

AN INTERNSHIP REPORT
ON
FUTURE WIRELESS COMMUNICATION 5G ADVANCED/ 6G
PROJECT

Submitted

In the partial fulfillment of the requirements for

the award of the degree of

Bachelor of Technology in
Electronics and Communication Engineering

By

VAKKANTI LAKSHMI REVANTH KUMAR -201FA05043

Under the guidance of
Mr. Satish Kanapala, M. Tech
Assistant Professor

Under the guidance of
Dr. G. V V Sharama
Associate Professor



VIGNAN'S
Foundation for Science, Technology & Research
(Deemed to be University)
-Estd. u/s 3 of UGC Act 1956



(ACCREDITED BY NAAC WITH 'A+' GRADE)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

(ACCREDITED BY NBA)

VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY AND RESEARCH

(Deemed to be university)

VADLAMUDI, GUNTUR – 522 213, INDIA.

MAY-2024



Indian Institute of Technology
Hyderabad National Highway 9
Kandi, Sangareddy
Telangana - 502284
Phone: (040) 2301 6772
Fax: (040) 2301 6000

Future Wireless Communication 5G Advanced/6G Project
INTERNSHIP COMPLETION CERTIFICATE

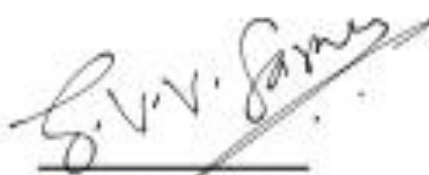
Date: 30 May 2024

This is to certify that **Mr. V Lakshmi Revanth Kumar**, has carried out "**Future Wireless Communications**" Internship at Indian Institute of Technology Hyderabad, under our supervision during the period from **20th -January-2024** to **29th -May-2024**.

His role and responsibilities include "**Data Handling and Hardware programming , INTERCHIP**" under **Prof. G. V. V. Sharma**.

He is extremely sincere and hardworking.

We wish you every success in all his future endeavors.



Prof. GVV Sharma
Program Coordinator
IIT Hyderabad

CERTIFICATE

This is to certify that the internship report entitled” **FUTURE WIRELESS COMMUNICATION 5G ADVANCED/ 6G PROJECT**” that is being submitted by **VAKKANTI LAKSHMI REVANTH KUMAR (201FA05043)** in partial fulfilment for the award of B. Tech degree in Electronics and Communication Engineering to Vignan’s Foundation for Science Technology and Research University, is a record of bonafide work carried out by them at Indian Institute of Technology under the supervision of Dr. G V V Sharma, Associate Professor of EE Department, IITH and the internal guidance of Mr. Satish Kanapala of ECE Department.

Signature of the guide

Mr. Satish Kanapala, M. Tech
Assistant Professor ECE.

Signature of Head of the Department

Dr. T. Pitchaiah, Ph.D., MIEEE, FIETE
Professor & HoD, ECE.

Signature of

INTERNAL EXAMINER

Signature of

EXTERNAL EXAMINER

DECLARATION

I am hereby declare that the internship work entitled **“FUTURE WIRELESS COMMUNICATION 5G ADVANCED/ 6G PROJECT”** is being submitted to Vignan’s Foundation for Science, Technology and Research, University, in partial fulfillment for the award of B. Tech degree in Electronics and Communication Engineering. The work was originally designed and executed by me under the guidance of my supervisor Dr. G V V Sharma with Mr. Satish Kanapala as faculty guide at Department /of Electronics and Communication Engineering, Vignan’s Foundation for Science Technology and Research University and was not a duplication of work done by someone else. I hold the responsibility of the originality of the work incorporated into this thesis.

Signature of the candidates

Vakkanti Lakshmi Revanth Kumar (201FA5043).

Place:

Date:

ACKNOWLEDGEMENT

Firstly, I would like to thank Dr. **Kiran Kuchi, Professor IIT-Hyderabad** for giving me the opportunity to do an internship within the organization.

I am highly indebted to **Dr. G V V Sharma, Associate Professor IITH** who monitored and trained me well personally throughout my internship.

I am greatly indebted to **Mr. Satish Kanapala**, our revered guide and Professor in the Department of Electronics and Communication Engineering, VFSTR (Deemed to be University), Vadlamudi, Guntur, for his valuable guidance in the preparation of this dissertation. He has been a source of great inspiration and encouragement to us. He has been kind enough to devote considerable amount of his valuable time in guiding us at every stage. This is our debut, but we are sure that we can do many more such studies, purely become of the lasting inspiration and guidance given by respectable guide

I would like to thank **Dr. T. Pitchaiah**, Professor & HoD VFSTR for his constructive criticism throughout my internship.

I would like to specially thank, **Prof. N. Usha Rani**, Dean, School of ECE for her help and support during the project work.

I thank our internship coordinators **Mr. M. Vamshi Krishna, Mr. Abhishek Kumar** for continuous support and suggestions in scheduling the reviews and report verifications.

I would like to thank all the people that worked along with me in **Indian Institute of Technology Hyderabad** with their patience and openness they created an enjoyable working environment. It is indeed with a great sense of pleasure and immense sense of gratitude that I acknowledge the help of these individuals.

I wish to express our gratitude to **Prof. P. Nagabhushan** Vice-Chancellor, VFSTR (Deemed to be University) for providing us the greatest opportunity to have a great exposure and to carry out the project

I would like to thank **Dr. Lavu Rathaiah**, chairman VFSTR for his support and advices to get and complete internship in above said organization.

I am extremely grateful to my department staff members and friends who helped me in successful completion of this internship.

Name of the Candidate

Vakkanti Lakshmi Revanth Kumar (201FA5043).

ABSTRACT

This internship report showcases the significance of the program in equipping participants with the necessary skills and knowledge to thrive in the rapidly evolving field of wireless communication, specifically in the context of 5G networks. It emphasizes the practical relevance of the program and its potential to contribute to the growth and advancement of wireless communication technology in Data Handling and Hardware Programming, Inter-Chip. The Vaman Board, an innovative product by Quick Logic, stands at the forefront of wireless communication and inter-chip connectivity technology. This report 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.

LEARNING OBJECTIVES

- Gain a comprehensive understanding of advanced wireless technologies, including 5G and beyond, and their applications in various industries.
- Develop proficiency in data handling and hardware programming, enabling the design and implementation of wireless communication systems.
- Acquire knowledge and skills in signal processing techniques and related hardware, essential for efficient data transmission and reception in wireless networks.
- Gain hands-on experience by working on real-world projects related to 5G/6G technology, applying the learned concepts and techniques to practical scenarios.
- Develop problem-solving and critical thinking abilities to address challenges in wireless communication.
- Enhance technical communication skills through interactions with industry experts and faculty members, presenting project findings, and collaborating within a diverse team environment.
- Understand the indigenization process of 5G networks in India and contribute to the country's goal of building a skilled workforce in wireless communication.
- Explore career opportunities in the field of wireless communication, including software development, hardware design, and research and development.
- Foster an understanding of the importance of continuous learning and staying updated with emerging trends and advancements in wireless communication technologies.

Major Design (Final Year Internship Work) Experience Information

Student Group	V. Lakshmi Revanth Kumar (201FA05043)	K. Sri Latha (201FA05070)	D. Siva Krishna (201FA05076)
Internship Title	FUTURE WIRELESS COMMUNICATION 5G ADVANCED/ 6G PROJECT		
Program Concentration Area	Implementation of Inter-chip communication using Vaman board		
Program Concentration Area	Implementation of Inter-chip communication using Vaman board		
Constraints - Examples			
Economic	Fixed budget		
Environmental	Friendly		
Sustainability	To save power and to reduce complexity in communication between two different devices in a communication system.		
Manufacturability	Yes		
Ethical	Followed the standard professional ethics		
Health and Safety	Guidelines are followed		
Social	Applicable for Communication		
Political	None		
Other	Turns to be more economical, less manufacturing required		
Standards			
1. IEEE 802.15.1	For establishing in range communication.		
2. IEEE 802.11	Used for establishing connection between other devices to Vaman.		
Previous Course Required for the Major Design Experience	1. Cellular Mobile Communication 2. Digital System Design 3. Internet of Things		

Supervisor

Project Co-Ordinator

Head of the department ECE

CONTENTS

	Page No
CHAPTER 1 INTRODUCTION	01- 03
INTRODUCTION	02
1.1 Standards	02
1.1.1 IEEE 802.15.1	02
1.1.2 IEEE 802.11	03
1.2 Protocols	03
CHAPTER 2 LITERATURE SURVEY	04-05
Literature survey	05
CHAPTER 3 DATA HANDLING	06-22
3.1 Android Applications	07
3.1.1 Termux	07
3.1.2 Platform.io	08
3.1.3 Arduino Droid	09
3.2 Installations	10
3.2.1 Termux Installations	10
3.2.2 Platform.io Installations	12
3.2.3 Arduino Droid Installations	13
3.3 LaTeX	14
3.4 Circuit designing for gate questions	19
3.4.1 Step-by-Step Guide for Call the Logic File	19
3.4.2 Logic for Designing Circuit Diagram	21
CHAPTER 4 HARDWARE PROGRAMMING	23-34

4.1	Components	24
4.2	Seven segment displays	24
4.3	7447	27
4.4	7474	30
4.5	State machine	32
CHAPTER 5	VAMAN BOARD	35-43
5.1	Introduction	36
5.2	Vaman ESP	37
	5.2.1 flashing vaman-esp32 using Arduino	38
5.3	Vaman ARM	39
5.4	Vaman FPGA	42
CHAPTER 6	UBUNTU, INSTALLATIONS AND MATH COMPUTATION	44-52
6.1	Ubuntu	45
6.2	Installations	47
6.3	Math Computing Using ESP32	51
CHAPTER 7	INTER-CHIP COMMUNICATION USING VAMAN BOARD	53-59
7.1	Inter-Chip Communication	54
	7.1.1 Seven Segment Display	54
	7.1.2 LCD	57
CHAPTER 8	CONCLUSION AND FUTURE SCOPE	60-61
8	Conclusion and Future scope	61
	REFERENCES	62

LIST OF FIGURES

FIG NO	NAME OF FIGURE	PAGE NO
3.1	Termux Home Page	12
3.2	Instructions for opening file	12
3.3	Circuit Diagram	19
3.4	Design Circuit Diagram	22
4.1	Seven Segment Display Pin Diagram	24
4.2	Output for Seven Segment Display	25
4.3	Output Zero Displaying	26
4.4	Output One Displaying	26
4.5	Output Two Displaying	26
4.6	Output Three Displaying	26
4.7	Output Four Displaying	26
4.8	Output Five Displaying	26
4.9	Output Six Displaying	26
4.10	Output Seven Displaying	26
4.11	Output Eight Displaying	27
4.12	Output Nine Displaying	27
4.13	7447 Pin Diagram	28
4.14	Output of 7447	29
4.15	Output of 7447	29
4.16	Output of 7447	29
4.17	Output of 7447	29

4.18	Output of 7447	29
4.19	Output of 7447	29
4.20	7474 Pin Diagram	30
4.21	Output of 7474	31
4.22	Output of FSM	32
5.1	Vaman LC Pin Diagram	34
5.2	Vaman LC Board	36
5.3	Circuit Connections	37
5.4	Example Output	38
5.5	Memory system design from A ₁₁ to A ₁₅	39
5.6	Output of ARM	41
5.7	Output of ARM	41
5.8	Output of ARM	41
5.9	Output of ARM	41
5.10	Output of ARM	42
5.11	Output of ARM	42
5.12	Output of ARM	42
5.13	Output of ARM	42
5.14	Interconnection of T-flip flop	43
5.15	Output of FPGA	43
6.1	Features of Ubuntu	45
6.2	Specifications of Ubuntu	46
6.3	Pros of Ubuntu over Windows	47
6.4	Math computing using Vaman ESP32	52

7.1	Seven segment pins	55
7.2	Web page for selecting output to be displayed	56
7.3	Seven segment display output	57

LIST OF TABLES

TABLE NO	TABLE NAME	PAGE NO
4.1	Component Required	24
4.2	Arduino to Seven Segment Display Connections	25
4.3	7447 to Seven Segment Display Connections	28
4.4	7447 to Arduino Connections	28
4.5	7474 and 7447 to Arduino connections for 7474	31
4.6	7474 and 7447 to Arduino connections for FSM	34
5.1	Connections b/w Vaman and Arduino	38
5.2	Seven segment display to Vaman connections	40
7.1	Connections b/w Vaman ESP to PYGMY	54
7.2	Seven segment display components	55
7.3	Connections b/w Vaman and seven segment display	55
7.4	Components for LCD	57
7.5	Connections b/w LCD and Vaman	57

LIST OF ACRONYMS

APT- Advanced Package Tool

ARM- Advanced RISC Machine

BCD- Binary Coded Decimal

CLB- Configurable Logic Block

DIP- Dual In-line Package

ESP- Event Stream Processing

FPGA- Field Programmable Gate Array

IDE- Integrated Development Environment

IEEE- Institute of Electrical & Electronics Engineers

LCD- Liquid Crystal Display

LED- Light Emitting Diode

MAC- Medium Access Control

OTG- On The Go

RBI- Ripple Blanking Input

RISC- Reduced Instruction Set Computer

SOC- System On Chip

SPI- Serial Peripheral Interface

USB- Universal Serial Bus

UART- Universal Asynchronous Receiver Transmitted

CHAPTER 1

INTRODUCTION

INTRODUCTION

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.

1.1 Standards

1.1.1 IEEE 802.15.1

IEEE 802.15.1 is a standard for Wireless Personal Area Networks (WPANs) that focuses on low-power, low-data-rate applications. It is primarily used for Bluetooth technology, which enables wireless connectivity between devices within a personal

operating space. The standard defines physical layer (PHY) and medium access control (MAC) specifications for wireless connectivity with fixed, portable, and moving devices. The key features of IEEE 802.15.1 include: Low power consumption: Designed for battery-powered devices, it ensures long battery life. Low data rate: Suitable for applications that require low data transfer rates. Personal operating space: Focuses on connectivity within a personal operating space, typically within a few meters.

