

Exploring Wireless Communication and Inter-Chip Connectivity with Vaman Board

A Major Project Report Submitted
In partial fulfillment of the requirement for the award of the degree of

Bachelor of Technology
in
CSE- Artificial Intelligence and Machine Learning

by

K Pavan kumar Goud

21N35A6605

M Arun kumar

20N31A6634

Kakarla Rajinikanth

20N31A6624

Under the esteemed guidance of

U RAKESH

Asst. Professor



Department of Computational Intelligence

Malla Reddy College of Engineering & Technology

(Autonomous Institution- UGC, Govt. of India)

(Affiliated to JNTUH, Hyderabad, Approved by AICTE, NBA & NAAC with 'A' Grade)

Maisammaguda, Kompally, Dhulapally, Secunderabad – 500100

website: www.mrcet.ac.in

2020-2024



Malla Reddy College of Engineering & Technology

(Autonomous Institution- UGC, Govt. of India)

(Affiliated to JNTUH, Hyderabad, Approved by AICTE, NBA & NAAC with 'A' Grade)

Maisammaguda, Kompally, Dhulapally, Secunderabad – 500100

website: www.mrcet.ac.in

CERTIFICATE

This is to certify that this is the bonofide record of the project titled **“Exploring Wireless Communication and Inter-Chip Connectivity with Vaman Board”** submitted by **K Pavan kumar Goud (21N35A6605), M. Arun kumar (20N31A6634), Kakarla Rajinikanth (20N31A6624)** of B.Tech in the partial fulfillment of the requirements for the degree of **Bachelor of Technology** in **CSE-AI&ML** Dept.of Computational Intelligence during the year 2023-2024. The results embodied in this project report have not been submitted to any other university or institute for the award of any degree or diploma.

INTERNAL GUIDE

U Rakesh

Asst.Professor

HEAD OF THE DEPARTMENT

Dr. D. Sujatha

Professor

EXTERNAL EXAMINER

DECLARATION

I hereby declare that the major project entitled “**Exploring Wireless Communication and Inter-Chip Connectivity with Vaman Board**” submitted to **Malla Reddy College of Engineering and Technology**, affiliated to Jawaharlal Nehru Technological University Hyderabad (JNTUH) for the award of the degree of **Bachelor of Technology in Computer Science and Engineering (AIML)** is a result of original research work done by me.

It is further declared that the project report or any part thereof has not been previously submitted to any University or Institute for the award of degree or diploma.

K Pavan Kumar Goud

21N35A6605

M. Arun Kumar

20N31A6634

Kakarla Rajinikanth

20N31A6624

ACKNOWLEDGEMENT

We feel honored and privileged to place our warm salutation to our college Malla Reddy College of Engineering and technology (UGC-Autonomous), our Director ***Dr. VSK Reddy*** who gave us the opportunity to have experience in engineering and profound technical knowledge.

We are indebted to our Principal ***Dr. S. Srinivasa Rao*** for providing us with facilities to do our project and his constant encouragement and moral support which motivated us to move forward with the project.

We would like to express our gratitude to our Head of the Department ***Dr. D. Sujatha*** for encouraging us in every aspect of our application development and helping us realize our full potential.

We would like to thank our application development guide as well as our internal guide ***U Rakesh***, Professor for his regular guidance and constant encouragement. We are extremely grateful for valuable suggestions and unflinching co-operation throughout application development work.

We would also like to thank all supporting staff of department of CI and all other departments who have been helpful directly or indirectly in making our application development a success.

We would like to thank our parents and friends who have helped us with their valuable suggestions and support has been very helpful in various phases of the completion of the application development.

K Pavan Kumar Goud 21N35A6605

M Arun Kumar 20N31A6634

Kakarla Rajinikanth 20N31A6624

ABSTRACT

Exploring Wireless Communication and Inter-Chip Connectivity with Vaman Board

The Vaman Board, an innovative product by Quick Logic, stands at the forefront of wireless communication and inter-chip connectivity technology. This document presents a comprehensive exploration of the Vaman Board's capabilities and applications, showcasing its potential across diverse fields. From facilitating seamless wireless code dumping and Bluetooth control to enabling inter-chip communication through protocols like SPI, the Vaman Board empowers users to unlock new possibilities in integrated systems and applications. With integrations like Termux and ESP32-FPGA modules, alongside the user-friendly Dabble app for wireless control, the Vaman Board emerges as a versatile platform for experimentation and innovation. Through this abstract, we provide insights into the transformative power of the Vaman Board, offering a glimpse into its role as a catalyst for technological advancement and development.

TABLE OF CONTENTS

S.NO	TOPIC	PAGE NO.
1	Introduction	2 – 30
	1.1 Purpose	3
	1.2 Project Features	3 – 4
	1.3 Limitations	5
	1.4 Math computing using Vaman	6 – 7
	1.5 Interchip Communication	8 – 9
	1.6 Speech to Text Converter using ESP32 and INMP441	10 – 12
	1.7 Creating website using ESP32 and ARM microcontroller	13 – 14
	1.8 5G Technology	15
2	System Requirements	30 – 31
	2.1 Hardware Requirements	30
	2.2 Software Requirements	31
3	Technologies Used	32 – 33
4	System Design	34 – 38
	4.1 System Architecture	34 – 36
	4.2 Flowchart	37
	4.3 Sequence Diagram	38
5	Implementation	39 – 62
	5.1 Codes	39 – 59
	5.2 OUTPUT	60 – 62
6	Conclusion	63
7	References	64

CHAPTER 1: INTRODUCTION

- Introduction to Vaman Board

The Vaman Board, a cutting-edge product from Quick logic, represents a leap forward in wireless communication and inter-chip connectivity technology. This documentation aims to provide a comprehensive overview of the Vaman Board's capabilities and applications, showcasing its potential in various fields.

- Wireless Code Dumping and Bluetooth Control

The Vaman Board facilitates seamless wireless code dumping and Bluetooth control, enabling users to effortlessly manage and manipulate connected devices. With its intuitive interface and robust functionality, users can execute commands and operations with ease, enhancing efficiency and productivity.

- Inter-Chip Communication

One of the key features of the Vaman Board is its ability to facilitate inter-chip communication between different boards, fostering collaboration and synergy across platforms. Through protocols like Serial Peripheral Interface (SPI), the Vaman Board empowers users to exchange data and commands seamlessly, opening up a world of possibilities for integrated systems and applications.

- Utilizing Termux and SPI Communication

To further extend the Vaman Board's capabilities, this documentation explores the integration of Termux and SPI communication. By leveraging Termux commands and SPI protocols, users can maximize the potential of the Vaman Board, unlocking advanced functionalities and expanding its utility in diverse environments.

- ESP32 and FPGA Integration

The integration of ESP32 and FPGA modules on the Vaman Board opens up a myriad of opportunities for experimentation and innovation. This section delves into various experiments and applications that leverage the combined power of ESP32 and FPGA, showcasing the versatility and adaptability of the Vaman Board.

- Dabble App and Wireless Control

The inclusion of the Dabble app, developed by STEM pedia, enhances the Vaman Board's usability and accessibility. By harnessing the features of the Dabble app, users can control connected devices wirelessly, utilizing virtual joystick interfaces and real-time control functionalities to streamline operations and enhance user experience.

1.1 PURPOSE

- The primary purpose of the Vaman Board project is to demonstrate the advanced capabilities of wireless communication and inter-chip connectivity in a real-world context, leveraging the unique features of the Vaman Board. This project aims to explore the potential applications of the Vaman Board in various domains, such as robotics, wireless control, and educational tools, by providing hands-on examples and practical applications. The objectives outlined below further delineate the purpose of this project.

1.2 PROJECT FEATURES

The Theft Detection System employing CNN for small retail establishments offers a variety of straightforward yet highly effective security enhancements:

- The Vaman Board project stands out due to its innovative integration of hardware capabilities, wireless communication, and inter-chip connectivity. The following features highlight the project's uniqueness and its potential impact on educational and developmental applications in electronics and wireless communication fields.
- Comprehensive Wireless Communication:
 1. Wireless Code Dumping: Enables users to upload code wirelessly to the Vaman Board, simplifying the development process and enhancing the flexibility of deployment.
 2. Bluetooth Control: Utilizes Bluetooth technology for remote control applications, including robotics and device management, demonstrating the power and versatility of wireless communication.

- **Advanced Inter-Chip Connectivity:**
 1. **Diverse Chip Integration:** Showcases the seamless communication between various chips and modules on the Vaman Board, such as ESP32, ARM, and FPGA, highlighting the board's capability to support complex, multi-platform projects.
 2. **SPI Communication:** Demonstrates the Serial Peripheral Interface (SPI) communication between the Vaman-ESP32 and the onboard FPGA, enabling high-speed data transfer and synchronization between different components of the system.
- **Educational and Developmental Applications:**
 1. **Practical Learning Tool:** Serves as an excellent educational tool for students and hobbyists to learn about wireless communication, programming, and electronics design in a hands-on manner.
 2. **Innovative Experimentation:** Encourages innovative experimentation with technology through real-world applications, such as controlling a mini vehicle wirelessly or solving mathematical problems using the ESP32.
- **User-Friendly Interfaces and Applications:**
 1. **Dabble App Integration:** Leverages the Dabble app for easy and intuitive control of projects via a smartphone, offering features like a virtual joystick for real-time control of robotic devices.
 2. **Termux Support:** Explains how to use Termux commands to run applications on the Vaman Board, making it accessible for users with varying levels of technical expertise.
- **Robotic and Electronic Device Control:**
 1. **Versatile Robotic Applications:** Enables the creation and control of robotic vehicles and electronic devices through simple, user-friendly interfaces, demonstrating the practical applications of the Vaman Board in robotics.
 2. **Customizable Control Options:** Offers customizable control options through the Dabble app, allowing users to tailor the control schemes to their specific projects and needs.

1.3 Limitations

While the Vaman Board project showcases a range of impressive features and applications, there are inherent limitations that users and developers should consider. These limitations highlight areas for future improvement and development, ensuring that the project continues to evolve to meet the needs of its users more effectively.

Hardware Dependency:

- **Specific Hardware Requirements:** The project is designed around the Vaman Board and specific auxiliary components, which may limit accessibility for those without direct access to these specific hardware platforms.
- **Limited Expansion Capabilities:** Given the integrated nature of the Vaman Board, there may be limitations in expanding or customizing the hardware for projects that require capabilities beyond what the board and its components can offer.

Wireless Communication Limitations:

- **Range and Interference:** The effectiveness of wireless communication, including Bluetooth control, can be limited by physical range, obstacles, and interference from other wireless devices, potentially affecting the reliability of remote-control applications.
- **Security Concerns:** Wireless communications are susceptible to security vulnerabilities, including unauthorized access and data interception. The project might need more robust security measures for applications requiring secure communications.

1.4 Math Computing using ESP32

This section demonstrates how to leverage the computational capabilities of the ESP32 microcontroller to solve mathematical problems. Follow the steps below to execute the provided code and interact with the ESP32 firmware:

1. Code Execution:

- Navigate to the `src` directory.
- Execute the `pio run` command to build the ESP32 firmware.

2. Firmware Upload:

- After building the firmware, flash it to the Vaman-ESP board.
- Use the following command:

```
pio run -t nobuild -t upload --upload-port 192.168.210.X
```


Replace `192.168.210.X` with the appropriate IP address.

3. WiFi Connection:

- Once the firmware is flashed, the Vaman-ESP board will connect to the WiFi network using the provided credentials.
- Ensure that your mobile phone is connected to the same WiFi network.

4. Accessing IP Address:

- Determine the IP address assigned to the Vaman-ESP board by running the following commands:

```
ifconfig
```



```
nmap -sP 192.168.x.x/24
```
- Replace `192.168.x.x` with the appropriate IP address range obtained from the first command.

5. Accessing Web Server:

- With the IP address of the Vaman-ESP board identified, open a web browser on your device.
- Enter the IP address of the Vaman-ESP board in the address bar and press Enter.

6. Interacting with Web Interface:

- The website hosted by the Vaman-ESP board will load in the web browser.
- Explore the web interface to interact with the mathematical computations performed by the ESP32.
- The output of the mathematical computations will be displayed on the website, providing insight into the ESP32's computational capabilities.

By following these steps, you can effectively utilize the ESP32 microcontroller for mathematical computations and access the results through a web interface hosted by the Vaman-ESP board.

1.5 Inter-Chip Communication

This section showcases the capability of the LC Vaman platform to facilitate communication between different integrated circuits, including the ESP32 microcontroller, ARM processor, and FPGA, using various communication protocols such as Serial Peripheral Interface (SPI).

ESP32 and FPGA Communication

This subsection details experiments conducted to establish communication between the ESP32 microcontroller and the embedded FPGA (eFPGA) on the LC Vaman platform.

Onboard LED Control

A specific demonstration of inter-chip communication involves toggling the LEDs onboard the Vaman-Pygmy FPGA using SPI communication with the Vaman-ESP32.

Implementation Steps:

1. Build the PlatformIO Project:

- Navigate to the `inter-chip/esp32-fpga/led/codes/esp32` directory.
- Use PlatformIO or ArduinoDroid to compile the project.

