OPEN ACCESS

Greenfort International Journal of Applied Medical Science

Abbreviated Key Title: Grn Int J Apl Med Sci | ISSN 3048-6904(P) | ISSN 2584-2420(O) |
Greenfort International Publisher, Assam, India
Journal homepage: https://gijams.in

Review Article DOI: 10.62046/gijams.2025.v03i02.007

Web Interface for Image Processing Using Android

Heber Anandan[1], Dhanisha JL[1,] Vinitha B[1], Tamilselvi M[1], Jenita Manickam[1]

[1]Dr. Agarwal's Eye Hospital, Tirunelveli, Tamil Nadu, India

***Corresponding Author:** Heber Anandan | **Received:** 09.02.2025 | **Accepted:** 17.03.2025 | **Published:** 22.03.2025

Abstract: The main goal of this project is to develop an Android application that visualizes the results of a simulation performed in MATLAB/Simulink. The project, titled "Web Interface for Image Processing Using Android," proposes a system where image processing tasks are executed on MATLAB via a web service, with the Android application serving as the user interface. In this system, the user provides an input image from the Android application. The image is then transmitted to a server, where MATLAB performs the required image processing operations. Once the processing is complete, the output is sent back and displayed in the Android application. This setup allows computationally intensive tasks to be handled by MATLAB on a PC, ensuring that the mobile application remains lightweight and efficient. The Android application is developed using custom libraries created during this project, facilitating smooth communication with a MATLAB/ Simulink-based server over an internet connection. The backend of the system is implemented using PHP and My SQL, with XAMPP being used as the local server environment.

Keywords: Simulation, MATLAB (Matrix Laboratory), Simulink, Web, Android application, Image Processing.

Citation: Heber Anandan *et al*. The Web Interface for Image Processing Using Android. Grn Int J Apl Med Sci, 2025 Mar-Apr 3(2): 65-69.

INTRODUCTION

Technical advances have had a great impact on medical practice. Research in medical field image processing is still ongoing, for which equipment's are developed for segmentation, detection, classification and visualization of such images.The main aim of this project was to create an Android application which would communicate with simulation or control system running on the PC. The work was done in the laboratory provided by the organization. MATLAB and Android Studio were used for programming [1].

The purpose of this work is to research and develop methods in order to interface with MATLAB for an Android application running on a tablet or on a mobile phone. The expected results in working libraries that can be used to create Android applications capable of receiving data from MATLAB as well as sending data back for control purposes and visualizing the result of the simulation or the output of the actual physical device. The communication between devices should be established over the air via a Wi-Fi network rather than the data cable. By which we can access the matlab running on the computer.

The main disadvantages the previous, similar works had which were the limited supported devices, only worked for screens with specific sizes, and lack of ability to reuse solutions and apply them to different applications, have been solved. Applications build using the libraries created during this work support screens with wide variety of resolutions and sizes. The applications work in both, landscape and portrait mode [2].

ANDROID

Android OS is a free and open-source platform primarily designed for touch screen smartphones and tablets. It features a user-friendly interface based on direct manipulation, using touch gestures like tapping, swiping, pinching, and long-pressing to interact with on-screen content. Beyond touch input, Android supports various external devices, including keyboards, joysticks, trackballs, and mice, which can connect via USB or Bluetooth. Android apps are software applications built specifically for this platform, mainly targeting smartphones and tablets but also extending to smart watches (Wear OS), smart TVs (Android TV), car systems (Android Auto), and IoT devices. These apps are typically developed using Java or Kotlin in Android Studio, though other languages like C++, Dart (Flutter), and JavaScript (React Native) are also supported. The Android Software Development Kit (SDK) provides powerful tools and libraries to streamline app creation. Android stands out with its high level of customization, allowing users to personalize widgets, themes, and interfaces — even

supporting custom ROMs. It offers robust multitasking capabilities, enabling smooth switching between apps and efficient background process handling. The OS supports a range of connectivity options, including Wi-Fi, Bluetooth, NFC, GPS, and 5G, ensuring seamless communication across devices. Android also prioritizes security with biometric authentication like fingerprint and face unlock, app permission controls, and Google Play Protect for malware detection. Regular updates ensure the system stays secure and optimized, while its wide hardware compatibility makes it a top choice for devices ranging from budget phones to high-performance flagships [3].