1.1.2 IEEE 802.11

IEEE 802.11 is a standard for Wireless Local Area Networks (WLANs) that supports higher data rates and longer transmission ranges. It is commonly used for Wi-Fi technology, which enables wireless connectivity between devices within a local area network. The key features of IEEE 802.11 include: Higher data rates: Supports higher data transfer rates, typically up to several hundred megabits per second. Longer transmission range: Designed for connectivity over longer distances, typically within a building or campus. Local area network: Focuses on connectivity within a local area network, often used for internet access and data transfer. In summary, IEEE 802.15.1 is designed for low-power, low-data-rate applications within a personal operating space, while IEEE 802.11 is designed for higher data rates and longer transmission ranges within a local area network.

1.2 Protocols

SPI: The Serial Peripheral Interface (SPI) protocol is a synchronous, full-duplex serial communication protocol that allows multiple slave devices to connect to a single master device. It's commonly used to send data between microcontrollers and small peripherals like sensors, shift registers, and SD cards.

CHAPTER 2

LITERATURE SURVEY

LITERATURE SURVEY

The academic community usually refers to UGVs (especially UGVs possessing significant autonomous capabilities) as mobile robots. There is a certain irony in this terminology, since many of the key research issues (e.g., "inverse kinematics") addressed in "traditional" robotics (oriented to the control of industrial manipulators) are completely irrelevant to mobile robots. There is some commonality in issues relating to path planning, obstacle avoidance, and sensor-based control, but results have tended to flow more from mobile robots to manipulators, rather than in the other direction. The focus of mobile robotic research has in fact evolved from the discipline of artificial intelligence (AI).

The first major mobile robot development effort was Shakey. Developed in the late 1960s to serve as a testbed for DARPA-funded AI work at Stanford Research Institute

Shakey was a wheeled platform equipped with steerable TV camera, ultrasonic range finder, and touch sensors, connected via an RF link to its SDS-940 mainframe computer that performed navigation and exploration tasks. As befit an AI testbed, the Shakey system could accept English sentence commands from the terminal operator, directing the robot to push large wooden blocks around in its lab environment "world". While Shakey was considered a failure in its day because it never achieved autonomous operation, the project established functional and performance baselines for mobile robots, identified technological deficiencies, and helped to define the AI research agenda in such areas as planning, vision, and natural language processing [Flynn, 1985]. From 1973 to 1981, Hans Moravec led the Stanford Cart project at the Stanford University AI Lab, exploring navigation and obstacle avoidance issues using a sophisticated stereo vision system. The Cart's single TV camera was moved to each of 9 different positions atop its simple mobility base, and the resulting images were processed by the offboard KL-10 mainframe. Feature extraction and correlation between images allowed reconstruction of a model of the 3-D scene, which was used to plan an obstacle-free path to the destination. The system was incredibly slow, taking up to 15 minutes to make each one-meter move.

CHAPTER 3

DATA HANDLING

3.1 Android Applications

3.1.1 Termux:

Termux is an Android terminal emulator and Linux environment app that allows users to run command-line programs and scripts directly on their Android devices. It provides a full-fledged Linux environment on your smartphone or tablet, giving you access to various tools and utilities typically found on desktop or server Linux distributions.

Here are some key features and details about Termux:

1. **Terminal Emulator:** Termux provides a terminal emulator for Android devices, allowing users to interact with a command-line interface just like they would on a traditional computer.
2. **Linux Environment:** Termux brings a Linux environment to Android, allowing users to install and run Linux packages and tools. It's based on the Debian GNU/Linux distribution.
3. **Package Management:** Termux includes its own package manager called APT (Advanced Package Tool), which allows users to easily install, update, and remove software packages. Users can install a wide range of packages, including programming languages, development tools, utilities, and more.
4. **Support for Programming and Scripting:** With Termux, users can write, compile, and execute code directly on their Android device. It supports various programming languages such as Python, Ruby, Perl, Node.js, and more. Users can also run shell scripts and automate tasks using tools like Bash.
5. **Access to Device Hardware and Sensors:** Termux provides access to various device hardware and sensors, allowing users to interact with the device's camera, microphone, accelerometer, and more through command-line tools and scripts.
6. **Customization and Configuration:** Users can customize their Termux environment by installing additional packages, configuring shell preferences, setting up custom scripts, and more.
7. **Community and Support:** Termux has an active community of users and developers who contribute to its development, share tips and tricks, and provide support through forums, social media, and other channels.
8. **Security:** Termux runs in an isolated environment on Android devices, which helps to enhance security by sandboxing the command-line programs and scripts executed

within it. However, users should still exercise caution when installing and running software from untrusted sources.

Overall, Termux is a powerful tool that extends the capabilities of Android devices, allowing users to perform a wide range of tasks and development activities directly from their smartphones or tablets. Whether you're a developer, hobbyist, or enthusiast, Termux provides a convenient way to access a Linux environment on the go.

3.1.2 Platform.io:

Platform.io is an open-source ecosystem for developing, testing, and deploying embedded systems. It provides a unified and customizable platform for various development boards and microcontrollers, making it easier for developers to manage their embedded software projects. Here are some key features and components of Platform.io in detail:

1. Integrated Development Environment (IDE):

Platform.io integrates with popular code editors like Visual Studio Code, Atom, and Eclipse. It also offers its standalone IDE, Platform.io IDE, which comes with all the necessary tools pre-installed.

Key IDE Features:

- **Code Completion and IntelliSense:** Helps in writing code more efficiently by providing suggestions and auto-completions.
- **Library Manager:** Facilitates easy installation and management of libraries from a large repository of pre-configured libraries.
- **Built-in Terminal:** Allows direct interaction with the system's command line within the IDE.
- **Integrated Debugger:** Supports debugging for a wide range of boards and frameworks.

2. Cross-Platform Build System:

Platform.io build system supports multiple platforms and frameworks. It abstracts away the complexities of different toolchains, providing a seamless build process.

Key Build System Features:

- **Multiple Frameworks and Platforms:** Supports a variety of development frameworks like Arduino, mbed, Zephyr, and many others, alongside numerous microcontroller platforms (e.g., Espressif, STMicroelectronics).
- **Automatic Dependency Management:** Automatically handles the dependencies required for a project, ensuring that all necessary tools and libraries are included.
- **Customizable Build Scripts:** Allows customization of the build process through the platformio.ini configuration file.

3. Library Management:

Platform.io includes a robust library management system, enabling developers to easily find, install, and update libraries.

Key Library Management Features:

- **Library Registry:** A comprehensive database of libraries that can be searched and added to projects.
- **Version Control:** Ensures compatibility by allowing specific versions of libraries to be installed.
- **Automatic Updates:** Notifies and updates to the latest library versions when available.

3.1.3 Arduino Droid:

Arduino Droid is an integrated development environment (IDE) for Android that allows users to write, compile, and upload Arduino sketches directly from their mobile devices. This makes it a handy tool for on-the-go development and testing of Arduino projects. Below is a detailed explanation of its features, installation process, and usage.

Features of Arduino Droid:

1. Full Arduino IDE Functionality:

- Write, edit, and compile Arduino sketches.
- Upload sketches to Arduino boards directly from your Android device.
- Supports popular Arduino libraries and board definitions.

2. File Management:

- Create, open, save, and manage Arduino sketches.

- Organize sketches in folders for better project management.
3. Library Management:
 - Integrated library manager to download and install libraries.
 - Supports custom libraries.
 4. Board Support:
 - Compatible with a wide range of Arduino boards, including Arduino Uno, Mega, Nano, Leonardo, and others.
 - Supports various third-party boards.
 5. Serial Monitor:
 - Monitor serial output from Arduino boards in real-time.
 - Send data to Arduino boards via the serial interface.
 6. Code Assistance:
 - Syntax highlighting and code suggestions for easier coding.
 - Line numbering and error highlighting for debugging.

3.2 INSTALLATIONS

3.2.1 Termux Installations:

1. Download and Install Termux:
 - Open the Google Chrome on your Android device.
 - In the search bar, type "F-Droid" and press Enter.
 - Tap on the F-Droid app from the search results.
 - Tap on the "Install" button to download and install the app onto your device.
 - In the F-Droid app, search for Termux and then install it.
 - Once the installation is complete, tap on the "Open" button to launch Termux.

2. Grant Permissions (Optional):

- Depending on your device and Android version, Termux may request certain permissions such as storage access, microphone access, etc. Follow the on-screen prompts to grant the necessary permissions if prompted.

3. Set Up Storage Access (Optional):

- Termux operates within its own isolated file system by default, but you can grant access to your device's storage if you want to manipulate files outside of Termux's sandbox.
- To grant storage access, you can run the command `termux-setup-storage` within Termux. This will prompt you to grant storage access to the app.

4. Update and Upgrade Packages (Optional):

- It's a good practice to update and upgrade the Termux packages to the latest versions.
- To do this, run the following commands within Termux:
 - `pkg update`
 - `pkg upgrade`

5. Explore Termux:

- Once installed, Termux provides a full-fledged Linux environment on your Android device. You can start exploring various command-line tools, programming languages, and utilities available within Termux.
- To get started, try running some basic commands like `ls` to list files, `cd` to change directories, `apt` to manage packages, etc.

6. Customize Your Setup (Optional):

- Termux is highly customizable. You can install additional packages, configure your shell environment, set up aliases, and more to tailor Termux to your preferences.
- Explore the available packages by running `pkg list` and install any packages you need using `pkg install`.

```
Welcome to Termux!

Community forum: https://termux.com/community
Gitter chat:    https://gitter.im/termux/termux
IRC channel:    #termux on libera.chat

Working with packages:

* Search packages:  pkg search <query>
* Install a package: pkg install <package>
* Upgrade packages: pkg upgrade

Subscribing to additional repositories:

* Root:    pkg install root-repo
* X11:     pkg install x11-repo

Report issues at https://termux.com/issues

~ $ █

ESC  /  -  HOME  ↑  END  PGUP
⇧   CTRL  ALT  ←  ↓  →  PGDN
```

Figure3.1: Termux Home Page

```
~ $ proot-distro login debian
root@localhost:~# cd /sdcard
root@localhost:/sdcard# cd digital-design/
root@localhost:/sdcard/digital-design# cd vaman
root@localhost:/sdcard/digital-design/vaman# cd esp32
root@localhost:/sdcard/digital-design/vaman/esp32# cd codes
root@localhost:/sdcard/digital-design/vaman/esp32/codes# cd ide
root@localhost:/sdcard/digital-design/vaman/esp32/codes/ide# cd ota
root@localhost:/sdcard/digital-design/vaman/esp32/codes/ide/ota# cd setup/
root@localhost:/sdcard/digital-design/vaman/esp32/codes/ide/ota/setup# cd src
root@localhost:/sdcard/digital-design/vaman/esp32/codes/ide/ota/setup/src# nvim main.cpp
root@localhost:/sdcard/digital-design/vaman/esp32/codes/ide/ota/setup# pio run █
```

Figure3.2: Instructions for opening the file

3.2.2 Platform.io Installations:

1. Update and Upgrade Termux Packages Open Termux and run the following commands to ensure that all packages are up to date:
 - `pkg update && pkg upgrade`
2. Install Python Platform.io requires Python. Install Python by running:
 - `pkg install python`
3. Install Pip: Pip is the package installer for Python. It should come pre-installed with Python in Termux, but you can ensure its up-to-date:

- `pip install --upgrade pip`
4. Install Platform.io Core Use pip to install Platform.io Core:
- `pip install platform.io`
5. Verify Platform.io Installation After installation, verify that Platform.io is installed correctly by checking its version:
- `pio --version`
- You should see an output similar to:
- Platform.io Core, version X.X.X

3.2.3 Arduino Droid Installations:

1. Install Arduino Droid App
 - Open Chrome and download APK on your Android device.
 - Search for "Arduino Droid - Arduino IDE".
 - Select the app from the search results.
 - Tap on "Install" to download and install the app.
2. Prepare Your Arduino Board
 - Ensure your Arduino board is compatible with Arduino Droid. Most common boards like Arduino Uno, Nano, and Mega are supported.
 - Get a USB OTG (On-The-Go) cable. This is necessary to connect your Arduino board to your Android device. The OTG cable has a micro-USB or USB-C connector on one end (to connect to your Android device) and a standard USB-A port on the other (to connect your Arduino).
3. Connect Your Arduino Board
 - Connect the OTG cable to your Android device.
 - Connect your Arduino board to the OTG cable using a standard USB cable.
 - Ensure the Arduino board is powered either through the USB connection or an external power source if needed.
4. Open Arduino Droid
 - Launch the Arduino Droid app from your app drawer.
 - Grant any necessary permissions the app requests, such as access to USB devices.

5. Set Up Arduino Droid

- Configure the board type:
 - Tap on the menu icon (three horizontal lines) or go to Settings within the app.
 - Select Board Type and choose the correct board type for your connected Arduino (e.g., Arduino Uno).
- select the serial port:
 - Tap on the USB icon (usually in the top menu).
 - Select the correct USB device from the list. This is your connected Arduino board.

6. Write Your Sketch

- Create a new sketch:
 - Tap on the + (plus) icon or select New Sketch from the menu.
 - Write your Arduino code in the editor.

7. Compile the Sketch

- Tap the compile button (usually a checkmark icon) to compile your sketch.
- Wait for the compilation process to complete. Any errors will be displayed in the console.

8. Upload the Sketch to Arduino

- Tap the upload button (usually an arrow icon pointing right) to upload the compiled sketch to your Arduino board.
- Monitor the upload process. The app will notify you once the upload is successful or if there are any errors.

9. Monitor Serial Output (Optional)

- Open the serial monitor by tapping the serial monitor icon (usually depicted as a magnifying glass over a text icon) to view serial output from your Arduino.
- Set the baud rate to match the baud rate specified in your sketch (e.g., Serial.Begin(9600)).

3.3 Latex:

LaTeX is a typesetting system commonly used for creating high-quality documents that contain complex elements such as mathematical formulas, tables, and figures. It's widely used in academia, particularly in the fields of mathematics, physics, computer science,

engineering, and economics, but it's also useful for any documents that require precise formatting.

