed reality, virtual reality, augmented reality, smartphone-based simulation, human-computer interaction.*

Abstrak

Teknologi Mixed Reality (MR) semakin mendapat perhatian dalam bidang simulasi dan pelatihan karena memungkinkan interaksi imersif antara lingkungan virtual dan fisik. Namun, banyak sistem MR masih bergantung pada perangkat head-mounted display dan perangkat keras pelacakan khusus yang mahal, sehingga membatasi aksesibilitasnya untuk penelitian pendidikan dan pengembangan konsep awal. Penelitian ini bertujuan mengembangkan dan mengevaluasi secara eksperimental simulator senjata berbasis smartphone berbiaya rendah yang beroperasi dalam mode Virtual Reality (VR) dan Augmented Reality (AR). Kebaruannya terletak pada integrasi perangkat keras konsumen ke dalam platform MR terpadu yang mendukung interaksi imersif dengan biaya terjangkau. Prototipe ini mengintegrasikan smartphone, pengendali genggam khusus, sensor tertanam, computer vision, dan lingkungan simulasi waktu nyata. Evaluasi eksperimental mencakup akurasi pemetaan crosshair, latensi respons, performa frame rate sistem, dan keandalan deteksi objek pada mode AR. Hasil pengujian menunjukkan bahwa prototipe memiliki akurasi interaksi yang memadai, respons waktu nyata yang baik, rendering yang stabil pada mode VR dan AR, serta deteksi objek yang andal selama evaluasi. Temuan ini menunjukkan bahwa sistem MR berbasis smartphone berbiaya rendah memiliki performa yang memadai untuk penelitian proof-of-concept dan aplikasi pendidikan, sekaligus menjadi landasan praktis bagi pengembangan sistem pelatihan imersif dan interaksi manusia-komputer.

Kata kunci: *mixed reality, virtual reality, augmented reality, simulasi berbasis smartphone, human-computer interaction.*

Paper accepted July 31st, 2026 – paper revised October 12th, 2026 – approved October 13th, 2026

This paper is open access with [CC BY-SA](#) license.



1 Introduction

Mixed Reality (MR) technologies have gained increasing attention due to their capability to integrate virtual and physical environments into interactive experiences. By combining the immersive characteristics of Virtual Reality (VR) and the contextual awareness of Augmented Reality (AR), MR enables users to interact with digital content in a more natural and spatial manner. These capabilities have encouraged the adoption of immersive technologies in various domains, including education, medical training, industrial maintenance, cultural heritage, and collaborative environments [1],[2],[3],[4]. In particular, simulation-based MR systems provide safe, repeatable, and controllable environments that allow users to practice complex tasks without requiring direct access to expensive equipment or hazardous situations [4],[5].

The application of VR and AR for training and simulation has been extensively explored. Previous studies have demonstrated that immersive environments can support procedural learning, rehabilitation, and technical skill development by providing realistic interaction scenarios [3],[5],[6]. Industrial applications have also utilized MR systems to improve assembly and maintenance training by combining real-world objects with digital guidance and visualization [4]. These developments demonstrate the potential of immersive simulation as an alternative approach for improving human-computer interaction and skill-based learning.

However, developing effective MR systems remains challenging, particularly in achieving accurate, responsive, and accessible interaction methods. Existing high-performance MR platforms often depend on dedicated hardware such as commercial head-mounted displays, depth cameras, and specialized tracking devices to achieve reliable spatial interaction [7],[8],[9]. Although these technologies provide accurate tracking capabilities, their cost and hardware complexity may limit accessibility for early-stage research, educational purposes, and rapid prototyping. Therefore, developing affordable MR platforms using consumer-grade devices remains an important research direction.

Recent studies have investigated alternative interaction approaches based on computer vision, hand tracking, and embedded sensing to reduce dependence on specialized hardware. Hand tracking systems using Leap Motion, depth cameras, and vision-based approaches have demonstrated promising results for virtual interaction, rehabilitation, and training applications [10], [11], [12]. Furthermore, machine learning-based vision frameworks have enabled markerless gesture recognition using conventional cameras, providing opportunities for lightweight and portable interaction systems [13],[14],[15].

Smartphone-based VR and MR systems provide a promising approach to address accessibility challenges because modern smartphones already integrate cameras, inertial sensors, displays, and computational capabilities within a widely available platform. Previous research has demonstrated the feasibility of using smartphones as interaction devices and immersive displays in VR applications [16],[17],[18].

This research proposes the development and experimental evaluation of a smartphone-based Mixed Reality gun simulator as a proof-of-concept platform for immersive interaction research. Unlike previous works that primarily evaluate specific interaction techniques or application usability, this study focuses on the integration of a complete low-cost MR system consisting of physical interaction, embedded sensing, communication, virtual simulation, and real-time feedback. The proposed system supports both Virtual Reality and Augmented Reality modes and is evaluated through quantitative parameters including interaction accuracy, response latency, rendering performance, and object detection reliability.

Rather than relying on a fully self-contained smartphone application, the proposed architecture adopts a hybrid approach in which the smartphone works alongside a desktop computer. Although smartphones integrate displays, cameras, and inertial sensors in a compact and accessible device, a gun simulation involving interactive virtual environments, real-time rendering, and computer-vision processing can impose considerable computational demands. Distributing these tasks between the smartphone and a desktop computer provides a practical way to develop the prototype without requiring all processing to be performed on the mobile device alone.

The objective of this work is not to develop an optimal MR headset or to compete with commercial MR headsets and fully self-contained mobile VR/AR systems, whose architectures and computational resources differ from the proposed platform. Instead, the primary success criterion is to demonstrate that the proposed hybrid architecture can operate successfully and reproduce the essential interaction functions of VR and AR in a gun simulation. Performance measurements provide additional baseline assessments of system responsiveness and aiming behavior, informed by considerations relevant to cybersickness in immersive environments [19],[20],[21]. These measurements are intended to characterize the prototype's operational feasibility rather than establish superiority over other MR platforms.

2 Method

2.1 Operational Flowchart

The operational flow of the proposed smartphone-based Mixed Reality system is illustrated in Figure 1 for the Virtual Reality (VR) mode and Figure 2 for the Augmented Reality (AR) mode. Although both modes utilize the same hardware platform, they employ different data processing pipelines according to their interaction mechanisms.

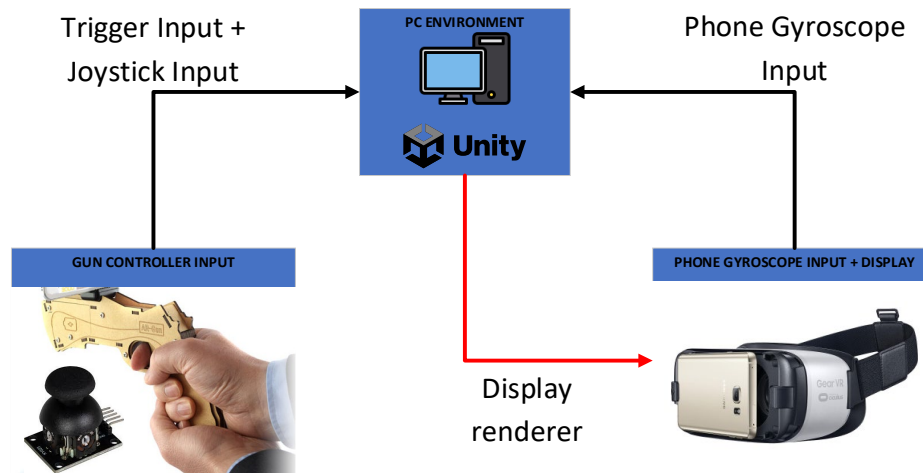


Figure 1. VR mode operational flowchart of the system

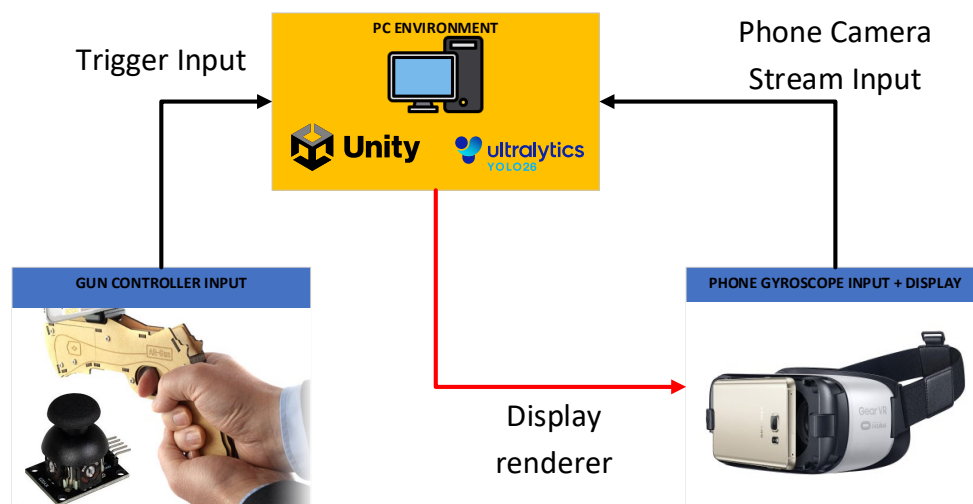


Figure 2. AR mode operational flowchart of the system

In the VR mode, the system begins by initializing the Unity Desktop application, the Android application, and the custom gun controller. During operation, movement commands generated from the controller joystick are transmitted to the Unity Desktop application to control player translation, while the trigger button is used to perform shooting interactions. Simultaneously, the smartphone continuously measures its gyroscope orientation and transmits the data to Unity, allowing the virtual camera orientation to follow the user's head movement in real time. After processing all interaction inputs, Unity renders the virtual environment and streams the resulting display back to the smartphone, completing one operational cycle before repeating the process continuously.