MATLAB

MATLAB, which signifies "Matrix Laboratory," is a multi-paradigm computational system and fourth-generation language for programming.Developed by Math Works, it is widely used in engineering, science, and mathematics for tasks involving data analysis, simulations, and algorithm development. MATLAB supports matrix manipulations, function and data plotting, algorithm implementation, creation of user interfaces, and integration with other programming languages such as C, C++, Java, Fortran, and Python — making it highly versatile for both academic research and industrial applications.

One of MATLAB's most powerful extensions is Simulink, a block diagram-based environment specifically designed for modeling, simulating, and analyzing dynamic systems. Simulink works seamlessly alongside MATLAB, providing an intuitive way to design control systems, signal processing algorithms, robotics systems, embedded systems, and real-time simulations. It supports hardware integration, enabling communication with microcontrollers, sensors, and Android device-making it a preferred choice for mobile-based control systems and IoT projects [4].

The Android libraries developed in this project rely on Simulink models to simulate and control systems while the Android device serves as the user interface for data input, real-time control, and result visualization. This setup ensures efficient wireless communication between the Android app and MATLAB/Simulink models, leveraging Wi-Fi or Bluetooth connectivity. Additionally, MATLAB's ability to generate standalone code from Simulink models allows for embedded deployment, enabling the control logic to run directly on hardware like Raspberry Pi, Arduino, or Android devices for portable, real-world implementations [5].

DEVELOPMENT

Communication Between Devices

(A) User Datagram Protocol

User Datagram Protocol (UDP) uses a simple connectionless transmission model with a minimal protocol mechanism, making it ideal for applications that require low-latency communication. Unlike Transmission Control Protocol (TCP), which establishes a connection before data transmission and ensures packet delivery, UDP transmits data without prior setup, allowing for faster and more efficient communication.

With UDP, computer applications can send messages, referred to as datagrams, to other hosts on an Internet Protocol (IP) network without needing to establish a dedicated connection. This reduces overhead and improves transmission speed, making it suitable for real-time applications such as live streaming, VoIP, online gaming, and IoT communications [6].

Since the applications developed in this thesis are time-sensitive, UDP is preferred over TCP because it prioritizes speed over reliability. In real-time systems, dropping packets is preferable to waiting for delayed packets, as retransmissions could introduce latency and disrupt system performance. Additionally, UDP supports multicast and broadcast transmission, allowing a single sender to communicate with multiple receivers simultaneously, which is advantageous for networked control systems, sensor data transmission, and real-time monitoring applications.

(B) UDP in Android

To send and receive UDP datagrams in an Android OS application, the Datagram Socket class is used. This class implements a UDP socket for sending and receiving Datagram Packet objects over a network. It allows communication without establishing a dedicated connection, making it ideal for real-time applications.

The Datagram Socket can be bound to a specific IP address and port, enabling it to listen for incoming data while sending packets to a target destination IP and port. It supports timeouts to prevent indefinite waiting and allows adjusting buffer sizes for optimized performance. Additionally, it works efficiently with both unicast and broadcast transmissions, making it suitable for applications like IoT, gaming, and real-time monitoring systems [7].

Sending Data from Android

The need for sending data to MATLAB arises only when a specific action is triggered in MATLAB. Since this is not a continuous process, an Async Task was implemented as a solution to handle data transmission efficiently. Async Task allows performing background operations without blocking the main thread — ensuring the user interface (UI) remains responsive.