Here's a detailed overview of LaTeX:

1. What is LaTeX?

LaTeX (pronounced "Lay-tech" or "Lah-tech") is a markup language and document preparation system. It's not a word processor but a typesetting system that uses plain text files with embedded commands to control the formatting.

2. Basic Structure of a LaTeX Document

A typical LaTeX document starts with a preamble followed by the document content. Here's a simple example:

```
\title{MATHEMATICS}
\author{SECTION A}
\date{\today}
\begin{document}
\maketitle
\begin{enumerate}
\item This is my first LaTeX document
\end{enumerate}
\end{document}
```

3. Key Features of LaTeX

- Text Formatting

LaTeX allows for extensive text formatting options:

1. Bold: `\textbf{bold text}`
2. Italics: `\textit{italic text}`
3. Underline: `\underline{underlined text}`

- Mathematical Typesetting

One of LaTeX's strengths is its ability to handle complex mathematical typesetting. LaTeX uses a special mode for mathematical expressions, enabling precise formatting of equations and symbols. Here are some examples:

Inline Math: Mathematical expressions can be included within the text using dollar signs. For example, $E=mc^2$.

Displayed Equations: For larger equations that need to be displayed on their own lines, you can use the ‘equation’ environment:

```
\begin{equation}
```

$$E = mc^2$$

```
\end{equation}
```

Mathematical Symbols: LaTeX supports a wide range of mathematical symbols, such as Greek letters (α , β , γ), operators (\times , \div , \pm), and functions (\sin , \cos , \log).

4. Including Figures and Tables

Figures: LaTeX makes it easy to include figures and graphics in your document. You can use the ‘graphicx’ package to include images:

```
\usepackage{graphicx}
```

```
\begin{figure}[h]
```

```
\centering
```

```
\includegraphics[width=0.5\textwidth]{example-image}
```

```
\caption{An example image}
```

```
\label{fig:example}
```

```
\end{figure}
```

Tables: LaTeX provides robust tools for creating tables with the `table` and `tabular` environments. Here's an example of a simple table:

```
\begin{table}[h]
```

```
\centering
```

```
\begin{tabular}{|c|c|c|}
```

```
\hline
```

```
Column 1 & Column 2 & Column 3 \\\
```

```
\hline
```

Data 1 & Data 2 & Data 3 \\\

Data 4 & Data 5 & Data 6 \\\

\hline

\end{tabular}

\caption{An example table}

\label{tab:example}

\end{table}

5. Referencing and Citations

Cross-referencing: LaTeX allows you to create cross-references to figures, tables, sections, etc., using labels and references. For example:

See Figure \ref{fig:example} for an example image.

Bibliography and Citations: LaTeX can handle bibliographies and citations using packages like biblatex or natbib. Here's a basic example using biblatex:

```
\usepackage[backend=biber] {biblatex}
```

```
\addbibresource{references.bib}
```

... According to \textcite{einstein1905}, mass and energy are equivalent

```
\printbibliography
```

6. Packages

LaTeX's functionality can be extended through the use of packages. Packages are add-ons that provide additional features or enhance existing ones. Some commonly used packages include:

- amsmath for advanced mathematical typesetting.
- geometry for flexible document layout and margins.
- hyperref for creating hyperlinks within the document.

7. Custom Commands and Environments

LaTeX allows users to define their own commands and environments to simplify repetitive tasks. For example, you can create a custom command for frequently used text:

```
\newcommand{\vect}[1]{\mathbf{#1}}
```


...

The vector `\vect{v}` is defined as follows.

8. Conclusion

LaTeX is a powerful tool for creating high-quality documents with precise formatting. Its extensive range of features makes it ideal for academic and technical writing. While it has a steeper learning curve compared to traditional word processors, the benefits of using LaTeX for complex documents are significant, particularly in terms of consistency, automation, and the ability to handle intricate formatting requirements.

Example:

1. Find the order and degree (if defined) of the differential equation.

$$\frac{d^2y}{dx^2} + x(dy/dx)^2 = 2x^2 \log \frac{d^2y}{dx^2}$$

In Latex:

```
\item Find the order and degree (if defined) of the differential equation
\begin{align*}
\frac{d^2y}{dx^2} + x\brak{\frac{dy}{dx}}^2 = 2x^2 \log\brak{\frac{d^2y}{dx^2}}
\end{align*}
```

2. Using properties of determinants, show that

$$\begin{vmatrix} 3a & -a+b & -a+c \\ -b+a & 3b & -b+c \\ -c+a & -c+b & 3c \end{vmatrix} = 3(a+b+c)(ab+bc+ca)$$

In Latex:

```
\item Using properties of determinants, show that
\begin{vmatrix}
3a & & -a+b & & -a+c \\
-b+a & & 3b & & -b+c \\
-c+a & & -c+b & & 3c
\end{vmatrix} = 3\brak{a+b+c}\brak{ab+bc+ca}
```

3.4 Circuit Designing for Gate questions:

In the circuit shown, the clock frequency, i.e., the frequency of the clk signal, is 12 KHz.

The frequency of the signal at Q2 is ____ KHz. (GATE-EC2019,25)

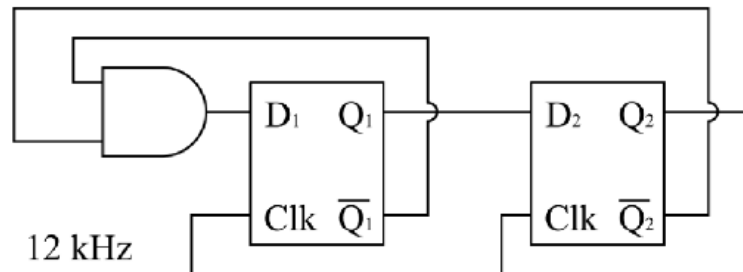


Fig3.3: Circuit Diagram

In Latex:

```
\begin{enumerate}
```

```
\item In the circuit shown, the clock frequency, i.e., the frequency of the clk signal, is  
12 KHz. The frequency of the signal at  $\mathbf{Q2}$  is  
                     KHz.
```

```
\hfill(GATE-EC2019,25)
```

```
\end{enumerate}
```

```
\begin{figure}[H]
```

```
\input{figs/fig.tex}
```

```
\caption{Circuit Daigram}
```

```
\end{figure}
```

```
\end{document}
```

3.4.1 Step-by-Step Guide for call the logic file:

Step 1: Create the Folder

Create a Folder: First, create a folder named figs in your project directory. This folder will store your LaTeX file containing the circuit diagram.

- On most operating systems, you can do this by right-clicking in your project directory and selecting "New Folder," then naming it figs.

Step 2: Create the LaTeX File

Create a File: Within the figs folder, create a new file named fig.tex. This file will contain the LaTeX code for your circuit diagram.

- You can create this file using a text editor such as VS Code, Sublime Text, or even a dedicated LaTeX editor like TeXShop or Overleaf.

Step 3: Write the LaTeX Code

Write the Logic in LaTeX: Open the fig.tex file and write your LaTeX code for the circuit diagram. You can use the circuitikz package for drawing circuit diagrams in LaTeX. Here's an example of how your fig.tex file might look:

```
\documentclass{standalone}
\usepackage{circuitikz}

\begin{document}
\begin{circuitikz}
  \draw
    (0,0) to[battery1] (0,4)
    to[resistor, l=$R_1$] (4,4)
    to[resistor, l=$R_2$] (4,0)
    -- (0,0);
\end{circuitikz}
\end{document}
```

- This code uses the standalone class, which is useful for creating standalone images or diagrams.
- The circuitikz package is used for drawing circuits.

Step 4: Include the Diagram in Your Main Document

Call the File in Your Main Code: In your main LaTeX document (e.g., main.tex), you need to include the figs/fig.tex file. Ensure you have the standalone package installed and included in your preamble. Here's an example of how to do this:

```
\documentclass{article}
\usepackage{standalone}
\usepackage{circuitikz}
\begin{document}
\section{Circuit Diagram}
Here is the circuit diagram:
\input{figs/fig.tex}
\end{document}
```

- The `\input` command is used to include the contents of the figs/fig.tex file in your main document.
- Make sure the path is correct relative to your main document

3.4.2 Logic for Designing Circuit Diagram:

1. `\documentclass{article}`
2. `\usepackage{circuitikz}`
3. `\begin{document}`
4. `\begin{circuitikz}`
5. `\draw (7,2)coordinate (E) -- (9,2)coordinate (F) -- (9,-1)coordinate (G) -- (7,-1)coordinate (H) -- (7,2)coordinate (E);`
6. `\draw (11,2)coordinate (I) -- (13,2)coordinate (J) -- (13,-1)coordinate (K) -- (11,-1)coordinate (L) -- (11,2)coordinate (I);`
7. `\draw (6,1.5) node[and port] (and) {};`
8. `\draw (and.in 1) node[left] {};`
9. `\draw (and.in 2) node[left] {};`
10. `\draw (and.out) |- ($(E)!0.2!(H)$) -- ++(0:0) node[right] {$D1$};`
11. `\draw ($(L)!0.5!(K)$) node[anchor=south] {clk};`
12. `\draw ($(H)!0.5!(G)$) node[anchor=south] {clk};`
13. `\draw(6,-2) node[above] {12 KHz} |- (12,-2);`
14. `\draw($(H)!0.5!(G)$) node[anchor=south,xshift=6] {} -- ++(90:-1) -- ++(0:0) node[left] {};`

```

15. \draw($(L)!0.5!(K)$)node[anchor=south,xshift=6]{ }--++(90:-1)--
    ++(0:0)node[left]{ };
16. \draw($(F)!0.2!(G)$)node[left]{$Q1$}
    ($ (I)!0.2!(L)$)node[right]{$D2$};
17. \draw($(F)!0.8!(G)$)node[left]{$\overline{Q1}$};
18. \draw($(J)!0.2!(K)$)node[left]{$Q2$};
19. \draw($(J)!0.8!(K)$)node[left]{$\overline{Q2}$};
20. \draw ($ (J)!0.2!(K)$)--++(0:1)node[right]{ };
21. \draw(and.in 1) --++(90:2)-|(9.5,1)-($ (F)!0.8!(G)$)(0:0)node[left]{ };
22. \draw(and.in 2) --++(-90:4)-|(13.5,-0.5)- ($ (J)!0.8!(K)$);
23. \end{circuitikz}
24. \end{document}

```

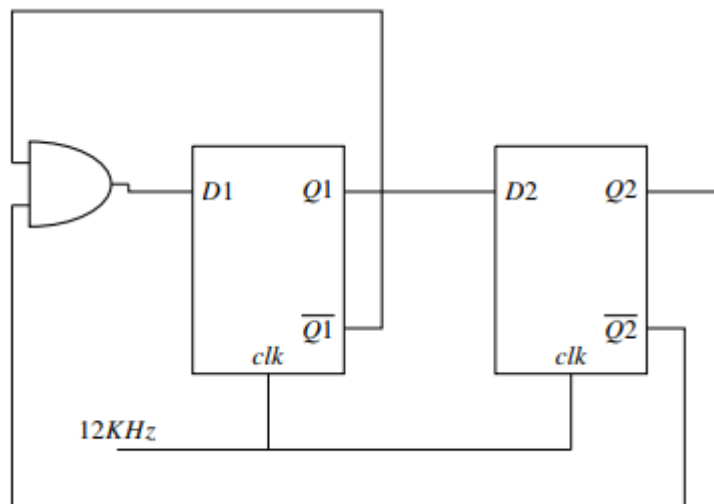


Figure3.4: Designed Circuit Diagram

CHAPTER 4

HARDWARE PROGRAMMING

4.1 Components:

Table4.1: Component Required

Component	Value	Quantity
Arduino	Uno	1
Bread Board		1
Seven Segment Led	Common Anode	1
Lcd	16x2	1
Resistor	$\geq 220\Omega$	1
Led		As Required
Vaman		1
Jumper Wires		As Required

4.2 Seven Segment Display:

A seven-segment display is an electronic display device used to display decimal numerals and some alphabetic characters. It consists of seven LED segments arranged in a pattern to form numbers and certain letters. Here's a detailed explanation of how a seven-segment display works, its types, and how to interface it with an Arduino.

1. Structure:

A typical seven-segment display has seven LEDs arranged in a rectangular fashion, each forming part of a digit (segments labeled A to G) and sometimes an additional dot segment (DP) for decimal points.

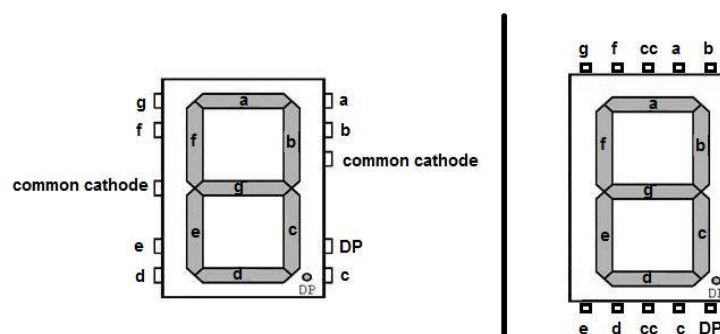


Figure 4.1 Seven Segment Display Pin Diagram

- **Segments:** Each segment can be illuminated individually to form numerals or characters.
- **Common Types:** There are two main types of seven-segments displays:
 - **Common Anode (CA):** All anode connections of the LEDs are connected together and need to be connected to the positive voltage. Individual segments are turned on by grounding their cathode.
 - **Common Cathode (CC):** All cathode connections of the LEDs are connected together and need to be connected to the ground. Individual segments are turned on by applying positive voltage to their anode.

1. Pin Configuration:

A standard seven-segment display usually has 10 pins, arranged to connect each segment individually and to provide common anode or cathode connections.