2. Flash the ESP32 Firmware:

- Connect a USB-UART interface to the Vaman-ESP32.
- Flash the generated `.bin` file to the Vaman-ESP32.

3. Build the FPGA Project:

- Set the `QORC_SDK_PATH` variable to the path of the Pygmy SDK.
- Navigate to the `inter-chip/esp32-fpga/led/codes/` directory.
- Execute the `make` command to build the FPGA project.

- Upon successful build, the `.bin` file is generated.

4. Flash the FPGA Firmware:

- Place the Vaman-Pygmy FPGA into bootloader mode.
- Execute the following command in a terminal window:

```

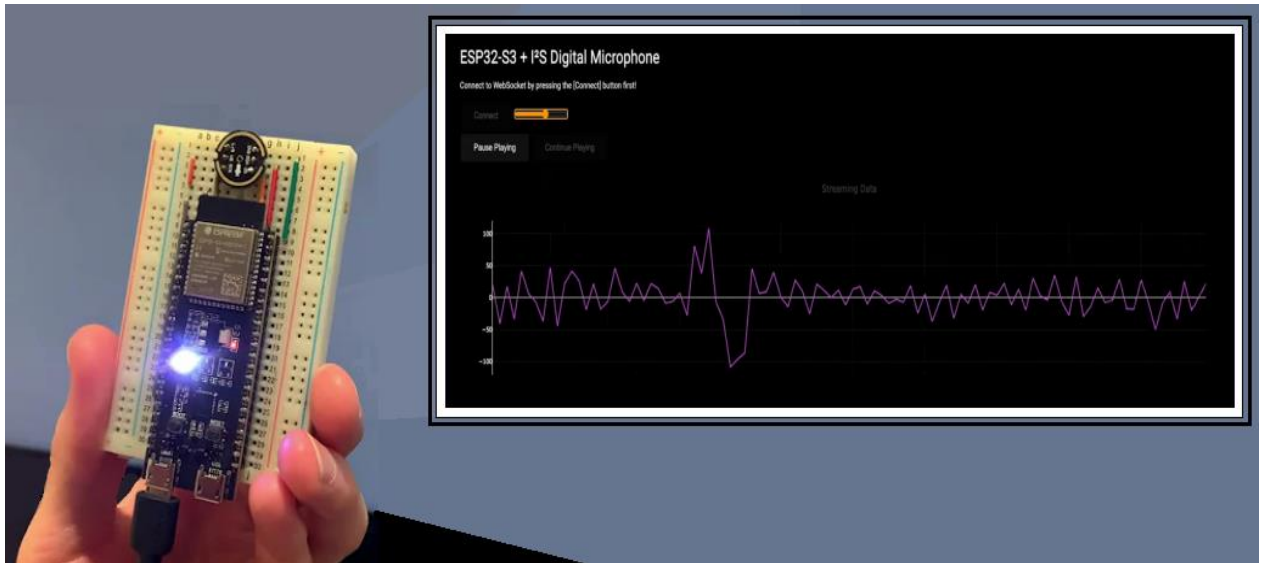
```
python3 /path/to/tinyfpga-programmer.py --mode fpga --app /path/to/AL4S3BFPGA_Top.bin
--reset
```

```

- Ensure to replace `/path/to/` with the appropriate file paths.
- The port at which the Vaman board is available (`/dev/ttyACMxx`) can be obtained using the `ls /dev/ttyACM*` command with root/sudo privileges.

By following these steps, the ESP32 and FPGA on the LC Vaman platform can effectively communicate with each other via SPI, enabling functionalities such as onboard LED control and fostering seamless inter-chip communication

1.6 Speech to Text Converter using ESP32 and INMP441



Introduction:

In the realm of IoT and embedded systems, the integration of voice recognition capabilities has become increasingly prevalent. This project explores the development of a Speech to Text Converter leveraging the ESP32 microcontroller and the INMP441 microphone module. The ESP32, renowned for its versatility and performance, serves as the core processing unit, while the INMP441 microphone provides high-quality audio input, crucial for accurate speech recognition.

About ESP32:

The ESP32 is a powerful microcontroller widely utilized in IoT applications due to its dual-core architecture, onboard Wi-Fi and Bluetooth connectivity, and ample processing capabilities. Its versatility makes it an ideal candidate for projects requiring real-time processing and communication with other devices or the cloud.

About INMP441:

The INMP441 is a low-power, high-performance digital microphone with built-in I2S interface, suitable for capturing clear audio signals. Its small form factor and low power consumption make it suitable for integration into compact electronic devices, making it an excellent choice for voice

recognition applications.

Hardware Setup:

Connect the INMP441 microphone module to the ESP32 microcontroller via the I2S interface.

Ensure proper power supply and signal connections according to the datasheets of both components.

Software Development:

Utilize the ESP32 development environment (e.g., Arduino IDE or PlatformIO) to write and upload the firmware.

Implement the necessary libraries for I2S communication and audio processing.

Integrate a suitable speech recognition library compatible with the ESP32 platform.

Speech Recognition Algorithm:

Develop or integrate a speech recognition algorithm capable of converting audio input from the microphone into text.

Consider factors such as noise cancellation, language support, and accuracy in selecting the algorithm.

Testing and Optimization:

Test the Speech to Text Converter in various environments to assess its performance and accuracy. Optimize the system parameters and algorithms to improve recognition accuracy and reduce processing latency.

Audio Capture:

The INMP441 microphone captures audio signals from the surrounding environment.

The captured audio data is transmitted to the ESP32 microcontroller through the I2S interface.

Speech Recognition:

The ESP32 processes the incoming audio data using the speech recognition algorithm.

The algorithm analyzes the audio input, identifies speech patterns, and converts them into textual output.

Text Output:

The converted text is either displayed on a connected display or transmitted to a designated output device. Users can access the recognized text for further processing or action.

Conclusion:

The Speech to Text Converter utilizing the ESP32 microcontroller and INMP441 microphone module presents a versatile solution for voice-controlled applications in IoT and embedded systems. By harnessing the processing power of the ESP32 and the high-quality audio input of the INMP441, developers can create efficient and accurate speech recognition systems suitable for a variety of use cases, ranging from home automation to industrial automation and beyond. With further refinement and optimization, such systems hold immense potential in enhancing human-machine interaction and advancing the capabilities of smart devices

1.7 Creating website using ESP32 and ARM microcontroller

Understanding the Components:

ESP32: Familiarize yourself with the ESP32 microcontroller's capabilities, pinout, and available libraries for Wi-Fi communication, sensor interfacing, and more.

ARM Microcontroller: Identify the specific ARM microcontroller model on the Vaman board and understand its architecture, peripheral interfaces, and development tools.

Vaman Board: Research the Vaman board's specifications, pin configurations, and any additional features it offers for hardware integration.

Setting Up Development Environment:

Install the necessary software tools and drivers for programming the ESP32 and ARM microcontroller. This include Arduino IDE for ESP32, ARM development toolchains like Keil μ Vision or ARM GCC, and USB drivers for the Vaman board.

Configure the development environment to support both ESP32 and ARM microcontroller development. Ensure you have the required libraries and board definitions for each platform.

Designing the Website:

Creating a detailed plan for your website's functionality and layout. Consider using wireframing tools to sketch out the user interface and interactions.

Determining the data flow between the microcontrollers and the website. Identify the types of data to be exchanged (e.g., sensor readings, control commands) and the communication protocols to be used.

Integrating Microcontrollers with Website:

Develop firmware for the ESP32 to interface with sensors or actuators and establish communication with the web server via Wi-Fi. Utilize libraries such as ArduinoJSON for handling JSON data and libraries for specific sensors or actuators.

Program the ARM microcontroller to perform tasks such as data processing, device control, or interfacing with external peripherals. Implement communication protocols compatible with the ESP32 and the web server.

Building the Backend:

Set up a web server environment using frameworks like Node.js with Express.js or Django with Django REST Framework.

Create RESTful API endpoints to handle incoming requests from the microcontrollers, including routes for receiving sensor data and sending control commands.

Implement middleware for request validation, authentication, and error handling to ensure the security and reliability of the backend.

Implementing Frontend:

Develop the frontend of the website using HTML, CSS, and JavaScript. Use modern frontend frameworks like React.js or Vue.js for building dynamic and responsive user interfaces.

Design UI components to visualize sensor data, display device status, and provide interactive controls for users to interact with connected devices.

Implement client-side logic to fetch data from the backend API endpoints asynchronously and update the UI in real-time using AJAX or WebSocket connections.

Testing and Debugging:

Perform thorough testing of each component of your system, including the microcontroller firmware, backend API endpoints, and frontend user interface.

Use debugging tools and techniques to identify and resolve any issues encountered during testing, such as incorrect data transmission, server errors, or UI glitches.

Deployment:

Deploy your website on a web hosting service or set up a dedicated server for hosting both the frontend and backend components.

Configure security measures such as HTTPS encryption, firewall rules, and access controls to protect your website and connected devices from unauthorized access and attacks.

1.8 5G Technology

Wireless communication has undergone remarkable advancements over the years, culminating in the emergence of 5G technology. Understanding the foundational principles and technical aspects of 5G is essential to comprehend its transformative capabilities. In this comprehensive guide, we delve into the basics of 5G technology, covering its fundamental principles, architecture, and key features.



Foundations of Wireless Communication

Wireless communication is the cornerstone of modern connectivity, enabling the transmission of data without the constraints of physical cables. To understand the evolution to 5G technology, it's imperative to grasp the foundational principles:

- 1. Transmission Mediums and Propagation:** The medium through which signals travel, whether it's air, space, or other materials, significantly impacts signal propagation. Understanding how signals behave in different mediums is vital for optimizing network design and performance.
- 2. Frequency Spectrum Allocation:** Wireless communication relies on allocated frequency bands for transmitting data. Different frequency bands offer varying capacity and performance characteristics. Efficient allocation of spectrum is crucial for maximizing network capacity and accommodating increasing data demands.

3. Modulation Techniques: Modulation schemes convert digital data into analog signals suitable for transmission over wireless channels. Techniques such as amplitude modulation (AM) and frequency modulation (FM) alter the properties of the carrier signal to encode digital information.

4. Signal Encoding and Decoding: Encoding and decoding methods translate digital data into a format suitable for transmission and reception. These processes ensure that data is accurately transmitted and received without errors.

Understanding these foundational concepts lays the groundwork for comprehending the complexities of wireless communication systems and the advancements brought about by 5G technology.

Radio Signal Properties and Factors Affecting Data Rate

Radio signals serve as the medium for transmitting data wirelessly. Exploring their properties and understanding the factors influencing data rate is crucial for optimizing network performance:

1. Frequency and Wavelength: Frequency determines the rate at which a signal oscillates, while wavelength refers to the physical distance between successive peaks of a signal. These properties directly affect signal propagation and coverage area.

2. Amplitude and Phase Modulation: Modulation techniques manipulate the amplitude or phase of the carrier signal to encode digital information. By varying these properties, multiple bits of data can be encoded into a single signal, enhancing spectral efficiency.

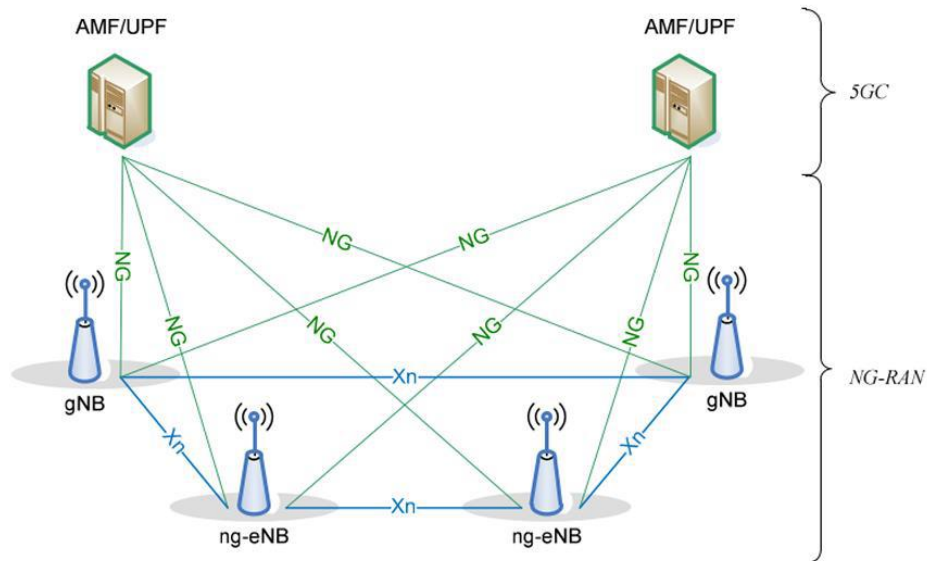
3. Signal Attenuation and Propagation Loss: As signals propagate through the environment, they experience attenuation, leading to a decrease in signal strength. Various factors, including distance, obstacles, and interference, contribute to signal attenuation and propagation loss.

4. Signal-to-Noise Ratio (SNR) and Signal Strength: SNR measures the ratio of signal power to noise power, indicating the quality of the received signal. Signal strength refers to the power of the received signal, influenced by factors such as distance from the transmitter and environmental conditions.

