Smart Traffic Light System

MAJOR PROJECT REPORT

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY

(Computer Science and Engineering)

Submitted By:

Roshni (2203593)

Simranjeet Kaur (2203596)

Tanveer Singh Puniah (2203599)

Submitted To:

Prof. *Satinderpal Singh*

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GURU NANAK DEV ENGINEERING COLLEGE

LUDHIANA, 141006

May, 2025

ABSTRACT

This project presents the development of a Smart Traffic Light System aimed at improving traffic management at a standard four-way intersection. The prototype demonstrates the sequential operation of red, yellow, and green signals for all four directions, simulating how real-world intersections regulate traffic flow. The primary objective is to reduce congestion and ensure road safety by maintaining an organized pattern of vehicle movement. The system is developed using basic electronic components and microcontroller logic, offering a practical understanding of traffic signal automation.

A standout feature of this project is its integration with the **Blynk IoT application**, which enables real-time remote control of the traffic lights via a smartphone. This functionality allows traffic personnel or system administrators to manually override or adjust the signal flow in case of emergencies, public events, or malfunctioning situations. With Blynk's user-friendly interface and cloud connectivity, the system becomes not only more accessible but also adaptable for future smart city infrastructure.

The current prototype serves as a scalable foundation for advanced implementations. Future versions can incorporate **vehicle detection sensors** to automate signal timing based on traffic density, giving longer green time to heavily crowded lanes. Additionally, **priority access for emergency vehicles** like ambulances and fire trucks can be added using RFID or GPS-based recognition. Integration with **machine learning algorithms** could further optimize traffic flow by predicting congestion patterns and adjusting signals dynamically.

ACKNOWLEDGEMENT

We are highly grateful to the Dr. Sehijpal Singh, Principal, Guru Nanak Dev Engineering College (GNDEC), Ludhiana, for providing this opportunity to carry out the minor project work.

The constant guidance and encouragement received from Dr. Kiran Jyoti H.O.D. CSE Department, GNDEC Ludhiana has been of great help in carrying out the project work and is acknowledged with reverential thanks.

We would like to express a deep sense of gratitude and thanks profusely to Project Guide Er. Satindepal Singh, without his wise counsel and able guidance, it would have been impossible to complete the project in this manner.

We express gratitude to other faculty members of computer science and engineering department of GNDEC for their intellectual support throughout the course of this work.

Finally, we are indebted to all whosoever have contributed in this report work.

Roshni

Simranjeet Kaur

Tanveer Singh Puniah

LIST OF FIGURES

FIG NO.	FIGURE DESCRIPTION	PAGE NUMBER
3.1	Node Mcu Workflow Diagram	31
3.2(a)	Flow Chart Diargam of Traffic Light Sys.	32
3.2(b)	Circuit Diagram	34
5.1	Traffic Light Control Iot App	50
5.2	Traffic Lights at Initial Stage	50
5.3	Data Flow Processing	50
5.4	Green Light on Lane1	51
5.5	Yellow Light at Lane1.	51
5.6	Green light at Lane2	51
5.7	Yellow light at Lane2	51
5.8	Green light at Lane3	51
5.9	Yellow light at Lane3	51
5.10	Green light at Lane4	52
5.11	Green Light on Lane1 (Automatic)	52
5.12	Yellow Light at Lane1 (Automatic)	52
5.13	Green light at Lane2 (Automatic)	52
5.14	Yellow light at Lane2 (Automatic)	52
5.15	Green light at Lane3 (Automatic)	53
5.16	Yellow light at Lane3 (Automatic)	53
5.17	Green light at Lane4 (Automatic)	53
5.18	Yellow Light on Lane4 (Automatic)	53

5.19	Green Light on Lane1 (Timer)	53
5.20	Yellow Light at Lane1 (Timer)	53
5.21	Green light at Lane2 (Timer)	54
5.22	Yellow light at Lane2 (Timer)	54
5.23	Green light at Lane3 (Timer)	54
5.24	Yellow light at Lane3 (Timer)	54
5.25	Green light at Lane4 (Timer)	54
5.26	Yellow Light on Lane4 (Timer)	54
5.27	Green Light on Lane1 (Prototype Manual)	55
5.28	Yellow Light at Lane1 (Prototype Manual)	55
5.29	Green light at Lane2 (Prototype Manual)	55
5.30	Yellow light at Lane2 (Prototype Manual)	55
5.31	Green light at Lane3 (Prototype Manual)	55
5.32	Yellow light at Lane3 (Prototype Manual)	55
5.33	Green light at Lane4 (Prototype Manual)	56
5.34	Green Light on Lane1 (Prototype Auto.)	56
5.35	Yellow Light at Lane1 (Prototype Auto.)	56
5.36	Green light at Lane2 (Prototype Auto.)	56
5.37	Yellow light at Lane2 (Prototype Auto.)	56
5.38	Green light at Lane3 (Prototype Auto.)	57
5.39	Yellow light at Lane3 (Prototype Auto.)	57
5.40	Green light at Lane4 (Prototype Auto.)	57
5.41	Yellow light at Lane4 (Prototype Auto.)	57
5.42	Green Light on Lane1 (Prototype Timer)	57
5.43	Yellow Light at Lane1 (Prototype Timer)	57

5.44	Green light at Lane2 (Prototype Timer)	58
5.45	Yellow light at Lane2 (Prototype Timer)	58
5.46	Green light at Lane3 (Prototype Timer)	58
5.47	Yellow light at Lane3 (Prototype Timer)	58
5.48	Green light at Lane4 (Prototype Timer)	58
5.49	Smart Traffic Light System	58

TABLE OF CONTENTS

CONTENTS	PAGE NUMBER
Abstract	***i***
Acknowledgement	***ii***
List Of Figures	***iii***
Table of Contents	***iv-v***
Chapter 1: Introduction	**1-16**
1.1 Introduction to Project	
1.2 Project Category	
1.3 Problem Formulation	
1.4 Identification/Recognition of Need	
1.5 Existing Systems	
1.6 Objectives	
1.7 Proposed System	
1.8 Unique Features of the Proposed System	
Chapter 2: Requirements Analysis and System Specification	**17-27**
2.1 Feasibility Study	
2.2 Software Requirement Specification	
2.3 SDLC Model to be used	
Chapter 3: System Design	**28-36**
3.1 Design Approach	
3.2 Detailed Design	
3.3 User Interface Design	

3.4 Methodology

Chapter 4: Implementation and Testing **37-48**

4.1 Introduction to Languages, IDEs, Tools and Technologies used for Project Work

4.2 Algorithm Used

4.3 Testing Techniques in context of project work

4.4 Test Cases designed for project work

Chapter 5: Results and Discussions **49-58**

5.1 User Interface Representation

5.2 Snapshots of the System with discussion

Chapter 6: Conclusion and Future Scope **59-60**

6.1 Conclusion

6.2 Future Scope

References **61**

CHAPTER 1 : INTRODUCTION

1.1 Introduction to Project

The Smart Traffic Light System aims to revolutionize traffic management by introducing a dynamic, real-time solution that adapts to traffic flow, ensuring optimal traffic control and minimizing congestion. This project leverages cutting-edge technologies to create an intelligent traffic light system that can detect and respond to real-time traffic conditions. By utilizing sensor networks, IoT (Internet of Things) devices, and the Blynk mobile application, the system will analyze traffic data and adjust signal timings accordingly. The project's scope lies in the specialized field of smart transportation systems, where technology is integrated with infrastructure to improve urban mobility. The Smart Traffic Light Project falls under the domain of intelligent transportation systems (ITS), a rapidly growing field that incorporates advanced technologies to enhance the efficiency and safety of transportation networks.

It is particularly relevant in urban areas facing challenges of increasing traffic volumes and limited road infrastructure. The Smart Traffic Light Project aims to revolutionize traffic management by introducing a dynamic, real-time solution that adapts to traffic flow, ensuring optimal traffic control and minimizing congestion. This project integrates advanced technologies like IoT (Internet of Things), sensor networks, by providing adaptive traffic management, it enhances urban mobility and addresses the challenges posed by increasing traffic volumes in cities with limited infrastructure. Urban areas worldwide are grappling with escalating traffic congestion, leading to increased travel times, fuel consumption, and environmental pollution. Traditional traffic management systems, often reliant on fixed signal timings, fail to adapt to the dynamic nature of urban traffic, resulting in inefficiencies and heightened commuter frustration. The Smart Traffic Light System aims to revolutionize traffic management by introducing a dynamic, real-time solution that adapts to traffic flow, ensuring optimal traffic control and minimizing congestion.

This project leverages cutting-edge technologies to create an intelligent traffic light system that can detect and respond to real-time traffic conditions. By utilizing sensor networks, IoT (Internet of Things) devices, and the Blynk mobile application, the system will analyze traffic data and adjust signal timings accordingly.

To fulfill the rising demands of the populace, alternative energy sources are in high demand. There are two ways to go about it: first, identify new energy sources, and second, cut back on how much energy you use from the currently accessible supplies. The second goal, which is to reduce energy use, is covered in this essay. Basically, this is a research of streetlights with microcontroller-based control lighting.

LED light, a motion sensor, and communication network make up the major components of the smart street light system. When cars arrive, the lights come on, and when no one is there, they go off. It will be challenging for pedestrians and car drivers to differentiate between smart street lighting and the standard street lights, as all street lights switch on prior to their arrival. About 30% of the energy used in cities is used for public lighting, including that in streets, tunnels, ports, and other public spaces. Cities require energy- saving smart lighting systems based on economic and environmental considerations.

IoT based street lighting system ensures various metrics of the system like automatic switching of lights from ON state to OFF state or vice versa based on the condition provided for the system. For example, an IoT based system which comprises sensors, actuators, relays, IoT hardware and software with some networking and communication protocols enable the automatic street light monitoring and controlling system to function properly. This IoT based system will be installed on each pole for controlling the lights and monitoring the street lights energy consumption on IoT based smart street light monitoring and controlling system is demonstrated in which different components of IoT like IoT devices, Gateways, Cloud, web browser, big data, sensors play essential role. All the smart systems are not intelligent but all the intelligent systems are smart it is because intelligent systems work on larger amount of Smart Street Light Monitoring and Controlling System data means there will be good decisions whereas poor amount of data means there would be wrong predictions.

So, for making intelligent systems more and more reliable big data is essential requirement. Street Lights are having very high contribution for providing the safety of transportation system and smart cities development. The existing street lighting system uses old techniques and it is facing so many problems like: Existing Street lighting systems are needed to be Turned ON and OFF manually. It has a high- power consumption and their maintenance is also quite expensive. More manpower is required to handle the functioning of the existing street light system. There is a growing demand of IoT based Smart intelligent systems for street light monitoring and

controlling operations. Recently, new technologies evolved for smart city development, smart healthcare, smart agriculture, smart transportation system are also exploring for smart street lighting systems for cities. These technologies can resolve the challenges faced by existing systems such as with the help of IoT, street lights can be switched ON and OFF automatically.

