Research Article / Review Article

FUPRE JOURNAL 10(2): 75-92(2026)
https://doi.org//10.60787/FJSIR.v10i2.75-92

FUPRE Journal

of

Scientific and Industrial Research

ISSN: 2579-1184(Print) ISSN: 2578-1129 (Online)

http://fupre.edu.ng/journal

Rfid-Based Intelligent Traffic Control System for Emergency Vehicle Priority and Congestion Reduction

AVWORO, O. L. [1,*]

[1]*Department of Electrical Engineering, Federal University of Petroleum Resources, Effurun, Delta State, Nigeria*

ARTICLE INFO

Received: x/x/2026
Accepted: x/x/2026

Keywords

RFID, Arduino Mega 2560, emergency vehicle pre-emption, adaptive traffic control, intelligent transportation system, congestion management

ABSTRACT

Traffic congestion and delayed emergency vehicle response remain critical challenges in urban areas. Traditional fixed-time traffic signals fail to adapt to real-time traffic conditions and cannot provide instant priority to ambulances and other emergency vehicles. This project presents an intelligent traffic management system that integrates Radio Frequency Identification (RFID) technology with an Arduino Mega 2560 microcontroller to address both issues simultaneously. The proposed system operates in two modes. In the Adaptive Traffic Control Mode, low-cost infrared (IR) sensors detect vehicle density on each approaching lane. The Arduino processes this data and dynamically adjusts green light duration, allocating more time to heavier lanes to minimize queue length and waiting time. In the Emergency Vehicle Pre-emption Mode, authorized emergency vehicles equipped with unique passive RFID tags are detected by an RC522 reader. Upon valid tag detection, the system immediately overrides normal operation, turning the signal green in the vehicle's direction and red in all others, ensuring unobstructed passage. Additional components include 74HC595 shift registers for efficient control of multiple traffic light LEDs, seven-segment displays for countdown timers, and a stable 12V-to-5V power supply. The complete prototype was successfully designed, constructed, and tested, demonstrating reliable performance in both adaptive timing and emergency priority scenarios. Results confirm that the system significantly improves traffic flow under normal conditions while guaranteeing rapid, safe passage for emergency vehicles. This low-cost solution offers a practical approach toward smarter, safer urban traffic management.

1. INTRODUCTION

Traffic congestion in urban areas generates severe difficulties for efficient transportation systems, mainly affecting the timely delivery of emergency services. With cities growing denser, the increase in vehicle volume leads to longer delays at intersections, which can intensify healthcare crises by hindering ambulances from reaching patients on time (Belend and Brent 2020). Studies have shown that traffic congestion increases emergency response times directly, with some urban environments experiencing delays of up to 10 minutes or more during rush hour. These delays are particularly dangerous because, in cases of severe trauma or cardiac arrest, the survival rate drops significantly for every minute without medical intervention (Rahman et al. 2019).

*Corresponding author, : leslieavworo@gmail.com
https://doi.org//10.60787/FJSIR.v10i2.75-92
©Scientific Information, Documentation and Publishing Office at FUPRE Journal

Traditional traffic signal systems operate with fixed timing or simple sensors that cannot adapt to real-time traffic conditions, causing ambulances to wait or navigate through congested lanes unsafely (Bullen K. 2018). This not only endangers patients but also increases pressure on emergency responders and contributes to overall traffic congestion when other drivers refuse to grant them access (Hillsborough County Fire and Rescue 2025).

Various technologies have been developed to give priority to emergency vehicles at traffic lights. The most common include infrared systems (e.g., Opticom), sound-based sirens with microphone triggers, GPS-based systems, and camera-based solutions (Miovision 2025). While these solutions have reduced response time by 15–25% in many cities, they come with limitations. Optical systems are affected by weather and line-of-sight; GPS-based systems require continuous connectivity; camera systems are expensive and unreliable under poor lighting; and all tend to be expensive for large-scale deployment, especially in developing countries (Wang et al. 2013).

Radio Frequency Identification (RFID) has gained recognition as a reliable, low-cost, and weather-independent alternative for emergency vehicle pre-emption. In RFID-based systems, ambulances are equipped with RFID tags while readers are placed at intersections with traffic signal controllers (Dobkin D. 2013). When an ambulance approaches, the reader detects it and switches to emergency pre-emption mode (Techpacs Canada Limited 2025). Recent studies have successfully integrated RFID with Arduino and IoT platforms to create intelligent traffic systems that prioritise ambulances and restore normal operation after passage, showing high accuracy and significantly reduced waiting times (International Journal of Research and Analytical Reviews 2019).