Understanding these properties and factors allows network engineers to design robust wireless communication systems capable of delivering reliable and high-performance connectivity.

General Concepts Behind Cellular Communication and 5G NR Vision

Cellular communication revolutionized connectivity by dividing geographical areas into cells served by base stations. Understanding the concepts behind cellular communication and the vision of 5G NR is essential:



1. Cell Layout and Frequency Reuse: Cellular networks divide coverage areas into cells, each served by a base station. By arranging cells in a grid pattern and employing frequency reuse techniques, cellular networks maximize spectral efficiency and capacity.

2. Handover and Mobility Management: Handover mechanisms ensure seamless transition between cells as mobile devices move within the network. Mobility management protocols facilitate efficient handovers while maintaining uninterrupted connectivity.

3. 5G NR Vision: 5G technology promises transformative capabilities, including ultra-high data rates, ultra-low latency, massive device connectivity, and enhanced reliability. These features enable a wide range of applications, from high-definition video streaming to mission-critical IoT deployments.

Understanding these concepts provides insights into the evolution of wireless communication networks and the revolutionary potential of 5G technology in addressing diverse connectivity requirements.

Why We Need 5G and 5G Layers and Architecture

The proliferation of smart devices, IoT applications, and emerging technologies necessitates the deployment of 5G networks. Additionally, understanding the layers and architecture of 5G is essential:

1. Growing Bandwidth Demand: The increasing usage of bandwidth-intensive applications, such as streaming media and cloud services, drives the need for higher data rates and network capacity. 5G technology addresses these demands by offering enhanced performance and throughput.

2. 5G Layers and Architecture: The architecture of 5G networks comprises both the radio access network (RAN) and the core network. Understanding the layered architecture allows network designers to optimize network performance and scalability.

By comprehensively understanding the drivers behind the adoption of 5G technology and the architectural components that comprise it, stakeholders can effectively deploy, manage, and leverage the transformative capabilities of next-generation networks.

5G Wireless Communication

The architecture of 5G networks is meticulously crafted to cater to the evolving demands of modern communication, promising to revolutionize connectivity across various sectors. At the core of 5G lies the Radio Access Network (RAN), driven by the innovative New Radio (NR) technology. NR operates across both sub-6 GHz and mmWave frequency bands, offering unprecedented data rates, ultra-low latency, and enhanced spectral efficiency. Base Stations, referred to as gNBs (Next-Generation NodeBs), form the backbone of 5G RAN, utilizing advanced antenna technologies like massive MIMO and beamforming to ensure extensive coverage and deliver high-quality wireless connectivity to user devices.

Complementing the RAN is the Core Network (CN), characterized by its Service-Based Architecture (SBA) that enables modular deployment of network functions as scalable services. Network Functions Virtualization (NFV) and Software-Defined Networking (SDN) principles are

integral to the 5G core, facilitating dynamic resource allocation, efficient management, and rapid service deployment. The Core Network comprises essential components such as the User Plane Function (UPF), Session Management Function (SMF), and Authentication and Authorization Function (AAF), ensuring seamless connectivity, mobility management, and secure access for users.

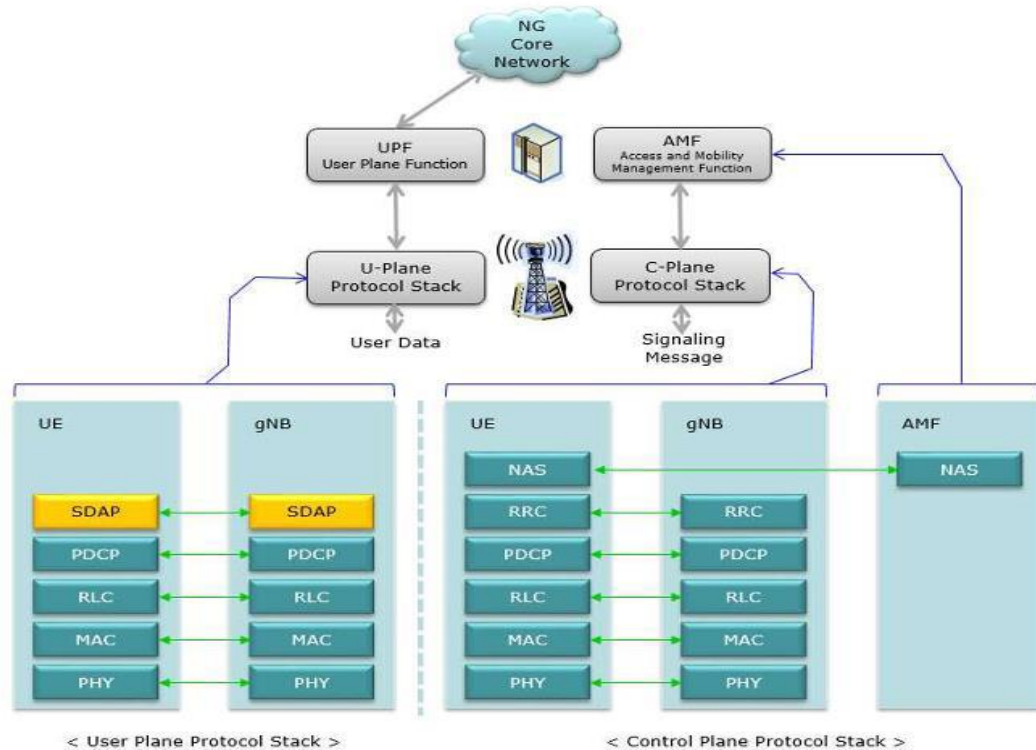
Moreover, 5G architecture incorporates Edge Computing and Mobile Edge Computing (MEC) to bring computation and service execution closer to end-users. MEC platforms deployed at the network edge enable low-latency applications, IoT services, and content caching, unlocking new possibilities for real-time interactions and immersive experiences. Interworking with legacy networks is another hallmark of 5G architecture, ensuring smooth migration paths and backward compatibility with existing LTE and legacy systems.

Central to the management and orchestration of 5G networks are sophisticated network management systems that provide centralized control, automation, and monitoring capabilities. These systems optimize resource allocation, enhance network efficiency, and ensure reliable operation through proactive fault detection and resolution. Security and privacy are paramount in 5G architecture, with robust measures implemented to safeguard against cyber threats, protect user data, and uphold privacy rights.

In summary, the architecture of 5G networks represents a paradigm shift in telecommunications, offering unparalleled performance, flexibility, and scalability to support a myriad of applications and services. By embracing cutting-edge technologies and standards compliance, 5G architecture sets the stage for a connected world where seamless connectivity, immersive experiences, and transformative innovations become the new norm.

ARCHITECTURE:

In the context of 5G architecture, the User Plane and Control Plane represent two distinct components responsible for different aspects of network operation and management.



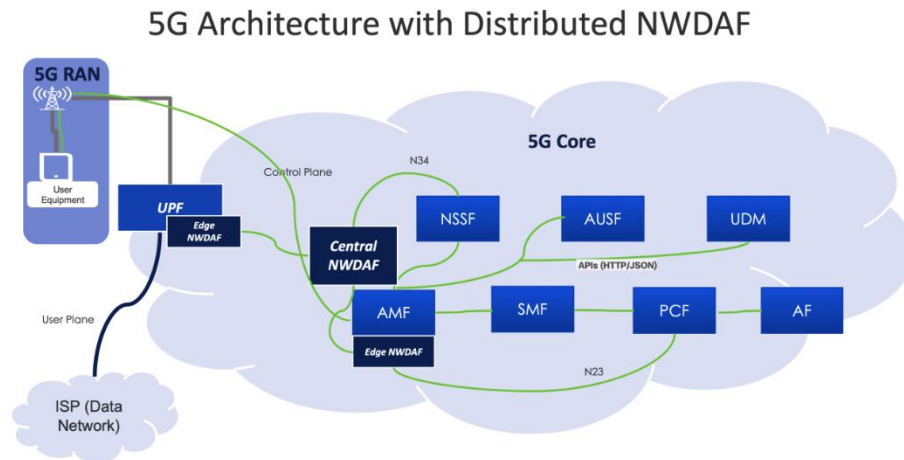
User Plane:

The User Plane, also known as the Data Plane, primarily handles the transmission of user data packets across the network. It focuses on efficient and reliable delivery of data from the source to the destination with minimal latency. Here's a breakdown of key elements within the User Plane:

- 1. UPF (User Plane Function):** The UPF is a critical component of the User Plane responsible for processing and forwarding user data packets. It performs functions such as packet routing, forwarding, and Quality of Service (QoS) enforcement to ensure optimal data delivery.
- 2. User Data:** User data packets contain information generated or requested by end-user devices (UEs), such as smartphones, IoT devices, or computers. These packets traverse the User Plane, undergoing various processing stages before reaching their destination.
- 3. U-Plane Protocol Stack:** The User Plane Protocol Stack comprises layers of protocols responsible for handling user data packets. These layers typically include the Packet Data Convergence Protocol (PDCP), Radio Link Control (RLC), Medium Access Control (MAC), and Physical (PHY) layer. Each layer performs specific functions to facilitate data transmission and reception.

Control Plane:

In contrast, the Control Plane focuses on network control and management functions, including signaling, authentication, and mobility management. It handles the establishment, maintenance, and termination of connections between network elements. Here are the key components of the Control Plane:



- 1. AMF (Access and Mobility Management Function):** The AMF is a vital element of the Control Plane responsible for managing user access and mobility within the network. It handles tasks such as user authentication, mobility management, and session establishment.
- 2. Signaling Messages:** Signaling messages are exchanged between network elements to facilitate control and management functions. These messages carry essential information for establishing connections, managing resources, and maintaining network integrity.
- 3. Control Plane Protocol Stack:** The Control Plane Protocol Stack comprises protocols responsible for handling signaling and control functions within the network. Protocols such as Non-Access Stratum (NAS), Radio Resource Control (RRC), and Service Data Adaptation Protocol (SDAP) are part of this stack. They facilitate procedures related to mobility management, session establishment, and QoS management.

In summary, while the User Plane focuses on the efficient transmission of user data packets, the Control Plane handles network control and management functions to ensure the smooth operation

of the 5G network. By segregating these functionalities, the architecture optimizes resource utilization, enhances network performance, and enables seamless connectivity for end users.

Components in 5G:

User Equipment (UE):

A UE, or User Equipment, refers to the device used by end-users to access cellular networks such as 5G. It encompasses a wide range of devices including smartphones, tablets, IoT gadgets, and more, capable of connecting to the network infrastructure. UE serves as the interface between users and the network, enabling communication, data exchange, and access to various services and applications. It operates across multiple layers of the protocol stack, from the physical layer responsible for transmitting data over the air interface to the application layer providing user-facing functionalities. UE plays a crucial role in facilitating seamless connectivity, supporting high-speed data transmission, low-latency communication, and diverse use cases ranging from mobile broadband to IoT deployments.

Layers in UE/ gNB:

1. PHY (Physical Layer):

- The Physical Layer is the lowest layer of the protocol stack and is responsible for transmitting raw binary data over the air interface.
- It handles tasks such as modulation, coding, and transmission of data bits using techniques like OFDMA (Orthogonal Frequency Division Multiple Access) or SC-FDMA (Single Carrier Frequency Division Multiple Access).
- PHY also manages aspects like power control, channel estimation, and beamforming to optimize the wireless link performance.

2. MAC (Medium Access Control):

- MAC is responsible for managing access to the shared radio resources and scheduling the transmission of data between the UE and the network.
- It coordinates the allocation of resources such as time slots, frequency bands, and power levels to multiple UEs within the cell.
- MAC protocol supports functionalities like random access, contention-based access, and

hybrid automatic repeat request (HARQ) for error recovery.

3. RLC (Radio Link Control):

- RLC operates above the MAC layer and ensures reliable and ordered delivery of packets over the radio interface.

- It performs segmentation and reassembly of data packets to fit into the size of radio frames and retransmits lost or corrupted packets using Automatic Repeat request (ARQ) mechanisms.

4. PDCP (Packet Data Convergence Protocol):

- PDCP operates above the RLC layer and provides various functionalities such as header compression, encryption, and integrity protection.

- It compresses the IP header to reduce overhead and optimize the transmission efficiency, especially for small data packets.

5. SDAP (Service Data Adaptation Protocol):

- SDAP is a new addition in the 5G protocol stack

and is responsible for adapting different types of service data to the requirements of the underlying layers.

- It provides service-specific QoS (Quality of Service) handling and mapping of traffic flows to appropriate QoS bearers.

gNodeB:

G node Band base stations are integral components of cellular networks, each serving as a pivotal link between user equipment (UE) and the network infrastructure. While traditional base stations have been utilized in previous generations of mobile networks like 4G LTE, g Node Bs represent the evolution of base station technology specifically tailored for 5G networks. Both g Node Bs and base stations are responsible for establishing and managing the radio interface, facilitating communication between UEs and the core network. They employ advanced radio technologies such as OFDMA (Orthogonal Frequency Division Multiple Access) and MIMO (Multiple-Input Multiple-Output) to optimize spectral efficiency, enhance coverage, and increase data rates.