In the AR mode, the operational sequence is extended by incorporating computer vision processing. After system initialization, the smartphone continuously streams images of the surrounding environment to a Python-based object detection module. The detected object coordinates are then transmitted to Unity, where corresponding virtual objects are generated and aligned with the detected positions. Unlike the VR mode, player translation and camera orientation through the phone gyro sensor reading are not required because users interact directly within the physical environment; therefore, the controller is primarily used for shooting interactions and crosshair recalibration. Unity combines the camera visualization with the generated virtual objects and streams the augmented scene back to the smartphone, enabling continuous real-time interaction throughout the experiment.

2.2 System Architecture

The proposed Mixed Reality platform was designed with two operational architectures corresponding to its Virtual Reality (VR) and Augmented Reality (AR) modes. The system adopts a hybrid smartphone–desktop architecture rather than a fully self-contained mobile VR/AR system, allowing computationally demanding tasks to be handled by the desktop computer while the smartphone provides display, sensing, and camera capabilities as required by each mode. This design was chosen to facilitate the development of an interactive gun simulation without requiring all processing to be performed on the smartphone alone. Although both modes share the same

smartphone, custom handheld controller, embedded hardware, and Unity-based simulation environment, they employ different data processing pipelines to accommodate their respective interaction paradigms. The VR mode focuses on fully virtual interaction by utilizing smartphone inertial sensing for camera orientation and controller inputs for navigation and shooting within a virtual environment. In contrast, the AR mode integrates real-world camera streaming and computer vision-based object detection to overlay virtual objects onto the physical environment while maintaining interaction through the same handheld controller.

2.2.1 Virtual Reality (VR) Mode

The architecture of the proposed VR mode is illustrated in Figure 3. The system consists of four main modules: a custom handheld controller, an Android smartphone, a Unity Desktop application, and a wireless communication network connecting all components. Each module performs a dedicated function to support real-time immersive interaction.

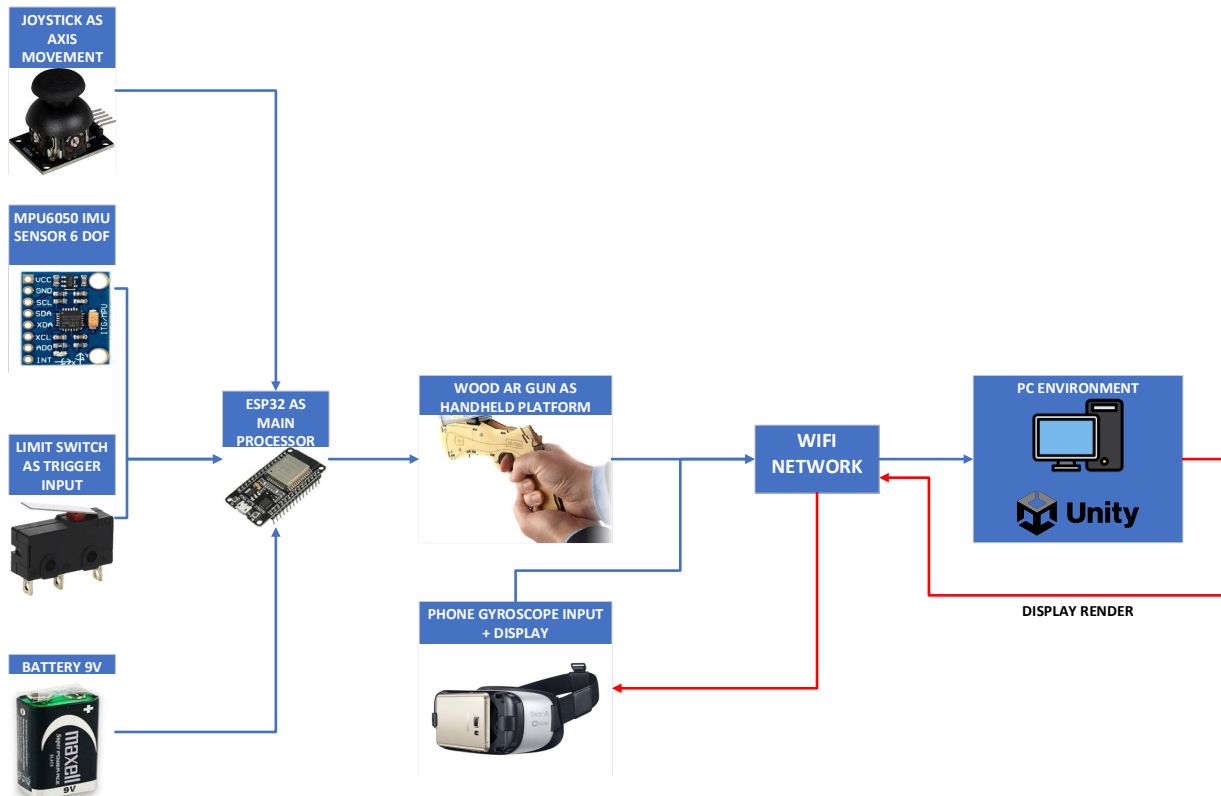


Figure 3 VR mode architecture of the system

The custom handheld controller serves as the primary interaction device. It incorporates a joystick for player translation, a trigger button for shooting actions, and a joystick button for additional interaction functions. The controller transmits these input signals to the Unity Desktop application through an ESP32 microcontroller, allowing the virtual player to navigate and interact with the simulation environment.

The smartphone functions as both the display device and the orientation sensing module. The built-in gyroscope continuously measures the user's head orientation and transmits the rotational data to the Unity Desktop application. The received orientation is applied to the virtual camera, enabling the displayed viewpoint to follow the user's head movement in real time. After updating the virtual environment based on both controller input and gyroscope data, Unity renders the scene and streams the resulting display back to the smartphone.

2.2.2 Augmented Reality (AR) Mode

The architecture of the proposed AR mode is illustrated in Figure 4. Similar to the VR architecture, the system consists of a custom handheld controller, an Android smartphone, a Unity Desktop application, and a wireless communication network. In addition, the AR mode incorporates a computer vision module that enables interaction with the real-world environment through object detection.

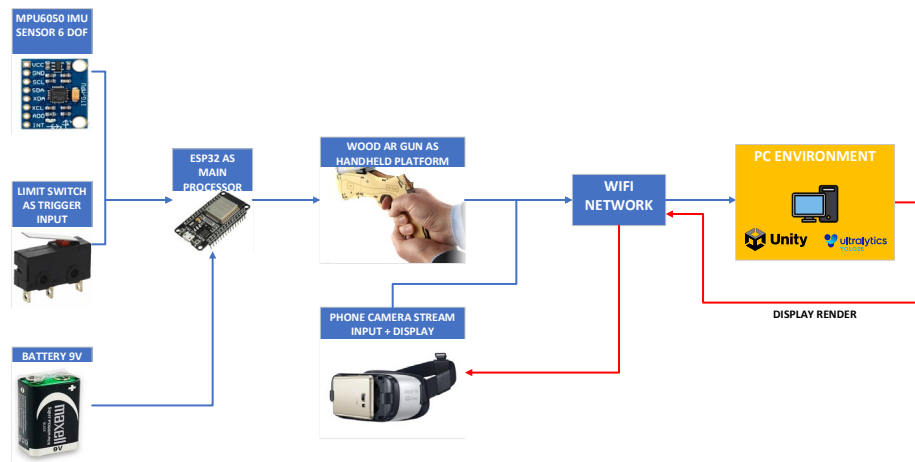


Figure 4. AR mode architecture of the system

In this mode, the smartphone serves as the primary image acquisition device by continuously streaming camera frames of the surrounding environment. The captured images are transmitted to a Python-based object detection module using YOLO26, where detected object coordinates are extracted and forwarded to the Unity Desktop application. Unity then generates corresponding virtual objects based on the detection results and overlays them onto the streamed camera view, forming the augmented environment displayed to the user.

Unlike the VR mode, player translation and camera orientation is not required because users navigate naturally within the physical environment. Consequently, the handheld controller is primarily used for shooting interactions and crosshair recalibration to compensate for accumulated inertial sensor drift. After integrating the camera stream, object detection results, and user interactions, Unity renders the augmented scene and streams the final display back to the smartphone.

2.3 Component and Materials

The list of hardware components used in this study, along with their technical specification, is presented in Table 1.