- **Energy Efficiency and Cost Reduction in IoT-Based Street Lighting:**

  Maintenance of street lights using IoT is quite less which leads to cost reduction. Power consumption is quite low in these street lights using IoT which also leads to energy conservation. No large manpower is required to maintain these street lights using IoT technologies. Monitoring the usage of street lights using IoT system is quite easy. Nowadays, it is observed that the sodium lamps are replaced with LED lights in streets of a city because one of the major factors is power consumption which is less and cost is another issue compared to sodium lamps. Further LED lights are eco-friendly and avoids greenhouse gas emission.

- **IoT-Based Smart Prototypes for Street Light Monitoring and Energy Management:**

  Our proposed street light monitoring and controlling system can conserve fair amount of energy we can also monitor it on the web browser. The major contribution of this research work is as follows: Firstly, aprototype is developed for monitoring and controlling of street lights using IoT platforms. Along with this, another prototype is proposed for monitoring electricity consumption at each pole and sending this electricity consumption information to IoT platform, i.e., on web browser. With the first prototype the street lights will TURNED ON/OFF depending on the presence of a moving object. With the implantation of this prototype, nearly 60-70 percent of energy saving can be achieved for a day which is demonstrated in discussion section thoroughly.

  The developed prototype 2 can be installed at each pole to measure electricity consumption or it could be separately installed for remote monitoring of electricity consumption of home appliances. The integration of both the prototypes can be utilized to collect real-time data of electricity consumption and with large amount of data by applying machine leaning algorithms future electricity consumption demands can be predicted and other controlling actions can be taken with the analysis of the real time data. This work presents an innovative

approach to improving the efficiency and monitoring capabilities of street lighting systems using IoT technology.

- **Real-time adaptability:** Unlike traditional systems, the proposed solution can adjust signal timings instantaneously based on current traffic conditions.
- **IoT integration:** Utilizing IoT devices allows for seamless data collection and communication between system components.
- **Remote monitoring and control:** The Blynk mobile application provides authorized users with the ability to oversee and adjust traffic signals remotely.
- **Cost-effectiveness:** By leveraging affordable hardware and open-source software, the system offers a financially viable solution for cities with limited budgets.
- **Scalability:** The modular design enables easy expansion to accommodate additional .

1.2 Project Category

This project falls under the category of Intelligent Transportation Systems (ITS), which utilizes modern technologies to enhance the efficiency and safety of transportation networks. The system uses real-time traffic data, sensor networks, and wireless communication to optimize traffic flow, making it particularly useful for urban areas facing heavy congestion. This category includes projects that apply existing and emerging technologies to real-world environments, specifically aiming to improve efficiency, responsiveness, and automation in public infrastructure systems. Our project draws from both engineering application and smart city innovation, addressing practical challenges in traffic regulation using cost-effective microcontrollers and cloud-based control mechanisms.This system is primarily focused on real-time signal control and fault monitoring using NodeMCU ESP8266 and Blynk, whereas the referenced research employs ESP32 for detecting traffic light failures. Both systems align under the broader goal of urban traffic management automation, showing how embedded hardware and wireless technologies can be combined to enhance infrastructure resilience and reduce human intervention.

The project is best categorized under Intelligent Transportation Systems (ITS), a sub-discipline of smart infrastructure focused on applying data-driven and automated solutions to improve traffic flow, safety, and road efficiency. ITS systems are typically complex and government-led,

but the proposed model presents a scaled-down prototype that demonstrates the core features of ITS in a modular and affordable format.

- Within ITS, several operational needs are addressed by this system:

  - Dynamic control of traffic signals using live input from IR or ultrasonic sensors.
  - Remote access and monitoring via the Blynk cloud dashboard.
  - Real-time notification and status reporting in case of component failure or congestion.
  - Priority handling for emergency vehicles and fault tolerance through automated fallback procedures.

The referenced paper describes a traffic light damage monitoring system using ESP32, with the main aim of helping technicians detect malfunctioning lights via ADC signal changes and Blynk notifications. Although this differs slightly in application, both projects highlight a trend toward proactive traffic infrastructure where the system can not only operate independently but also report its operational status, helping city administrators optimize performance with minimal manual effort.

Our proposed system takes this a step further by integrating traffic density detection and adaptive light timing, focusing on mobility improvement rather than just fault detection. While the referenced system uses ADC value readings to determine light functionality, our system utilizes sensor input to control traffic light logic dynamically. Both utilize the Blynk mobile application for cloud-based visibility and interface control, proving its versatility in ITS applications.

Target Applications

This project can be practically deployed or simulated in the following environments:

- **Educational Institutions:** To demonstrate the feasibility of IoT-based automation, providing students hands-on experience with smart infrastructure design.
- **Municipal Pilot Projects:** Local governments with limited budgets can use this scalable design to test ITS functionality in select intersections before city-wide rollouts.
- **Private Campuses or Industrial Zones:** Facilities with internal roads and intersections

can benefit from smart signal control, enhancing safety and traffic flow.

- **R&D and Prototyping:** Research labs or innovation hubs working on smart city projects can adopt this system as a testbed for integrating additional features like AI, CCTV, or ML- based traffic pattern recognition.

Though this project is application-based and prototype-level, it has high potential for scaling into a full-fledged ITS module. Additional features like:

- RFID-based emergency vehicle detection.
- Machine learning models for traffic forecasting.
- Edge computing using ESP32 with AI accelerators can be integrated in the future. Moreover, since it is built using open-source tools and standard communication protocols, the system is interoperable, allowing third-party integration and easy updates.

1.3 Problem Formulation

The core issue addressed is the inefficiency of traditional, time-based traffic lights, which operate on pre-set intervals regardless of real-time traffic volume. These systems:

- Cause long wait times during low traffic hours.
- Lead to gridlocks during peak hours.
- Cannot prioritize emergency vehicles.
- Require manual interventions for changes.

➢ **To overcome these challenges, an intelligent system is needed that can:**

- Detect vehicle count and adjust signal duration accordingly.
- Provide remote control for traffic authorities.
- Ensure smooth transition between signals and manage emergency situations efficiently.

One of the persistent challenges faced by urban transportation systems is the inefficiency and

rigidity of conventional traffic signal mechanisms, which rely on pre-programmed, time-based intervals. These traditional traffic lights operate based on static schedules that fail to respond to real- time traffic conditions, leading to multiple inefficiencies that directly affect vehicular flow, fuel consumption, travel time, and road safety.

In the current systems widely deployed across developing and even many developed cities, traffic lights change signals at fixed intervals — regardless of whether there is traffic or not on a particular lane. This often results in one or more of the following consequences:

- Unnecessary delays during off-peak hours, where vehicles may wait at a red signal despite the road being empty.
- Severe congestion during rush hours, as fixed cycles do not adapt to increased traffic flow, creating long queues and delays.
- Inability to manage emergency scenarios, such as ambulances or fire trucks needing a green signal on priority, which traditional systems cannot identify or accommodate.
- Dependency on manual intervention, where traffic personnel need to override systems physically during festivals, construction work, or special events — which is not only inefficient but prone to human error.
- Energy wastage and pollution, as vehicles idle longer, increasing fuel use.

These limitations make it clear that traditional traffic control systems are not scalable or efficient for modern urban transportation needs, particularly in smart city initiatives where data-driven decision- making and automation are key components.

- **The Need for a Smarter Approach**

With increasing urbanization, there is a pressing need for an intelligent traffic control system that not only automates signal timing but also reacts dynamically to real-time road conditions. The proposed IoT-based Smart Traffic Light System addresses these gaps through a sensor-driven, cloud-connected approach that empowers both local control and remote monitoring.

Key problems identified in current systems, and how they can be solved through smart infrastructure:

1. **Traffic Signal Inflexibility:**

   Traditional lights follow rigid signal cycles, regardless of lane-specific traffic intensity. In contrast, a smart system utilizes infrared (IR) or ultrasonic sensors to measure the number of vehicles on each road segment. Based on this data, the system intelligently adjusts the green-light duration to prioritize the busiest lanes, ensuring more efficient traffic dispersal.

2. **Lack of Emergency Vehicle Handling:**

   In conventional systems, no distinction is made for ambulances or fire brigades. A smart system could integrate RFID or sound-based emergency detection, enabling it to temporarily override normal signal logic and give an instant green light to approaching emergency vehicles.

3. **No Remote Operability:**

   Old systems require manual intervention at the signal box to make any adjustments. In contrast, smart systems utilize mobile-based platforms like the Blynk app to allow remote configuration, override, and live monitoring of traffic signal status by authorized personnel.

4. **High Human Dependency:**

   Deployment of traffic personnel for manual signal changes is resource intensive and unsustainable. IoT-based systems can function autonomously, with minimal human supervision, and can self-diagnose faults in the traffic light components using internal voltage/current monitoring sensors.

1.4 Identification/Recognition of Need

As cities continue to expand and vehicle populations increase, urban transportation networks are being pushed beyond their designed limits. Traffic congestion, poor signal coordination, and lack of dynamic response are common characteristics of most conventional traffic light systems,

particularly in developing regions. The resulting inefficiencies are no longer just minor inconveniences—they contribute to economic losses, increased environmental impact, and compromised public safety.

1. Rising Pressure on Urban Infrastructure

Traditional traffic control systems rely on fixed-timing Traditional traffic control systems rely on fixed-timing mechanisms that are not responsive to actual road conditions. Whether traffic is dense or sparse, the signal cycle remains unchanged. This rigid setup is unable to meet the growing needs of modern urban ecosystems, where vehicle flow varies dramatically throughout the day and unexpected events (accidents, breakdowns, or VIP movements) require instant adaptations. The limitations of outdated systems emphasize the growing need for intelligent and adaptable infrastructure that supports modern urban demands. According to global smart city development frameworks, transportation is one of the core pillars of urban digitization, and traffic signal control plays a vital role in this transformation.

2. Shift Toward Automation and Real-Time Control

The transition from conventional infrastructure to intelligent systems is no longer optional—it is an essential upgrade. Automated, sensor-driven traffic management solutions, enabled by the Internet of Things (IoT), offer the potential to replace static, time-based control with real-time, context-aware signaling. Such systems use live data collected through sensors (IR, ultrasonic, or camera-based) and communicate this information to cloud platforms, enabling dynamic decision-making.

There is a critical demand for systems that not only manage traffic but also adapt to it, ensuring smoother vehicular movement and better utilization of available road space. In this context, a smart traffic system is not merely a technological innovation it becomes a strategic necessity for:

- Reducing congestion
- Improving commuter experience
- Streamlining emergency response
- Supporting sustainability goals

3. Alignment with Smart City Objectives:

As India and other nations progress toward the implementation of Smart Cities, traffic

systems must evolve in parallel to meet new standards of performance, efficiency, and connectivity. Smart traffic systems align directly with urban policy goals such as:

- Environmental sustainability (by reducing vehicle idling time and associated CO_2 emissions)
- Public safety (through smoother traffic flows and prioritized emergency response)
- Governance and accountability (with data-based monitoring and performance tracking)
- Digital service delivery (enabling mobile app-based control and notifications for city administrators)

By embracing IoT-driven traffic solutions, municipalities can build a foundation for future innovations, such as AI-assisted route prediction, smart parking systems, and integrated public transportation coordination.