Additionally, both g Node Bs and base stations support features like beamforming, which concentrates radio signals towards specific UEs, improving signal quality and capacity. However, g Node Bs typically offer enhanced capabilities compared to traditional base stations, such as support for network slicing, low latency communication, and cloud-native architectures, making them essential components for delivering the full potential of 5G networks.

RRC (Radio Resource Control):

In the 5G protocol stack, the Radio Resource Control (RRC) is a key protocol that resides at the third layer, known as the RRC layer. The RRC protocol is responsible for the control of the radio resources between the User Equipment (UE) and the Radio Access Network (RAN), which includes the gNodeB (gNB) in the 5G context.

Here's a brief overview of the RRC at level 3 in the 5G protocol stack:

- 1. Connection Establishment and Release:** The RRC protocol manages the establishment, maintenance, and release of the connection between the UE and the gNB. This includes procedures like initial access, handover, and connection re-establishment.
- 2. Radio Bearer Control:** RRC controls the setup, modification, and release of radio bearers, which are logical channels used to transfer user and control data between the UE and the gNB.
- 3. Mobility Procedures:** RRC manages mobility-related procedures such as handover, which involves transferring the UE's connection from one gNB to another without interrupting the ongoing services.
- 4. Quality of Service (QoS) Management:** The RRC protocol is responsible for managing and maintaining the QoS parameters, ensuring that the required quality of service is provided to the UE based on its requirements and network conditions.
- 5. Security:** RRC handles the security-related procedures, including authentication, ciphering, and integrity protection, to ensure the confidentiality and integrity of the data transmitted between the UE and the gNB.
- 6. System Information Broadcast:** RRC is responsible for broadcasting system information to the UEs, which includes information about the network configuration, available services, and cell parameters.

7. UE Capability Negotiation: RRC facilitates the negotiation and exchange of capabilities between the UE and the gNB, ensuring that both entities are aware of each other's capabilities and can operate efficiently.

NAS (Non-Access Stratum):

1. NAS – Non Access Stratum Registration and Mobility Management: The NAS layer manages the registration of the UE with the core network and handles mobility-related procedures such as tracking area updates, location updates, and roaming.

2. Session Management: NAS is responsible for managing the EPS (Evolved Packet System) sessions, including the activation, modification, and deactivation of the bearers for the data transfer between the UE and the core network.

3. Security Management: The NAS layer handles security-related functions, including authentication, key management, ciphering, and integrity protection, to ensure the confidentiality, integrity, and authenticity of the communication between the UE and the core network.

4. Paging and Notification: NAS manages the paging and notification procedures, allowing the core network to contact the UE when there is incoming data or signaling messages.

5. SMS and Supplementary Services: NAS supports Short Message Service (SMS) and supplementary services, allowing the UE to send and receive SMS messages and access additional services provided by the core network.

6. Service Request and Release: NAS handles the procedures for initiating, modifying, and terminating services requested by the UE, ensuring smooth communication and resource utilization within the network.

7. Network Selection and Connection Establishment: The NAS layer facilitates the selection of the appropriate core network and the establishment of connections with the selected network, based on the UE's location, preferences, and network conditions.

8. User Identity Handling: NAS manages the handling and protection of the user's identity information, ensuring privacy and security when the UE communicates with the core network.

Overall, the NAS layer plays a crucial role in managing the signaling and control functions for the core network connectivity and services in the 5G network. It ensures seamless communication, mobility, security, and service provisioning for the User Equipment (UE) within the network.

VAMAN TESTING PHASE

In the realm of embedded systems development, where efficiency, security, and reliability are paramount, the Vaman Testing Team stands as a stalwart guardian of integrity. Comprising dedicated experts in software engineering, and system optimization, it is tasked with ensuring the seamless and secure operation of the Vaman board – a cutting-edge amalgamation of ESP32 and Pygmy platforms.

It lies with the imperative to facilitate secure code execution on the Vaman board, a multifaceted device designed for a myriad of applications, including wireless communication, inter-chip connectivity, and real-time data processing. With the ESP32 and Pygmy coexisting on a single platform, optimizing memory management, mitigating vulnerabilities, and fortifying cryptographic protocols to safeguard against potential threats.

The Vaman Testing Team operates at the intersection of innovation and security, leveraging our collective expertise to conduct rigorous assessments and implement robust solutions. Our workflow encompasses several key aspects:

First and foremost, comprehensive analyses of the ESP32 and Pygmy platforms to identify potential vulnerabilities and areas for optimization.

Furthermore, in the development of tailored optimization strategies to maximize the efficiency and reliability of code execution on the Vaman board. This involves fine-tuning memory allocation, optimizing data structures, and implementing secure coding practices to fortify the integrity of embedded software.

In addition to proactive approach to system optimization, the Vaman Testing excels in vulnerability assessment and threat modeling. Through meticulous testing and simulation of potential attack scenarios. In essence, the Vaman Testing serves as a beacon of assurance in the realm of embedded systems security.

The Vaman testing team plays a pivotal role in ensuring the security and reliability of

cryptographic algorithms, particularly focusing on the evaluation of the ZUC algorithm. Also conduct comprehensive assessments of ZUC's strength and vulnerabilities.

One of the main roles of the Vaman testing team is to perform thorough testing and analysis of the ZUC algorithm. This involves simulating various attack scenarios, such as differential cryptanalysis, related-key attacks, and side-channel attacks, to identify potential weaknesses and vulnerabilities. By meticulously examining ZUC under different conditions, we aim to uncover any security flaws that could be exploited by malicious actors.

Furthermore, the Vaman testing team is tasked with providing valuable insights and recommendations based on our findings. We work closely with cryptographic experts and developers to communicate our observations and suggest potential improvements to enhance the security of the ZUC algorithm. Our analyses contribute to the ongoing refinement and optimization of cryptographic protocols, ultimately strengthening the security posture of telecommunications systems that rely on ZUC.

In addition to evaluating ZUC's security, the Vaman testing team also plays a crucial role in validating its performance and compatibility with existing systems. We conduct extensive testing to ensure that ZUC operates efficiently and seamlessly within telecommunications infrastructure. This involves assessing factors such as computational efficiency, resource utilization, and interoperability with other cryptographic components.

The ZUC algorithm is a stream cipher used for encryption, mainly in the field of telecommunications, particularly in 4G and 5G mobile networks. It was developed as part of the Chinese National Standard for Wireless LAN WAPI (WLAN Authentication and Privacy Infrastructure). ZUC stands for "ZU Chongzhi," named after the Chinese mathematician Chongzhi Zu.

In simple terms, the ZUC algorithm works by generating a stream of pseudorandom bits, known as a keystream, based on a secret key and an initialization vector (IV). This keystream is then combined with the plaintext data using bitwise XOR operation to produce encrypted ciphertext. The ZUC algorithm is designed to provide confidentiality and integrity for wireless communications. It ensures that data transmitted over the airwaves remains secure from eavesdroppers and unauthorized access. It achieves this by generating a unique keystream for each data packet, making it difficult for attackers to predict or decipher the encrypted messages without knowledge of the secret key.

Overall, ZUC is a crucial component in ensuring the security of mobile communication networks, protecting sensitive information exchanged between devices and base stations from potential threats and unauthorized interception.

ZUC

The stream cipher ZUC is a world-oriented stream cipher, taking a 128-bit secret key and a 128-bit IV as input, and outputs a keystream of 32-bit words, which is used to encrypt or decrypt the data.

Stream cipher ZUC plays a crucial role in the next generation of mobile communication as it has already been included by the 3GPP LTE-Advanced, which is a candidate standard for the 4G network. Through a long-time evaluation program, ZUC algorithm is thought to be robust enough to resist many existing cryptanalyses, but not for DPA, one of the most powerful threat of SCAs(Side Channel Analysis).Up to the present, almost all the work on DPA is for block ciphers, such as DES and AES, a very few work has been done on stream ciphers, such as ZUC algorithm,

for particular reasons that would be illustrated in the later section. In this paper, we generally study the security of unprotected ZUC hardware implementation against DPA. Our theoretical analysis and experimental results show that ZUC algorithm is potentially vulnerable to this kind of attack. Furthermore, kinds of common countermeasures are discussed when we try to apply them to ZUC hardware implementations, both the security and tradeoffs are considered. The experiments are given in the last section to verify our conclusions, which would undoubtedly provide some guidance to the corresponding designers.

The execution of ZUC is composed of two stages: the initialization stage and working stage. During the initialization stage, the cipher algorithm runs the following operations 32 times to finish the initialization:

1. *Bitreorganization()* ;
2. $W = F(X0, X1, X2)$;
3. *LFSRWithInitialisationMode(w>>1)*.

After the initialization stage, the algorithm moves into the working stage. At the beginning of this stage, the algorithm executes the following operations once, and discards the output W of nonlinear function F :

1. *Bitreorganization()* ;
2. $W = F(X0, X1, X2)$;
3. *LFSRWithInitialisationMode()*.

Then the algorithm goes into the stage of producing keystream, i.e., for each iteration, the following operations are executed once, and a 32-bit word Z is produced as an output:

1. *Bitreorganization()* ;
2. $Z = F(X0, X1, X2) \oplus X3$;
3. *LFSRWithInitialisationMode()*.

CHAPTER 2: SYSTEM REQUIREMENTS

2.1 HARDWARE REQUIREMENTS

- Vama board: A combination of Vaman-ESP32 and Vaman-Pygmy boards is required. These boards facilitate the core functionality, including wireless communication and inter-chip connectivity.
- Computer or Development Environment: A computer running Windows, MacOS, or Linux with sufficient processing power and RAM to support development environments such as PlatformIO or Arduino IDE.
- US-UART Adapter: Needed for programming the Vaman-ESP32 and Vaman-Pygmy boards.
- L293D Motor Driver IC: For projects involving motor control, such as the mini vehicle controlled via Bluetooth.
- Wifi Router: For projects requiring a wireless connection, a stable WiFi network is necessary.

2.2 SOFTWARE REQUIREMENTS

- The application1. Development Tools:
 - PlatformIO or Arduino IDE: For compiling and uploading firmware to the Vaman boards.
 - Python: Required for certain tools and scripts, such as the FPGA programming tool.
- Dabble App:
 - An Android application developed by STEMpedia, used for Bluetooth control and accessing various modules like the virtual joystick.
- Termux App (Optional):
 - For running command-line tools on Android devices, facilitating a portable development environment.
- Web Browser:
 - Any modern web browser (e.g., Chrome, Firefox, Safari) for accessing the web server hosted on the Vaman-ESP32.
- Network Tools:
 - Tools like ``ifconfig`` (Linux/Mac) or ``ipconfig`` (Windows) and ``nmap`` for identifying the IP address of the Vaman-ESP32 within the netwo

CHAPTER 3: TECHNOLOGIES USED

- **Vaman Board:**

- The Vaman board serves as the primary hardware platform for executing the projects. It comprises the Vaman-ESP32 and Vaman-Pygmy boards, offering capabilities for wireless communication, inter-chip connectivity, and interfacing with peripherals.

- **ESP32 Microcontroller:**

- The ESP32 microcontroller is the core processing unit on the Vaman-ESP32 board. It provides WiFi and Bluetooth connectivity, making it suitable for IoT applications and wireless communication protocols.

- **ARM Processor:**

- The ARM processor, likely present in the Vaman-Pygmy board, facilitates additional computational tasks and acts as an auxiliary processing unit in inter-chip communication scenarios.

- **FPGA (Field-Programmable Gate Array):**

- The FPGA on the Vaman-Pygmy board offers programmable logic resources, enabling developers to implement custom digital circuits and perform hardware-accelerated computations.

- **Serial Peripheral Interface (SPI):**

- SPI is a synchronous serial communication protocol used for transmitting data between the ESP32 microcontroller and the FPGA. It facilitates high-speed data transfer and configuration between the two chips.

- **PlatformIO and Arduino IDE:**

- PlatformIO and Arduino IDE are integrated development environments (IDEs) used for writing, compiling, and uploading firmware to the Vaman boards. They offer convenient toolchains and libraries for embedded development.

- **C and C++:**

- C and C++ programming languages are used extensively for firmware development on the Vaman boards. They offer low-level control and efficient resource utilization, making them ideal for embedded systems programming.

- **Dabble App:**

- Dabble is an Android application developed by STEMpedia, providing a user-friendly interface for controlling electronic devices, including the Vaman board, via Bluetooth. It offers features like virtual joystick control and sensor monitoring.

- **Networking Tools (ifconfig, nmap):**

- Networking tools such as ifconfig and nmap are used to identify and configure network settings, allowing seamless communication between the Vaman board and external devices over WiFi.

- **Web Browser:**

- Standard web browsers are employed for accessing the web server hosted on the Vaman-ESP32, enabling users to interact with web-based interfaces and retrieve data from connected sensors or peripherals.