Table 1. Technical specification of each component

Component	Specification	Function
ESP32 DEVKIT	ESP-WROOM-32U, Bluetooth Classic + BLE, Operating voltage: 3.3V, Input USB: 5V, GPIO: 38 pin	Main Gun Controller processor, receiving all input from connected sensor, and sending it to Unity app in PC
Joystick KY-023	Dual-axis XY Joystick Module, Input Power: 5V, Dimensions: Size: 34x26x32mm, 2 potentiometers for 2 axes	X and Y axis movement input for translation movement in VR mode, Joystick button also provided bonus input if needed
Sensor IMU MPU6050	MPU-6050, Input Power: 3-5V, 6 DOF	Main sensor to control the movement of crosshair to be translated into 2D plane (display)
Augmented Reality Wood Gun-Shaped	Simple handheld made from wood to play VR-AR games, modified for this experiment	Main handheld platform to resemble gun controller and playing all other electronics components
9V Battery	9.0V power, 400 mAh to 600 mAh, 45 grams	Main power source for all components in the system
Limit Switch	Input Power: 3-5V, NO, non-latching	Simple button input for Gun Controller trigger
Phone: Infinix Note 30(May 2023 Released)	6.78 Inch IPS LCD, 1080x2460 Pixel Res, 120Hz Refresh Rate, MediaTek Helio G99 (6nm), Octa-core, GPU Mali-G57 MC2. 8GB RAM, 256GB internal memory, 64MP main camera, 5000 mAh battery with 45W wired fast charging	Main display of the Mobile MR Headset. Gyro sensor source for the VR system, and capturer of IRL environment through back camera for AR system
PC	CPU: Intel Core i7-13700KF (13th Gen) GPU: NVIDIA GeForce RTX 4070 Ti, 12 GB VRAMRAM: 32 GB	Main hub and process for the entire system

2.4 System Design

The proposed Mixed Reality platform was implemented by integrating four primary components: a desktop computer, an Android smartphone, a custom handheld controller, and a wireless communication network. Each component performs a dedicated role in supporting real-time interaction while maintaining a low-cost and accessible hardware configuration suitable for proof-of-concept research.

The desktop computer serves as the main processing unit responsible for executing the Unity simulation environment, handling interaction logic, and rendering the virtual scene. The Android smartphone functions as the user Mixed Reality main display while simultaneously providing sensing and image acquisition capabilities depending on the selected operating mode. The custom handheld controller enables user interaction through movement and shooting commands, whereas the wireless communication network facilitates real-time data exchange among all system components. Together, these components provide sufficient functionality to implement both Virtual Reality and Augmented Reality operation without requiring specialized immersive hardware.

2.4.1 Hardware Setup and Wiring

The hardware implementation of the proposed system is illustrated in Figure 5, which presents the wiring configuration of the custom handheld controller together with the supporting devices used in both VR and AR operating modes. The complete hardware setup consists of a desktop computer, an Android smartphone, a custom handheld controller, and a wireless communication network.

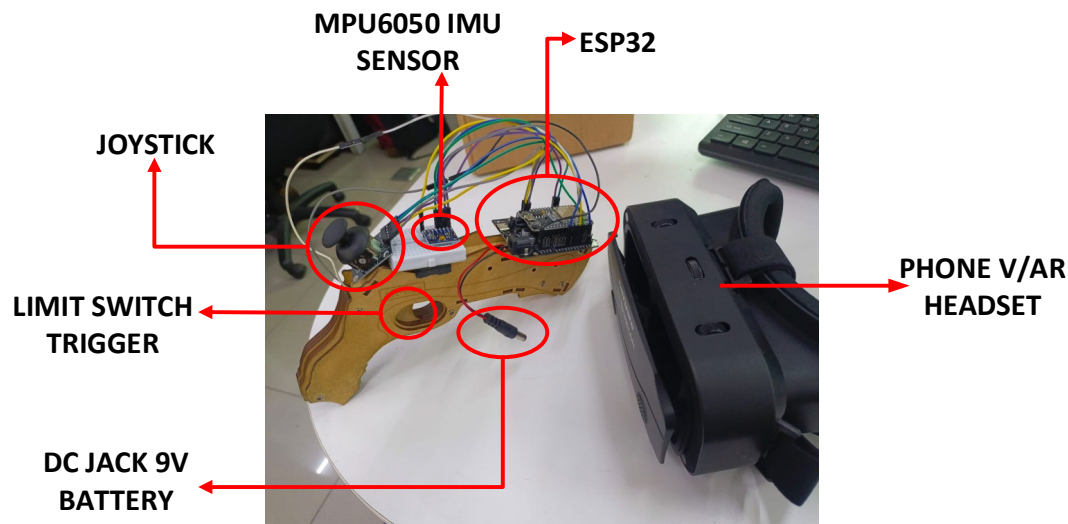


Figure 5. Gun controller wiring and V/AR Headset

The custom handheld controller was developed using an ESP32 microcontroller as the primary processing and communication unit. User input is provided through a two-axis analog joystick for movement control, an integrated joystick push button for additional interaction, and a trigger button for shooting commands. An MPU6050 inertial measurement unit (IMU) is incorporated to measure the controller orientation, while a 9 V battery supplies power for standalone operation.

The Android smartphone serves as the primary user display for both operating modes. During operation, the rendered output generated by the Unity Desktop application is streamed to the smartphone, allowing the device to function as the visualization unit. The smartphone is subsequently mounted inside a mobile VR headset to provide an immersive viewing experience for both VR and AR operation. In addition to visualization, the smartphone also functions as the wireless access point, establishing a dedicated 2.4 GHz Wi-Fi network that connects the ESP32, desktop computer, and mobile application. This configuration simplifies deployment by eliminating dependence on external wireless infrastructure while ensuring compatibility with the ESP32 wireless module.

The desktop computer executes the Unity simulation environment and performs the primary computational tasks required for rendering and interaction processing. Together, these hardware components establish a compact and low-cost experimental platform capable of supporting both Virtual Reality and Augmented Reality interaction using consumer-grade devices.

2.4.2 Data Workflow

The data workflow describes how information is exchanged among the hardware and software modules during system operation. Although both operating modes utilize the same hardware platform, the sequence of data processing differs because the VR mode relies on inertial sensing for immersive visualization, whereas the AR mode additionally incorporates computer vision for object detection and virtual object generation.

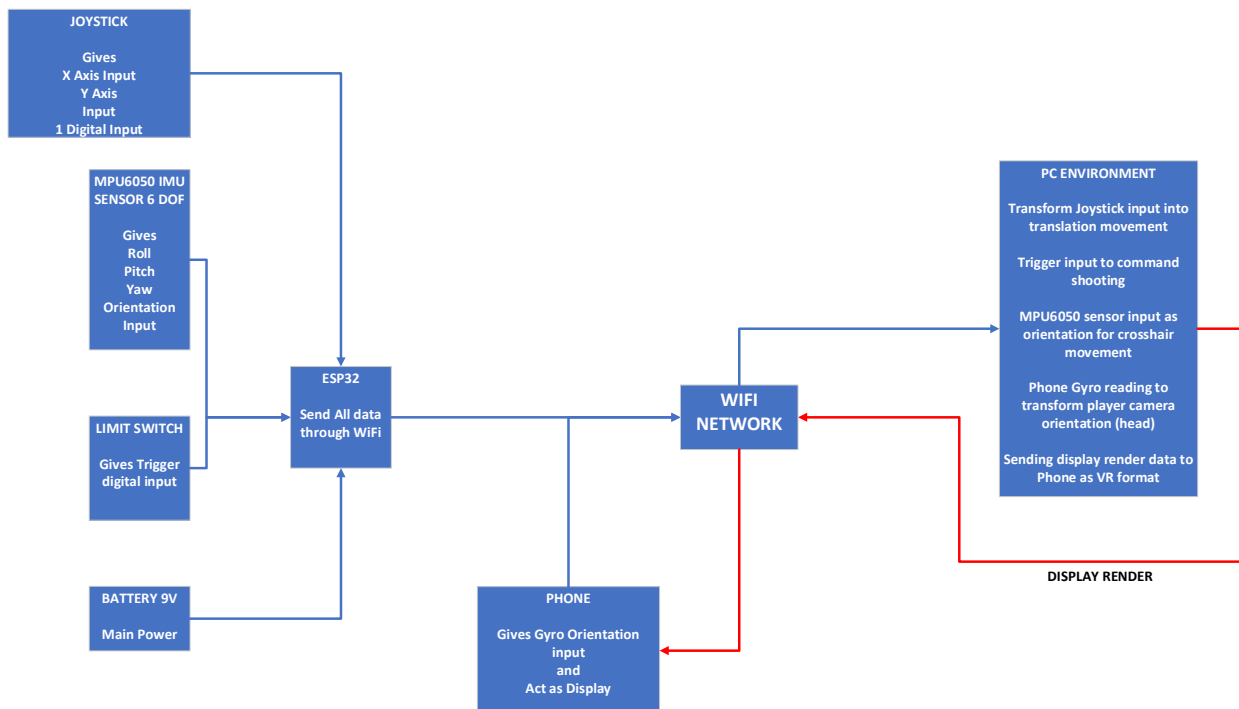


Figure 6 Data workflow in VR mode of the system

In the VR mode, the interaction process begins with the custom handheld controller and the Android smartphone. The ESP32 continuously acquires joystick, trigger button, and controller orientation data before transmitting them wirelessly to the Unity Desktop application. Simultaneously, the smartphone measures the user's head orientation using its built-in gyroscope and sends the orientation data to Unity. The Unity application combines both information sources to control player movement, camera orientation, and interaction within the virtual environment. After updating the simulation, Unity renders the resulting scene and streams the display back to the smartphone, where it is presented through the mobile VR headset. This closed-loop process is repeated continuously to maintain real-time immersive interaction.