2. Interfacing a Seven-Segment Display with an Arduino:

Table4.2: Arduino to Seven segment display connections

Arduino	2	3	4	5	6	7	8
Display	a	b	c	d	e	f	g

3. Outputs:

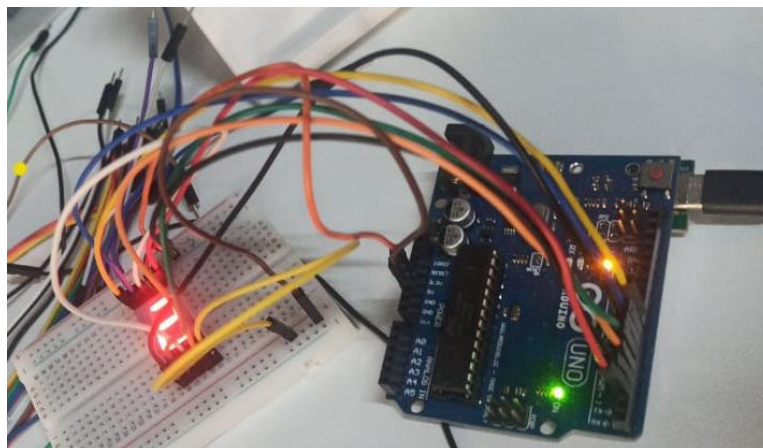


Figure 4.2: Output for Seven Segment Display

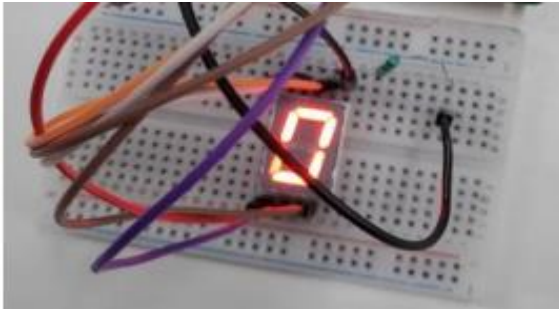


Figure4.3: Output Zero Displaying

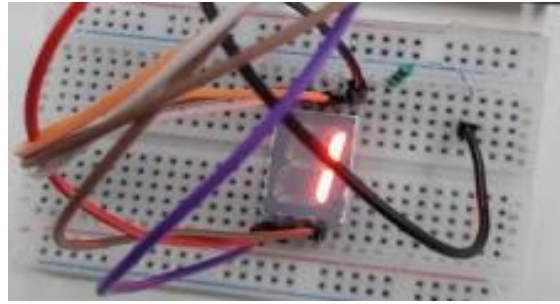


Figure4.4: Output One Displaying

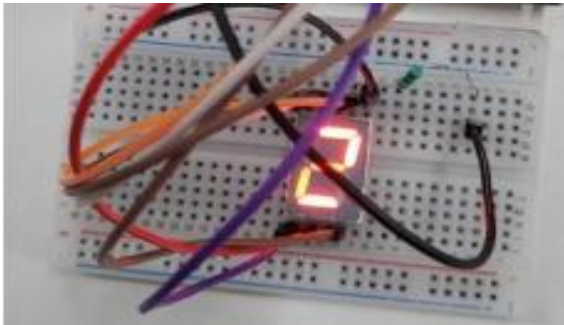


Figure4.5: Output Two Displaying

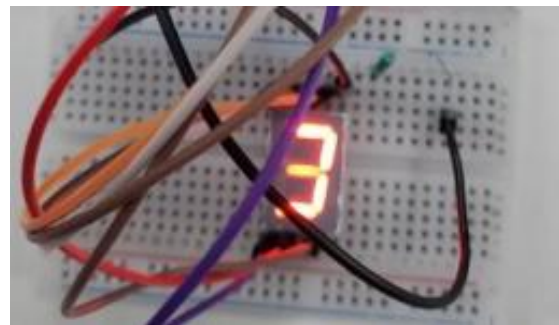


Figure4.6: Output Three Displaying

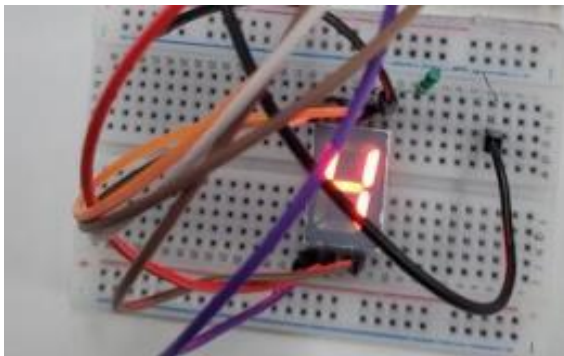


Figure4.7: Output Four Displaying

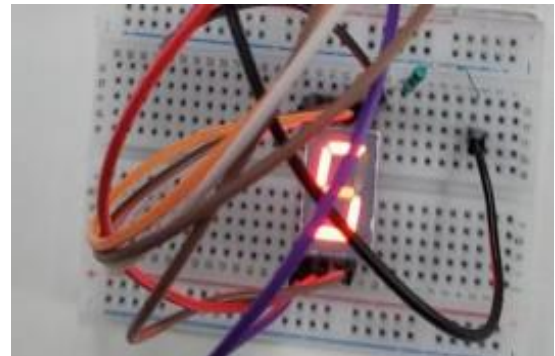


Figure4.8: Output Five Displaying

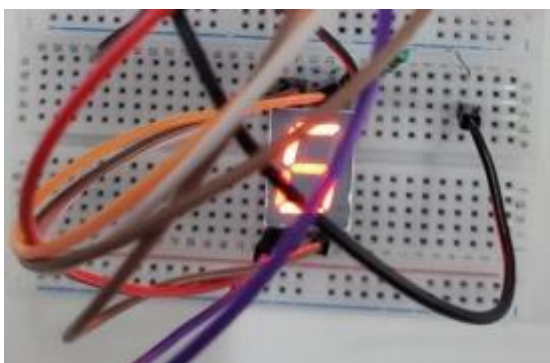


Figure4.9: Output Six Displaying

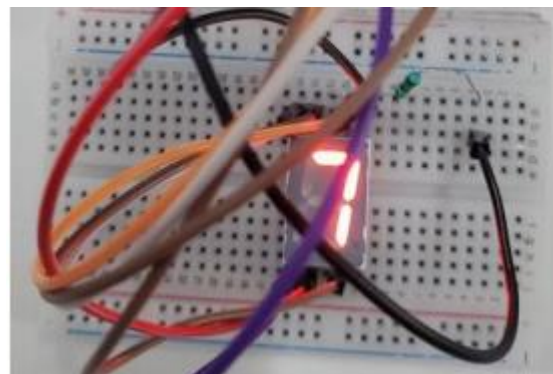


Figure4.10: Output Seven Displaying

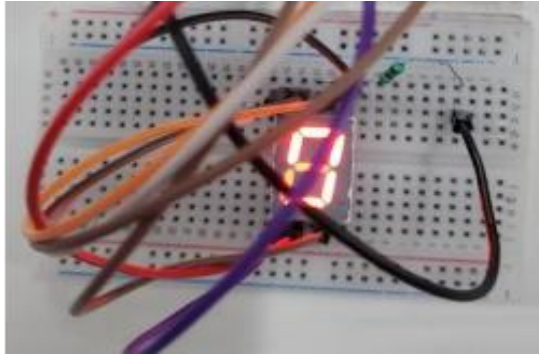


Figure4.11: Output Eight Displaying

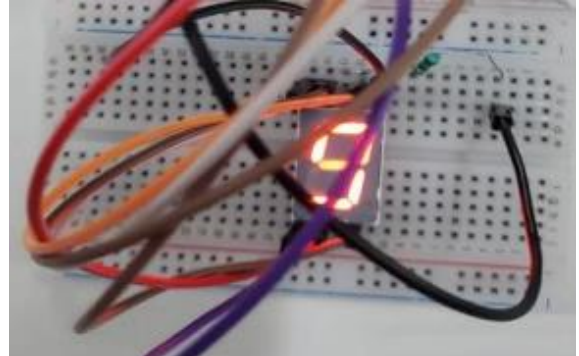


Figure4.12: Output Nine Displaying

4. Conclusion:

A seven-segment display is a versatile component for displaying numerical data and simple characters in many electronics projects. By understanding its structure and how to interface it with microcontrollers like Arduino, you can create a wide range of applications, from simple clocks to advanced data displays.

4.3 7447:

The 7447 is a BCD (Binary Coded Decimal) to 7-segment display driver. It takes a 4-bit BCD input and converts it to signals that drive a common anode 7-segment display.

1. Key Features:

- BCD to 7-segment decoding: Converts binary-coded decimal input into 7-segment display output.
- Common Anode: Designed to drive common anode 7-segment displays.
- Open-collector outputs: Requires external pull-up resistors to function correctly.

2. Pin Configuration:

- Inputs (A, B, C, D): 4-bit BCD input.
- Outputs (a-g): Connect to the segments of the 7-segment display.
- LT (Lamp Test): Turns all segments on when low.
- BI (Blank Input): Turns all segments off when low.
- RBI (Ripple Blanking Input): Used to blank leading zeros.
- GND and Vcc: Power supply pins.

3. Pinout Diagram:

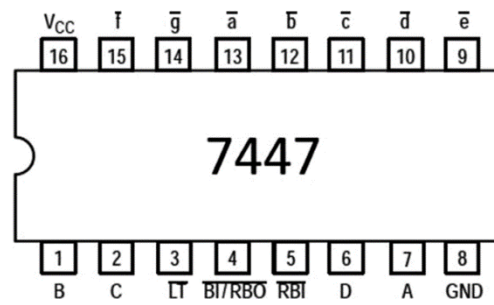


Figure 4.13: 7447 Pin Diagram

4. Functionality:

- Inputs (A-D): Provide the binary-coded decimal value (0000 to 1001).
- Outputs (a-g): Correspond to the 7 segments of the display. Each output controls one segment.
- Lamp Test (LT): When low, all segments are illuminated.
- Blank Input (BI): When low, all segments are off regardless of inputs.
- Ripple Blanking Input (RBI): Used to suppress leading zeroes.

5. Connections:

Table4.3: 7447 to Seven segment display connections

7447	\bar{a}	\bar{b}	\bar{c}	\bar{d}	\bar{e}	\bar{f}	\bar{g}
DISPLAY	A	b	c	d	e	f	g

Table4.4: 7447 to Arduino connections

7447	D	C	B	A
ARDUINO	5	4	3	2

6. Output:

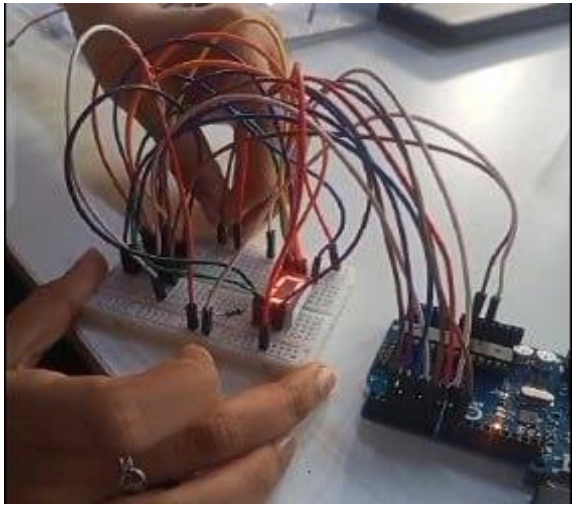


Figure4.14: output of 7447

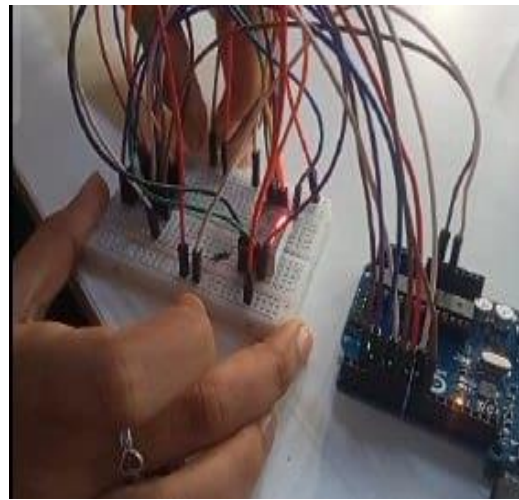


Figure4.15: output of 7447

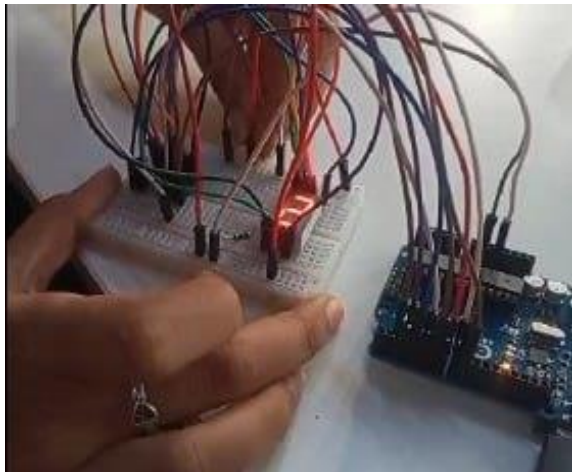


Figure4.16: output of 7447

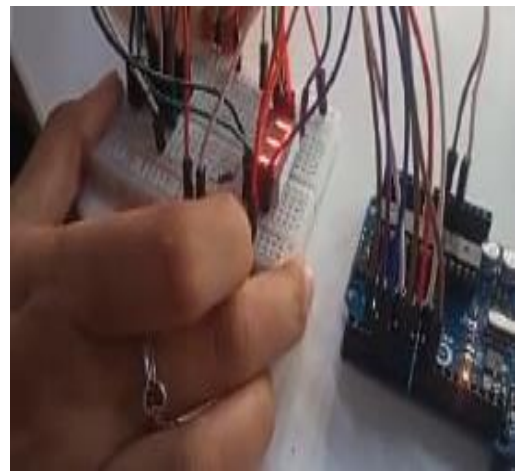


Figure4.17: output of 7447

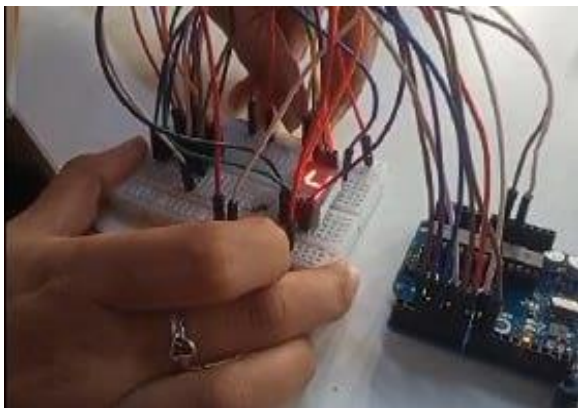


Figure:4.18: output of 7447

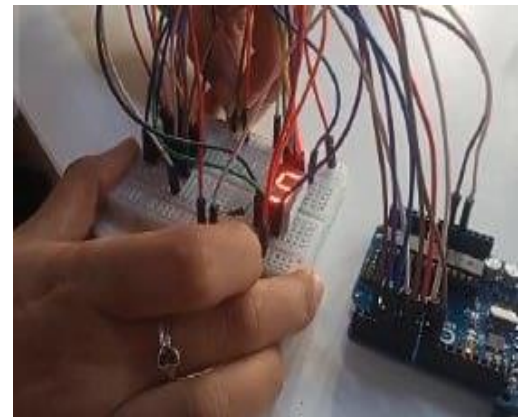


Figure4.19: output of 7447

7. Conclusion: The 7447 is a useful IC for driving 7-segment displays from BCD inputs, simplifying the process of displaying numeric data on these displays. The 7447 is a useful IC for driving 7-segment displays from BCD inputs, simplifying the process of displaying numeric data on these displays.