When an Android application is launched, the system creates a "main thread" (also known as the UI thread), responsible for handling user interactions, dispatching events, and rendering UI elements. This thread is critical to the app's responsiveness. Any long-running or time-consuming tasks — like network communication or data processing — must run on a

separate thread to avoid freezing the UI or causing "Application Not Responding" (ANR) errors.
Async Task works by splitting the task into three main phases: on Pre Execute(prepares the UI before the task starts), do In Background (handles background processing), and on Post Execute (updates the UI after the task completes). This ensures that the MATLAB communication happens smoothly, without interrupting the user's interaction with the app [8].

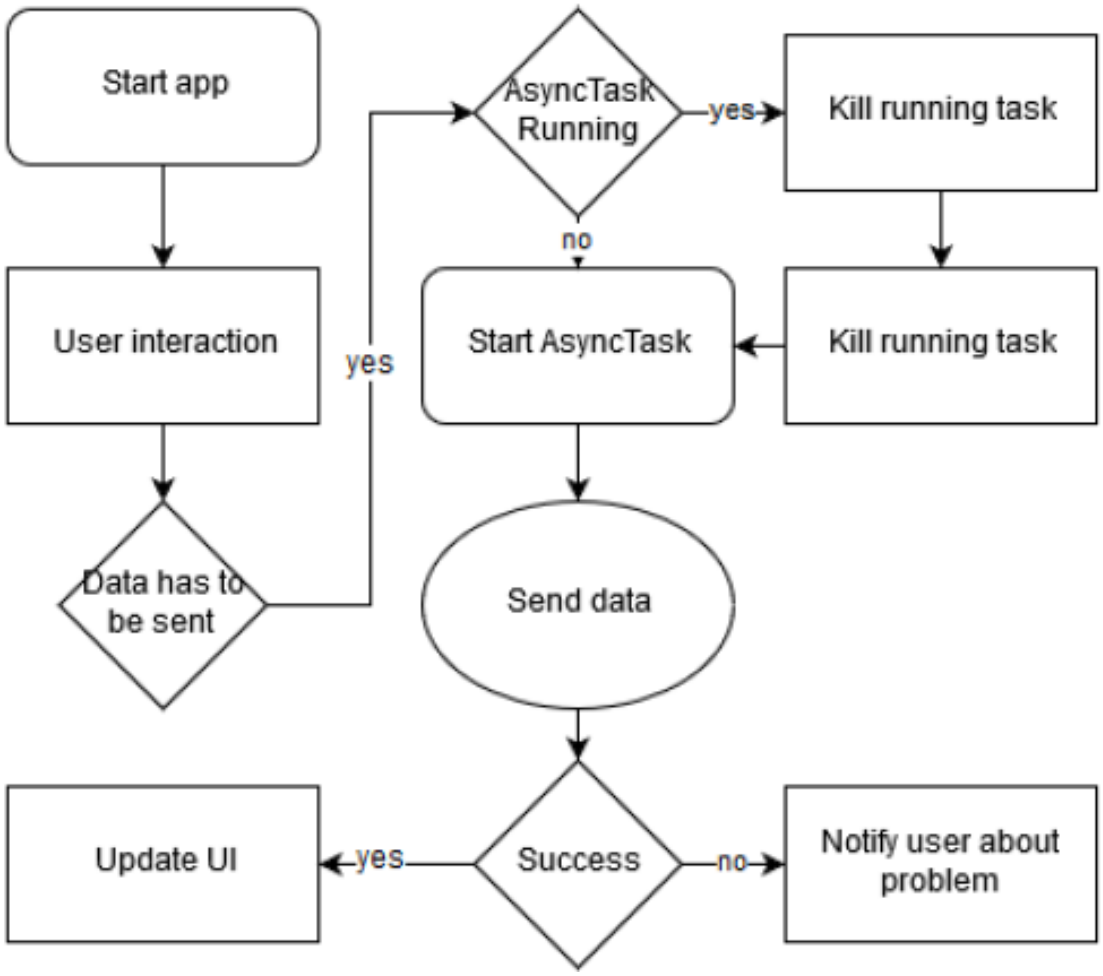

Fig:1 Data sending on Android

Receiving Data in Android

The approach used for sending data couldn't be applied to receiving incoming UDP packets from MATLAB, especially in systems like magnetic levitation, where data reception is a continuous, non-stop process. To handle this efficiently, an Android Service was implemented.