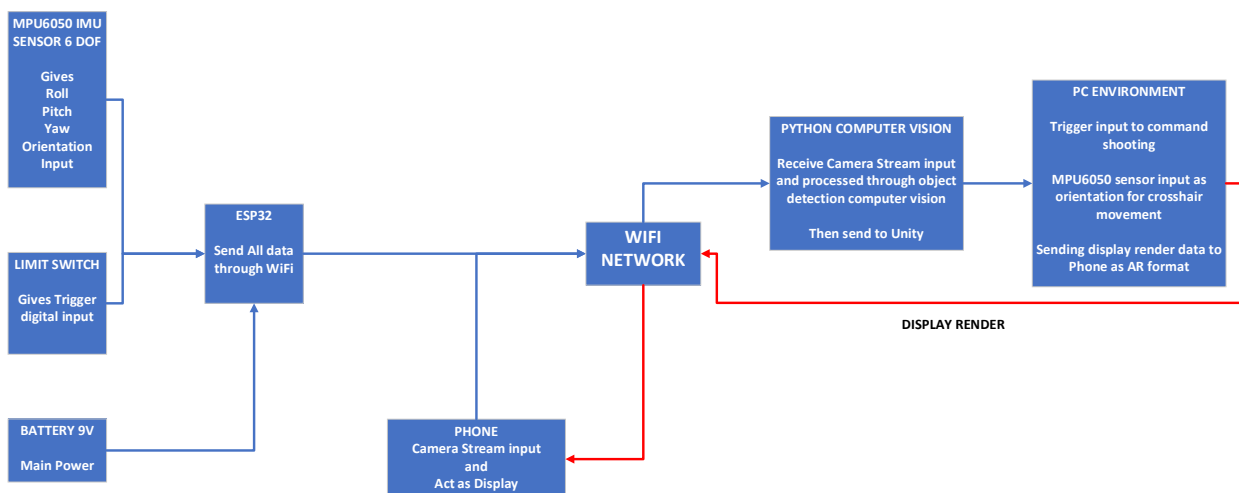


Figure 7 Data workflow in AR mode of the system

In the AR mode, the interaction workflow incorporates an additional computer vision stage. The smartphone continuously captures images of the surrounding environment and streams the camera frames to a Python-based object detection module YOLO26. The detected object coordinates are then transmitted to the Unity

Desktop application, where corresponding virtual targets are generated and positioned according to the detection results. At the same time, the ESP32 sends trigger and controller button inputs to Unity for user interaction. Unlike the VR mode, joystick-based player movement and camera orientation are not utilized because navigation is naturally achieved by the user's physical movement within the environment. After combining the camera image, detected objects, and user interactions, Unity renders the augmented scene and streams the final visualization back to the smartphone for display. Similar to the VR mode, this process executes continuously to provide real-time augmented interaction.

2.4.3 Testing Procedure

The developed system was experimentally evaluated in both Virtual Reality (VR) and Augmented Reality (AR) modes. The evaluation focused on three quantitative parameters: rendering performance, response latency, and crosshair mapping accuracy. These parameters were selected to characterize the prototype's operational performance and to provide baseline measurements relevant to cybersickness considerations in immersive environments [19],[20],[21].

Frame rate (FPS) was measured separately for the Unity desktop application and the Unity Android application because each application has its own rendering workload and operating conditions. For the desktop application, FPS testing was performed under three scenarios: Idle, Crosshair Movement, and Shooting/Respawn. Each scenario was executed for approximately 60 seconds and repeated five times in both VR and AR modes. For the Android application, FPS was recorded during system operation in each mode to characterize the rendering performance experienced on the smartphone display.

Response latency was measured internally within each application rather than across the complete system. Measuring total system latency would require synchronizing timestamps across heterogeneous components, communication interfaces, and software environments, which was beyond the scope of this prototype evaluation. For the Unity desktop application, latency was defined as the elapsed time between the receipt of a trigger input from the handheld controller and the corresponding shooting event being processed in Unity. This interval was selected to characterize the responsiveness of the desktop application's shooting interaction without including the additional delays associated with smartphone sensor input, camera streaming, or display transmission. For the Unity Android application, latency was measured as the interval between receiving a display frame from the desktop application and completing its display on the smartphone. This measurement represents the responsiveness of the smartphone's primary display function within the proposed architecture. For each mode and application, the latency test consisted of 50 consecutive shooting attempts and was repeated five times for the desktop application. The Android application's latency was recorded during operation using the corresponding application-side measurement procedure.

Crosshair mapping accuracy was evaluated by comparing the target position with the resulting crosshair position during repeated shooting trials. For every successful shot, the screen coordinates of the target center and crosshair center were recorded to calculate the spatial mapping error. Because crosshair positioning and shooting interaction are processed within the desktop application, this measurement was performed on the PC side for both VR and AR modes; the resulting visual output was then transmitted to the smartphone.

The recorded data were processed using descriptive statistical analysis, including the average, minimum, maximum, and standard deviation where applicable. These measures provide a quantitative basis for characterizing the rendering performance, internal response latency, and aiming behavior of the proposed hybrid Mixed Reality system.

3 Results and Discussion

Following the system design described in the previous section, the hardware and software components were successfully integrated into a complete prototype capable of operating in both Virtual Reality (VR) and Augmented Reality (AR) modes. The successful integration enables real-time interaction between the custom handheld controller, smartphone, desktop simulation, and wireless communication modules. The implemented prototype subsequently serves as the basis for evaluating rendering performance, interaction responsiveness, and crosshair mapping accuracy under different operating conditions.

3.1 View Display

Figure 8 and Figure 9 present the visual appearance of the developed Mixed Reality gun simulator in Virtual Reality (VR) and Augmented Reality (AR) modes, respectively. Both modes utilize the same hardware platform but provide different visualization and interaction environments according to their intended operation.

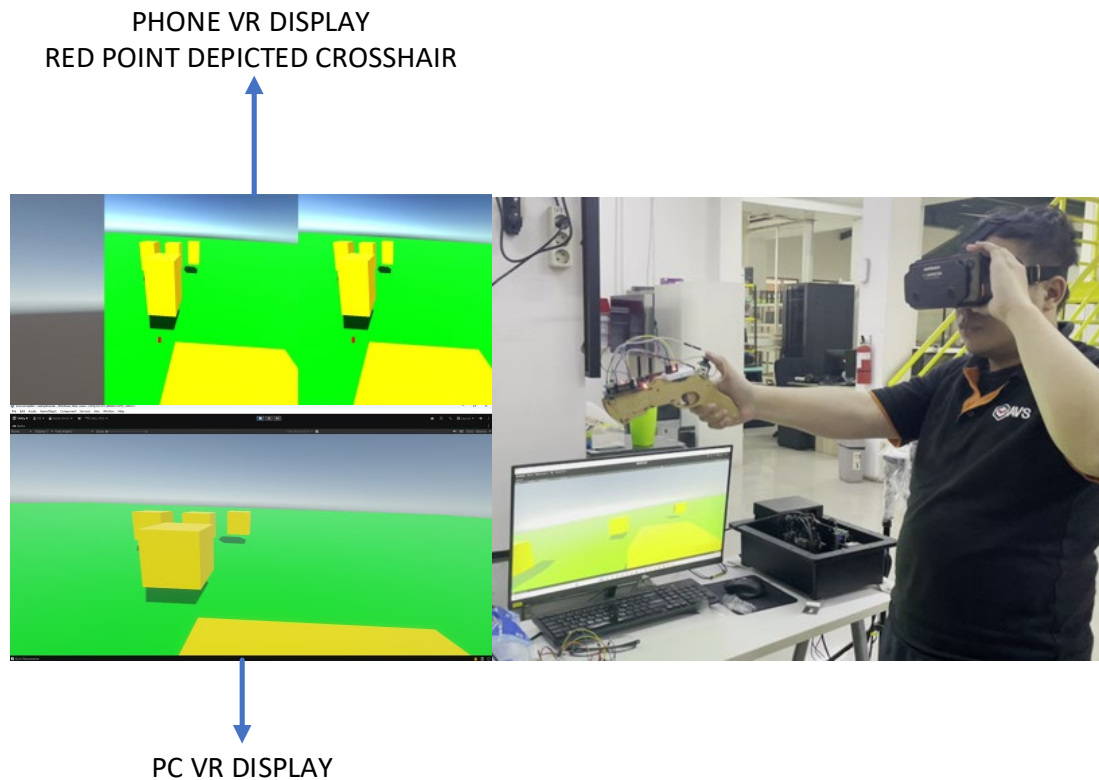


Figure 8 Interface of VR mode gun simulator

In the VR mode (Figure 8), the user is immersed in a fully virtual environment developed in Unity. The scene consists of a simple ground plane, a sky environment, and multiple cube objects that serve as interactive shooting targets. A red crosshair is displayed at the center of the screen and moves according to the orientation of the custom handheld controller measured by the IMU sensor. When a target is successfully hit, it is automatically regenerated at another random position while maintaining a constant number of targets within the virtual environment.