4.4 7474:

The 7474 integrated circuit (IC) is a dual D-type flip-flop, which is a fundamental digital storage device used in various electronic circuits. Let's break down its components and functionality in detail:

1. The 7474 IC

The 7474 IC is a specific type of flip-flop known as a dual D-type flip-flop. It contains two independent D-type flip-flops in a single package. This means it can store and manipulate two bits of data independently.

2. Pin Configuration and Functions

- The 7474 IC typically comes in a 14-pin Dual In-line Package (DIP). Here is a detailed description of each pin's function:
 - Vcc (Pin 14): This is the power supply pin. The typical operating voltage is 5V.
 - GND (Pin 7): This is the ground pin.
 - Data Input (D): This pin is used to input the data that needs to be stored in the flip-flop.
 - Clock Input (CLK): This pin is used to trigger the flip-flop. The flip-flop captures the input data on the rising edge (low to high transition) of the clock signal.
 - Preset (PRE): This asynchronous input, when activated (usually with a low signal), sets the flip-flop output to high (1) regardless of the clock or data input.
 - Clear (CLR): This asynchronous input, when activated (usually with a low signal), resets the flip-flop output to low (0) regardless of the clock or data input.
 - Q Output: This is the primary output of the flip-flop.
 - Q' (Q-bar) Output: This is the inverted output of the flip-flop (the opposite of Q).

3. Functionality:

The primary function of the 7474 D-type flip-flop is to store the data present at the D input and transfer it to the Q output on the rising edge of the clock pulse. The asynchronous preset and clear inputs allow for immediate setting or resetting of the flip-flop, independent of the clock signal.

- When CLR is low: The flip-flop is reset, and Q is 0.
- When PRE is low: The flip-flop is set, and Q is 1.
- When both PRE and CLR are high: The flip-flop follows the clock signal.
 - On the rising edge of the clock signal, the data at the D input is transferred to the Q output.
 - If D is 1, Q becomes 1.
 - If D is 0, Q becomes 0.

4. Connections:

Table4.5: 7474 AND 7447 to Arduino connections for 7474

	INPUT				OUTPUT				Clock		5V			
	W	X	Y	Z	A	B	C	D						
Arduino	D6	D7	D8	D9	D2	D3	D4	D5	D13					
7474	5	9			2	12			Clk1	Clk2	1	4	10	13
7474			5	9			2	12	Clk1	Clk2	1	4	10	13
7447					7	1	2	6			16			

5. Pin Diagram:

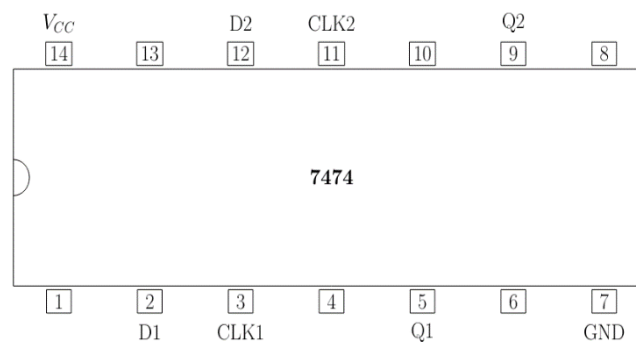


Figure4.20: 7474 Pin Diagram

6. Output:

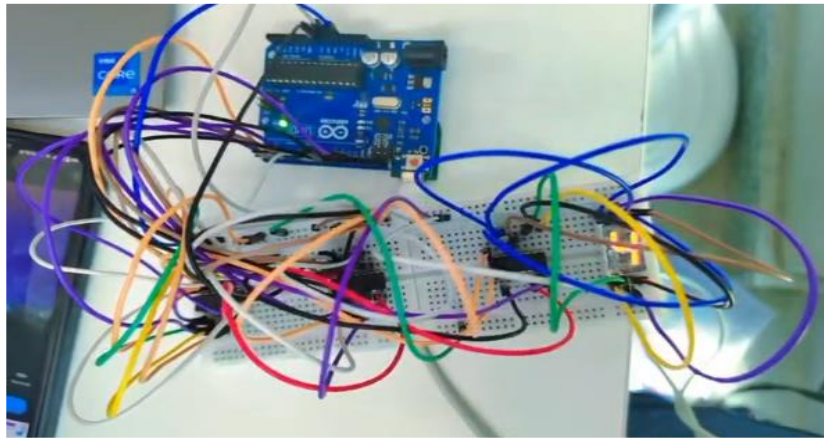


Figure4.21: Output for 7474

7. Conclusion:

The 7474 dual D-type flip-flop IC is a versatile and essential component in digital electronics, enabling data storage, synchronization, and processing in various applications. Understanding its pin configuration, functionality, and applications is crucial for designing and troubleshooting digital circuits.

4.5 State Machine:

A decade counter is a type of counter that counts from 0 to 9 (or in reverse, from 9 to 0) and then resets to 0 (or 9). When designing a reverse decade counter (counting down from 9 to 0), using the 7474 IC (which contains D-type flip-flops), we need to focus on the following aspects:

- State Representation: Representing each number (9 to 0) as a state.
- State Transitions: Moving from one state to the next (9 to 8, 8 to 7, etc.).
- Clock Signal: Driving the transitions.
- Output Representation: Representing the current state/output.

1. State Representation:

In a reverse decade counter, we have 10 states, corresponding to the numbers 9 to 0. For simplicity, we can represent these states in binary form, which is suitable for digital circuits:

- $9 = 1001$
- $8 = 1000$
- $7 = 0111$
- $6 = 0110$
- $5 = 0101$
- $4 = 0100$
- $3 = 0011$
- $2 = 0010$
- $1 = 0001$
- $0 = 0000$

2. State Transitions:

The counter needs to move from one state to the previous one on each clock pulse:

- From 9 to 8
- From 8 to 7
- ...
- From 1 to 0
- From 0 back to 9 (to make it a continuous loop)

3. Using 7474 IC for Implementation:

The 7474 IC has dual D-type flip-flops. To implement a decade counter, we'll use 4 flip-flops (since 2 ICs are needed for 4 flip-flops), as we need to represent states from 0 to 9 in binary (which requires 4 bits).

4. Modifications in Hardware:

Here's a step-by-step approach to modifying the hardware:

- **Connecting Flip-Flops:** Connect the four D-type flip-flops in series to form a 4-bit register.
- **Feedback Mechanism:** Implement feedback to set the next state based on the current state.
- **Clock Input:** Use a common clock signal for all flip-flops to ensure they transition simultaneously.
- **Reset Logic:** Ensure that when the counter goes from 0 (0000), it transitions to 9 (1001).

5. Pin Connections for Two 7474 ICs (total 4 flip-flops):

- Clock (CLK): Connect a common clock signal to the CLK inputs of all four flip-flops.
- Data (D) Inputs: Connect the D inputs to the appropriate logic that determines the next state based on the current state.
- Outputs (Q): The Q outputs represent the current state (4-bit binary value).
- Clear (CLR) and Preset (PRE): Typically, these are held inactive (high) unless an asynchronous reset or set is required.

6. Connections:

Table4.6: 7474 AND 7447 to Arduino connections for FSM

	INPUT				OUTPUT				Clock		5V			
	W	X	Y	Z	A	B	C	D						
Arduino	D6	D7	D8	D9	D2	D3	D4	D5	D13					
7474	5	9			2	12			Clk1	Clk2	1	4	10	13
7474			5	9			2	12	Clk1	Clk2	1	4	10	13
7447					7	1	2	6			16			

7. Output:

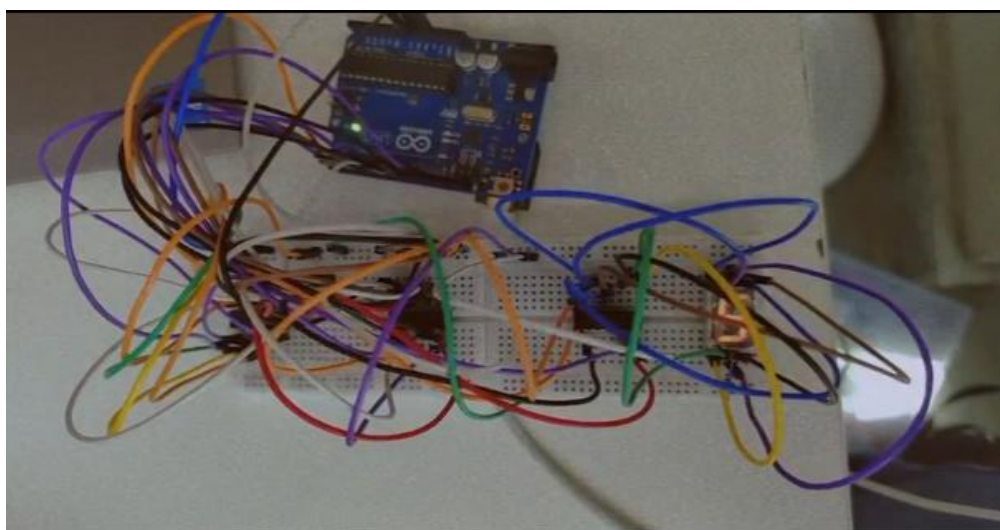


Figure 4.22: Output for FSM

CHAPTER 5

VAMAN BOARD

5.1 INTRODUCTION

The Indian Institute of Technology Hyderabad (IIT Hyderabad) has been at the forefront of technological innovation, contributing significantly to various fields of science and engineering. One of its notable achievements is the development of the Vaman LC, a Liquid Crystal display module that exemplifies advanced technology and engineering excellence. The term "Vaman" denotes something small yet powerful, and this is reflected in the Vaman LC's compact and efficient design. IIT Hyderabad's research team has focused on creating a Liquid Crystal display that not only saves space but also enhances performance and energy efficiency. The Vaman LC leverages cutting-edge materials and innovative engineering techniques to achieve high resolution, superior brightness, and low power consumption, making it an ideal choice for modern electronic devices.

The Vaman LC's energy-saving features not only improve the performance and usefulness of electronics, but they also advance sustainable technology. It also has several advanced features such as high resolution to provide clear images, essential for applications requiring detailed visual output it also designed to consume minimal power, extending battery life in portable devices. This board consists of different modules like arm, esp, fpga it has 50+50 (100) 2.54mm Pins for a wide variety of uses and flexibility (esp32+pygmy). The module used is Espressif ESP-32 WiFi+BLE to provide wireless connectivity and programming an micro-USB port for Power, Programming and wired Interface.