➢ **Key Needs Identified**

To meet the demands of a modern traffic environment, the following core needs have been identified:

- Secure and scalable device control
- Mobile app integration

1.5 Existing System

In most urban environments, the control of traffic signals still depends on outdated infrastructure that operates on predetermined cycles. These systems are typically based on electromechanical timers or basic programmable logic controllers (PLCs) that follow a fixed schedule, regardless of the actual traffic density on the roads. While functional, these traditional systems are inherently limited in their ability to respond to dynamic or unexpected situations, making them less effective in managing modern traffic volumes.

❖ **Need for Responsiveness**

Urban traffic patterns are highly dynamic and unpredictable. Static systems fail to respond to changing traffic loads during rush hours, weekends, or emergencies. A responsive system can analyze sensor input and adjust signal durations in real-time, reducing wait times and bottlenecks.

- **Need for Safety and Discipline**

  Signal failures, poorly coordinated lights, and gridlocks contribute to accidents and road rage. With real-time control and remote override functionality, traffic management becomes more precise, enforcing better road discipline and improving commuter safety.

- **Need for Energy and Environmental Efficiency**

  Fuel waste from idling engines at non-optimized signals is a silent contributor to pollution and economic inefficiency. Intelligent systems can minimize idle time and reduce fuel consumption, making them environmentally responsible and cost-effective.

- **Need for Centralized Control and Monitoring**

  In conventional setups, troubleshooting requires physical inspection, and there's no real-time visibility into system performance. An IoT-based platform provides centralized dashboards where traffic officers can monitor signal status, respond to failures, and override settings as needed.

- **Need for Responsiveness**

  Urban traffic patterns are highly dynamic and unpredictable. Static systems fail to respond to changing traffic loads during rush hours, weekends, or emergencies. A responsive system can analyze sensor input and adjust signal durations in real-time, reducing wait times and bottlenecks.

- **Need for Safety and Discipline**

  Signal failures, poorly coordinated lights, and gridlocks contribute to accidents and road rage. With real-time control and remote override functionality, traffic management becomes more precise, enforcing better road discipline and improving commuter safety.

- **Static and Non-Portable Infrastructure**

  In most urban areas, traffic light control systems still rely on traditional infrastructure that operates on fixed, pre-programmed schedules. These systems are commonly managed through electromechanical timers or basic programmable logic controllers (PLCs), which function without adapting to real-time traffic conditions.

  A key limitation of such systems is their **static and non-portable nature**. Once installed, they remain fixed and cannot be easily relocated or adjusted based on changing traffic demands. This rigidity makes them ineffective in dealing with dynamic road situations such as sudden congestion, emergencies, or special events. Although stable in performance, these systems lack the intelligence and flexibility required to optimize modern traffic flow

efficiently.

- **Technological Relevance in IoT and Cloud Computing**

  The advancement in IoT microcontrollers (such as NodeMCU ESP8266 and ESP32), along with robust cloud platforms (like Blynk, Firebase, or ThingSpeak), has made real-time, remotely accessible systems viable for implementation even in mid-tier cities and educational environments. These platforms offer (Low latency communication).

- **IoT (Internet of Things)**

  IoT involves connecting physical devices to the internet to collect, transmit, and act on data in real time. These devices, embedded with sensors and actuators, are used in various domains like smart homes, healthcare, agriculture, industrial automation, and transportation.

  The technological relevance of IoT lies in its ability to generate large volumes of real-time data from the physical world, which can be analyzed to optimize operations, reduce costs, and improve user experience. For example, in a smart traffic system, sensors can monitor vehicle movement and adjust traffic signals dynamically to reduce congestion.

- **Cloud Computing**

  Cloud computing offers on-demand access to computing resources such as storage, processing power, and networking through the internet. It eliminates the need for heavy infrastructure investment and allows services to scale efficiently.

  In the context of IoT, cloud computing provides a robust platform to store, process, and analyze the massive data generated by IoT devices. It supports big data analytics, artificial intelligence, and machine learning models that derive insights from IoT data.

- **Integration of IoT and Cloud Computing**

  The integration of IoT and cloud computing—often referred to as **Cloud of Things (CoT)** has significant technological relevance:

  Scalability: Cloud services can handle the growing number of connected devices and data volumes.

  - **Remote Management:** IoT devices can be monitored and managed from anywhere via cloud platforms.
  - **Data Analytics:** Cloud platforms provide tools to analyze IoT data for informed decision- making.
  - **Cost Efficiency:** Reduces the need for local data storage and processing units.

- **Security and Updates:** Cloud services facilitate centralized updates and security management. Conventional signal systems must be configured manually. Any change in signal duration, such as increasing the green time for a busier lane, requires physical access to the control unit and manual reprogramming. This process is not only time-consuming but also.

One of the major drawbacks of these legacy systems is their complete reliance on human intervention for any form of adaptive behavior. If an accident occurs or a sudden spike in traffic is observed due to a public event, traffic officers must step in to manually manage the signals or redirect vehicles, which increases workload and reduces efficiency. Such systems also fail to prioritize emergency vehicles, like ambulances or fire trucks, which may be delayed in congested intersections due to the system's inability to identify and accommodate urgent traffic needs.

Although there have been efforts in some advanced cities to incorporate technologies such as closed-circuit television (CCTV) cameras and artificial intelligence for traffic monitoring, these implementations are often prohibitively expensive. They involve high installation and maintenance costs, require continuous internet connectivity, and depend on powerful processing units to handle video data and predictive algorithms. As a result, such sophisticated systems are generally out of reach for smaller municipalities, educational campuses, or developing regions where budget constraints are significant.

Furthermore, these newer systems often lack modularity, meaning they cannot be easily scaled or upgraded without overhauling the entire infrastructure. This presents a major barrier to wider adoption in areas that require flexible and expandable solutions. In addition, the majority of traditional traffic systems do not offer remote control capabilities, making it difficult for traffic authorities to intervene or monitor operations without being physically present at the signal control point.

Overall, the existing traffic management infrastructure is outdated, rigid, and insufficient for the growing complexity of urban transportation. It does not offer the adaptability, affordability, or automation needed to cope with today’s traffic patterns. There is a pressing

need for a more intelligent system that combines real-time sensing, remote control, and automation one that can improve efficiency, ensure safety, and provide scalable options for future growth.

1.6 Objectives

The objectives of IOT – based smart traffic light are:

- To design an IoT-based smart traffic light system to manage traffic.
- To design a mobile application to control lights remotely.
- To enable seamless wireless communication for efficient and responsive control.

1.7 Proposed System

The system proposed in this project is built on the concept of automation, decentralization, and remote accessibility, enabled through the integration of Internet of Things (IoT) technologies. It aims to resolve the limitations of traditional traffic control systems by implementing real-time monitoring and adaptive signal management at road intersections. The core of this solution is a microcontroller-based setup that interacts with both hardware sensors and a cloud-based interface, allowing it to operate with minimal human intervention.

At the heart of this system lies the NodeMCU ESP8266, a microcontroller that is both compact and equipped with built-in Wi-Fi capabilities. This device is responsible for executing the control logic that manages the timing of the traffic signals. Connected to this controller are multiple sensors placed at the entry point of each lane in the intersection. These sensors, which can either be infrared-based or ultrasonic, are responsible for detecting the presence and volume of vehicles waiting at the signal. By collecting real-time traffic data, the system is capable of determining which lane is experiencing the highest congestion and dynamically allocating the green light to that lane for a longer period.

To replicate the functionality of a real traffic signal, three different colored LEDs—red, yellow, and green—are connected to the microcontroller for each lane. These LEDs light up in a sequence that mimics actual road signal behavior, with the duration and transition between lights being controlled programmatically based on the sensor input. The lights operate continuously in a loop, with sensor feedback serving as the key variable that determines how long each light

remains active.

An essential feature of this system is its ability to connect to the internet using Wi-Fi, made possible by the NodeMCU's onboard module. This connection allows the system to sync with cloud services and mobile applications, thus enabling remote access and monitoring. Through the Blynk mobile platform, users—such as traffic control personnel or municipal engineers—can view the current status of all signal lights, override the automatic system manually if necessary, and make adjustments without needing to be physically present at the intersection.

In scenarios where an emergency arises—such as an ambulance needing immediate passage or an unexpected traffic blockage—the manual override function becomes particularly useful. From the Blynk application, an authorized user can halt all traffic, prioritize one lane, or reconfigure the system instantly to respond to evolving traffic needs. Despite this level of control, the system remains autonomous under normal conditions, making decisions continuously based on live sensor data without external commands.

The proposed system is not only intelligent but also efficient and affordable. It uses components that are readily available and cost-effective, allowing it to be implemented as a prototype for institutional, research, or municipal purposes. The simplicity of the hardware combined with the flexibility of the software architecture makes it ideal for both small-scale and scalable deployments.

1.8 Unique Features of the Proposed System

What sets this system apart from existing solutions is its combination of intelligent automation, real-time data processing, and user-driven control, all within a compact and efficient architecture. One of its most important features is its ability to modify traffic light durations dynamically, based on actual traffic conditions. Unlike traditional systems that follow a preset schedule, this design reacts to real-time data, making the flow of traffic more efficient and reducing idle times significantly. The dynamic nature of the system allows it to favor more congested lanes automatically while maintaining orderly transitions between signals. Another advantage lies in its mobile-based accessibility.

Through a user-friendly application interface, authorities can monitor the functioning of traffic signals and take direct control when needed. This level of accessibility allows for faster responses to on-ground issues, such as malfunctioning hardware, emergency rerouting, or sudden congestion. It also reduces the dependency on manual fieldwork, saving both time and human effort. Emergency handling is another built-in functionality that ensures the system can adapt to critical situations. In such scenarios, the user can activate emergency modes that override the normal logic of the system and provide green signals exclusively to a particular lane. This is especially beneficial when dealing with ambulances, police vehicles, or fire services that require quick and unobstructed access through intersections.

Additionally, the system is engineered with power efficiency in mind. The use of low-energy LEDs, combined with the minimal power consumption of the NodeMCU, makes it suitable for continuous operation without imposing a heavy load on energy resources. This makes it ideal for institutions and municipalities that seek energy-conscious solutions.

Scalability is another notable characteristic. The modular design allows additional lanes or intersections to be added without overhauling the core logic or hardware. This means that the same control algorithm and mobile interface can be adapted for wider implementation across multiple traffic points with minimal effort.

CHAPTER 2 : REQUIREMENT ANALYSIS & SYSTEM SPECIFICATION

2.1 Feasibility Study

Before beginning development, it is essential to assess the viability of the Smart Traffic Light System using a structured feasibility study. This ensures that the system is not only practical from a technological perspective but also cost-efficient and suitable for real-world deployment. The feasibility study evaluates the project under three critical dimensions: technical feasibility, economic feasibility, and operational feasibility.