A Service is an application component designed for long-running background tasks without a user interface. It can run independently of user interactions, ensuring the data reception continues even if the user switches to another app or locks the screen. The main UI thread initiates the Service, which then continuously listens for and processes incoming data from MATLAB through UDP packets.

This setup ensures the UI remains responsive while data handling happens in the background. Additionally, a Foreground Service can be used with a persistent notification to prevent the service from being stopped by Android's power management (e.g., Doze Mode). For more reliability, Wake Locks can be applied to keep the CPU awake during critical data reception periods.

The service can also broadcast the received data to the UI thread using Local Broadcast Manager or Handler, ensuring smooth real-time data visualization without interrupting user interactions. This approach is essential for real-time control systems where data delay or loss could disrupt the system's stability [9].

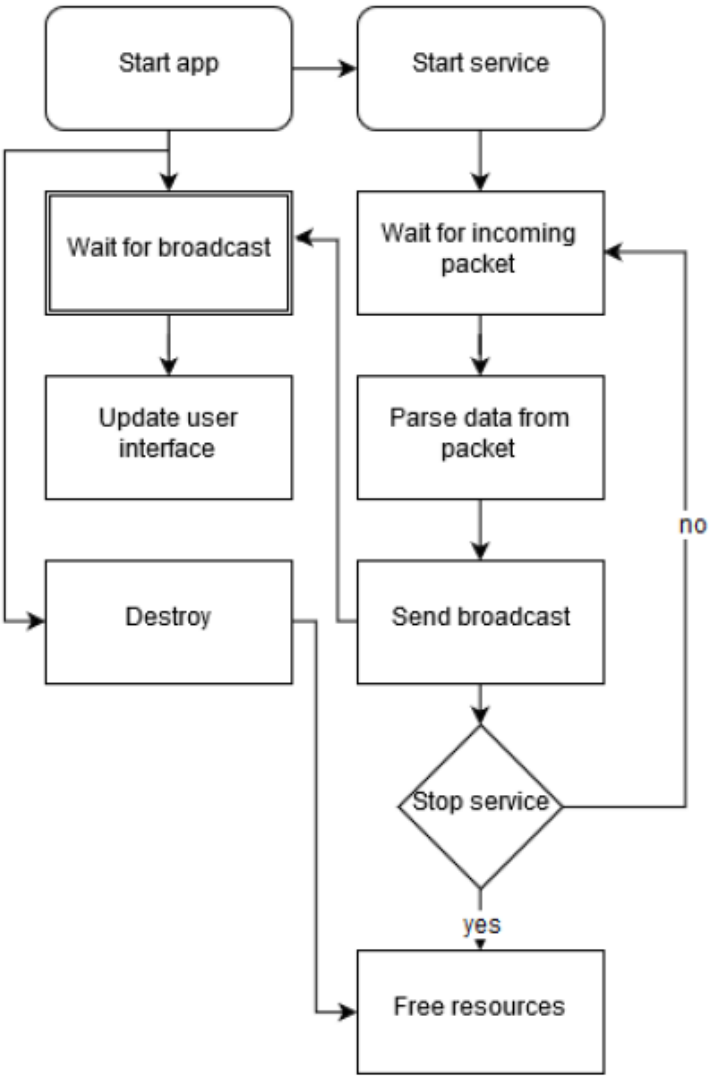

Fig:2 Data receiving on Android

Communication in MatLab

Packet Input/Output blocks in MATLAB/Simulink facilitate UDP communication using binary encoding, enabling data transfer between devices. The block sends data from one computer's UDP port to another device while simultaneously receiving incoming data on a separate UDP port. This setup ensures bidirectional data flow, essential for real-time systems like control applications.