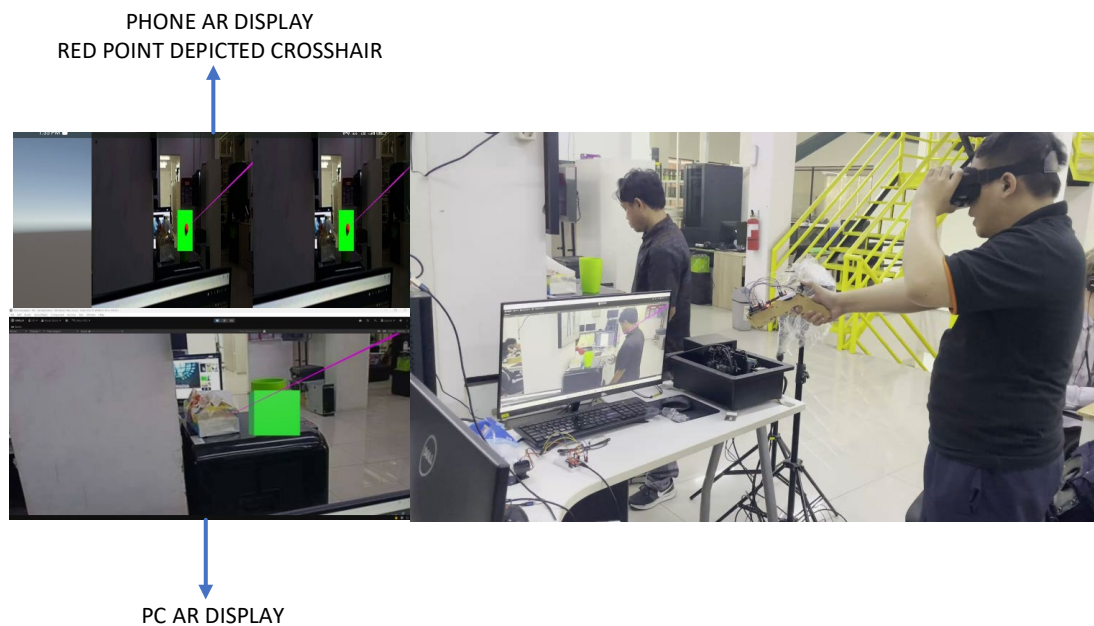


Figure 9. Interface of AR gun simulator

In the AR mode (Figure 9), the smartphone camera provides a live view of the surrounding environment, which is displayed together with virtual elements generated by the Unity application. Detected real-world objects obtained through the computer vision module are represented as virtual cube targets that can be interacted with

using the handheld controller. Due to differences between the smartphone camera characteristics and the virtual camera projection, slight discrepancies in field of view and depth perception may be observed. Nevertheless, the displayed scene demonstrates the successful integration of live camera visualization with real-time object detection and virtual object rendering.

The visual results shown in both operating modes confirm that the developed system successfully provides interactive visualization for VR and AR environments.

3.2 Mode Explanation

Both operating modes were developed to demonstrate the feasibility of the proposed smartphone-based Mixed Reality platform through a simple first-person shooting simulation. Rather than constructing a complex game environment, the implementation focuses on validating the interaction pipeline, including user input, aiming, shooting, object interaction, and real-time visual feedback. Therefore, both modes employ a simple cube as the primary interactive object.

In both VR and AR modes, the cube functions as a hittable target. When the user successfully shoots a target, the cube is automatically regenerated to maintain continuous interaction throughout the experiment. Although the target regeneration mechanism differs between the two operating modes according to their respective environments, the interaction principle remains identical. This simplified implementation provides sufficient functionality to evaluate rendering performance, interaction responsiveness, and crosshair mapping accuracy without introducing unnecessary complexity from game mechanics or scene design.

The operational characteristics of each mode are described in the following subsections.

3.2.1 Virtual Reality (VR) Mode

In the VR mode, the user interacts within a fully virtual environment rendered by the Unity engine and displayed on a smartphone mounted inside a mobile VR headset. The smartphone gyroscope controls the viewing direction, allowing natural head movement to change the camera orientation. Player translation is performed using the analog joystick integrated into the custom handheld controller, enabling forward, backward, left, and right movement within the virtual environment. Shooting is executed using the trigger button, while the joystick button is used to recenter the controller whenever orientation drift becomes noticeable.

To demonstrate immersive interaction while maintaining a simple experimental environment, the virtual scene consists of a ground plane, a sky environment, and five cube targets positioned within a predefined spawning region above the ground. The total number of targets remains constant throughout the simulation. When a cube is successfully hit, it immediately disappears and a new cube is generated at a random position inside the designated spawning area. This regeneration mechanism enables continuous interaction without increasing scene complexity, making the environment suitable for repeated experimental evaluation.

A red crosshair is displayed at the center of the screen and serves as the primary aiming indicator. Unlike conventional VR systems that directly rotate the virtual camera using the controller orientation, the proposed system interprets the IMU measurements as two-dimensional cursor movement on the display plane. The MPU6050 continuously measures the controller's roll and pitch angles, which are transmitted by the ESP32 to the Unity application through a wireless connection. After filtering and sensitivity scaling, these angular measurements are converted into horizontal and vertical screen displacements to determine the crosshair position.

Assuming the controller rotates by roll angle θ_r and pitch angle θ_p relative to its calibrated neutral orientation, the corresponding screen displacement is approximated as

$$\begin{aligned}x &= r \sin(\theta_r) \\y &= r \sin(\theta_p)\end{aligned}\tag{1}$$

where r denotes the sensitivity scaling factor that converts angular displacement into screen displacement, while x and y represent the horizontal and vertical offsets relative to the screen center. The final crosshair position is obtained by equation (2)

$$\begin{aligned}X_{\text{crosshair}} &= X_{\text{center}} + x \\Y_{\text{crosshair}} &= Y_{\text{center}} - y\end{aligned}\tag{2}$$

where X_{center} and Y_{center} denote the center coordinates of the display. Dead-zone filtering and smoothing are subsequently applied to suppress small sensor fluctuations and reduce hand tremor, resulting in more stable aiming during interaction.

Once the desired target is aligned with the crosshair, pressing the trigger button initiates a ray-casting operation from the virtual camera through the current crosshair position. If the ray intersects one of the cube targets, the target is removed from the scene and a new cube is immediately generated at another random location within the predefined spawning region. This mechanism emulates the basic interaction of a first-person shooting application while maintaining a controlled environment for quantitative evaluation.

Although this mapping provides an intuitive aiming mechanism, its accuracy is influenced by the characteristics of the MPU6050. Since the sensor combines only a three-axis gyroscope and a three-axis accelerometer, it does not provide an absolute heading reference and is therefore susceptible to orientation drift during prolonged operation. Software-based filtering, dead-zone implementation, and manual re-centering help reduce this effect but cannot eliminate it completely. Furthermore, the proposed mapping assumes a simplified relationship between controller rotation and screen displacement, whereas the true perspective projection of the virtual camera is inherently nonlinear.

3.2.2 Augmented Reality (AR) Mode

Unlike the VR mode, the AR mode combines the real-world environment with virtual objects generated from real-time object detection. Instead of rendering a fully virtual scene, the smartphone continuously captures the surrounding environment using its camera and transmits the video stream to the desktop computer through a wireless connection. The captured frames are processed by a Python-based computer vision module implementing the YOLO26 object detection algorithm. The detection results are then transmitted to the Unity application, where virtual targets are generated according to the detected object coordinates.

Unlike marker-based AR approaches that rely on specific development frameworks or predefined reference objects, the proposed approach uses YOLO26 object detection to identify objects dynamically without requiring each target to be individually registered within a vendor-specific AR platform. This design provides greater flexibility in selecting real-world objects for interaction. However, it introduces additional computational and communication demands because camera frames must be captured by the smartphone, transmitted to the desktop computer, and processed by the computer-vision module before the corresponding virtual targets can be generated in Unity. Consequently, the camera frame resolution and transmission rate must be configured carefully to balance detection input quality, communication overhead, and smartphone rendering performance.

To simplify the interaction while preserving the concept of mixed reality, each detected object is represented by a virtual cube that serves as a shootable target. The cube is positioned according to the image coordinates obtained from the YOLO26 detection process, allowing the virtual object to appear aligned with its corresponding real-world object. Multiple object detections are supported simultaneously, enabling several virtual targets to exist within the scene whenever multiple objects are recognized.

The interaction mechanism remains similar to that of the VR mode, where the user aims using the crosshair and fires using the trigger button. However, unlike the randomly generated targets in the virtual environment, the cube positions in AR mode are determined entirely by the object detection results. When a virtual cube is successfully hit, it immediately disappears from the scene and is automatically regenerated after a delay of approximately three seconds at the same detection coordinate. This mechanism allows repeated interaction with the same detected object while preserving the correspondence between the real-world object and its associated virtual target.