- **Technical Feasibility**

  The technical feasibility of the Smart Traffic Light System is one of its strongest attributes. The architecture is based on standard, well-documented hardware and software platforms that are highly compatible and easy to integrate. The central processing unit of the system is the NodeMCU ESP8266, a microcontroller that comes with built-in Wi-Fi capabilities. This device serves both as the signal processor and the communication controller, reducing the need for additional networking modules. Its popularity among IoT developers ensures wide availability of online resources and community support, which simplifies troubleshooting and accelerates the development cycle. The NodeMCU is programmed using the Arduino IDE, an open-source integrated development environment that supports the C/C++ programming languages. Arduino IDE offers numerous libraries, including those for sensor integration and Wi-Fi communication, making the development process straightforward.

  The sensors used—infrared (IR) or ultrasonic sensors—are chosen for their reliability, low power consumption, and ability to detect vehicle presence or movement with reasonable accuracy. These sensors are also easy to calibrate and interface with the NodeMCU via its GPIO pins. Signal indication is handled using light-emitting diodes (LEDs), which are used to represent the standard red, yellow, and green traffic lights. These LEDs are cost-effective, long- lasting, and require minimal current, making them ideal for extended usage. To simulate real traffic light behavior, the system uses logic-based switching controlled by traffic density data collected through sensors. An essential component of the system is the Blynk IoT platform, which acts as a cloud-based bridge between the hardware and the user interface. It allows users to monitor the state of traffic signals and issue override commands

via a smartphone application. The platform supports real-time data synchronization, meaning any change in the system status is reflected immediately in the app. This capability ensures that traffic managers can monitor and intervene, if necessary, from any location with internet access. The system's design is flexible and modular. Each lane and signal light operates as a distinct unit controlled by the main microcontroller. Additional lanes or intersections can be added to the system with minor changes to the code and minimal increase in hardware complexity. This scalability proves that the system is technically viable not only as a prototype but also as a deployable solution in small-scale municipal or campus environments.

- **Economic Feasibility**

  From a cost perspective, the Smart Traffic Light System is highly viable and attractive, particularly for developing regions or organizations with limited budgets. One of the significant advantages of this system is its reliance on low-cost, off-the-shelf components that do not compromise on performance. The NodeMCU microcontroller, IR sensors, LEDs, resistors, and jumper wires can all be purchased at minimal expense. The overall bill of materials is significantly lower compared to conventional traffic management systems, which often rely on proprietary hardware and centralized control centers. Furthermore, the system benefits from the use of open-source software tools.

  Overall, the economic feasibility study indicates that the Smart Traffic Light System is well-suited for environments that seek cost-effective automation without sacrificing core functionality.

- **Operational Feasibility**

  In terms of real-world operation, the system has been designed with user accessibility, reliability, and adaptability in mind. It performs automatically under standard conditions, using data collected by sensors to adjust signal durations in real time. This ensures a smoother traffic flow and reduces congestion at intersections. The switching logic takes into account the number of vehicles waiting in each lane and adjusts the green light duration accordingly. This behavior mimics intelligent traffic control without requiring expensive computing infrastructure. What makes the system truly flexible is its ability to operate in both automatic and manual modes. In automatic mode, the system relies entirely on sensor data and executes decisions using pre-coded logic. In manual mode, a user can override the

system remotely using the Blynk mobile app, which provides a visual interface with signal control buttons. This feature is especially useful during special events, emergencies, or temporary traffic diversions where manual intervention is necessary.

2.2 Software Requirement Specification (SRS) Document

The Software Requirement Specification (SRS) defines the operational expectations and constraints of the Smart Traffic Light System. This document provides a comprehensive overview of the software behaviors, data interactions, reliability goals, and user interface expectations. In the context of the Agile methodology used in this project, the SRS also serves as a dynamic guideline that can evolve across multiple iterations based on performance reviews, user feedback, and functional testing.

- **Overview and Purpose**

The aim of this system is to create an intelligent traffic light controller that adjusts signal timings dynamically based on real-time traffic conditions. The system should be able to manage lane priorities autonomously while also providing manual override features to authorized personnel via a mobile app interface. Additionally, it must ensure a high degree of system stability, responsiveness, and security under real-world constraints, such as varying network availability or hardware
inconsistencies.

- **Data Requirements**

The Smart Traffic Light System relies on multiple data inputs and outputs that enable real-time traffic analysis and control decisions. The input data is sourced primarily from **IR or ultrasonic sensors,** which detect the presence or volume of vehicles at each intersection lane. This data is continuously fed into the **NodeMCU ESP8266**, which processes the input and determines signal behavior. Each sensor provides binary or numerical feedback (e.g., vehicle detected or not, count of vehicles), which is evaluated in each cycle of the microcontroller's loop. The output data includes the status of the LEDs (Red, Yellow, Green), which indicate traffic permissions. These statuses are also sent to the **Blynk mobile app** for remote display and monitoring. Commands initiated from the mobile interface—such as override instructions or manual signal changes—are treated as incoming data for the

NodeMCU to process, ensuring two-way communication between the device and the user interface. Importantly, this data is processed in real-time and not stored, maintaining data simplicity and minimizing memory load on the microcontroller.

- **Functinal Requirements**

The system must be capable of autonomous operation based on programmed traffic logic and real- time input. The basic functional expectations include:

- **Traffic density analysis:** The system evaluates input from all lanes to determine which one has the most vehicles and assigns that lane a longer green light duration.
- **Sequential light control:** The system must execute traffic light transitions in the correct sequence (Green → Yellow → Red) across all lanes to avoid conflicts.
- **Manual override:** Authorized users must be able to control signal timing remotely through the Blynk app interface.
- **Fail-safe behavior:** In the absence of sensor data, the system must follow a default signal cycle to avoid full shutdown.

These functions must operate in both connected and offline modes, with the latter ensuring basic signal operations even if cloud connectivity fails temporarily.

- **Performance Requirements**

To qualify as responsive and effective in real-time scenarios, the Smart Traffic Light System must meet the following performance metrics:

- **Response time:** The system must process sensor input and update LED states within 2 to 3 seconds.
- **Remote control latency:** Commands from the mobile app should reflect changes within 2–4 seconds.
- **Stability:** The microcontroller should function without crash or reset for extended periods (8–10 hours of continuous operation at minimum).
- **Sync accuracy:** Signal timing and transitions must be precisely timed to avoid two lanes receiving green signals simultaneously.

Real-world testing indicates that the system maintains smooth operation under typical indoor lab conditions. In later stages, performance tuning can be introduced using advanced features like timer interrupts or real-time clocks for microsecond-level timing.

- **Dependability Requirements**

  The Smart Traffic Light System must demonstrate high reliability, especially in urban or institutional

  - **Handle network interruptions** by reverting to an autonomous fallback mode where fixed signal timing cycles are followed.
  - **Prevent logic conflicts** such as green signals on conflicting lanes or skipped yellow phases.
  - **Correctly reflect signal states** on both physical LEDs and the mobile app, ensuring synchronization across all interfaces.deployments. The system must:

  The logic must include checks for sensor failure or abnormal data (such as a sensor stuck in high- output state), and revert to default behavior to avoid traffic disruptions.

- **Maintainability Requirements**

  A modular approach was followed during both hardware and software development. Each component—be it the IR sensor, NodeMCU, or a single LED—can be tested, removed, and replaced independently without requiring a complete system rebuild. This significantly simplifies maintenance in the field. The codebase is written using the **Arduino programming language,** with clear function definitions and inline comments that describe logic steps. Future contributors to the project can easily understand and enhance the code without needing to rewrite existing modules. Libraries used (like Blynk, ESP8266WiFi) are well-supported and documented, further simplifying updates. Additionally, settings such as default signal durations, sensor thresholds, and override permissions are defined as constants or variables at the start of the script, making them easy to modify.

- **Security Requirements**

  Security is a critical concern in any system that allows remote control, especially those related to infrastructure like traffic management. In the proposed solution:

  - Access to the Blynk app is token-based, meaning only users with a valid authentication token can send commands to the NodeMCU..
  - The system limits access to specific users defined during the app setup phase.
  - Future improvements may include network-based access controls, IP-based filtering, or

end-to-end encryption for sensitive data packets.

While advanced encryption was not implemented in this prototype, the system is built on platforms that support secure communication protocols if needed for larger-scale deployment.

- **Look and Feel Requirements**

The **user interface** provided via the Blynk app must be visually intuitive and simple to operate, especially for traffic personnel without technical backgrounds. Interface elements should follow standard color and label conventions used in traffic systems.

- Each signal light (Red, Yellow, Green) must be clearly represented with real-time indicators.
- Manual control buttons must be clearly labeled with the respective lane numbers and functions.
- Feedback should be instantaneous, with status updates provided for each action (e.g., "Lane 1: Green Activated").

The layout should be clean, with minimal clutter, and compatible with all Android and iOS smartphones. Accessibility considerations such as high-contrast colors and large button sizes improve usability in outdoor or low-light environments.

- **Software Requirements**

1. **Arduino IDE: Code Development and Programming for Node MCU**
   - Used to write, compile, and upload code to the ESP8266 microcontroller.
   - Provides debugging tools for testing the traffic light system.

2. **Blynk App: Remote Control and Monitoring of Traffic Lights**
   - A cloud-based mobile application that connects with Node MCU over Wi-Fi.
   - Allows users (e.g., traffic authorities) to remotely monitor and override traffic signals if needed.
3. **ESP8266WiFi Library: Enables Wi-Fi Communication**
   - Allows the microcontroller to connect to the internet.
   - Facilitates real-time data exchange with the Blynk cloud server.
4. **Blynk Library: Integrates microcontrolar with the Blynk Mobile Application**

- Provides an interface to send and receive data between ESP8266 and the Blynk app.
- Allows users to control and monitor traffic lights remotely.

5. **C/C++ (via Arduino IDE) – Most Common**
   - You are likely using this if you're uploading code via Arduino IDE.
   - It uses a simplified version of C++.
   - You write functions like setup() and loop().
   - Supports many IoT libraries like WiFi.h, ESP8266WiFi.h, PubSubClient.h (for MQTT), etc.

- **Hardware Requirements**

1. **Microcontroller: NodeMCU ESP8266 (Wi-Fi Enabled)**
   - Acts as the brain of the system, processing sensor data and controlling LED lights.
   - Wi-Fi enabled, allowing communication with the Blynk app for remote control.
   - Supports multiple GPIO pins for connecting sensors and LEDs.
2. **LEDs: Red, Yellow, Green (Traffic Light Representation)**
   - Acts as the brain of the system, processing sensor data and controlling LED lights.
   - Wi-Fi enabled, allowing communication with the Blynk app for remote control.
   - Supports multiple GPIO pins for connecting sensors and LEDs.
3. **Sensors: IR Sensors or Ultrasonic Sensors for Traffic Detection**
   - Infrared (IR) Sensors: Detect vehicle presence based on infrared light reflection.
   - Ultrasonic Sensors: Measure distance to detect traffic density and adjust signal timing accordingly.
4. **Power Supply: 5V DC Adapter**
   - Provides power to the NodeMCU ESP8266 and connected components.
   - Ensures stable operation of traffic light LEDs and sensors.
5. **Resistors: 220Ω for LED Control**
   - Limits current flow to prevent LED burnout.
   - Ensures proper brightness for visibility.
6. **Wi-Fi Module: Integrated with ESP8266**
   - Allows real-time data communication between the traffic light system and the Blynk app.