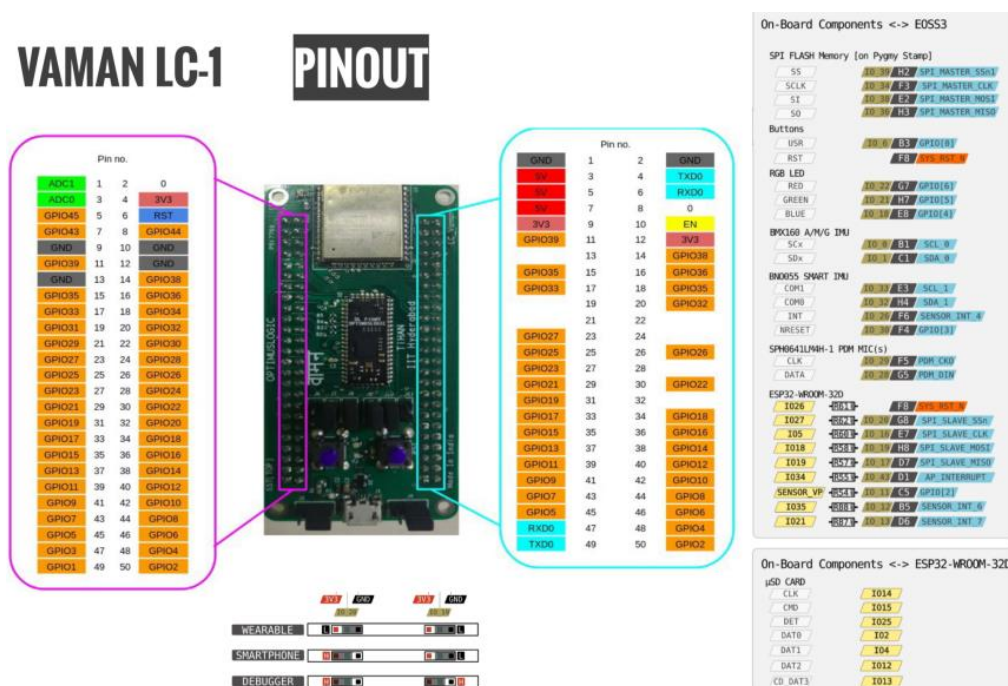


Figure 5.1: Vaman LC Pin Diagram

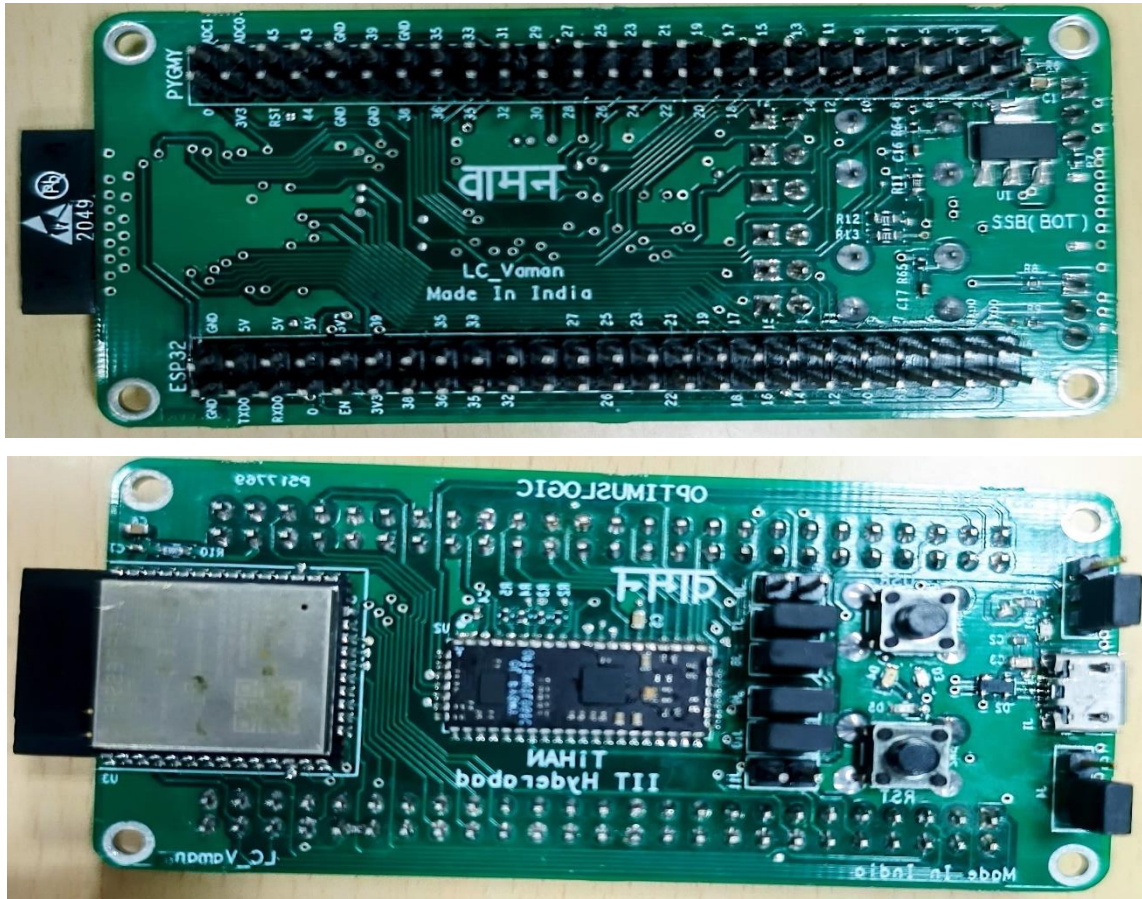


Figure5.2: Vaman LC Board

5.2 VAMAN- ESP32

The ESP32 microcontroller, a versatile and powerful component that enhances the board's capabilities. the Vaman-ESP32 board lies the ESP32 microcontroller, a robust and efficient system-on-chip (SoC) solution. The ESP32 is renowned for its dual-core architecture, which allows for high processing power and multitasking capabilities. This dual-core design enables the Vaman-ESP32 board to handle complex computations and multiple tasks simultaneously, making it ideal for sophisticated applications. The ESP32 also features a rich set of input/output (I/O) pins, facilitating the connection of various sensors, actuators, and other peripherals. This versatility in I/O options allows developers to create a wide range of projects, from simple sensor monitoring systems to complex automation solutions. One of the standout features of the ESP32 microcontroller is its built-in Wi-Fi and Bluetooth connectivity. The integrated Wi-Fi module supports the 802.11 b/g/n standards, providing robust and reliable wireless networking capabilities. This makes the Vaman-ESP32 board an excellent choice for applications that require internet access, such as remote monitoring, data logging, and cloud-based services.

3.2.1 FLASHING VAMAN-ESP32 USING ARDUINO

Table 5.1: Connection between Vaman and Arduino

VAMAN LC PINS	RPI PINS
3.3	3.3
GND	GND
TXD0	TXD
RXD0	RXD
0	GND
EN	GND

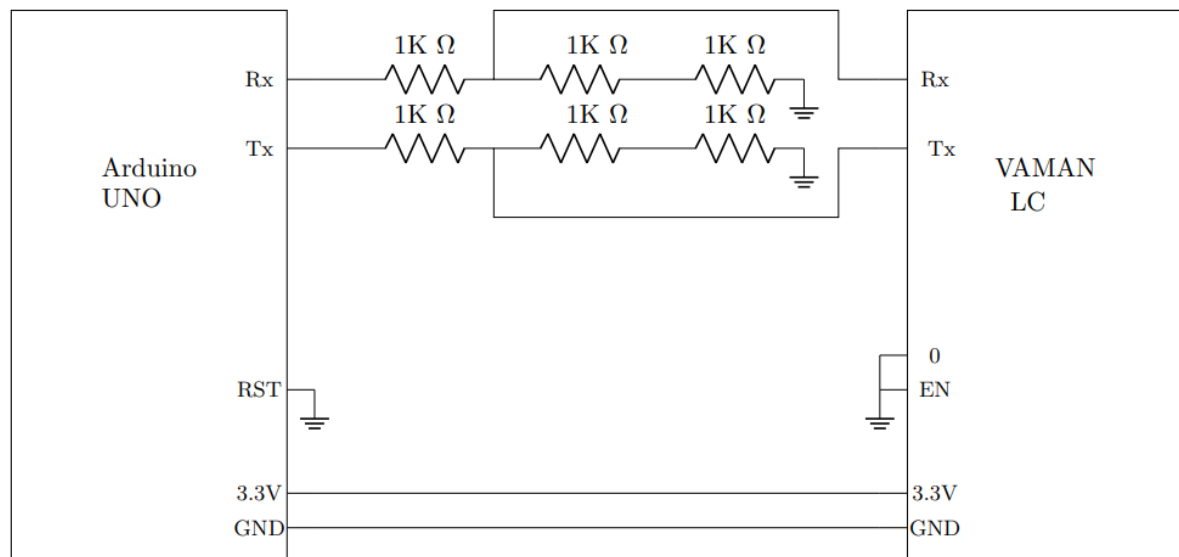


Figure 5.3: Circuit Connections

Steps required to execute code with ESP in Vaman

- Connect the ESP32 pins to Arduino UNO as per table 3.1
- Execute the code after editing the wifi credentials and path of code vaman/esp32/codes/setup
- Open the Droid Application
- Click the three dots in the top right corner
- Navigate to settings → DOIT ESP32 DEVKIT V1

- f) Change the upload speed to 115200
- g) Upload the generated .bin file

For example,

A 6 $\frac{1}{2}$ digit timer – counter is set in the ‘time period’ mode of operation and the range is set as ‘ns’. For an input signal, the timer-counter displays 1000000. With the same input signal, the timer-counter is changed to ‘frequency’ mode of operation and the range is set as ‘Hz’. The display will show the number ____



Figure 5.4: Example Output

5.3 VAMAN ARM

The Vaman LC ARM is designed to be extremely compact, addressing the increasing demand for smaller and more efficient electronic components. Its small form factor makes it ideal for integration into devices where space is limited, such as portable electronics, wearable technology, and miniature IoT devices. The compact size does not compromise performance, making the Vaman LC ARM a versatile choice for various applications. Energy efficiency is a critical feature of the Vaman LC ARM. Both the Liquid Crystal display and the ARM microcontroller are optimized for low power consumption, which is essential for battery-powered devices. This feature ensures that devices using the Vaman LC ARM can operate longer between charges, enhancing their usability and sustainability. It is particularly beneficial for applications in remote or mobile environments where recharging options are limited.

The Vaman LC ARM is designed with ease of prototyping in mind. Its compatibility with various development tools and environments simplifies the process of designing, testing, and iterating new products. Developers can quickly prototype and refine their ideas, accelerating the development cycle and reducing time to market. This ease of use is crucial for innovation, enabling rapid experimentation and optimization of electronic solutions. Scalability is one of the standout features of the Vaman LC ARM. Its modular design and extensive peripheral support allow it to be used in a wide range of applications, from simple gadgets to complex systems. This scalability ensures that the Vaman LC ARM can grow with the needs of a project, supporting initial prototyping stages through to full-scale commercial production. This adaptability makes it an attractive option for developers looking to future-proof their designs.

Table 5.2: Seven segment Display-Vaman connection

Display	Vaman
a	IO_4
b	IO_5
c	IO_6
d	IO_7
e	IO_8
f	IO_10
g	IO_11
COM	3.3 V

Steps required to execute code with ARM in Vaman

- Connect the Vaman pins to display as per table 3.2
- Execute the code after editing the wifi credentials and path of code
vaman/arm/codes/setup
- Follow the instructions at vaman/installation/termuxdebian/termuxdebian_arm.txt
- Open the Droid Application
- Click the three dots in the top right corner
- Navigate to settings → DOIT ESP32 DEVKIT V1
- Change the upload speed to 115200
- Upload the generated .bin file

For example,

The chip logic for a certain DRAM chip in a memory system design is shown below. Assume that the memory system has 16 address lines denoted by A_{15} to A_0 . What is the range of addresses (in hexadecimal) of the memory system that can get enabled by the chip select (CS) signal?