CHAPTER 4: SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

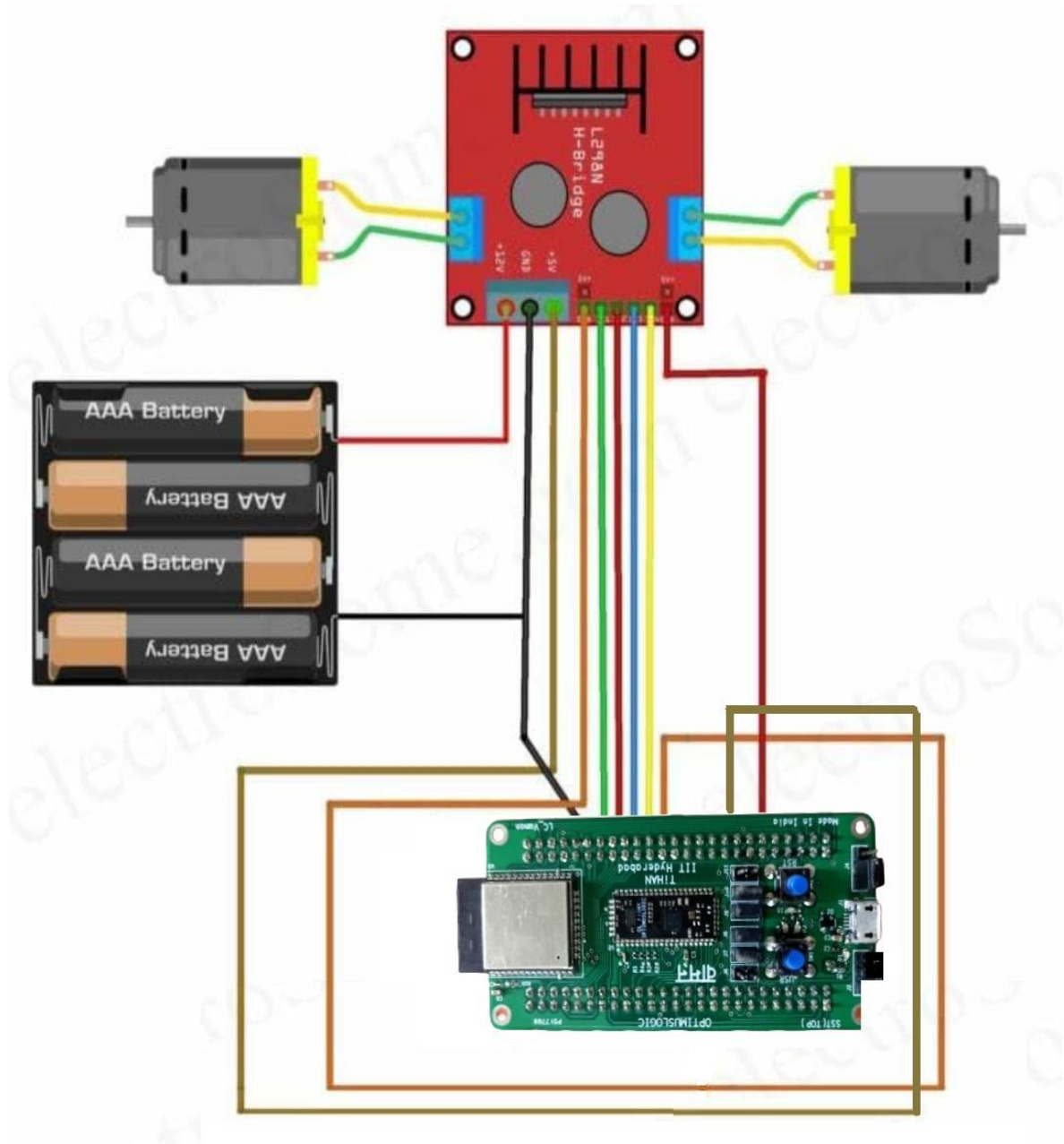


FIG 1: System Architecture

FIG 2: Vaman PIN Diagram

VAMAN LC-1

PINOUT

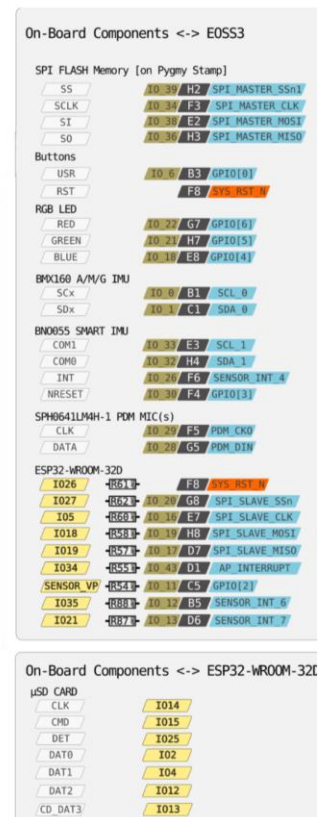
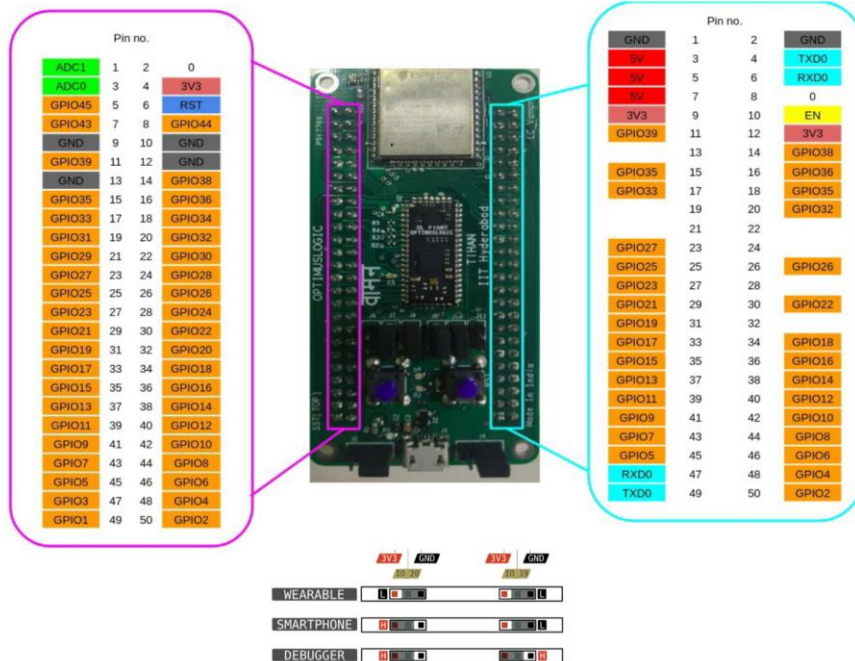
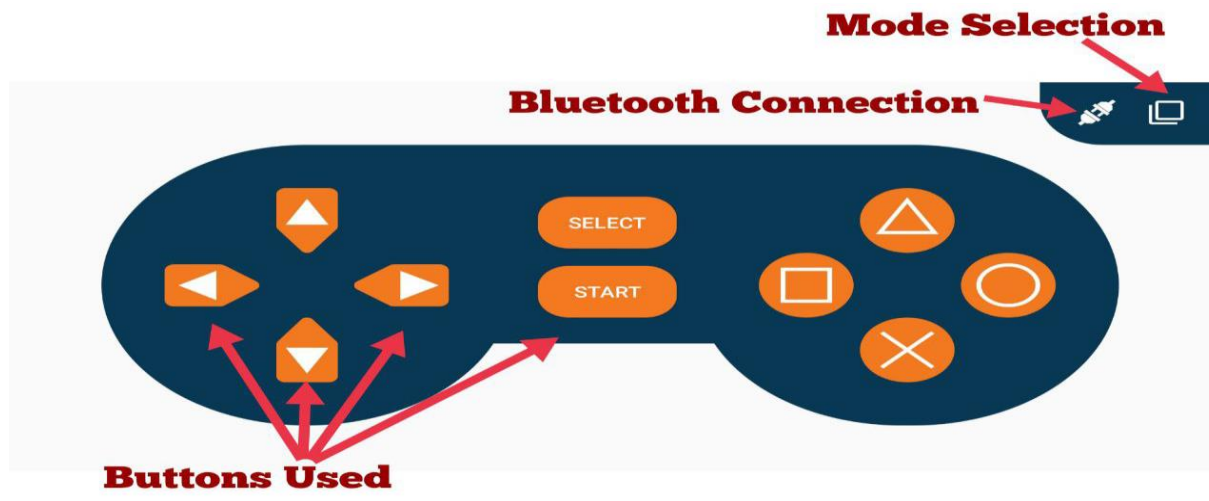