- Enables wireless monitoring and control of traffic lights.

7. **Breadboard & Wires: For Circuit Connections**
   - Used to connect and test hardware components.
   - Facilitates easy prototyping and modifications without soldering.
8. **74HC595 IC**
   - It's an 8-bit shift register with a latch, allowing you to control 8 output lines using just 3 pins from your microcontroller (Data, Clock, Latch).
   - You can cascade multiple 74HC595s to control even more outputs, useful if you have multiple LEDs, displays, or light signals.
   - Perfect for traffic light LEDs, where each light (Red, Yellow, Green) at multiple junctions can be controlled with fewer pins.

9. **Breadboard & Wires: For Circuit Connections**
   - Used to connect and test hardware components.
   - Facilitates easy prototyping and modifications without soldering.

➢ **Laptops/Desktops**

For the smooth execution of the Smart Traffic Light (IoT Based) project, the development team relies on efficient computing systems such as high-performance laptops or desktops. These systems must be capable of running multiple software tools and environments without performance lags.

- **Processor:** A machine powered by at least an Intel Core i5 or an equivalent AMD processor is ideal. This ensures stable performance when using development tools like Arduino IDE, compiling code, or running backend simulations.
- **RAM:** To manage simultaneous tasks like database operations, real-time monitoring, and running virtual servers, 8 GB of RAM is considered the baseline. However, 16 GB RAM is more suitable for smoother multitasking, especially when handling larger data streams from sensors.
- **Storage:** A Solid-State Drive (SSD) with at least 200 GB capacity is recommended. SSDs significantly speed up read/write operations, which is critical during tasks like compiling firmware, loading datasets, or deploying system components.
- **Graphics:** Although a dedicated GPU is not a strict requirement for this project, having a

moderate graphics card can enhance productivity when working with user interfaces or handling visual simulation tools related to traffic system modelling.

- **How These Components Work Together**
  - The microcontroller (NodeMCU ESP8266) receives traffic data from IR/ultrasonic sensors.
  - The system processes the data and adjusts traffic light timing dynamically.
  - The Wi-Fi module enables communication with the Blynk app, allowing remote monitoring and manual control.
  - The Arduino IDE is used for developing and uploading code to the NodeMCU.
  - The Blynk app acts as the user interface, displaying traffic light status and allowing remote modifications.

2.3 SDLC Model to be Used

For the development of the Smart Traffic Light (IoT Based) project, the Iterative SDLC Model has been selected. This model is highly suitable for IoT systems, as it promotes the gradual development and integration of hardware and software components over multiple development cycles. Each iteration results in a working version of the product, allowing improvements and refinements based on continuous testing and feedback.

- **What is the Iterative Model?**

  The Iterative Model is a development approach where the system is built incrementally through repeated cycles. Each cycle includes planning, designing, coding, and testing, and results in a functional version of the product. Unlike linear models, it encourages returning to earlier stages if required, allowing flexibility and continuous enhancement.

- **Why Iterative for This Project?**

  The Smart Traffic Light system involves:
  - Hardware-software integration,
  - Cloud connectivity,
  - Real-time decision-making.

  These dynamic components make the Iterative Model a strong choice.

Key Benefits:

- **Modular Development**: Allows separate development of hardware, cloud services, and logic.
- **Flexibility:** Changes in logic or hardware configuration can be incorporated in the next cycle without affecting the entire system.
- **Progressive Improvement:** Each iteration improves the system based on real-world feedback and testing.
- **Low Risk:** Issues are identified and addressed early, preventing full system failure.

- **Why Iterative for Smart Traffic Light Project?**

  This project benefits from the Iterative approach in several ways:

  - **Multiple subsystems** (e.g., sensors, traffic logic, cloud dashboard) can be developed and tested independently in cycles.
  - Iterative development suits the **trial-and-error nature** of configuring traffic algorithms and sensor placements.
  - Continuous testing after each cycle ensures **system stability and responsiveness**.

  - Feedback from mentors or testing simulations can be applied in the next iteration quickly.

- **Iterative Workflow for the Project**

  1. **Requirement Analysis and Planning**

     Identify core system needs:

     - Detect vehicle presence and traffic density.
     - Automate signal change based on traffic conditions.
     - Monitor signals remotely through a cloud platform.
     - Define development goals for each cycle or iteration.

  2. **Iteration-Based Design and Development Iteration Development Focus**

     - **1st:** Basic prototype with microcontroller and LED lights for signals.
     - **2nd:** Integration of sensors (IR/Ultrasonic) for traffic detection.
     - **3rd** Develop logic to adjust signal timing based on sensor data.
     - **4th:** Connect system to a cloud platform for remote access.

- **5th:** Design a web or mobile dashboard for monitoring and control.
- **6th:** Final system optimization and user feedback incorporation.

3. **Continuous Testing in Each Iteration**

   After each cycle:

   - Conduct unit testing of modules (e.g., sensor input, cloud sync).
   - Perform integration testing to ensure components work together.
   - Real-world simulation or prototype demo to validate performance.

4. **Review and Feedback**

   After each cycle:

   - At the end of each cycle, present the working version to mentors.
   - Note feedback and adjust the next cycle plan accordingly.
   - Examples:
     - "Sensor takes too long to detect vehicle" → Tune detection range in next cycle.
     - "Dashboard needs clearer status labels" → Update UI in next iteration.

5. **Final Delivery**

By the end of all iterations, deliver a complete and fully tested system that includes:

- Smart signal control based on live traffic conditions.
- Cloud-enabled dashboard for monitoring and updates.
- Modular and maintainable codebase for future enhancements.
- Reliable hardware integration with real-time performance.

CHAPTER 3: SYSTEM DESIGN

3.1 Design Approach

The design approach of the "Smart Traffic Light (IoT Based)" project focuses on creating an intelligent traffic management system that adapts to real-time conditions to improve traffic flow and reduce congestion. Our goal is to design a system that not only automates traffic control but also optimizes it based on actual traffic density and emergency situations.

The Smart Traffic Light system was built using a combination of embedded programming, IoT communication, and basic electronic components. The project not only handles automatic light transitions but also offers manual lane switching using a mobile interface via the Blynk platform.

1. **Problem Analysis and Requirement Gathering**

   The design approach for this project begins with analyzing the issues commonly associated with traditional traffic light systems. We noticed that fixed-timer lights often cause unnecessary waiting or inefficient flow during low or high traffic conditions. Based on real-world observations and additional research, we determined that a dynamic signal timing system would be far more effective.

2. **System Architecture Design**

   We developed a modular system architecture to ensure flexibility and easy upgrades. The core components include vehicle-detection sensors like IR or ultrasonic sensors, a microcontroller such as an Arduino or NodeMCU, and a cloud-based platform for data storage and monitoring. Each traffic junction operates independently using a smart controller that communicates with the central cloud server.

3. **Sensor Integration and Data Collection**

   Sensors are installed at each lane to detect and count vehicles. These readings are continuously sent to the microcontroller, which uses this data to determine how long each signal should stay green, adapting in real time to traffic volume.

4. **Decision-Making Logic Implementation**

   We implemented an algorithm that prioritizes lanes with higher traffic density by extending

their green light duration. Emergency vehicle detection is also included using sound or image processing, allowing the system to override standard operations when ambulances or fire trucks are detected.

5. **IoT Connectivity and Cloud Integration**

   Using Wi-Fi-enabled controllers like NodeMCU, the system connects to an IoT cloud platform such as ThingSpeak or Blynk. This enables real-time data access, remote monitoring, and trend analysis for traffic authorities to manage flow more efficiently.

6. **Prototype Development and Testing**

   A user-friendly dashboard was created for traffic control personnel. This interface allows for real-time monitoring of signal states and traffic levels, displays alerts, and offers manual override capabilities when needed.

3.2 Design Approach

This section provides detailed design documentation for the project using structured analysis and various diagrams. The design addresses each of the project's objectives through detailed illustrations. The **NodeMCU ESP8266 Flow Diagram** for your **Smart Traffic Light System** will illustrate how the microcontroller interacts with sensors, traffic lights, and a cloud-based system. Below is a structured breakdown of the flow:

Flow Diagram Breakdown

1. **Power On & Initialization**

   - The ESP8266 initializes and connects to the Wi-Fi network.
   - All connected sensors (IR sensors, cameras, ultrasonic, or RFID) are initialized.

2. **Traffic Data Collection**

   - Sensors and cameras collect real-time traffic data.
   - IR/ultrasonic sensors count vehicles at the intersection.
   - Cameras provide live feeds for image processing.

3. **Data Processing**

    - The ESP8266 processes sensor input locally or sends data to a cloud server.
    - If cloud processing is used, data is transmitted via MQTT or HTTP requests.

4. **Decision Making**

    - The microcontroller determines the optimal signal timing based on traffic density.
    - If an emergency vehicle is detected (RFID, siren detection, etc.), priority is given.
    - Pedestrian crossing requests are processed.

5. **Traffic Light Control**

    - The ESP8266 controls LED traffic lights according to the processed data.
    - Signal duration is dynamically adjusted to optimize traffic flow.

6. **Remote Monitoring & Alerts**

    - Traffic data is sent to a dashboard for remote monitoring.
    - Alerts are triggered for anomalies like traffic jams or sensor failures.

7. **Continuous Loop**

    - Traffic data is sent to a dashboard for remote monitoring.
    - Alerts are triggered for anomalies like traffic jams or sensor failures.