Despite these advances, most existing solutions either lack hardware validation (Ali and Iqbal 2021), depend on internet connectivity (Patel et al. 2022), or do not integrate adaptive congestion control with emergency pre-emption in a single low-cost package. Furthermore, none of the previous works provide explicit timing logic tables for multiple congestion scenarios or use shift register-based I/O expansion for seven-segment countdown displays in a four-lane intersection model (Hussain, R. Ali, and M. Shah 2018).

This paper addresses these gaps by presenting a fully constructed and tested RFID-based intelligent traffic control system that combines:

- Adaptive green time adjustment using IR sensors (with mathematical formulation and timing tables for six congestion scenarios).
- RFID-based emergency vehicle pre-emption with dual-authentication (lane button + tag).
- Efficient I/O expansion using six 74HC595 shift registers to drive 12 traffic LEDs and four dual-digit seven-segment displays.
- A robust power management unit (Li-ion battery, BMS, buck converter).
- A physical T-junction model that simulates real intersection behavior.

The system is designed for deployment in Nigerian cities and other developing regions where cost, reliability, and offline operation are paramount.

2 METHODOLOGY AND

COMPONENTS

This section outlines all the different components utilized in the design and construction of this project. It covers the hardware specifications and the software environment used to develop the system's operational logic.

The task development involves two parts: Hardware design and Software design. The hardware is the physical part of the system while the software part consists of the algorithm of the traffic light system and the codes written to implement the logic, process sensor data, and regulate the timing sequences of the Arduino.

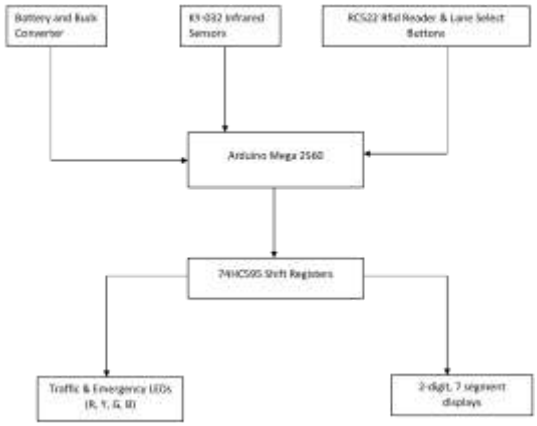


Figure 1: The System Block Diagram

2.1 Hardware Design

The hardware design for the traffic control system is categorized into four functional parts: the power unit, the input unit, the central processing unit, and the output/display unit as shown in Figure 1. Each part plays an important role in ensuring the system operates efficiently and reliably. The following sections provide a detailed analysis of these components, outlining their specific roles and technical specifications.

2.1.1 The Power Unit

The power unit as shown in figure 2, was designed to provide a regulated and centralized power distribution network for the whole system. It comprises of an SPST Rocker switch, 12v lithium-ion battery, Battery Management System (3s 20A), and DC-DC Buck Converter (LM2596).

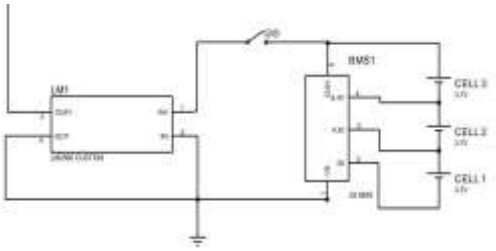


Figure 2: The Power Unit Circuit

- Lithium-ion battery: is the main supply of the circuit. It was chosen for this project due to its high energy density and portability. The battery pack was formed by connecting six individual 3.7v cells in a 3S2P (3-series, 2-parallel) configuration to achieve a 12.6v maximum output.
- Battery Management System (3s BMS): acts as a protection circuit, constantly monitoring voltage levels to prevent over charging, deep discharging, and short circuits.
- SPST Rocker switch: acts as the main on/off control, physically cutting off the power between the battery and the rest of the circuit.
- DC-DC Buck Converter (LM2596): was used as an interface between the battery pack and the rest of the circuit. Almost all components in the circuit require 5v which the buck converter steps down efficiently from 12.6v input. The regulated output was connected to the 5v pin on the Arduino Mega. Unlike other regulators like the LM7805, the LM2596 performs the conversion efficiently, minimizing

heat loss while providing a stable supply for the Arduino Mega, which in turn powers the rest of the components.