The AR mode introduces additional processing stages that are not present in the VR mode, including wireless image transmission, real-time object detection, coordinate conversion, and synchronization between the computer vision module and the Unity application. Consequently, the overall system must maintain both spatial consistency and temporal synchronization between the real environment and the rendered virtual objects.

3.3 Evaluation Metric

The performance evaluation of the proposed smartphone-based Mixed Reality gun simulator focused on three quantitative parameters: frame rate (FPS), response latency, and crosshair mapping accuracy. FPS and latency were selected because rendering performance and system responsiveness are relevant considerations in cybersickness research [19], [20], [21]. Although these parameters do not directly establish cybersickness, they provide baseline indicators of the system's operational performance.

Crosshair mapping accuracy was selected because aiming precision is fundamental to gun simulator interaction, reflecting the spatial correspondence between the operator's intended target and the displayed crosshair.

Together, these three parameters characterize the system's rendering performance, responsiveness, and aiming accuracy.

3.3.1 Frame Rate (FPS) Evaluation

Frame rate (FPS) was measured separately for the Unity desktop application and the Unity Android application to characterize their rendering performance during system operation. On the PC, measurements were conducted in both VR and AR modes under three scenarios: Idle, Crosshair Movement, and Shooting/Respawn. Each scenario ran for approximately 60 seconds and was repeated five times, with Unity recording instantaneous FPS throughout each trial.

On the smartphone, FPS was recorded during operation in each mode using a single overall scenario rather than the three PC-specific scenarios. This distinction reflects the different roles of each device: the PC handles simulation and, in AR mode, computer-vision processing, while the smartphone primarily displays the transmitted visual output and provides mobile sensing and camera input. The recorded data were analyzed using average FPS, minimum FPS, and standard deviation.

Table 2. FPS evaluation results

Mode	Scenario	Avg FPS	Min FPS	Standard deviation
VR (PC)	Idle	149.44	3	59.88
VR (PC)	Movement	162.23	18.07	50.12
VR (PC)	Shooting	169.04	10.95	45.41
VR(Mobile)	All	56.20	50.47	2.78
AR(PC)	Idle	167.72	74.33	36.53
AR(PC)	Movement	163.67	45.53	38.52
AR(PC)	Shooting	162.6	51.76	43.59
AR(Mobile)	All	23.13	19.41	1.39

Table 2 presents the frame rate evaluation results. The results show that the average FPS values ranged from 149.44 to 169.04 FPS in VR(PC) mode and from 162.60 to 167.72 FPS in AR(PC) mode across all tested scenarios. Meanwhile for the mobile FPS calculation the FPS values got average around 56.20 with minimum value at 50.47 with small standard deviation that is 2.78 all for VR(Mobile) mode which is indicate a stable result. Meanwhile for the mobile AR FPS we got 23.13 as average FPS and 19.41 as minimum FPS with small standard deviation of 1.39, a stable one yet a rather below standard FPS.

3.3.2 Response Latency Evaluation

Response latency was measured separately within the Unity desktop and Android applications to characterize their respective response times, rather than the end-to-end latency of the complete system. On the PC, latency was defined as the interval between receiving the trigger input from the handheld controller and processing the corresponding shooting event in Unity. On the smartphone, latency was measured from receiving a visual frame transmitted by the PC to completing its processing for display. These measurements reflect different stages of the hybrid architecture and should not be interpreted as the total system latency.

PC latency measurements consisted of 50 consecutive shooting attempts, repeated five times in both VR and AR modes. Smartphone latency was recorded during operation in each mode using the corresponding application-side measurement procedure. The collected values were statistically analyzed using the average, minimum, maximum, and standard deviation.

Table 3 Latency evaluation results

Mode	Avg latency (ms)	Minimum latency (ms)	Maximum latency (ms)	Standard deviation (ms)
VR(PC)	2.216	0.2	517.31	30.62
VR(Mobile)	29.595	10.04	86.28	19.251
AR(PC)	2.273	0.23	593.036	33.883
AR(Mobile)	49.990	1.9	86.31	19.488

The results in Table 3 show that the average PC-side latency was approximately 2.2 ms in both VR and AR modes. However, the maximum values exceeded 500 ms due to a single isolated outlier in each mode, among hundreds of recorded attempts. Most measurements remained below 1 ms, indicating that these outliers were not representative of typical response behavior. They may have resulted from a temporary application or system pause while other processes were running on the PC, although the exact cause could not be confirmed. Reducing background processing and overall PC workload during experiments may help minimize the likelihood of such interruptions.

In comparison, smartphone-side latency averaged 29.595 ms in VR mode and 49.990 ms in AR mode. This difference reflects the distinct responsibilities of the smartphone in each mode. In VR mode, the phone primarily transmits gyroscope data to the PC for camera orientation updates. In AR mode, the phone must activate its camera, capture image frames, and process them into a suitable format and resolution for transmission to the PC. These additional processing and communication stages impose a greater workload, contributing to the higher measured smartphone-side latency in AR mode.

3.3.3 Crosshair Mapping Error Evaluation

Crosshair mapping accuracy was evaluated by measuring the spatial deviation between the intended target position and the displayed crosshair position during repeated shooting trials. This parameter was selected because accurate crosshair positioning directly affects target acquisition in both Virtual Reality (VR) and Augmented Reality (AR) modes. The error was quantified using the two-dimensional Euclidean distance between the target center and crosshair center in screen coordinates, following a measurement approach also used in targeting performance research [22]. This method provides a consistent measure of aiming discrepancy on the displayed image.

For each shot, the screen coordinates of the target center (x_t, y_t) and crosshair center (x_c, y_c) were recorded. The horizontal and vertical deviations were first calculated as

$$\begin{aligned}\Delta x &= x_t - x_c \\ \Delta y &= y_t - y_c\end{aligned}\tag{3}$$

The overall crosshair mapping error was then computed as the Euclidean distance between the target position and the crosshair position,

$$\text{Error}_{\text{pixels}} = \sqrt{(\Delta x)^2 + (\Delta y)^2}\tag{4}$$

where a smaller error indicates better alignment between the target and the displayed crosshair. The experiment was conducted in both VR and AR modes using repeated shooting trials, and the recorded error values were statistically analyzed using the average, minimum, maximum, and standard deviation.

Table 4 Crosshair Error evaluation results

Mode	Avg error (pixels)	Minimum error (pixels)	Maximum error (pixels)	Standard deviation (pixels)
VR	33.61	0.19	193.34	26.06562
AR	135.4497	100.64	203.54	15.45017

Table 4 summarizes the crosshair mapping error obtained during the evaluation. The VR mode achieved an average mapping error of 33.61 pixels (approximately 1.05 cm), while the AR mode produced a larger average error of 135.45 pixels (approximately 4.22 cm). The minimum error recorded in VR was 0.19 pixels (approximately 0.006 cm), indicating that the crosshair could closely coincide with the intended target under favorable conditions. In contrast, the AR mode had a minimum error of 100.64 pixels (approximately 3.13 cm), reflecting the additional alignment challenges associated with the real-world camera image, computer vision detection, and virtual rendering pipeline.

The standard deviation also reveals different characteristics between the two operating modes. Although VR produced a lower average error, it exhibited greater variation, with a standard deviation of 26.07 pixels (approximately 0.81 cm), compared with 15.45 pixels (approximately 0.48 cm) for AR. This indicates that VR

aiming error fluctuated more throughout the experiment, whereas AR exhibited more consistent errors despite its larger average offset.

3.4 Analysis of Metric Result

The PC-side FPS evaluation showed average frame rates of 149.44–169.04 FPS in VR and 162.60–167.72 FPS in AR. These averages exceed the 120 FPS level identified as an important threshold in the VR experiments conducted by Wang et al. [23], suggesting that the desktop application achieved strong average rendering performance in both operating modes. However, the minimum FPS values fell to 3.00–18.07 FPS in VR and 45.53–74.33 FPS in AR, indicating occasional frame drops. Therefore, the PC-side results exceed the 120 FPS reference when assessed by average performance, but do not maintain this level consistently throughout every recorded trial. Furthermore, these measurements characterize desktop rendering performance rather than the frame rate ultimately experienced by the user on the smartphone display.

Smartphone-side measurements revealed a substantial difference between the two modes. VR achieved an average of 56.20 FPS, whereas AR achieved only 23.13 FPS. Z.Wang. [24] evaluated mobile VR performance and reported that frame rates below 50 FPS were perceived by participants as tolerable at best and nauseating at worst. Based on this study-specific reference, the smartphone VR result of 56.20 FPS exceeds the reported 50 FPS level, although it remains below the 120 FPS level associated with improved task performance and reduced simulator sickness symptoms in the VR experiments of Wang et al. [23]. Accordingly, smartphone VR exceeds the mobile-VR reference of 50 FPS on average, but does not reach the higher performance level identified in [23]. These comparisons provide context rather than a definitive prediction of user comfort because the hardware, application workload, and experimental conditions differ.