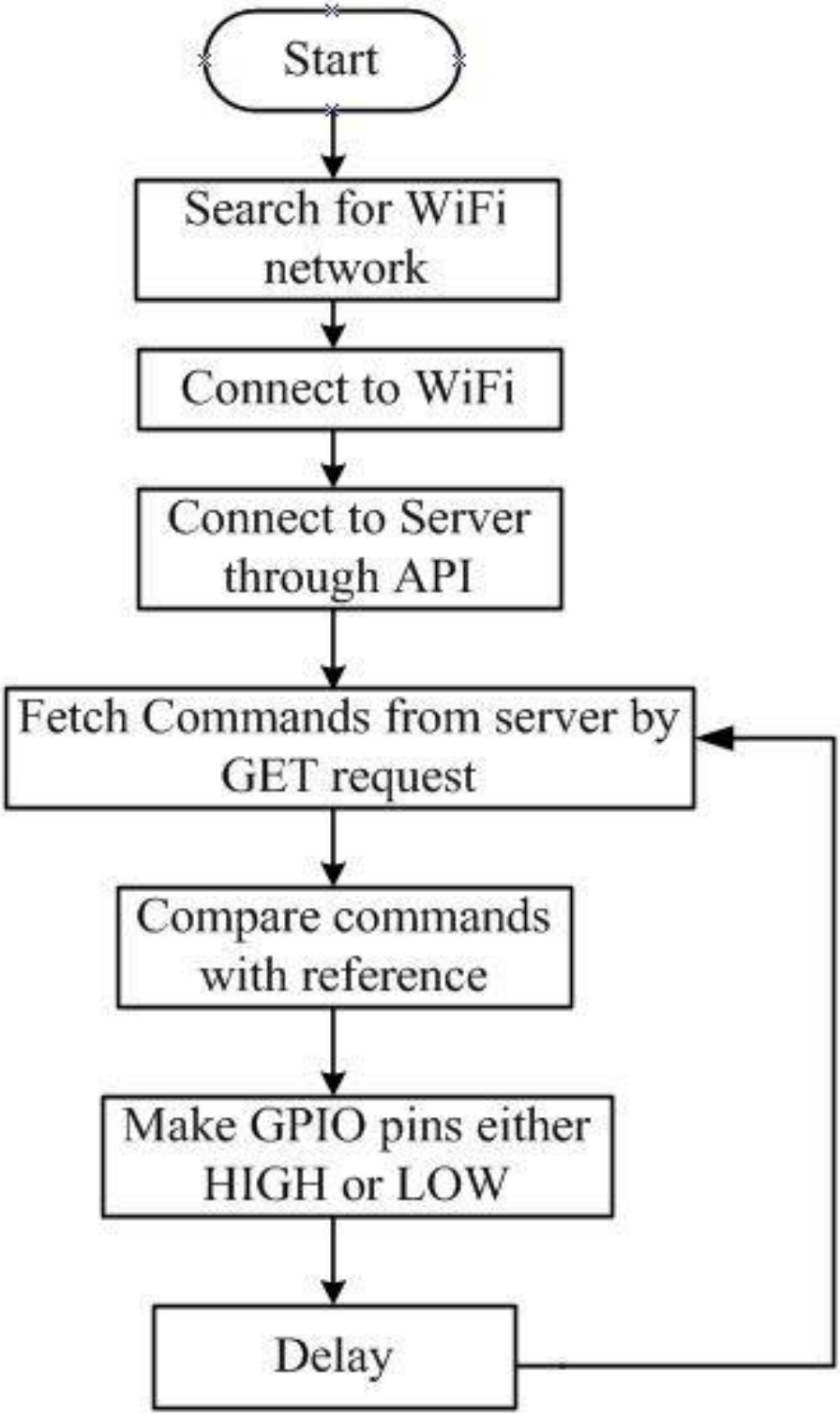

Figure 3.1 Node Mcu Workflow Diagram

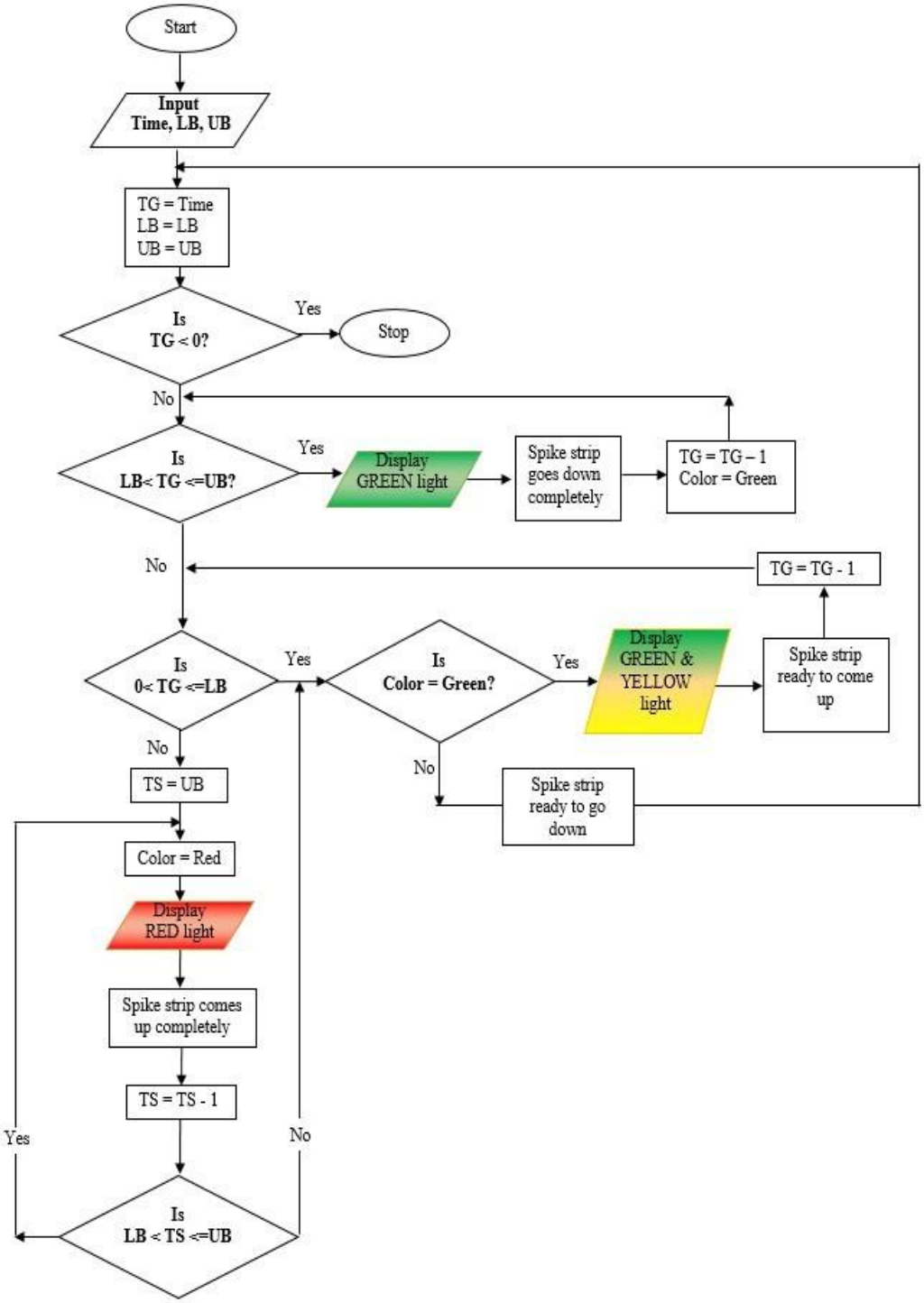

Figure 3.2(a) Flowchart Diagram of Traffic Light System

I. System Architecture

The system consists of:

- **Hardware Components:** NodeMCU ESP8266, Arduino, LEDs (representing traffic lights), and a Breadboard.

- **Software & Communication:**

  Arduino IDE: To program the Arduino controlling the lights.

  Blynk IoT App: To monitor and control the traffic signals remotely.

  Wi-Fi Communication: NodeMCU connects with Blynk via Wi-Fi for real-time operation.

II. Traffic Light Control Logic

- The **Arduino runs the core logic,** executing predefined traffic light sequences.
- The **NodeMCU ESP8266 communicates with Blynk,** allowing remote control of the lights.
- The **Blynk app acts as the user interface,** enabling administrators to manually switch lights ON/OFF or change their sequence.
- The system ensures a smooth **Red-Yellow-Green** transition for all 12 lights while avoiding conflicts.

➢ **Traffic Light Timing Rules:**

- **Normal Operation Mode:**

  Green → 30s, Yellow → 5s, Red → 30s
- **Manual Override Mode (via Blynk):**

  Admin can manually control each light from the app.
- **Emergency Mode:**

III. User Interface with Blynk IoT App

The Blynk IoT app serves as the remote interface for managing the traffic lights.

➢ **Virtual Buttons:**

Each button in the app corresponds to a traffic light, allowing ON/OFF control.

➢ **Status Monitoring:**

The app provides real-time feedback on the status of each light (Red/Yellow/Green).

➢ **Customization & Scheduling**

The admin can modify signal timings or schedule automatic operations.

IV. Wi-Fi & Communication Flow

- NodeMCU connects to Wi-Fi and syncs with the Blynk cloud server.
- Blynk app sends control commands to NodeMCU, which relays them to Arduino.
- Arduino executes the commands, turning the traffic lights ON/OFF accordingly.

Blynk IoT platform for remote control.

1. **Understanding the Problem and Planning the Solution**

   At the outset, we analyzed how conventional traffic lights operate and the challenges faced in busy intersections, such as unnecessary waiting time and lack of emergency overrides. Based on this, a plan was drawn up to create a simple yet effective model that could mimic a real-world traffic junction and allow both automated and manual control through a smartphone.

2. **Component Selection and Preparation**

   The hardware components were chosen based on availability, cost-effectiveness, and suitability for rapid prototyping. The key elements included:

   - A microcontroller (Arduino Uno) to serve as the central unit for controlling signal operations.
   - Sets of red, yellow, and green LEDs for each of the four directions.
   - A breadboard and jumper wires for circuit assembly.
   - A NodeMCU (ESP8266) module for enabling wireless connectivity.
   - A smartphone with the Blynk application installed to serve as the user interface for remote control.

3. **Circuit Assembly and Design**

   Each traffic signal side was connected to the microcontroller through its digital pins, with appropriate resistors to manage current flow. The traffic lights were arranged in such a way that only one direction would turn green at a time, while the others remained on red or transitioned to yellow before changing.

   The NodeMCU module was connected to the same circuit and programmed to interface with the Blynk app. Specific pins were assigned to virtual controls within the app, allowing real-time toggling of lights.

4. **Programming and Integration**

   The system was programmed using the Arduino IDE. Logic was written to manage the order and duration of signal changes using delay functions. Each phase—green,

yellow, and red—was timed appropriately to simulate realistic traffic control. Simultaneously, the Blynk app was configured with buttons and sliders, each mapped to a specific digital pin through virtual channels. This allowed the lights to be activated manually from the phone, providing an emergency override feature or the ability to test the system remotely.

5. **Testing the System**

   After setting up the hardware and software, multiple rounds of testing were carried out. The signal transition was observed to ensure there was no overlap or conflicting signals. The Blynk app was used to verify remote access and response time. Adjustments were made to correct timing mismatches or communication delays.

6. **Review and Future Planning**

   With the system functioning as expected, we reviewed the design to explore areas for improvement. Ideas were noted for including vehicle detection sensors, solar power integration, and emergency vehicle prioritization. The current structure provides a strong foundation for scaling into a more complex and intelligent traffic control system suitable for smart cities.

CHAPTER 4: IMPLEMENTATION AND TESTING

4.1 Introduction to Languages, IDEs, Tools, and Technologies Used

To ensure the reliability and efficiency of our **Smart Traffic Light System**, various testing methodologies are applied, including **unit testing, integration testing, end-to-end testing, and regression testing.** Since our project is based on **Arduino, NodeMCU ESP8266, and the Blynk IoT App,** testing is focused on verifying the hardware-software interaction, light switching logic, and remotecontrol functionality.

The Smart Traffic Light system was built using a combination of embedded programming, IoT communication, and basic electronic components. The project not only handles automatic light transitions but also offers manual lane switching using a mobile interface via the Blynk platform.

1. **Programming Language**

   The primary language used for this project is C++, specifically in the format used by the **Arduino platform.** This language is well-suited for hardware control, as it offers low-level access to GPIO pins and timing functions, which are essential for managing LEDs and sensors.

   The ESP8266 initializes and connects to the Wi-Fi network.