- (A) C800 to CFFF
- (B) CA00 to CAFF
- (C) C800 to C8FF
- (D) DA00 to DFFF

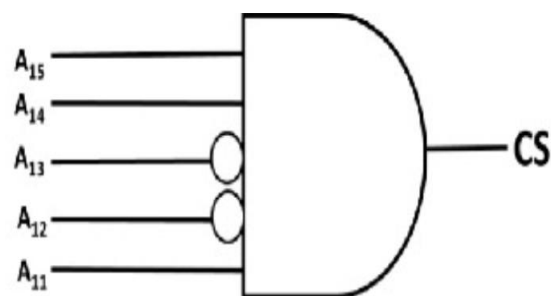


Figure 5.5: Memory system design from A_{11} to A_{15}

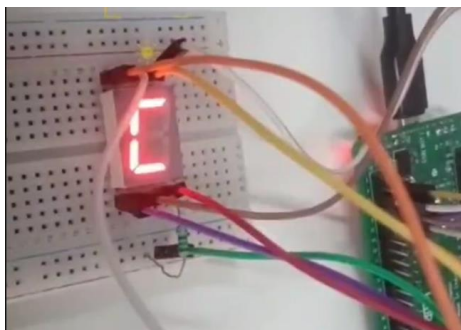


Figure 5.6: Output ARM

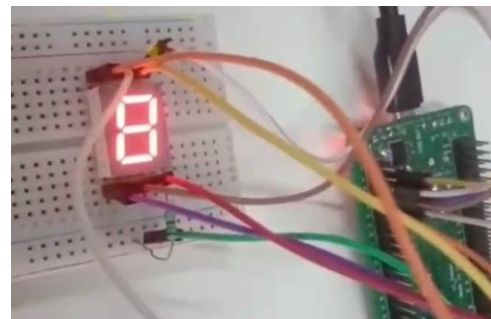


Figure 5.7: Output ARM

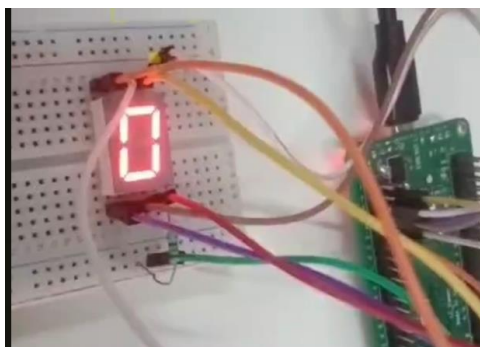


Figure 5.8: Output ARM

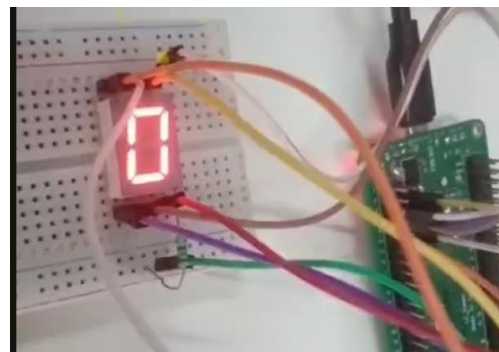


Figure 5.9: Output ARM

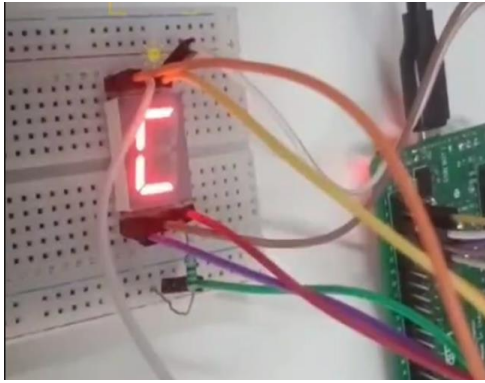


Figure 5.10: Output ARM

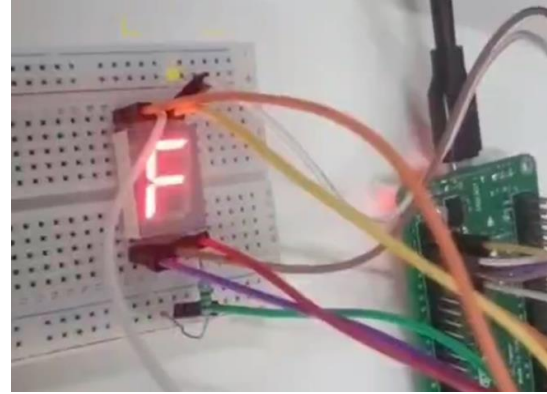


Figure 5.11: Output ARM

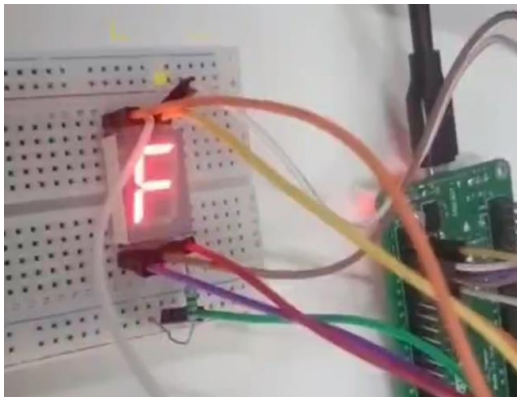


Figure 5.12: Output ARM

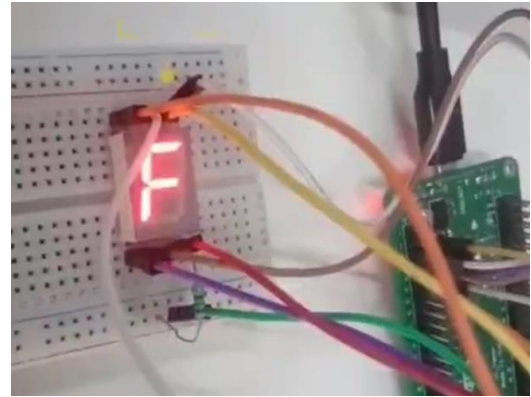


Figure 5.13: Output ARM

5.4 VAMAN FPGA

The Vaman-Pygmy board, a product of IIT Hyderabad's innovation, integrates a Field-Programmable Gate Array (FPGA) to provide advanced programmable logic resources. This essay elaborates on the capabilities and applications of the FPGA on the Vaman-Pygmy board, emphasizing its role in implementing custom digital circuits, performing hardware-accelerated computations, and utilizing Verilog coding for programming. An FPGA is a semiconductor device that can be configured by the user after manufacturing – "field-programmable." The FPGA on the Vaman-Pygmy board offers extensive programmable logic resources, which include a large array of configurable logic blocks (CLBs), interconnects, and input/output blocks. The reconfigurability of the Vaman LC FPGA is a key advantage, enabling the hardware to be reprogrammed and repurposed as needed. This reconfigurability provides several important benefits such as adaptability of changing requirements, cost-effective development and versatile application

Steps required to execute code with Pygmy in Vaman

- Connect the Vaman pins to display as per table 3.2
- Execute the code after editing the wifi credentials and path of code
vaman/fpga/codes/setup
- Follow the instructions at vaman/installation/termuxdebian/termuxdebian_fpga.txt
- Open the Droid Application
- Click the three dots in the top right corner
- Navigate to settings → DOIT ESP32 DEVKIT V1
- Change the upload speed to 115200
- Upload the generated .bin file

For example,

Two T-flip flops are interconnected as shown in the figure. The present state of the flip flops are: $A=1$, $B=1$. The input x is given as 1, 0, 1 in the next three clock cycles. The decimal equivalent of $(AB_y)_2$ with A being the MSB and y being the LSB, after the 3rd clock cycle is __

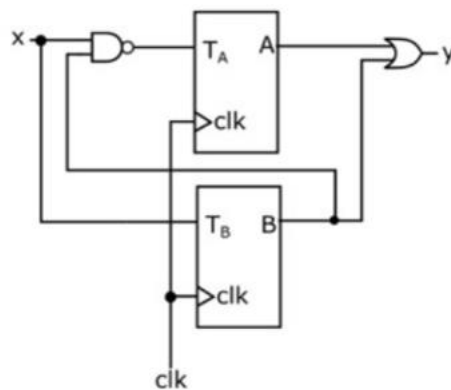


Figure 5.14: Interconnection of T-flip flop



Figure 5.15: Output of FPGA

CHAPTER 6

UBUNTU INSTALLATIONS AND MATH COMPUTATION

6.1 Ubuntu:

Ubuntu is officially released in multiple editions: Desktop, Server, and Core for Internet of things devices and robots.

Ubuntu is a free and open-source distribution of Linux. It is an OS for cloud computing to support Open Stack. Ubuntu is integrated by the Canonical Community and it's freely available. Canonical Ltd. is also liable for the Ubuntu funding.

It is designed for network servers, smartphones, and computers. The system is integrated by a UK-based enterprise which is known as Canonical Ltd. Every principle used for developing the Ubuntu software is based on open-source software development principles.

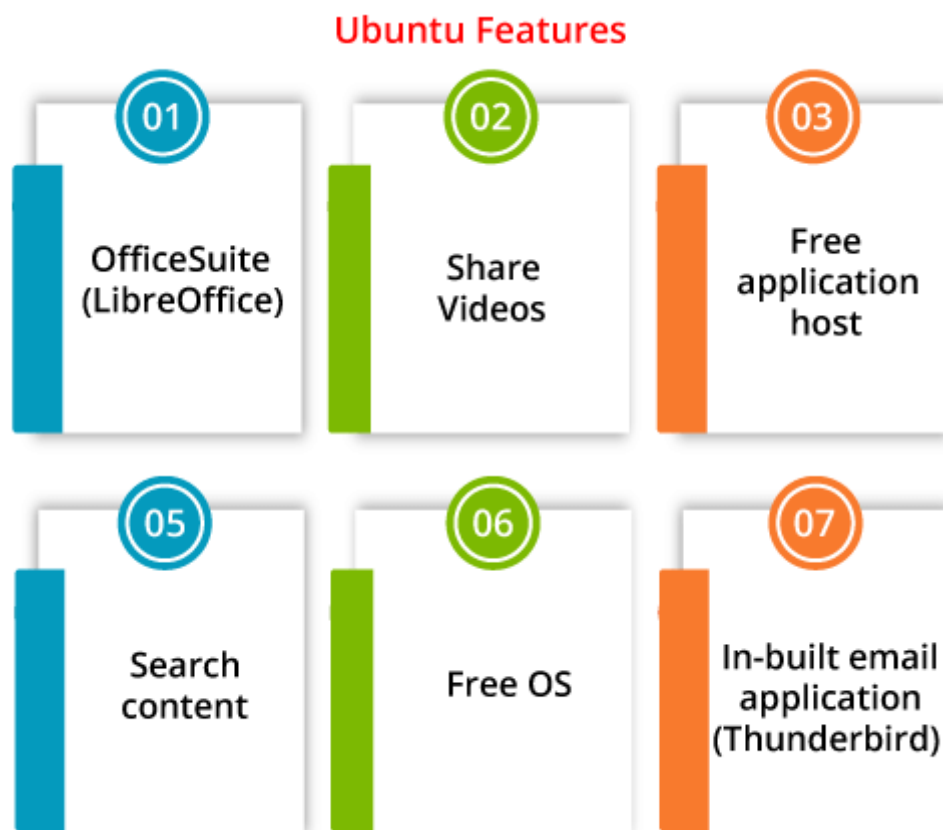


Figure6.1: Features of Ubuntu

- The desktop release of Ubuntu supports every basic software on Windows like VLC, Chrome, Firefox, etc.
- It supports *OfficeSuite* which is known as *LibreOffice*.
- Ubuntu contains an in-built email application which is known as *Thunderbird* which provides the user access to email like Hotmail, Gmail, exchange, etc.
- Also, there are many applications for managing videos and they permit the users for sharing videos as well.
- The free application's host is also available for users for viewing and editing photos.

- It is easy to search content in Ubuntu using the smart searching feature.
- The best aspect is that it is a free OS and is backed by a large open-source community.

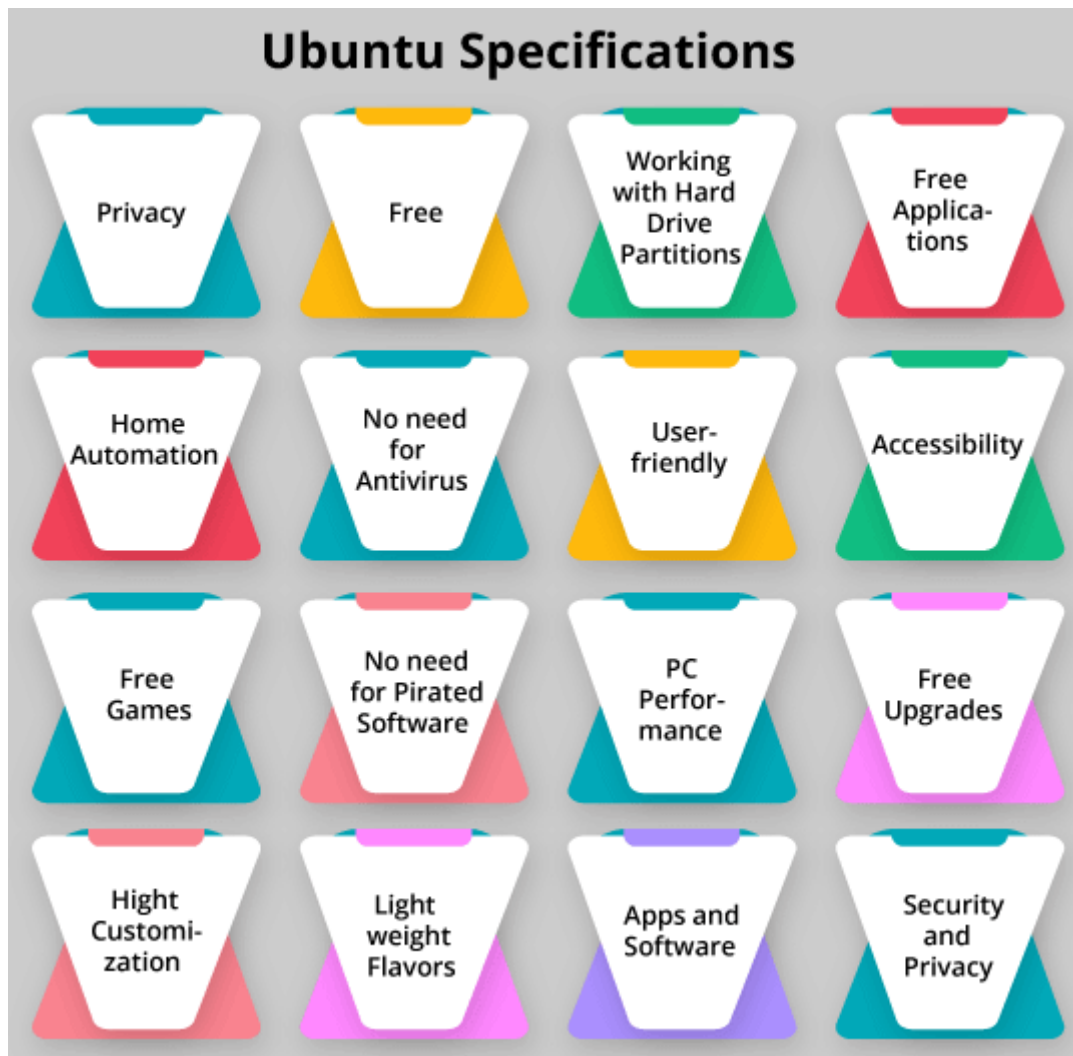


Figure6.2: Specifications of Ubuntu

Ubuntu offers a better option for security and privacy as compared to Windows. The best benefit of having Ubuntu is that we can inherit the needed privacy and extra security without having any solution from a third party. Hacking and several other attack risks can be decreased by having this distribution.

There is a popular myth about Ubuntu is that it is integrated for coders and developers only. But the real fact is that Ubuntu is a good desktop operating system and could be used by any normal computer user.

The risk of having malware or virus is minimal that decreases the anti-virus software cost. Also, anti-virus is a cause for the slowdown of our computer system and badly affects the performance. It may also lead the system to not efficiently function. Anti-virus also inherits a

lot of space in the system apart from this problem. But each of these problems and risks can be avoided in an affordable way with Ubuntu.

We may have heard people saying that Linux operating system is more secure than others, but they are referring to its lack of Linux-targeted viruses and open-source nature. The source code is open for everyone to make changes or add code when we say that software or operating system is open source. Several developers and people work together for fixing security loopholes and issues.

Also, Ubuntu can collect our hardware information (GPU, CPU, RAM), usage data, and location data. However, we can opt-out at the time of the installation process or within the setting when the installation is done.