For smartphone AR, Vuforia's developer documentation identifies 30 FPS as the recommended target render rate for Android devices, while noting that the appropriate target depends on the device and application requirements [25]. The measured AR average of 23.13 FPS falls below this recommendation, indicating that the smartphone AR mode does not meet the cited 30 FPS rendering target. Vuforia also notes that performance-intensive AR applications can increase device temperature and lead to operating-system throttling, highlighting the importance of balancing computational workload, rendering performance, and power consumption [25]. Although this documentation describes optimization practices for Vuforia Engine rather than the proposed YOLO26-based architecture, its recommendations provide a relevant mobile-AR performance reference. The result should therefore be interpreted as falling below a recommended development target, rather than as proof that the system is unusable or that all AR applications require a minimum of 30 FPS.

The lower smartphone AR frame rate is consistent with the additional workload associated with camera capture, image preparation, and transmission to the PC in the proposed architecture. Consequently, high PC-side FPS does not translate into equivalent smartphone performance, highlighting the importance of evaluating both devices in a hybrid system. Overall, the desktop application demonstrates high average rendering performance in both modes, while smartphone VR exceeds the cited mobile-VR reference of 50 FPS and smartphone AR falls below Vuforia's recommended 30 FPS target for Android AR rendering. The smartphone AR mode therefore represents the principal rendering-performance limitation and warrants further optimization. These comparisons characterize operational performance; FPS measurements alone cannot establish whether users experience cybersickness or whether the system meets a particular comfort threshold.

The latency evaluation revealed different response characteristics between the PC and smartphone components. The average PC-side latency was 2.216 ms in VR mode and 2.273 ms in AR mode, indicating nearly identical shooting response times within the Unity desktop application. However, the maximum values reached 517.31 ms in VR and 593.036 ms in AR, with standard deviations of 30.620 ms and 33.883 ms, respectively. These large deviations were influenced by isolated latency spikes, with one measurement exceeding 500 ms in each mode among hundreds of recorded attempts. Although temporary application or system interruptions may have contributed to these outliers, their exact cause could not be confirmed. Thus, the average values indicate rapid PC-side shooting response under typical conditions, while the maximum values highlight occasional interruptions that warrant further investigation.

In comparison, smartphone-side latency averaged 29.595 ms in VR and 49.990 ms in AR, with standard deviations of 19.251 ms and 19.488 ms, respectively. The average AR latency was therefore 20.395 ms higher than the VR latency. This difference is consistent with the distinct processing requirements of the two modes. In VR, the smartphone primarily transmits inertial sensor data for camera orientation updates. In AR, the smartphone additionally captures camera frames, prepares them for transmission, and sends them to the PC for computer-vision processing. These additional operations increase the processing and communication workload associated with the AR mode.

DAQRI [26] explains that latency in mobile AR and VR systems can accumulate across sensor acquisition, tracking, rendering, and display stages. Its analysis illustrates that a conventional pipeline can exceed 100 ms, while specialized techniques such as motion prediction and late-stage image correction can substantially reduce perceived latency. In this context, the measured smartphone AR latency of 49.990 ms is below the latency of the conventional pipeline illustrated in [26]. However, this comparison is illustrative rather than a direct performance benchmark, because DAQRI discusses motion-to-photon latency, whereas the present study measures the interval between receiving a frame from the PC and completing its processing for display on the smartphone. The two measurements therefore represent different stages and definitions of latency.

Prior research also identifies latency as an important consideration in immersive experiences and simulator sickness [19],[21]. Nevertheless, commonly discussed latency levels should not be treated as universal pass/fail thresholds for the present measurements. The PC-side results represent the interval between receipt of the controller trigger input and processing of the corresponding shooting event in Unity, while the smartphone-side results characterize frame reception and processing for display. Neither measurement represents the complete motion-to-photon latency of the hybrid system. Overall, the results demonstrate rapid average PC-side shooting response in both modes and lower smartphone-side latency in VR than in AR. The higher AR latency and the isolated PC-side spikes identify areas for further optimization, while measurement of the complete system latency remains outside the scope of the present evaluation.

The crosshair mapping evaluation indicates different accuracy characteristics between the two operating modes. The VR mode achieved a lower average mapping error of 33.61 pixels (approximately 1.05 cm) than the AR mode, which recorded 135.45 pixels (approximately 4.22 cm). These centimeter conversions are approximate and assume a 27-inch, 16:9 PC monitor with a resolution of 1920 × 1080 pixels. The results indicate higher spatial accuracy in VR, whereas the larger AR error may be associated with additional camera processing, object detection, and coordinate transformation required to align the crosshair with the target. However, these factors were not isolated experimentally and therefore cannot be confirmed as the sole causes of the observed difference.

Interestingly, AR exhibited a smaller standard deviation of 15.45 pixels (approximately 0.48 cm), compared with 26.07 pixels (approximately 0.81 cm) in VR. This indicates that AR mapping errors varied less across the recorded trials despite the substantially larger average error. The relatively consistent error pattern may indicate a systematic offset that could potentially be reduced through improved calibration. Overall, VR provided higher spatial accuracy, while AR exhibited more consistent mapping errors despite its larger average deviation.

Collectively, the three evaluation parameters demonstrate that the proposed system can operate as a proof-of-concept Mixed Reality simulation platform, although its performance varies across operating modes and processing components. The desktop application achieved high average frame rates and rapid average shooting response in both VR and AR modes, despite occasional frame drops and isolated latency spikes. On the smartphone, VR achieved an average frame rate above the cited mobile-VR reference, whereas AR remained below Vuforia's recommended 30 FPS target for Android AR rendering. Smartphone-side latency was also higher in AR than in VR. Crosshair mapping evaluation showed lower average spatial error in VR, while AR exhibited a larger but less variable error, suggesting potential for further calibration. These findings support the operational feasibility of the hybrid architecture for research and educational experimentation while identifying smartphone AR rendering performance, aiming alignment, and occasional response interruptions as areas for improvement. Since the evaluation did not measure complete end-to-end latency or directly assess user comfort, the results should not be interpreted as proof that the system meets universal immersive-performance requirements.

3.5 Research Limitations

This study has several limitations arising from the implementation of a low-cost proof-of-concept MR platform. First, the handheld controller relies on an MPU6050 inertial sensor, which is inherently susceptible to measurement drift over prolonged operation. Although software-based calibration, filtering, and compensation techniques were implemented to reduce this effect, residual orientation drift may still influence crosshair stability and aiming accuracy during extended use.

Second, the analog joystick integrated into the controller does not provide perfectly smooth omnidirectional input across the full 360° range. Mechanical limitations of the low-cost joystick introduce slight nonlinearities and dead-zone variations that may affect movement consistency, particularly during fine user adjustments.

Third, the communication performance depends on the quality and stability of the wireless network. Since sensor data, camera streams, and interaction commands are transmitted through Wi-Fi, variations in network latency, interference, or signal quality may occasionally introduce communication delays or temporary frame fluctuations.

Finally, the proposed prototype integrates multiple heterogeneous software and hardware components, including an Android smartphone running Unity, a Python-based computer vision module, an ESP32 embedded

controller, and a Unity application on a desktop computer. While this modular architecture facilitates rapid prototyping and flexible experimentation, it also requires multiple communication interfaces and data synchronization processes that may introduce additional overhead compared with a unified development environment. Future implementations could reduce system complexity by integrating computer vision directly into the Unity application or mobile platform, eliminating intermediate communication stages and potentially improving overall responsiveness and system robustness.

3.6 Implication and Future Development

This study indicates that a smartphone-based Mixed Reality platform can provide a feasible foundation for proof-of-concept research using consumer-grade hardware, despite performance differences between operating modes.

Future development can be carried out incrementally by improving individual system components. The MPU6050 could be replaced with a 9-DOF inertial sensor incorporating a magnetometer to reduce orientation drift. Alternative aiming methods, such as sensor fusion or vision-assisted tracking, could also be investigated to improve spatial accuracy.

On the smartphone side, future work should prioritize improving AR rendering performance through camera-frame optimization, more efficient frame transmission, and reduced processing overhead. These improvements could help increase frame rate and reduce latency while maintaining visual quality. The modular architecture also supports future integration of advanced tracking devices, perception algorithms, and communication frameworks without fundamentally changing the overall system design.

4 Conclusion

This study presented the development and experimental evaluation of a smartphone-based Mixed Reality gun simulator capable of operating in Virtual Reality (VR) and Augmented Reality (AR) modes using consumer-grade hardware. The proposed system integrates a smartphone, a custom handheld controller, embedded sensing, computer vision, wireless communication, and a Unity-based simulation environment to provide an accessible platform for immersive interaction research.

Experimental evaluation was conducted using frame rate (FPS), component-level response latency, and crosshair mapping accuracy. The desktop application achieved average frame rates of 149.44–169.04 FPS in VR and 162.60–167.72 FPS in AR. However, smartphone performance differed considerably, averaging 56.20 FPS in VR and 23.13 FPS in AR. Average PC-side shooting response latency was below 3 ms in both modes, while smartphone-side latency averaged 29.595 ms in VR and 49.990 ms in AR. Crosshair mapping errors averaged 33.61 pixels in VR and 135.45 pixels in AR. These results demonstrate the operational feasibility of the proposed hybrid architecture while identifying smartphone AR rendering performance and aiming accuracy as areas for improvement.