   - The Arduino IDE was employed for writing and uploading the code to the NodeMCU ESP8266 microcontroller. It supports a wide range of libraries, including the Blynk library, which facilitates IoT communication.
   - The IDE also provides a serial monitor for debugging, allowing real-time tracking of program behavior during development.

2. **Microcontroller and IoT Board**

   - The NodeMCU ESP8266 is a Wi-Fi-enabled development board that acts as the brain of the project. It handles communication with the Blynk app, processes logic for LED control, and interacts with shift registers when additional pins are needed.

- Its compact size and built-in Wi-Fi make it ideal for IoT applications like this 1 smart traffic control system.

3. IoT and Mobile Control Platform

Blynk was used for real-time interaction with the system. Through the app, a user can:

- Enable or disable **auto mode**.
- Switch traffic flow to specific lanes (Lane 1 to Lane 4).

Virtual pins V0 to V4 were mapped in the Blynk app and linked to logic in the code using the BLYNK_WRITE() functions.

4. Hardware Implementation (Practical Setup)

The Smart Traffic Light (IoT Based) prototype is built on a breadboard using the ESP8266 NodeMCU microcontroller. It simulates traffic light operations and integrates IoT features for smart control.

Key Components Observed:

- **NodeMCU (ESP8266):** This microcontroller manages traffic light logic and includes Wi-Fi for IoT functions.
- **Breadboard Setup:** Components are mounted on a breadboard, enabling easy testing without soldering.
- **LEDs (Red, Yellow, White):** Represent traffic lights for multiple directions, controlled via code.
- **Resistors:** Connected in series with LEDs to limit current and prevent damage.
- **Jumper Wires:** Used to establish connections between the NodeMCU and other components.
- **USB Power Connection:** Provides power to the NodeMCU and uploads code from a computer.

5. Additional Libraries and Features

- **SimpleTimer** or BlynkTimer was used to handle timed functions without blocking the loop, allowing smooth transitions and responsiveness.
- **millis() Function** is used for managing time-based events without relying on

delay(), which would pause system responsiveness.

4.2 Introduction to Languages, IDEs, Tools, and Technologies Used

In this section, we explain the logic, sequence, and decision-making flow behind the functioning of our Smart Traffic Light (IoT Based) system. Algorithms and pseudocode are crucial to understanding how the system behaves in both automatic and manual modes. Instead of using highly complex logic, this project adopts a simple but effective approach using time intervals, lane switching, and manual override options. The pseudocode outlined here represents the actual logic that is implemented in the system using the NodeMCU (ESP8266) microcontroller.The Smart Traffic Light system was built using a combination of embedded programming, IoT communication, and basic electronic components. The project not only handles automatic light transitions but also offers manual lane switching using a mobile interface via the Blynk platform.

- **Overview of the Algorithm** The algorithm used in the smart traffic light system can operate in two main modes: **Automatic Mode and Manual Mode**. In automatic mode, the system automatically cycles through the traffic lights of all lanes at fixed intervals. In manual mode, a user can select a particular lane using the Blynk app, and the system will allow that lane to have the green light, while all other lanes remain red.

  The algorithm uses basic conditional logic, digital pin control, and time delays to switch between different traffic light states. The core functions include:

  - Initializing the hardware pins (LEDs, control pins)
  - Setting all traffic lights to red
  - Automatically or manually switching one lane to green
  - Transitioning from green to yellow before turning red
  - Cycling through each lane based on time or user input

- **Initialization Logic**

  At the start, the system configures all connected LED pins (for red and yellow lights) as output. This step is important because it allows the microcontroller to control the LED lights on each lane. Green lights are simulated using a shift register or digital write logic through other pins or external modules.

The system also initializes communication with the Blynk application and starts the internal timer function to keep track of delays and intervals.

Pseudocode: Initialization

pgsql

CopyEdit

Start

Set all red and yellow LED pins as OUTPUT

Set all control pins for shift register as OUTPUT

Connect to App

Set timer to call auto traffic cycle function repeatedly

Set all lights to RED

Set initial green light to lane 1

- **Main Loop Function**

The main loop in the algorithm keeps checking whether automatic mode is enabled or if a manual button has been pressed. Depending on the input, the system either continues the automatic traffic cycle or responds to manual lane selection.

Pseudocode: Loop Function

mathematica

CopyEdit

Loop:

Run Blynk App continuously

Run timer functions continuously

This simple loop is important as it allows the system to stay responsive and update in real-time based on either pre-set timing or user input from the app.

- **Automatic Traffic Mode Algorithm**

In automatic mode, the system cycles through all four lanes. Each lane gets a turn to have a green light while the others remain red. The transition from green to yellow, and then to red, is based on fixed delays. After each cycle, the next lane is activated.

Pseudocode: Auto Mode

vbnet

CopyEdit

If Auto Mode is ON:

For each lane:

Turn RED lights ON for all lanes

Turn GREEN light ON for current lane

Wait for 5 seconds

Turn GREEN OFF, YELLOW ON

Wait for 2 seconds Turn YELLOW OFF

Move to next lane

Repeat cycle

This mode does not require any manual control and is ideal for use when traffic conditions are consistent across all lanes.

- **Manual Mode Algorithm**

Manual mode allows a user to control traffic lights for a specific lane using the Blynk app. This feature is useful for emergency vehicles, VIP movement, or real-time human control in special conditions.

Pseudocode: Manual Mode

vbnet

CopyEdit

If Manual Button is Pressed:

Identify which lane was selected

Turn RED lights ON for all lanes

Turn GREEN light ON for selected lane

Wait until user releases control or timeout

Turn GREEN OFF, YELLOW ON

Wait for 2 seconds

Turn YELLOW OFF

Each button on the Blynk app corresponds to one of the four lanes. When a button is pressed, that lane is activated, and others are set to red.

- **Combining Both Modes**

The system supports switching between auto and manual modes without restarting. This is made possible by a condition in the loop that checks the status of a toggle in the Blynk app.

Pseudocode: Mode Selection

mathematica

CopyEdit

Check status of Blynk toggle (V4):

If ON: enable Auto Mode Else:

enable Manual Mode

This design keeps the system flexible and adaptable for real-world usage.

- **Timing and Delay Handling**

The system uses the timer.setInterval() function to manage consistent delays. Instead of using delay(), which would block other operations, a non-blocking timer ensures that the Blynk app and traffic logic run smoothly in parallel.

The delay values are selected to provide realistic behavior:

- Green Light Duration: 5 seconds
- Yellow Light Duration: 2 seconds

➢ **Handling State Transitions**

The algorithm handles states using variables such as state, currentLane, and nextLane. These variables are updated after each cycle to track which lane is active and which comes next. This logic is simple but efficient in managing transitions across lanes.

Pseudocode: State Transition

mathematica

CopyEdit

Initialize currentLane = 1

Loop:

Activate currentLane

Set nextLane = currentLane + 1

If nextLane > 4: reset to 1

Update currentLane = nextLane

This ensures that the traffic lights rotate continuously among all lanes.

➢ **Scalability and Upgradability**

The current algorithm is scalable. It can be modified to handle more lanes by increasing the loop range and assigning more buttons in the Blynk app. The shift register can also help control more LEDs by extending GPIO pin availability.

Future upgrades may include:

- Adding vehicle sensors for dynamic green light timing

- Real-time traffic data processing
- Cloud-based analytics for traffic density

4.3 Introduction to Languages, IDEs, Tools, and Technologies Used

1. Unit Testing

Unit testing involves evaluating each component of the system independently to ensure it performs its intended function correctly. In the context of our project, we tested the individual modules as follows:

- The embedded code running on the NodeMCU was checked to verify the correct timing sequence of traffic lights, ensuring the transition from green to yellow and then to red occurred in a smooth and accurate manner.
- The Wi-Fi module integrated into the NodeMCU (ESP8266) was tested to confirm that it could connect to the Blynk platform without interruptions and respond to commands reliably.
- The Blynk mobile application interface was reviewed to ensure that button presses properly triggered the manual control functions for each lane.
- Debugging and validation were performed using tools like the Arduino Serial Monitor and the built-in debug console provided by Blynk, which helped monitor system responses and identify potential faults.

2. Integration Testing

Integration testing involves checking how well the different parts of the system work together when combined:

- We validated that the embedded code interacts seamlessly with the LEDs, turning them on and off according to the designed logic.
- The communication between the NodeMCU and the Blynk application was assessed to ensure remote control commands could be sent and executed without delay.

- Switching between automatic and manual operation modes was tested to verify that both modes function independently without disrupting each other.
- Prior to actual implementation, simulation tools such as Proteus and Fritzing were used to virtually connect the components and check the overall functionality of the integrated setup.

3. End-to-End Testing

End-to-end testing focuses on evaluating the complete system under real operating conditions:

- The full traffic light cycle was run continuously to confirm there were no unexpected interruptions or timing issues.
- Manual override via the Blynk app was tested by controlling each lane and ensuring that user commands took effect even while the system was running in automatic mode.
- The system's response to network disconnection was observed. In such cases, the microcontroller continued to run based on its internal logic without needing external input, demonstrating fault tolerance.
- These tests were carried out using a physical prototype set up on a breadboard, allowing for direct observation of the traffic light operation and app response.

4. Regression Testing

Regression testing is performed to make sure that updates or additions to the system do not interfere with existing functionality:

- Whenever changes were made—such as adjusting timing intervals or modifying the interface with the Blynk app—the system underwent a full test cycle to confirm that earlier functionalities still worked as expected.
- Newly added features, such as special traffic modes or emergency overrides, were tested extensively to ensure they did not interrupt the base logic of automatic and manual controls.
- Continuous testing and iterative development helped in refining the system’s behavior

and maintaining consistent, dependable operation.

For your Smart Traffic Light (IoT Based) project, the most suitable and commonly used testing technique.

5. **Integration Testing**

Why Integration Testing is Most Relevant:

Regression testing is performed to make sure that updates or additions to the system do not interfere with existing functionality:

- Your system involves multiple components working together — the NodeMCU, LEDs, Blynk App, Wi-Fi communication, and the embedded logic.
- Integration testing ensures that:
    - The code correctly controls hardware components.
    - Wireless communication between the NodeMCU and Blynk App is stable.
    - Switching between manual and automatic modes works smoothly.
    - All parts interact without conflict or failure.

4.4 Test Cases Designed for the Project

The following test cases have been carefully designed to evaluate the major functionalities of the Smart Traffic Light (IoT Based) system. These test scenarios assess hardware behavior, software logic, and remote communication features through the Blynk platform.

- **Test Case 1: Manual Mode Activation via Blynk App**

  Description: This test verifies whether pressing a manual control button on the Blynk application activates the correct traffic lane.

  Input: User presses the Lane 1 button (V0) on the Blynk interface.

  Expected Output: The LED for Lane 1 turns green, while all other lanes remain red.

  Result: Pass/Fail

- **Test Case 2: Automatic Mode Traffic Light Cycling**

  Description: Ensure that when automatic mode is turned on, all four lanes transition properly in the defined green → yellow → red pattern.

  Input: User activates automatic mode by pressing the V4 toggle button.