FIG3: Dabble Controller



4.2 FLOWCHART

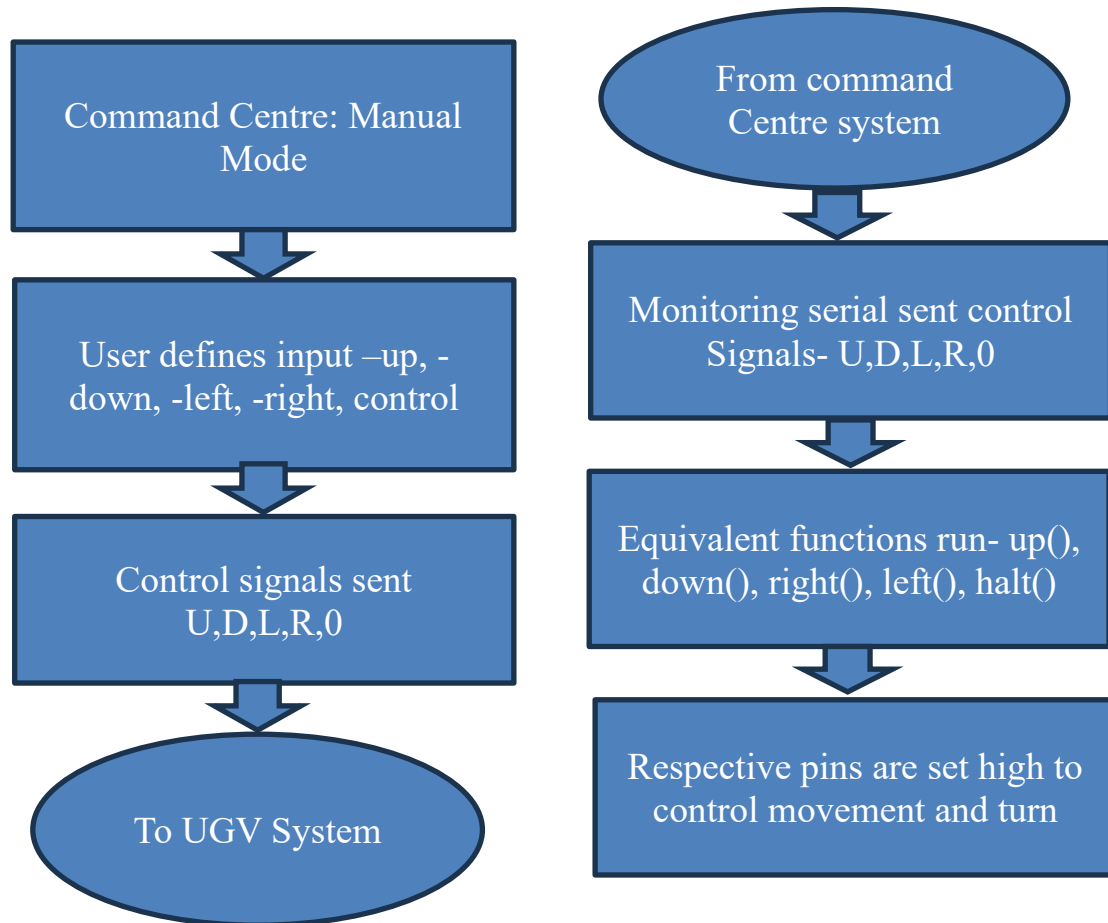


FIG 4: Flow Chart

4.3 SEQUENCE DIAGRAM

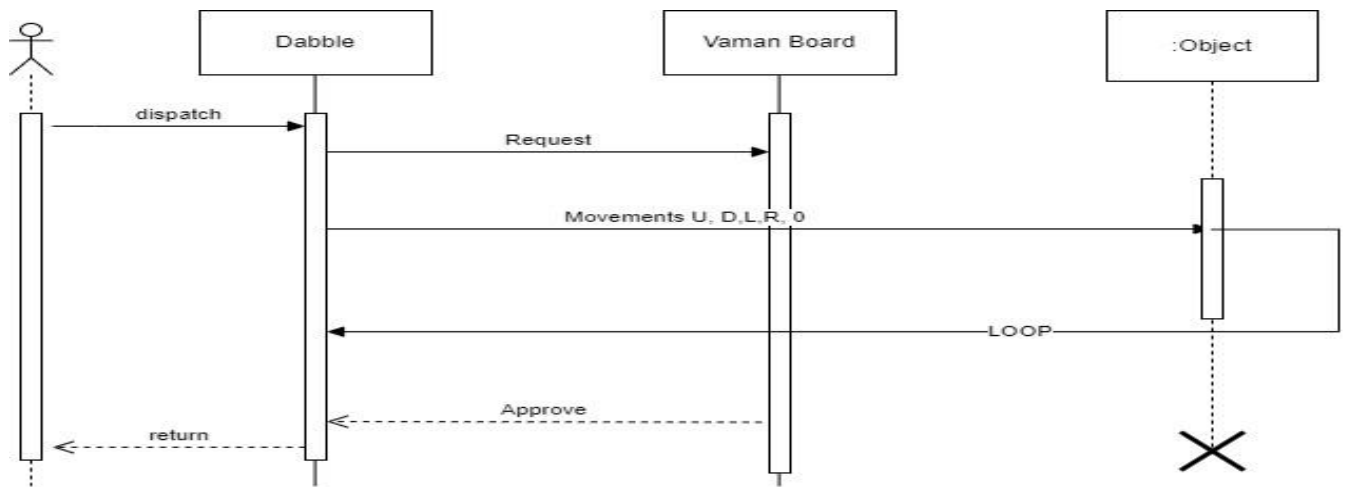


FIG 5: Sequence diagram

CHAPTER 5: IMPLEMENTATION

5.1 CODE

// LED ON/OFF CODE

```
#include <Arduino.h>
#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include "esp32_eoss3_spi.h"
#include "credentials.h"
#define GPIO_OUTPUT_VAL_REG 0x40021004
#define GPIO_OUTPUT_DIR_REG 0x40021008
#define PIN_BLUE 18
#define PIN_GREEN 21
#define PIN_RED 22
#define PIN_ALL (1<<PIN_GREEN) | (1<<PIN_BLUE) | (1<<PIN_RED)
AsyncWebServer server(80);
const char* PARAM_STATE = "state";
const char index_html[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML><html><head>
<title>Vaman LED Form</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
</head>
<body>
<h1>LED State</h1>
<form action="/led" method="POST">
<input type="radio" value="ON" name="state">
<label for="ON">ON</label>
<input type="radio" value="OFF" name="state">
<label for="OFF">OFF</label>
<input type="submit" value="Submit">
</form>
</body></html>)rawliteral";
void notFound(AsyncWebServerRequest *request) {
```

```

request->send(404, "text/plain", "Not found");
}

void setup() {
  esp32_eoss3_spi_init();
  uint32_t dirval = (1<<PIN_GREEN) | (1<<PIN_BLUE) | (1<<PIN_RED);
  uint32_t gpioval = 0;
  esp32_eoss3_spi_ahb_write(GPIO_OUTPUT_DIR_REG, (uint8_t *)&dirval, 4);
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(2000);
  }
  Serial.println();
  Serial.print("IP Address: ");
  Serial.println(WiFi.localIP());
  server.on("/led", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/html", index_html);
  });
  server.on("/led", HTTP_POST, [](AsyncWebServerRequest *request) {
    int params = request->params();
    for(int i=0;i<params;i++){
      AsyncWebParameter* p = request->getParam(i);
      if(p->isPost()){
        if (p->name() == PARAM_STATE) {
          if (p->value() == "ON") gpioval |= (PIN_ALL);
          else if (p->value() == "OFF") gpioval &= ~(PIN_ALL);
          Serial.println(gpioval, HEX);
          esp32_eoss3_spi_ahb_write(GPIO_OUTPUT_VAL_REG, (uint8_t *)&gpioval,
4);

```

```
    }  
  }  
}  
request->send(200, "text/html", index_html);  
  
});  
server.onNotFound(notFound);  
server.begin();  
}  
void loop() {}
```

// UGV CODE

```
#include <Arduino.h>  
#include <WiFi.h>  
#include <esp32PWMTilities.h>  
#include <DabbleESP32.h>  
Motor Motor1;  
Motor Motor2;  
const int irSensorPin = 2; // Assuming your IR sensor is connected to GPIO pin 2  
const int extraPin = 5; // Example pin to stay high  
void _setup() {  
  Motor1.attach(14, 16, 17);  
  Motor2.attach(15, 18, 19);  
  pinMode(irSensorPin, INPUT);  
  pinMode(extraPin, OUTPUT); // Set extra pin as output  
  digitalWrite(extraPin, HIGH); // Set extra pin high  
  Dabble.begin("MyEsp32");  
}  
void _loop() {  
  Dabble.processInput();  
}  
void setup() {
```

```
_setup();
}
void loop() {
  _loop();
  if (GamePad.isPressed(0)) {
    // Moving forward
    if (digitalRead(irSensorPin) == HIGH) {

      // Obstacle detected, stop the motors
      Motor1.lockMotor();
      Motor2.lockMotor();
    } else {
      // No obstacle detected, continue moving forward
      Motor1.moveMotor(2.55 * 100);
      Motor2.moveMotor(2.55 * 100);
    }
  } else if (GamePad.isPressed(1)) {
    // Moving backward

    Motor1.moveMotor(-2.55 * 100);
    Motor2.moveMotor(-2.55 * 100);
  } else if (GamePad.isPressed(3)) {
    // Turning left
    Motor1.moveMotor(2.55 * 100);
    Motor2.moveMotor(-2.55 * 100);
  } else if (GamePad.isPressed(2)) {
    // Turning right
    Motor1.moveMotor(-2.55 * 100);
    Motor2.moveMotor(2.55 * 100);
  } else {
    // If no GamePad button is pressed, stop the motors
    Motor1.lockMotor();
    Motor2.lockMotor();
  }
}
```

```
}  
}  
  
// TRIANGLE CALCULATOR CODE USING VAMAN  
#include <ArduinoOTA.h>  
#ifdef ESP32  
  #include <WiFi.h>  
  #include <AsyncTCP.h>  
#else  
  #include <ESP8266WiFi.h>  
  
  #include <ESPAsyncTCP.h>  
#endif  
#include <ESPAsyncWebServer.h>  
#include "libs/matfun.h"  
AsyncWebServer server(80);  
const char* ssid = "Iqooneo6";  
const char* password = "gps93093";  
const char* input_parameter00 = "input00";  
const char* input_parameter01 = "input01";  
const char* input_parameter10 = "input10";  
const char* input_parameter11 = "input11";  
const char* input_parameter20 = "input20";  
const char* input_parameter21 = "input21";  
const char index_html[] PROGMEM = R"rawliteral(  
  
<!DOCTYPE HTML><html><head>  
  <title>Angle Bisector</title>  
  <meta name="viewport" content="width=device-width, initial-scale=1">  
  <style>  
    html {font-family: Times New Roman; display: inline-block;}  
    h2 {font-size: 2.0rem; color: blue;}  
  </style>
```



```

<script>
    function updateResults(results) {
        document.getElementById("results").innerHTML = results;
    }
</script>
</head><body>
<h2>TO Find the Incenter and Inradius</h2>
<p>Enter the values of points A, B, C</p>
<form id="triangleForm" onsubmit="submitForm(); return false;">
    Enter the values of Point A: <input type="number" value="1"
        name="input00"><input type="number" value = "-1" name="input01"><br>
    Enter the values of Point B: <input type="number" value="-4"
        name="input10"><input type="number" value="6" name="input11"><br>

    Enter the values of Point C: <input type="number" value="-3"
        name="input20"><input type="number" value="-5" name="input21"><br>
    <input type="submit" value="Submit">
</form><br>
<div id="results"></div>
<br><a href="/">Return to Home Page</a>
<script>
    function submitForm() {
        var formData = new FormData(document.getElementById("triangleForm"));
        fetch('/get', { method: 'POST', body: formData })
            .then(response => response.text())
            .then(results => updateResults(results));
    }
</script>
</body></html>rawliteral";
void notFound(AsyncWebServerRequest *request) {
    request->send(404, "text/plain", "Not found");
}

```

```

void setup() {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  if (WiFi.waitForConnectResult() != WL_CONNECTED) {
    Serial.println("Connecting...");
    return;
  }
  Serial.println();
  Serial.print("IP Address: ");
  Serial.println(WiFi.localIP());
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send_P(200, "text/html", index_html);
  });

  server.on("/get", HTTP_POST, [] (AsyncWebServerRequest *request) {
    double Ax = request->arg(input_parameter00).toDouble();
    double Ay = request->arg(input_parameter01).toDouble();
    double Bx = request->arg(input_parameter10).toDouble();
    double By = request->arg(input_parameter11).toDouble();
    double Cx = request->arg(input_parameter20).toDouble();
    double Cy = request->arg(input_parameter21).toDouble();
    double A, B, C,eigen,a,b,c, D, E, F;
    int m1 = 2, n1 = 1;
    A = createMat(m1, n1);
    B = createMat(m1, n1);
    C = createMat(m1, n1);
    eigen = createMat(m1, n1);
    A[0][0] = Ax;
    A[1][0] = Ay;
    B[0][0] = Bx;
    B[1][0] = By;
    C[0][0] = Cx;

```

```

C[1][0] = Cy
double diff_AB = Matsub(B, A, 2, 1);
double distance_AB = Matnorm(diff_AB, 2);
double diff_AC = Matsub(C, A, 2, 1);
double distance_AC = Matnorm(diff_AC, 2);
double diff_BC = Matsub(C, B, 2, 1);

double distance_BC = Matnorm(diff_BC, 2);
a = distance_BC;
b = distance_AC;
c = distance_AB;
double l1 = (a + c - b) / 2;
double l2 = (a + b - c) / 2;
double l3 = (c + b - a) / 2;
D = Matscale(Matadd(Matscale(C, 2, 1, l1), Matscale(B, 2, 1, l2), 2, 1), 2, 1, 1.0 /
(l1 + l2));
E = Matscale(Matadd(Matscale(A, 2, 1, l2), Matscale(C, 2, 1, l3), 2, 1), 2, 1, 1.0 /
(l2 + l3));
F = Matscale(Matadd(Matscale(B, 2, 1, l3), Matscale(A, 2, 1, l1), 2, 1), 2, 1, 1.0 /
(l3 + l1));

double temp1 = Matscale(A, 2, 1, a);
double temp2 = Matscale(B, 2, 1, b);
double temp3 = Matscale(C, 2, 1, c);
double eigenvalues = Matadd(Matadd(temp1, temp2, 2, 1), temp3, 2, 1);
double eigendenominator = a + b + c;
eigen[0][0] = eigenvalues[0][0] / eigendenominator;
eigen[1][0] = eigenvalues[1][0] / eigendenominator;
String result =
"D: " + String(D[0][0],2) + ", " + String(D[1][0],2) + "<br>"
"E: " + String(E[0][0],2) + ", " + String(E[1][0],2) + "<br>"
"F: " + String(F[0][0],2) + ", " + String(F[1][0],2) + "<br>"
"distance of AB: " + c + "<br>"
"distance of BC: " + a + "<br>"

```

```

    "distance of CA: " + b + "<br>"
    "InCenter:" + String(eigen[0][0],2) + ", " + String(eigen[1][0], 2) + "<br>"
    "."
    request->send(200, "text/html", result);
});
server.onNotFound(notFound);
server.begin();
}
void loop() {
}

```

// ZUC Algorithm code

```

#include <iostream>
#include <vector>
#include <iostream>
#include <fstream>
#include <sstream> // For std::stringstream
#include <string>
#include "ZUC.h"

int main(int argc, char *argv[]){

    uint32_t COUNT = 0x2738cdaa;
    uint32_t BEARER = 0x1a;
    uint32_t DIRECTION = 0x0;
    uint32_t LENGTH = 4019;

    uint32_t L = (LENGTH + 31) / 32;

    //uint32_t C[512];

    //unsigned char iv[16] = {0x84, 0x31, 0x9a, 0xa8, 0xde,
    0x69, 0x15, 0xca, 0x1f, 0x6b, 0xda, 0x6b, 0xfb, 0xd8, 0xc7,
    0x66};
    unsigned char iv[16];

    unsigned char key[16] = {0xe1, 0x3f, 0xed, 0x21, 0xb4,

```

```

0x6e, 0x4e, 0x7e, 0xc3, 0x12, 0x53, 0xb2, 0xbb, 0x17, 0xb3,
0xe0};

    //uint32_t* z;

    //z = new uint32_t[L]; // Use new instead of malloc in
C++

    iv[0] = (COUNT >> 24) & 0xFF;
    iv[1] = (COUNT >> 16) & 0xFF;
    iv[2] = (COUNT >> 8) & 0xFF;
    iv[3] = COUNT & 0xFF;
    iv[4] = ((BEARER << 3) | ((DIRECTION & 1) << 2)) &
0xFC;
    iv[5] = 0;
    iv[6] = 0;
    iv[7] = 0;
    iv[8] = iv[0];
    iv[9] = iv[1];
    iv[10] = iv[2];
    iv[11] = iv[3];
    iv[12] = iv[4];
    iv[13] = iv[5];
    iv[14] = iv[6];
    iv[15] = iv[7];

    //std::string key_string;
    //std::cout << "Enter key (16 symbols only):\n>>";

    //getline(std::cin, key_string);
    //if(key_string.size() != 16){
        //std::cout << "Bad key size:" << key_string.size()
<< std::endl;
        //return -1;
    //}
    //unsigned char *key = (unsigned
char*)key_string.c_str();

    uint32_t keyStreamSize = L;
    //std::cout << "Enter key stream size:\n>>";
    //std::cin >> keyStreamSize;

    unsigned int *pKeyStream = new unsigned
int[keyStreamSize];