Some things that Ubuntu OS does better than Window

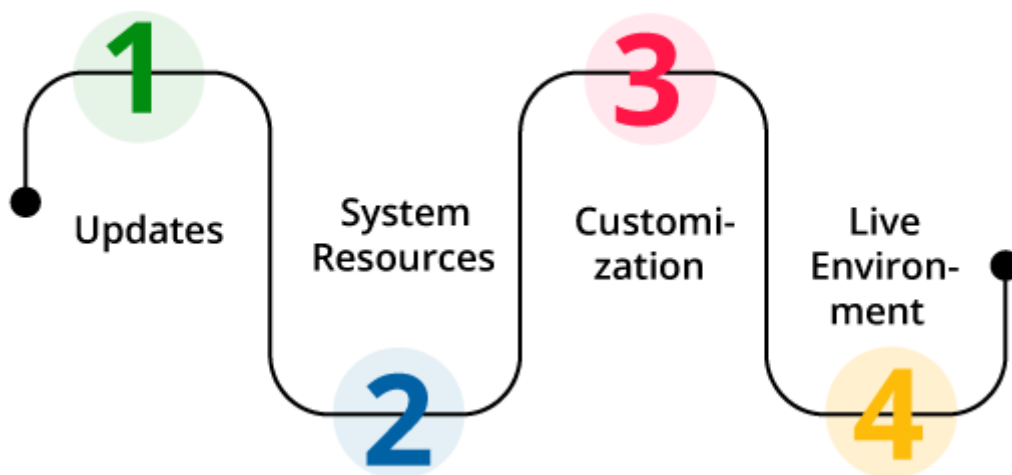


Figure6.3: Pros of Ubuntu over Windows

6.2 Installations:

This contains steps to install fpga and run successfully Steps to follow:

1. Install the zip provided i.e fpga.zip onto Desktop
2. now open terminal and go to the fpga folder
3. In here /Desktop/fpga type: `chmod +x setup.sh -->` This command is particularly useful for making scripts and programs executable. In this way, users can run them without typing their full path or using a dot slash (./) before their name.
4. `sudo bash setup.sh`

5. After running the above command, Go to: file Manager --> Other Locations --> Computer --> root . Copy the files Symbiflow and Pygmy-dev paste them in "/desktop/fpga"
6. Now download arch.tar.gz using this below link and also place it in /desktop/fpga:

https://iith-my.sharepoint.com/:u:/g/personal/gadepall_ee_iith_ac_in/Ebot5QHEYXBAo-7n4hmvJu0B8vMrTIdj_COHJC2cmDY1ww?e=bqDxHI
7. Make sure that /desktop/fpga contains pygmy-dev, setup.sh and symbiflow and zip file arch.tar.gz
8. Next: At Home:
9. `sudo apt update -y`
10. `sudo apt upgrade -y`
11. `sudo apt install openssh-server sshpass build-essential libssl-dev libffi-dev python3-dev bison flex git tcl-dev tcl tcl-tclreadline libreadline-dev autoconf libtool make automake texinfo pkg-config libusb-1.0-0 libusb-1.0-0-dev gcc-arm-none-eabi libnewlib-arm-none-eabi telnet python3 apt-utils libxslt-dev cmake curl python3-pip python3-venv -y`
12. `pip3 install gdown lxml simplejson`
13. `sudo apt install openssh-server sshpass`
14. `sudo apt install build-essential libssl-dev libffi-dev python3-dev bison flex git tcl-dev tcl tcl-tclreadline libreadline-dev autoconf libtool make automake texinfo pkg-config libusb-1.0-0 libusb-1.0-0-dev gcc-arm-none-eabi libnewlib-arm-none-eabi telnet python3 apt-utils libxslt-dev python3-lxml python3-simplejson cmake curl python3-setuptools python3-pip`
15. `cp arch.tar.gz /home/username/Desktop/fpga/arch.tar.gz`
16. `export INSTALL_DIR=/home/"Username"/Desktop/fpga/symbiflow`
17. `tar -C $INSTALL_DIR -xvf /home/username/Desktop/fpga/arch.tar.gz`

18. `export PATH="$INSTALL_DIR/quicklogic-arch
defs/bin:$INSTALL_DIR/quicklogic-arch-defs/bin/python3:$PATH"`
19. `cd /home/username/Desktop/fpga/pygmy-dev/tools/quicklogic-fasm`
20. `nvim requirements.txt`
21. Replace contents with these lines:
22. `-e git+https://github.com/symbiflow/fasm.git#egg=fasm`
23. `-e git+https://github.com/antmicro/quicklogic-fasm-utils.git#egg=fasm-utils`
24. `python3 -m venv venv`
25. `source venv/bin/activate`
26. `pip3 install -r requirements.txt`
27. `sudo python3 setup.py install`
28. `cd /home/username/Desktop/fpga/pygmy-dev/tools/quicklogic-yosys`
29. `make config-gcc`
30. `make -j4 install PREFIX=$INSTALL_DIR`
31. `cd /home/username/Desktop/fpga/pygmy-dev/tools/yosys-symbiflow-plugins`
32. `export PATH=$INSTALL_DIR/bin:$PATH`
33. `make -j4 install`
34. `cd /home/username/Desktop/fpga/pygmy-dev/tools/vtr-verilog-to-routing`
35. `make -j4`
36. `nvim ~/.bashrc`
37. #paste the following 3 lines at the end of the file
38. `export INSTALL_DIR=/home/username/Desktop/fpga/symbiflow`

39. `export PATH="$INSTALL_DIR/quicklogic-arch-defs/bin:$INSTALL_DIR/quicklogic-arch-defs/bin/python3:$PATH"`
40. `export PATH=/home/username/Desktop/fpga/symbiflow/bin:$PATH`
41. `source ~/.bashrc`
42. `vpr -h`
43. `yosys -h`
44. `qlfasm -h`
45. `ql_symbiflow -h`
46. `cd $INSTALL_DIR/quicklogic-arch-defs/tests/counter_16bit`
47. `ls`
48. `ql_symbiflow -compile -d ql-eos-s3 -P pd64 -v counter_16bit.v -t top -p chandalar.pcf -dump binary`
49. `pip3 install gdown lxml simplejson`
50. `ql_symbiflow -compile -d ql-eos-s3 -P pd64 -v counter_16bit.v -t top -p chandalar.pcf -dump binary`
51. `ls`
52. `cd /home/username/Desktop/fpga`
53. `ls`
54. `mkdir fpga-examples`
55. `cd fpga-examples`
56. `svn co https://github.com/gadepall/vaman/trunk/fpga/setup/codes/blink`
57. `cd blink`
58. `ls`

59. Command to compile code:
60. `ql_symbiflow -compile -src /home/username/Desktop/fpga/fpga-examples/blink -d ql-eos-s3 -P PU64 -v helloworldfpga.v -t helloworldfpga -p quickfeather.pcf -dump binary`
61. `ls`
62. `nvim helloworldfpga.v -->` This file contains the code
63. `nvim quickfeather.pcf -->` This file contains pin configurations
64. Make sure to compile the file after modifying .v or .pcf file
65. Command to upload code to vaman:
66. #Make sure that vaman is in upload mode(i.e Green Light)
67. # After Compiling the code make sure to copy the bin file and paste it in home and perform the upload command


```
sudo python3 /home/username/TinyFPGA-Programmer-Application/tinyfpga-programmer-gui.py --port /dev/ttyACM0 --appfpga /home/username/helloworldfpga.bin --mode fpga
```

6.3 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.

TO FIND IF POINTS ARE COLLINEAR OR NOT

Enter the values of points A,B,C

Enter the values of Point A:

Enter the values of Point B:

Enter the values of Point C:

Submit



▲ 192.168.111.186



The points are collinear B divides AC in the ratio: 0.67)
[Return to Home Page](#)

Figure6.4: Math computing using Vaman ESP32

CHAPTER 7

INTER-CHIP COMMUNICATION USING VAMAN BOARD

7.1 Inter-Chip Communication:

Inter-chip communication refers to the exchange of data between integrated circuits (ICs) within an electronic device. Inter-chip communication ensures these various components work together efficiently, exchanging data and control signals as needed. Crucial for the performance, reliability, and power efficiency of electronic devices.

Used in:

- Automotive systems
- Industrial automation
- Medical devices

• ESP-FPGA:

This section describes various experiments that interface the ESP32 and FPGA present on the LC Vaman.

Table 7.1: Connections between Vaman ESP to Pygmy

Pin	Vaman-ESP32	Vaman-Pygmy
CS/SS	27	IO_20
CIPO/MISO	19	IO_17
COPI/MOSi	18	IO_19
SCLK	5	IO_16

• ESP-ARM:

This section describes various experiments that interface the ESP32 and ARM present on the LC Vaman.

• ARM-FPGA:

This section describes various experiments that interface the FPGA and ARM present on the LC Vaman.

7.1.1 Seven segment Display:

This subsection illustrates the use of SPI Communication between the Vaman-ESP32 and the FPGA onboard the Vaman-Pygmy by controlling a seven-segment display remotely.

Components Required:

Table 7.2: Seven Segment Display Components

Component	Value	Quantity
Vaman	Lc	1
Uart		1
Usb-Cable	Type-B	1
7 Segment Led	Common Anode	1
Resistor	10k	1
Breadboard		1
Jumper Wires		Required

Connections:

Table 7.3: Connections between Vaman and Display

7 segment display	pygmy
a	IO_4
b	IO_5
c	IO_6
d	IO_7
e	IO_8
f	IO_10
g	IO_11
COM	3V3

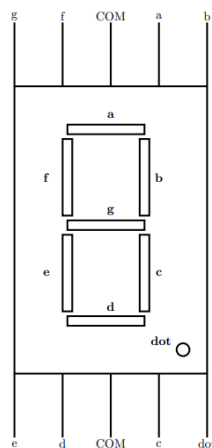


Figure7.1: Seven Segment Pins

Building and Flashing:

- Build and flash ESP32 project using PlatformIO or ArduinoDroid.
- Build FPGA project and flash using tinyfpga-programmer-gui.py.
- Flash the project .bin file using USB-UART connected to the Vaman-ESP32, via PlatformIO or ArduinoDroid.
- Build the FPGA project .bin file by entering the following commands at a terminal window.
- #The following variable can be sourced from .shellrc or .venv/bin/activate export QORC_SDK_PATH=/path/to/pygmy-sdk
- -cd inter-chip/esp32-fpga/led/codes/fpga
- make
- On building successfully, the .bin file is generated at
- inter-chip/esp32-fpga/led/codes/fpga/rtl/AL4S3B FPGA Top.bin
- Flash the .bin file to the Vaman-Pygmy by resetting it to bootloader mode and entering the following command at a terminal window,
- python3 /path/to/tinyfpga-programmer-gui.py --port /dev/ttyACMxx --mode fpga --appfpga
- /path/to/AL4S3B FPGA Top.bin --reset

Demonstration:

- Find the IP address of the Vaman-ESP32 by inspecting the output of the serial terminal, or by typing at a terminal window
- By the Commands:
- ifconfig
- nmap -sn xx.yy.zz.0/24

where xx.yy.zz represents the first three octets of the IP address of your device on the WiFi network interface, found using ifconfig.



Figure 7.2: web page for selecting output to be displayed

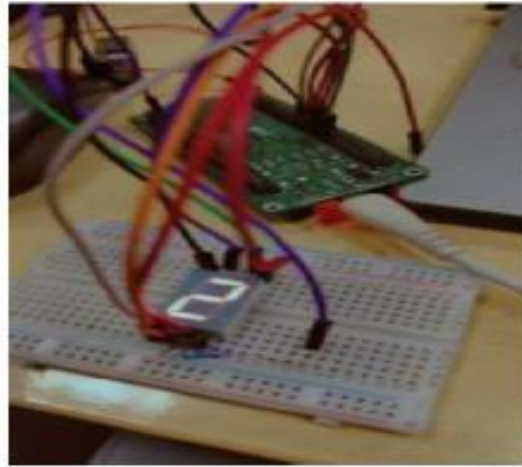


Figure7.3: Seven Segment display output

7.1.2 LCD

Components:

Table7.4: Components for LCD

Component	Value	Quantity
Vaman	Lc	1
Uart		1
Usb-Cable	Type-B	1
Lcd	16x2	1
Resistor	10k	1
Breadboard		1
Jumper Wires		Required

Connections:

Table 7.5: Connections Between LCD and Vaman

Pygmy pins	LCD Pins	LCD Pin Label	LCD Pin Description
GND	1	GND	
5V	2	Vcc	
GND	3	Vee	Contrast
10	4	RS	Register Select
GND	5	R/W	Read/Write
9	6	EN	Enable
14	11	DB4	Serial Connection
13	12	DB5	Serial Connection
12	13	DB6	Serial Connection
11	14	DB7	Serial Connection
5V	15	LED+	Backlight
GND	16	LED-	Backlight

Building and Flashing :

- Build and flash ESP32 project using PlatformIO or ArduinoDroid.
- Build FPGA project and flash using tinyfpga-programmer-gui.py.
- Flash the project .bin file using USB-UART connected to the Vaman-ESP32, via PlatformIO or ArduinoDroid.
- Build the FPGA project .bin file by entering the following commands at a terminal window.
- #The following variable can be sourced from .shellrc or .venv/bin/activate export QORC SDK PATH=/path/to/pygmy-sdk
- -cd inter-chip/esp32-fpga/lcd/codes/fpga
- make
- On building successfully, the .bin file is generated at
- inter-chip/esp32-fpga/lcd/codes/fpga/rtl/AL4S3B FPGA Top.bin
- Flash the .bin file to the Vaman-Pygmy by resetting it to bootloader mode and entering the following command at a terminal window,
- python3 /path/to/tinyfpga-programmer-gui.py --port /dev/ttyACMxx --mode fpga --appfpga
- /path/to/AL4S3B FPGA Top.bin -reset

Demonstration:

- Find the IP address of the Vaman-ESP32 by inspecting the output of the serial terminal, or by typing at a terminal window
- By the Commands:
- ifconfig
- nmap -sn xx.yy.zz.0/24

where xx.yy.zz represents the first three octets of the IP address of your device on

the Wi-Fi network interface, found using ifconfig.

LCD expected output:

The digit selected on the web page will be displayed on the LCD display in binary format, similar to a seven-segment display.

CHAPTER 8

CONCLUSION AND FUTURE SCOPE

Conclusion And Future Scope

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.

In Future there is a lot of scope the Vaman LC can be used as a learning platform to deliver industry-relevant content suitable for career development in the Semiconductor & Electronics industry across Graduate/Post-Graduate Engineering & Science Disciplines

References

- [1] Dr. G. V. V. Sharma. “Catalyzing Autonomous Navigation Design Thinking across India”, IITH research Dairy, vol. 4, issue 2, p.p. 10-12, 2022.
- [2] <https://github.com/gadepall>
- [3] <https://github.com/gadepall/fwc-1>
- [4] <https://github.com/gadepall/vaman>
- [5] <https://github.com/gadepall/embedded-system>