Despite limitations including inertial sensor drift, controller precision, occasional response interruptions, and the complexity of multi-platform communication, this work establishes a practical foundation for further development. Future work may investigate more accurate tracking methods and optimize smartphone-side image processing and transmission to improve AR performance. Overall, the proposed platform offers a foundation for educational applications, rapid prototyping, and early-stage research in immersive simulation and human–computer interaction.

For interest and open collaboration, the result of each mode can be seen through these link:

AR Mode: https://www.youtube.com/watch?v=6YFg2_IEMb4

VR Mode: <https://www.youtube.com/watch?v=7dlq-dlRXUc>

Acknowledgement

This research was conducted independently without external financial support from any institution.

References

- [1] S. Picard, N. Sun, and J. Botev, "XR MUSE: An open-source Unity framework for extended reality-based networked multi-user studies," *Virtual Worlds*, vol. 3, no. 4, pp. 404–417, 2024. <https://doi.org/10.3390/virtualworlds3040022>

- [2] V. De Luca, M. C. Barba, G. D'Errico, B. L. Nuzzo, and L. T. De Paolis, "A user experience analysis for a mobile Mixed Reality application for cultural heritage," *Virtual Reality*, vol. 27, pp. 2821–2837, 2023. <https://doi.org/10.1007/s10055-023-00840-w>
- [3] T. N. Fitria, "Augmented Reality (AR) and Virtual Reality (VR) technology in education: media of teaching and learning : A review," *International Journal of Computer and Information System (IJCIS)*, vol. 4, no. 1, pp. 14–25, 2023. <https://doi.org/10.29040/ijcis.v4i1.102>
- [4] F. A. Aziz, A. S. M. A. Alsaeed, S. Sulaiman, M. K. A. Mohd Ariffin, and M. F. Al-Hakim, "Mixed Reality improves education and training in assembly processes," *Journal of Engineering and Technological Sciences*, vol. 52, no. 4, pp. 598–607, 2020. <https://doi.org/10.5614/j.eng.technol.sci.2020.52.4.10>
- [5] J.-S. Kim, K.-W. Kim, H.-J. Kim, and S.-Y. Moon, "Development and usability evaluation of a leap motion-based controller-free VR training system for inferior alveolar nerve block," *Applied Sciences*, vol. 16, no. 3, Art. no. 1325, 2026. <https://doi.org/10.3390/app16031325>
- [6] D. Ergen and A. N. Yilmaz, "The effects of leap motion-based virtual reality training on performance-based dexterity in children with cerebral palsy: A randomized controlled trial," *Egyptian Pediatric Association Gazette*, vol. 74, no. 1, Art. no. 49, 2026. <https://doi.org/10.1186/s43054-026-00552-1>
- [7] Y. Lei, Y. Deng, L. Dong, X. Li, X. Li, and Z. Su, "A novel sensor fusion approach for precise hand tracking in virtual reality-based human-computer interaction," *Biomimetics*, vol. 8, no. 3, Art. no. 326, 2023. <https://doi.org/10.3390/biomimetics8030326>
- [8] M. Gil-Martín, M. R. Marini, R. San-Segundo, and L. Cinque, "Dual leap motion controller 2 : A robust dataset for multi-view hand pose recognition," *Scientific Data*, vol. 11, Art. no. 1102, 2024. <https://doi.org/10.1038/s41597-024-03968-9>
- [9] A. Houston, V. Walters, T. Corbett, and R. Coppack, "Evaluation of a multi-sensor leap motion setup for biomechanical motion capture of the hand," *Journal of Biomechanics*, vol. 127, Art. no. 110713, 2021. <https://doi.org/10.1016/j.jbiomech.2021.110713>
- [10] L. Caro, E. Villablanca, and B. Peralta, "A Virtual Reality application using hand motion tracking with leap motion sensor," in 2020 39th International Conference of the Chilean Computer Science Society (SCCC), Coquimbo, Chile, Nov. 2020. <https://doi.org/10.1109/SCCC51225.2020.9281210>
- [11] Y. Zhang and O. E. Meruvia-Pastor, "Operating virtual panels with hand gestures in immersive VR games : Experiences with the leap motion controller," in Augmented Reality, Virtual Reality, and Computer Graphics (Lecture Notes in Computer Science, vol. 10324), L. T. De Paolis and P. Bourdot, Eds. Cham, Switzerland: Springer, 2017, pp. 306–319. https://doi.org/10.1007/978-3-319-60922-5_24
- [12] K. Mahdikhani and H. Ebrahimnezhad, "Integration of leap motion sensor with camera for better performance in hand tracking," *Metaverse*, vol. 5, no. 2, Art. no. 3020, 2024. <https://doi.org/10.54517/m.v5i2.3020>
- [13] I. Mostachetti, D. Ferrari, E. Barcella, M. Corna, K. Zavareh, C. Pasini, P. Valentini, and A. Vitali, "Usability Comparison Between Leap Motion Controller and MediaPipe in Serious Games," in dHealth 2026: Proceedings of the 20th Health Informatics Meets Digital Health Conference, Studies in Health Technology and Informatics, vol. 331. Amsterdam, The Netherlands: IOS Press, 2026, pp. 131–136. <https://doi.org/10.3233/SHTI260069>
- [14] Indriani, M. Harris, and A. S. Agoes, "Applying hand gesture recognition for user guide application using MediaPipe," in Proceedings of the 2nd International Seminar of Science and Applied Technology (ISSAT 2021), Advances in Engineering Research, vol. 207. Paris, France: Atlantis Press, 2021, pp. 101–108. <https://doi.org/10.2991/aer.k.211106.017>
- [15] V. Patil, S. Sutar, S. Ghadage, and S. Palkar, "Gesture recognition for media interaction : A streamlit implementation with OpenCV and MediaPipe," *International Journal for Research in Applied Science and Engineering Technology*, vol. 11, no. 9, pp. 1039–1046, 2023. <https://doi.org/10.22214/ijraset.2023.55775>
- [16] H. Bai, L. Zhang, J. Yang, and M. Billingham, "Bringing full-featured mobile phone interaction into Virtual Reality," *Computers & Graphics*, vol. 97, pp. 42–53, 2021. <https://doi.org/10.1016/j.cag.2021.04.004>
- [17] S. S. Kyian and R. J. Teather, "Selection performance using a smartphone in VR with redirected input," in Proceedings of the ACM Symposium on Spatial User Interaction (SUI '21), Virtual Event, USA, Nov. 9–10, 2021, pp. 1–12. <https://doi.org/10.1145/3485279.3485292>
- [18] L. Schenck, C. Bäumer, B. Ross, G. Schäfer, N. Stember, H. Thomas, S. Stieglitz, and B. Timmermann, "Development of a smartphone virtual reality game to support the radiation therapy of children and adolescents in proton centers," *Frontiers in Pediatrics*, vol. 11, Art. no. 1163022, 2023. <https://doi.org/10.3389/fped.2023.1163022>
- [19] J.-P. Stauffert, F. Niebling, and M. E. Latoschik, "Latency and cybersickness: Impact, causes, and measures. A review," *Frontiers in Virtual Reality*, vol. 1, Art. no. 582204, 2020. <https://doi.org/10.3389/frvir.2020.582204>
- [20] T. Porcino, D. Reilly, E. Clua, and D. Trevisan, "A guideline proposal for minimizing cybersickness in VR-based serious games and applications," 2022, arXiv:2207.06346, <https://doi.org/10.48550/arXiv.2207.06346>

- [21] Tošić, D. Hoffman, and N. Balram, "Effect of latency on simulator sickness in smartphone virtual reality," *Journal of the Society for Information Display*, vol. 29, no. 7, pp. 561–572, 2021, <https://doi.org/10.1002/jsid.999>
- [22] T. Bourlai et al., "Development of a human performance baseline of lay error in targeting," *IEEE Access*, vol. 13, pp. 91182–91199, 2025, <https://doi.org/10.1109/ACCESS.2025.3571591>
- [23] J. Wang, R. Shi, W. Zheng, W. Xie, D. Kao, and H.-N. Liang, "Effect of frame rate on user experience, performance, and simulator sickness in Virtual Reality," *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 5, pp. 2478–2488, May 2023. <https://doi.org/10.1109/TVCG.2023.3247057>.
- [24] Z. Wang, K. Foo, S. Yan, V. A. Gonzalez, and N. Giacaman, "Refresh rate and graphical benchmarks for mobile VR application development," **Journal of Mobile Multimedia**, vol. 18, no. 6, pp. 1561–1598, 2022, <https://doi.org/10.13052/jmm1550-4646.1865>
- [25] PTC Inc., "Device Performance Optimization," *Vuforia Engine Library*. [Online]. Available: <https://developer.vuforia.com/library/vuforia-engine/platform-support/camera/device-performance-optimization/>. Accessed 12 October 2026
- [26] DAQRI, "Motion-to-Photon Latency in Mobile AR and VR," *Medium*. [Online]. Available: <https://medium.com/@DAQRI/motion-to-photon-latency-in-mobile-ar-and-vr-99f82c480926>. Accessed 12 October 2026