```

```

        Initialization(key, iv);
        GenerateKeyStream(pKeyStream, keyStreamSize);

        std::cout << "Generated key stream:" << std::endl <<
std::endl;

        std::stringstream ss;
        for (int i = 0; i < keyStreamSize; ++i){
            //std::cout << std::dec << i << "\t0x" << std::oct
<< pKeyStream[i] << std::endl;
            ss << std::hex << pKeyStream[i] << std::endl;
        }

        std::string output = ss.str();
        printf("%s",output.c_str());

        return 0;
    }

// ZUC.H

#ifndef ZUC_H
#define ZUC_H

void GenerateKeyStream(unsigned int *pKeyStream, unsigned
int KeyStreamLen);
void Initialization(unsigned char *k, unsigned char *iv);

#endif

// ZUC.cpp

#include "ZUC.h"

/* state registers LFSR */
unsigned int LFSR_S[16] ;

/* F registers */
unsigned int F_R1 ;
unsigned int F_R2 ;

/* output of BR procedure */

```

```

unsigned int BRC_X[4] ;

/* S-boxes */
unsigned char S0[256] = {

0x3e,0x72,0x5b,0x47,0xca,0xe0,0x00,0x33,0x04,0xd1,0x54,0x98
,0x09,0xb9,0x6d,0xcb,

0x7b,0x1b,0xf9,0x32,0xaf,0x9d,0x6a,0xa5,0xb8,0x2d,0xfc,0x1d
,0x08,0x53,0x03,0x90,

0x4d,0x4e,0x84,0x99,0xe4,0xce,0xd9,0x91,0xdd,0xb6,0x85,0x48
,0x8b,0x29,0x6e,0xac,

0xcd,0xc1,0xf8,0x1e,0x73,0x43,0x69,0xc6,0xb5,0xbd,0xfd,0x39
,0x63,0x20,0xd4,0x38,

0x76,0x7d,0xb2,0xa7,0xcf,0xed,0x57,0xc5,0xf3,0x2c,0xbb,0x14
,0x21,0x06,0x55,0x9b,

0xe3,0xef,0x5e,0x31,0x4f,0x7f,0x5a,0xa4,0x0d,0x82,0x51,0x49
,0x5f,0xba,0x58,0x1c,

0x4a,0x16,0xd5,0x17,0xa8,0x92,0x24,0x1f,0x8c,0xff,0xd8,0xae
,0x2e,0x01,0xd3,0xad,

0x3b,0x4b,0xda,0x46,0xeb,0xc9,0xde,0x9a,0x8f,0x87,0xd7,0x3a
,0x80,0x6f,0x2f,0xc8,

0xb1,0xb4,0x37,0xf7,0x0a,0x22,0x13,0x28,0x7c,0xcc,0x3c,0x89
,0xc7,0xc3,0x96,0x56,

0x07,0xbf,0x7e,0xf0,0x0b,0x2b,0x97,0x52,0x35,0x41,0x79,0x61
,0xa6,0x4c,0x10,0xfe,

0xbc,0x26,0x95,0x88,0x8a,0xb0,0xa3,0xfb,0xc0,0x18,0x94,0xf2
,0xe1,0xe5,0xe9,0x5d,

0xd0,0xdc,0x11,0x66,0x64,0x5c,0xec,0x59,0x42,0x75,0x12,0xf5
,0x74,0x9c,0xaa,0x23,

0x0e,0x86,0xab,0xbe,0x2a,0x02,0xe7,0x67,0xe6,0x44,0xa2,0x6c
,0xc2,0x93,0x9f,0xf1,

0xf6,0xfa,0x36,0xd2,0x50,0x68,0x9e,0x62,0x71,0x15,0x3d,0xd6
,0x40,0xc4,0xe2,0x0f,

```

```
0x8e,0x83,0x77,0x6b,0x25,0x05,0x3f,0x0c,0x30,0xea,0x70,0xb7
,0xa1,0xe8,0xa9,0x65,

0x8d,0x27,0x1a,0xdb,0x81,0xb3,0xa0,0xf4,0x45,0x7a,0x19,0xdf
,0xee,0x78,0x34,0x60
};

unsigned char S1[256] = {

0x55,0xc2,0x63,0x71,0x3b,0xc8,0x47,0x86,0x9f,0x3c,0xda,0x5b
,0x29,0xaa,0xfd,0x77,

0x8c,0xc5,0x94,0x0c,0xa6,0x1a,0x13,0x00,0xe3,0xa8,0x16,0x72
,0x40,0xf9,0xf8,0x42,

0x44,0x26,0x68,0x96,0x81,0xd9,0x45,0x3e,0x10,0x76,0xc6,0xa7
,0x8b,0x39,0x43,0xe1,

0x3a,0xb5,0x56,0x2a,0xc0,0x6d,0xb3,0x05,0x22,0x66,0xbf,0xdc
,0x0b,0xfa,0x62,0x48,

0xdd,0x20,0x11,0x06,0x36,0xc9,0xc1,0xcf,0xf6,0x27,0x52,0xbb
,0x69,0xf5,0xd4,0x87,

0x7f,0x84,0x4c,0xd2,0x9c,0x57,0xa4,0xbc,0x4f,0x9a,0xdf,0xfe
,0xd6,0x8d,0x7a,0xeb,


0x2b,0x53,0xd8,0x5c,0xa1,0x14,0x17,0xfb,0x23,0xd5,0x7d,0x30
,0x67,0x73,0x08,0x09,

0xee,0xb7,0x70,0x3f,0x61,0xb2,0x19,0x8e,0x4e,0xe5,0x4b,0x93
,0x8f,0x5d,0xdb,0xa9,

0xad,0xf1,0xae,0x2e,0xcb,0x0d,0xfc,0xf4,0x2d,0x46,0x6e,0x1d
,0x97,0xe8,0xd1,0xe9,

0x4d,0x37,0xa5,0x75,0x5e,0x83,0x9e,0xab,0x82,0x9d,0xb9,0x1c
,0xe0,0xcd,0x49,0x89,

0x01,0xb6,0xbd,0x58,0x24,0xa2,0x5f,0x38,0x78,0x99,0x15,0x90
,0x50,0xb8,0x95,0xe4,

0xd0,0x91,0xc7,0xce,0xed,0x0f,0xb4,0x6f,0xa0,0xcc,0xf0,0x02
```



```
,0x4a,0x79,0xc3,0xde,

0xa3,0xef,0xea,0x51,0xe6,0x6b,0x18,0xec,0x1b,0x2c,0x80,0xf7
,0x74,0xe7,0xff,0x21,

0x5a,0x6a,0x54,0x1e,0x41,0x31,0x92,0x35,0xc4,0x33,0x07,0x0a
,0xba,0x7e,0x0e,0x34,

0x88,0xb1,0x98,0x7c,0xf3,0x3d,0x60,0x6c,0x7b,0xca,0xd3,0x1f
,0x32,0x65,0x04,0x28,

0x64,0xbe,0x85,0x9b,0x2f,0x59,0x8a,0xd7,0xb0,0x25,0xac,0xaf
,0x12,0x03,0xe2,0xf2
};

/* D constants */
unsigned int EK_d[16] = {
    0x44D7, 0x26BC, 0x626B, 0x135E, 0x5789, 0x35E2, 0x7135,
    0x09AF,
    0x4D78, 0x2F13, 0x6BC4, 0x1AF1, 0x5E26, 0x3C4D, 0x789A,
    0x47AC
};

unsigned int AddM(unsigned int a, unsigned int b) {
    unsigned int c = a + b;
    return (c & 0x7FFFFFFF) + (c >> 31);
}

#define MulByPow2(x, k) (((x) << k) | ((x) >> (31 - k))) &
0x7FFFFFFF)

/* LFSR */
void LFSRWithInitializationMode(unsigned int u) {
    unsigned int f, v;

    f = LFSR_S[0];
    v = MulByPow2(LFSR_S[0], 8);
    f = AddM(f, v);

    v = MulByPow2(LFSR_S[4], 20);
    f = AddM(f, v);

    v = MulByPow2(LFSR_S[10], 21);
    f = AddM(f, v);

    v = MulByPow2(LFSR_S[13], 17);
    f = AddM(f, v);
}
```

```

        v = MulByPow2(LFSR_S[15], 15);

        f = AddM(f, v);

        f = AddM(f, u);

        /* update the state */
        for(int i = 0; i < 15; ++i) {
            LFSR_S[i] = LFSR_S[i + 1];
        }
        LFSR_S[15] = f;
    }

    /* LFSR with work mode */
    void LFSRWithWorkMode() {
        unsigned int f, v;

        f = LFSR_S[0];
        v = MulByPow2(LFSR_S[0], 8);
        f = AddM(f, v);

        v = MulByPow2(LFSR_S[4], 20);
        f = AddM(f, v);

        v = MulByPow2(LFSR_S[10], 21);
        f = AddM(f, v);

        v = MulByPow2(LFSR_S[13], 17);
        f = AddM(f, v);

        v = MulByPow2(LFSR_S[15], 15);
        f = AddM(f, v);

        /* update state */
        for(int i = 0; i < 15; ++i) {
            LFSR_S[i] = LFSR_S[i + 1];
        }
        LFSR_S[15] = f;
    }

    /* Bit Reorganization Procedure */
    void BitReorganization() {
        BRC_X[0] = ((LFSR_S[15] & 0x7FFF8000) << 1) |
        (LFSR_S[14] & 0xFFFF);
    }

```

```

        BRC_X[1] = ((LFSR_S[11] & 0xFFFF) << 16) | (LFSR_S[9]
>> 15);
        BRC_X[2] = ((LFSR_S[7] & 0xFFFF) << 16) | (LFSR_S[5] >>
15);
        BRC_X[3] = ((LFSR_S[2] & 0xFFFF) << 16) | (LFSR_S[0] >>
15);
    }

#define ROT(a, k) (((a) << k) | ((a) >> (32 - k)))

/* linear transformation L1 */
unsigned int L1(unsigned int X) {
    return (X ^ ROT(X, 2) ^ ROT(X, 10) ^ ROT(X, 18) ^
ROT(X, 24));
}

/* linear transformation L2 */
unsigned int L2(unsigned int X) {
    return (X ^ ROT(X, 8) ^ ROT(X, 14) ^ ROT(X, 22) ^
ROT(X, 30));
}

/* create 32-bit word */
#define MAKEU32(a, b, c ,d) (\
((unsigned int)(a) << 24) \
| ((unsigned int)(b) << 16) \
| ((unsigned int)(c) << 8) \
| ((unsigned int)(d)))

/* non-linear function F */
unsigned int F(void) {
    unsigned int W, W1, W2, u, v;

    W = (BRC_X[0] ^ F_R1) + F_R2;
    W1 = F_R1 + BRC_X[1];
    W2 = F_R2 ^ BRC_X[2];
    u = L1((W1 << 16) | (W2 >> 16));
    v = L2((W2 << 16) | (W1 >> 16));
    F_R1 = MAKEU32(S0[u >> 24], S1[(u >> 16) & 0xFF], S0[(u
>> 8) & 0xFF], S1[u & 0xFF]);
    F_R2 = MAKEU32(S0[v >> 24], S1[(v >> 16) & 0xFF], S0[(v
>> 8) & 0xFF], S1[v & 0xFF]);

    return W;
}

```

```

    }

#define MAKEU31(a, b, c) ( \
    ((unsigned int)((unsigned int)(0) \
    | (unsigned char)(a)) << 23) \
    | ((unsigned int)(b) << 8) \
    | (unsigned int)((unsigned int)(0) \
    | (unsigned char)(c)))

void Initialization(unsigned char *k, unsigned char *iv) {
    unsigned int w;

    /* expand key */
    for(int i = 0; i < 16; ++i) {
        LFSR_S[i] = MAKEU31(k[i], EK_d[i], iv[i]);
    }
    /* set F_R1 and F_R2 to zero */
    F_R1 = 0;
    F_R2 = 0;
    unsigned int nCount = 32;
    while (nCount > 0){
        BitReorganization();
        w = F();

        LFSRWithInitializationMode(w >> 1);
        nCount--;
    }

    BitReorganization();
    F();
    LFSRWithWorkMode();
}

void GenerateKeyStream(unsigned int *pKeyStream, unsigned
int KeyStreamLen){
    /* working cycles */
    for (int i = 0; i < KeyStreamLen; ++i){
        BitReorganization();
        pKeyStream[i] = F() ^ BRC_X[3];
        LFSRWithWorkMode();
    }
}

```

// Audio to Text using ESP32 and INMP441

- Device

ESP32-S3 DevKit-C

<https://docs.espressif.com/projects/esp-idf/en/latest/esp32s3/hw-reference/esp32s3/user-guide-devkitc-1.html>