The Standard Devices UDP Protocol is configured as the data acquisition board within the block's parameters. This setup involves three key parameters: the IP address of the remote device, the receiving UDP port, and the sending UDP port. These parameters ensure the correct routing of data packets between devices.

The same UDP configuration is applied to both input and output blocks, ensuring seamless communication. Additionally, packet size and data type configurations (e.g., double, int8, uint16) can be adjusted to match the data format used by the Android application or other connected devices, preventing data corruption [10,11].

CONCLUSION

This article introduces a cost-effective, portable, and low-maintenance approach to developing an image processing Android app. The app communicates with a PC-based simulation or control system via wireless protocols like Wi-Fi or UDP, enabling real-time data exchange while keeping the app lightweight by offloading heavy processing to the PC. It supports remote image processing and result display, making it suitable for medical analysis, object recognition, industrial monitoring, and educational simulations. The user-friendly interface ensures accessibility for non-experts, while the scalable design supports future upgrades - allowing integration with new simulations or machine learning models without hardware modifications.

Author contributions: Heber Anandan contributed to this paper; Anandan H designed the overall concept and outline of the manuscript; Dhanisha JL and Vinitha Babuselvan contributed to the development, design and writing of the manuscript; Tamilselvi Murugaraj and Jenita Manickam contributed to the writing, and editing the manuscript, illustrations, and conclusion.

REFERENCES

1. Na Yang, Yanan Hu. “Design of the Pedometer Based on MATLAB and Android Smartphone Sensor”, December 2017, 978-94-6252-430-9
2. Android, "Sensor Coordinate System"; https://developer.android.com/guide/topics/sens ors/sensors_overview.html /
3. Ma Z, Qiao Y, Lee B, Fallon E. Experimental evaluation of mobile phone sensors. In 24th IET Irish Signals and Systems Conference (ISSC 2013). 2013 doi: 10.1049/ic.2013.0047
4. H. Lu, W. Pan, N. Lane, T. Choudhury, A. Campbell, "Sound Sense: Scalable Sound Sensing for People-Centric Applications on Mobile Phones", Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services (MobiSys), pp. 165-178, 2009.
5. S. O. Madgwick, A. J. L. Harrison, and R. Vaidyanathan, "Estimation of IMU and MARG Orientation Using a Gradient Descent Algorithm", IEEE International Conference on Rehabilitation Robotics, pp. 1-7, 2011.
6. Passaro VMN, Cuccovillo A, Vaiani L, Carlo M, Campanella CE. Gyroscope Technology and Applications: A Review in the Industrial Perspective. Sensors (Basel). 2017 Oct 7;17(10):2284. doi: 10.3390/s17102284. PMID: 28991175; PMCID: PMC5677445.
7. M. Azizyan, I. Constandache, R. Choudhury, "Surround Sense: Mobile Phone Localization via Ambience Fingerprinting", Proceedings of the 15th Annual International Conference on Mobile Computing and Networking (MobiCom), pp. 261-272, 2009.
8. Sriskanthan, N., Tan, F., & Karande, A. (2002). Bluetooth based home automation system. *Microprocess. Microsystems*, *26*, 281-289.
9. Android, "Sensor Manager"; https://developer.android.com/reference/android/hardware/SensorManager.html/
10. J. Yang, S. Sidhom, G. Chandrasekaran, T. Vu, H. Liu, Y. Chen, "Detecting Driver Phone Use Leveraging Car Speakers", Proceedings of the 17th Annual International Conference on Mobile Computing and Networking (MobiCom), pp. 97-108, 2011.
11. J. Mantyjarvi, M. Lindholm, E. Vildjiounaite, S. Makela, and H. Ailisto, "Identifying Users of Portable Devices from Gait Pattern with Accelerometers", Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, pp. 973-976, 2005.Philadelphia, PA, USA, 2005, pp. ii/973-ii/976 Vol. 2, doi: 10.1109/ICASSP.2005.1415569.