  Expected Output: All lanes follow the correct cycle, one after the other, in a loop.

  Result: Pass/Fail

- **Test Case 3: Loss of Internet Connectivity**

  Description: To check if the NodeMCU continues running the programmed logic after losing Wi-Fi connection.

  Input: To check if the NodeMCU continues running the programmed logic after losing Wi-Fi connection.

  Expected Output: Traffic light sequence continues to run without any delay or interruption.

  Result: Pass/Fail

- **Test Case 4: LED Hardware Response**

  Description: This case tests whether the LED connected to a specific lane responds accurately when a command is given.

  Input: Press the manual button for Lane 3 (V2).

  Expected Output: The green LED on Lane 3 turns on; LEDs for the other lanes turn red.
  Result: Pass/Fail

- **Test Case 5: LED Hardware Response**

  Description: Verify that switching between manual and automatic modes doesn't result in any malfunction.

  Input: Activate Auto Mode, switch to Manual Mode for Lane 2, then return to Auto Mode.

  Expected Output: Manual mode overrides auto instantly and auto resumes correctly when

re- enabled.

Result: Pass/Fail

- **Test Case 6: LED Hardware Response**

  Description: This test measures the response time between issuing a command in the Blynk app and the hardware's response.

  Input: Press Lane 4 button (V3) on Blynk.

  Expected Output: Lane 4 traffic light responds within 1 second.

  Result: Pass/Fail

- **Test Case 7: System Initialization**

  Description: Ensure that the system starts in a stable state when powered on.

  Input: Power on the NodeMCU via USB.

  Expected Output: All lanes display red lights by default; automatic mode remains off until activated.

  Result: Pass/Fail

CHAPTER 5: RESULTS AND DISCUSSIONS

5.1 User Interface Representation

The system is designed to provide an interactive user interface for controlling hardware components (LEDs in this case) through a mobile application. The mobile interface facilitates manual and automated control of the connected devices via a clean, minimalistic layout. The interface includes toggle buttons to switch devices ON or OFF and additional increment/decrement buttons to control timing or sequencing functions.

As shown in the image, the smartphone app (most likely developed using platforms such as Blynk, MIT App Inventor, or a custom IoT application) is wirelessly connected to a microcontroller (such as Arduino or NodeMCU). The app sends commands to the microcontroller, which then controls the LEDs connected to a breadboard.

5.1.1 Brief Description of Various Modules of the System

1. **Mobile Application Module**
    - **Purpose:** Serves as the primary interface for the user.
    - **Functionality:** Provides ON/OFF buttons for LED control, "+/-" buttons for setting values (e.g., delays), and an "Auto Loop" toggle for continuous operation.
    - **Connectivity:** Communicates with the microcontroller over a shared Wi-Fi hotspot, ensuring wireless operation without needing internet access.
2. **Wi-Fi-Based Microcontroller Module**
    - **Purpose:** Acts as the brain of the system, processing input from the mobile app.
    - **Functionality:** Receives HTTP/MQTT/UDP commands from the app over Wi-Fi and controls the LEDs based on these inputs.
    - **Example:** NodeMCU or ESP8266 module, powered via USB cable from a mobile device or power bank.
3. **LED Driver and Circuit Module**
    - **Purpose:** Visual output system for the controlled devices.
    - **Functionality:** Contains multiple LEDs arranged on a breadboard, connected through resistors to protect from overcurrent.

- ***Manual Traffic Light Control Using Individual Buttons***

Figure 5.4: Green Light on Lane1

Figure 5.5: Showing Yellow Light at Lane1.

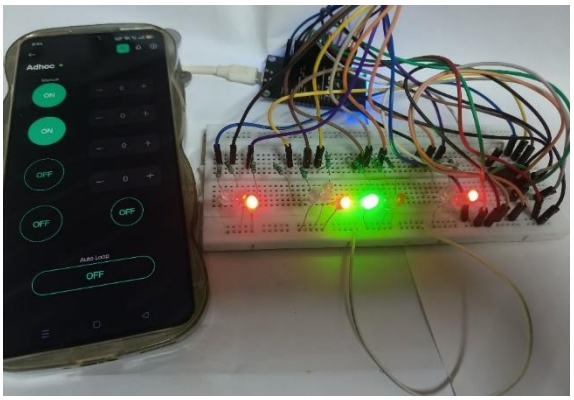

Figure 5.6:Green light at Lane2.

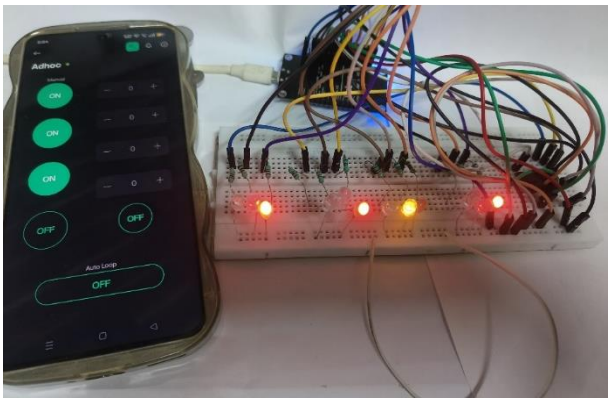

*Figure 5.7:*Y*ellow Light at Lane2*

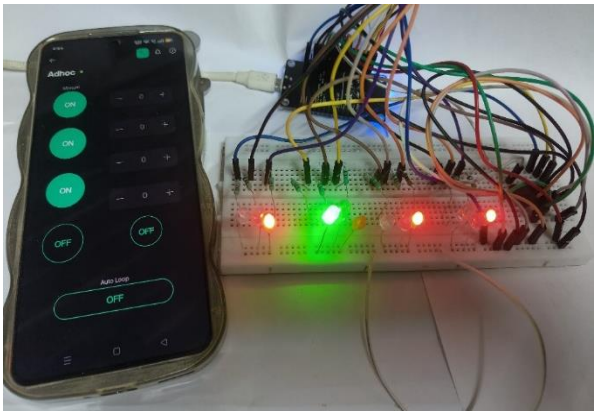

Figure 5.8: Green light at Lane3

Figure 5.9:Yellow Light at Lane3

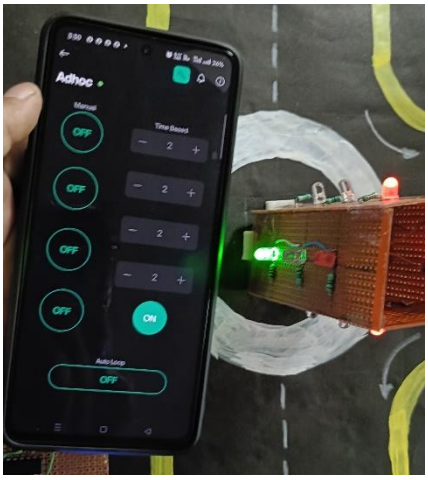

Figure 5.44: Green light at Lane2 (Prototype Timer)

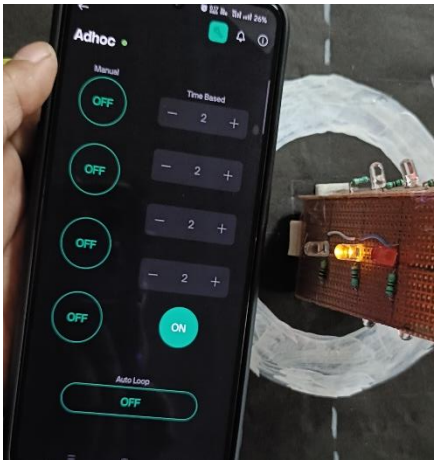

Figure 5.45: Yellow Light at Lane2 (Prototype Timer)

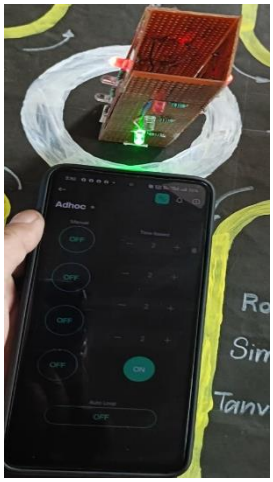

Figure 5.46: Green light at Lane3 (Prototype Timer)

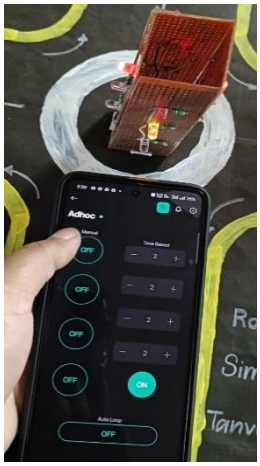

Figure 5.47: Yellow Light at Lane3 (Prototype Timer)

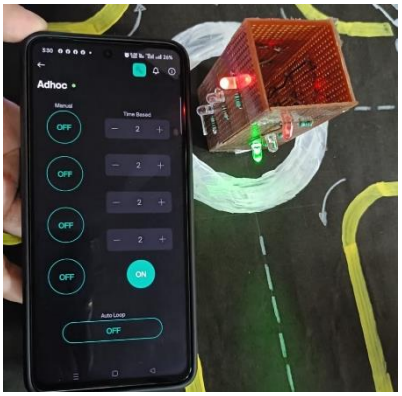

Figure 5.48: Green light at Lane4 (Prototype Timer)

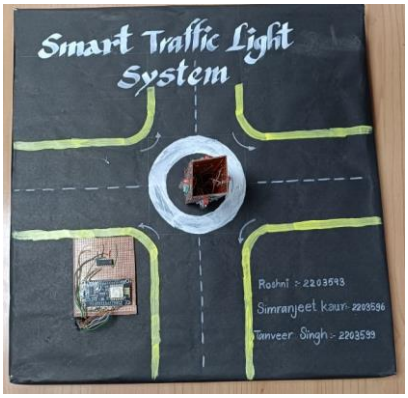

Figure 5.49: Smart Traffic Light System

REFERENCES

[1] AF Ahamed, Y Sukhi, S Anita, H Charulatha… - AIP Conference …, 2025 - pubs.aip.org" *IEEE Internet of Things Journal*, vol. 4, pp. 1125–1142, 2025

[2] Gough, M. B., et al., "Preserving privacy of smart meter data in a smart grid environment," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 1, pp. 707-718, 2021.

[3] Dizon, E., and Pranggono, B., "Smart streetlights in Smart City: a case study of Sheffield," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1-16, 2021.

[4] Avancini, D. B., et al., "A new IoT-based smart energy meter for smart grids," *International Journal of Energy Research*, vol. 45, no. 1, pp. 189-202, 2021.

[5] Bingöl, E., Kuzlu, M., and Pipattanasompom, M. A., "LoRa-based Smart Streetlighting system for Smart Cities," *7th International Istanbul Smart Grids and Cities Congress and Fair (ICSG)*, pp. 66–70, Apr. 2019.

[6] Lin, J., Yu, W., Zhang, N., Yang, X., Zhang, H., and Zhao, W., "A survey on Internet of Things: architecture, enabling technologies, security and privacy, and applications," *IEEE Internet of Things Journal*, vol. 4, pp. 1125–1142, 2020.