- Required Library

Arduino ESP32: 2.0.9

Arduino Websockets: 0.5.3

<https://github.com/gilmaimon/ArduinoWebsockets>

*/

```
#include <driver/i2s.h>
```

```
#include <WiFi.h>
```

```
#include <ArduinoWebsockets.h>
```

```
#define I2S_SD 10
```

```
#define I2S_WS 11
```

```
#define I2S_SCK 12
```

```
#define I2S_PORT I2S_NUM_0
```

```
#define bufferCnt 10
```

```
#define bufferLen 1024
```

```
int16_t sBuffer[bufferLen];
```

```
const char* ssid = "<YOUR_WIFI_SSID>";
```

```
const char* password = "<YOUR_WIFI_PW>";
```

```
const char* websocket_server_host = "<WEBSOCKET_SERVER_HOST_IP>";
```

```
const uint16_t websocket_server_port = 8888; // <WEBSOCKET_SERVER_PORT>
```

```

using namespace websockets;
WebsocketsClient client;
bool isWebSocketConnected;

void onEventsCallback(WebsocketsEvent event, String data) {
    if (event == WebsocketsEvent::ConnectionOpened) {
        Serial.println("Connnection Opened");
        isWebSocketConnected = true;
    } else if (event == WebsocketsEvent::ConnectionClosed) {
        Serial.println("Connnection Closed");
        isWebSocketConnected = false;
    } else if (event == WebsocketsEvent::GotPing) {
        Serial.println("Got a Ping!");
    } else if (event == WebsocketsEvent::GotPong) {
        Serial.println("Got a Pong!");
    }
}

void i2s_install() {
    // Set up I2S Processor configuration
    const i2s_config_t i2s_config = {
        .mode = i2s_mode_t(I2S_MODE_MASTER | I2S_MODE_RX),
        .sample_rate = 44100,
        //.sample_rate = 16000,
        .bits_per_sample = i2s_bits_per_sample_t(16),
        .channel_format = I2S_CHANNEL_FMT_ONLY_LEFT,
        .communication_format =
            i2s_comm_format_t(I2S_COMM_FORMAT_STAND_I2S),
        .intr_alloc_flags = 0,
        .dma_buf_count = bufferCnt,
        .dma_buf_len = bufferLen,

        .use_apll = false
    };
}

```

```
};

i2s_driver_install(I2S_PORT, &i2s_config, 0, NULL);
}

void i2s_setpin() {
    // Set I2S pin configuration
    const i2s_pin_config_t pin_config = {
        .bck_io_num = I2S_SCK,
        .ws_io_num = I2S_WS,
        .data_out_num = -1,
        .data_in_num = I2S_SD
    };

    i2s_set_pin(I2S_PORT, &pin_config);
}

void setup() {
    Serial.begin(115200);

    connectWiFi();
    connectWSServer();
    xTaskCreatePinnedToCore(micTask, "micTask", 10000, NULL, 1, NULL, 1);
}

void loop() {
}

void connectWiFi() {
    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
}
```

```
Serial.println("");

Serial.println("WiFi connected");
}

void connectWSServer() {
  client.onEvent(onEventsCallback);
  while (!client.connect(websocket_server_host, websocket_server_port, "/")) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("Websocket Connected!");
}

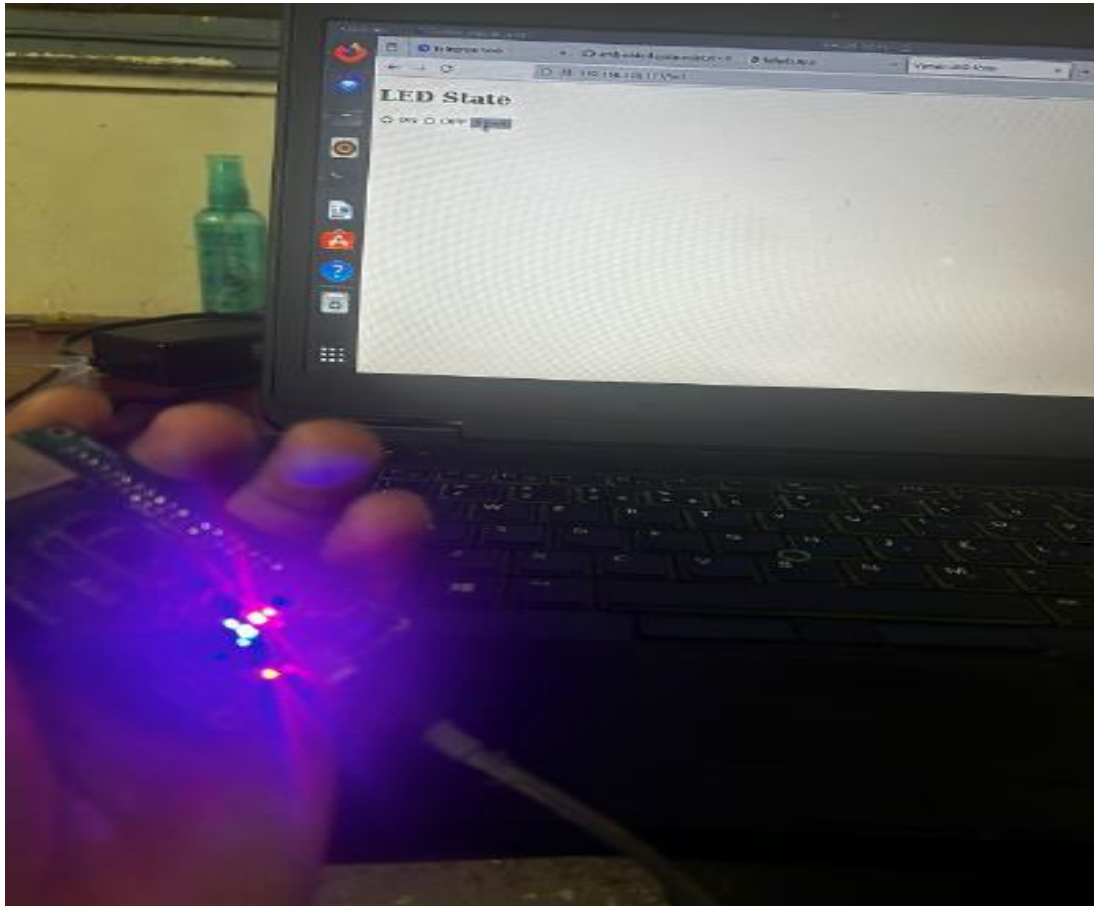
void micTask(void* parameter) {

  i2s_install();
  i2s_setpin();
  i2s_start(I2S_PORT);

  size_t bytesIn = 0;
  while (1) {
    esp_err_t result = i2s_read(I2S_PORT, &sBuffer, bufferLen, &bytesIn,
                                portMAX_DELAY);
    if (result == ESP_OK && isWebSocketConnected) {
      client.sendBinary((const char*)sBuffer, bytesIn);
    }
  }
}
```


5.2 OUTPUT SCREENS

Vaman Testing Sample LED ON/OFF:



Output 1

UGV :

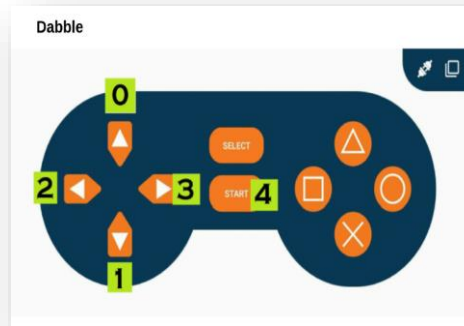
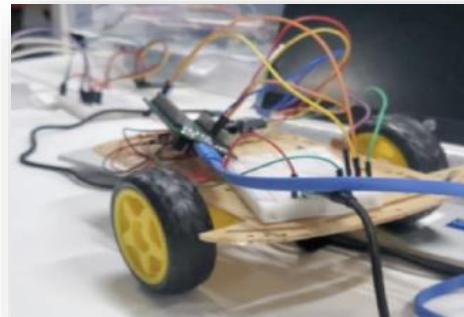
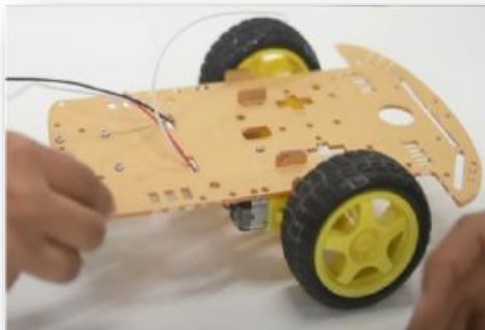
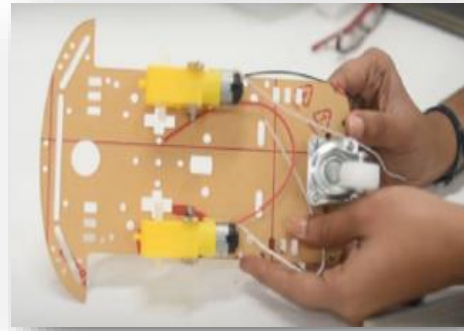
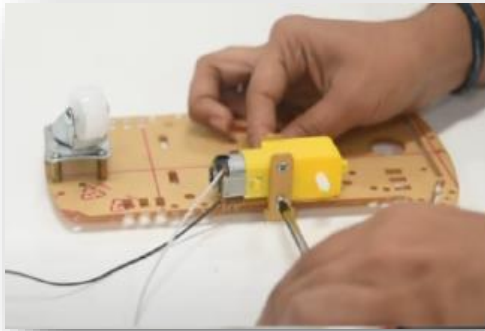
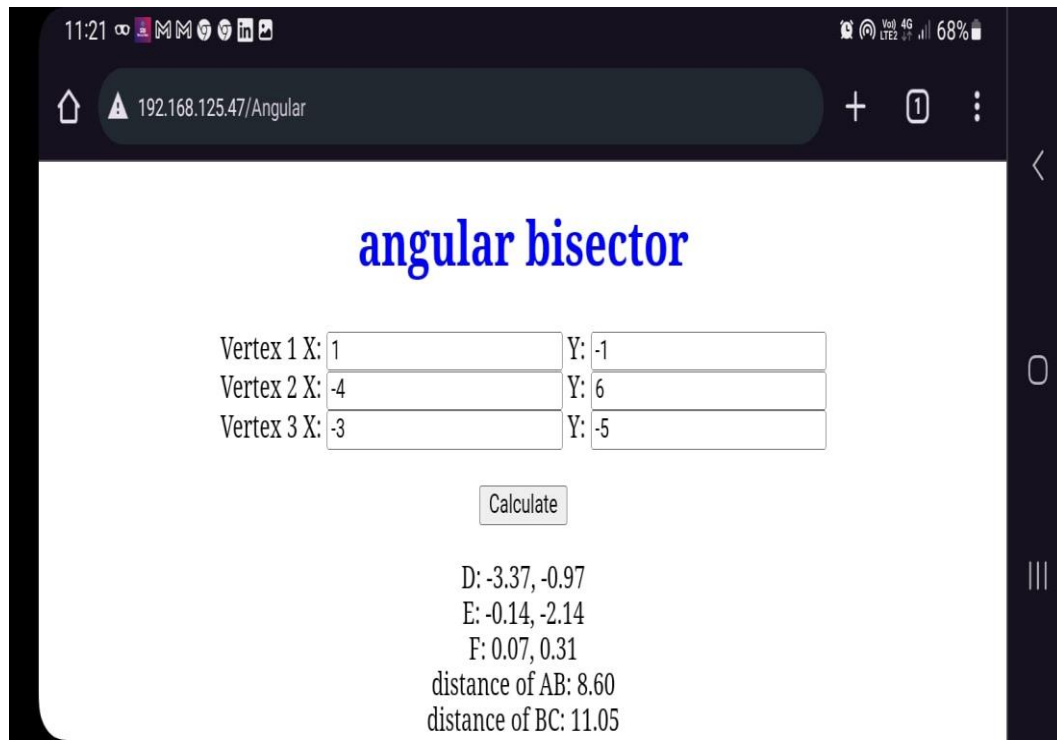


FIG 6: UGV

WEB Page generation using VAMAN Board:



11:21 192.168.125.47/Angular 68%

angular bisector

Vertex 1 X:	<input type="text" value="1"/>	Y:	<input type="text" value="-1"/>
Vertex 2 X:	<input type="text" value="-4"/>	Y:	<input type="text" value="6"/>
Vertex 3 X:	<input type="text" value="-3"/>	Y:	<input type="text" value="-5"/>

D: -3.37, -0.97
E: -0.14, -2.14
F: 0.07, 0.31
distance of AB: 8.60
distance of BC: 11.05

OUTPUT 2

CHAPTER 6: CONCLUSION

In summary, the introduction to the Vaman Board underscores its revolutionary role in wireless communication and inter-chip connectivity technology. It highlights the board's capacity for seamless wireless code dumping, Bluetooth control, and robust inter-chip communication via protocols like SPI. Additionally, the integration of Termux and the Dabble app enhances user accessibility and functionality. Moreover, the combination of ESP32 and FPGA modules offers immense potential for innovation and experimentation. While the Vaman Board exhibits impressive features, it also acknowledges inherent limitations, emphasizing the need for continuous improvement. Overall, the introduction sets a strong foundation for exploring the Vaman Board's capabilities and applications.

CHAPTER 7: REFERENCES

<https://github.com/gadepall/embedded-system>

https://www.optimuslogic.in/product_pygmy.html

<https://www.espressif.com/en/products/socs/esp3>

<https://www.linkedin.com/pulse/5g-analytics-nwdaf-satheesh-kumar-marappan/>

https://commons.wikimedia.org/wiki/File:Point-to-point_wireless_link_for_Internet_connectivity_with_FreedomBox.svg

https://commons.wikimedia.org/wiki/File:Point-to-point_wireless_link_for_Internet_connectivity_with_FreedomBox.svg