2.1.2 The Input Unit

The input unit as shown in figure 3, consists of components that send signals into the system and they include: infrared sensors (ky-032), RFID reader (rc522), and lane select buttons (SPST Momentary Tactile Push buttons). The input unit would be divided into two modules: traffic detection module and emergency priority module.

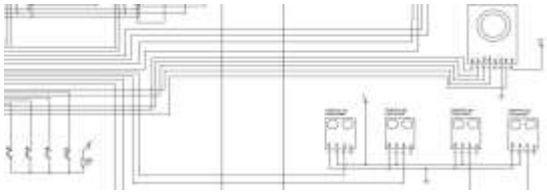


Figure 3: The Input Unit

- Traffic detection module (ky-032): to monitor vehicle presence, the system utilises four infrared sensors which are connected to digital pins 42, 43, 44, and 45 on the Arduino. These sensors were used because it uses an active transmitter-receiver pair which reliably detects the presence of a static vehicle. They operate on a digital "Active-LOW" logic protocol. What this means is that under normal conditions (no vehicle), the sensor outputs a logic 1 (HIGH), and when there is a vehicle which obstructs the sensor's beam, the sensor outputs a logic 0 (LOW), allowing the software to instantly flag congestion events and trigger the adaptive timing algorithm.
- Emergency priority module (rc522 + lane select buttons): To facilitate the specific intervention for emergency vehicles, the design makes use of an rc522 card reader which connects with the Arduino via the SPI protocol (SDA to pin 53, RST to pin 48, SCK to pin 52) with a set of four momentary pushbuttons (Pins 32, 33, 35, 36). These components operate as a dual-authentication mechanism: the buttons function as lane selectors that must be engaged simultaneously with a valid RFID card scan to grant the selected lane immediate priority.

2.1.3 The Central Processing Unit

The central processing unit of the system is the Arduino Mega 2560, a microcontroller board based on the ATmega2560. This unit as shown in Figure 4, acts as the brain of the traffic control system, responsible for executing the software instructions, processing sensor data, and generating the timing signals for the displays. It also supplies the power needed for the rc522 RFID reader via its 3.3v pin.

The Arduino Mega was specifically selected for this design over smaller boards like the Uno because of its extensive input/output (I/O) pins. The project requires a high number of digital connections to connect with the four infrared sensors, four manual push buttons, the RFID module and the control lines for the shift registers. The Arduino Mega provides 54 digital I/O pins and 256 kb of flash memory, which allows it to accommodate the complex wiring harness and store the extensive C++ code required for the string manipulation and countdown logic.

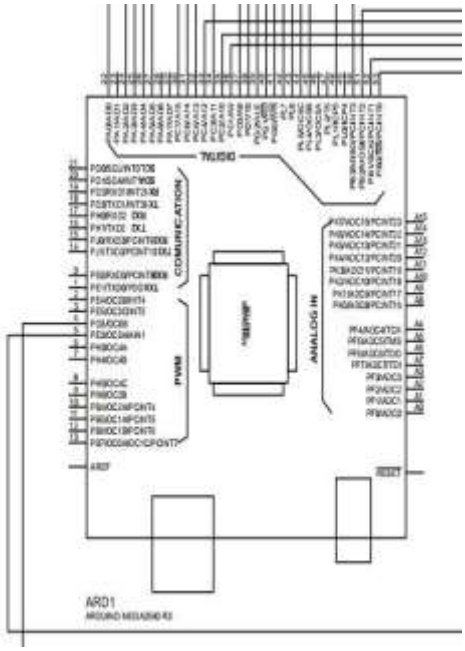


Figure 4: The Central Processing Unit

2.1.4 The Output and Display Unit

The output and display unit as shown in Figure 5, is responsible for translating the logic signals from the Arduino into visual traffic modules. Due to the high number of LEDs required for four traffic lanes (Red, Yellow, Green, and Blue emergency lights) and four sets of dual-digit seven segment displays, driving these components directly from the Arduino is not feasible.

To resolve this, the design uses 74HC595 8-bit shift registers. These integrated circuits function as serial-to-parallel converters and are configured in a parallel-driven configuration.

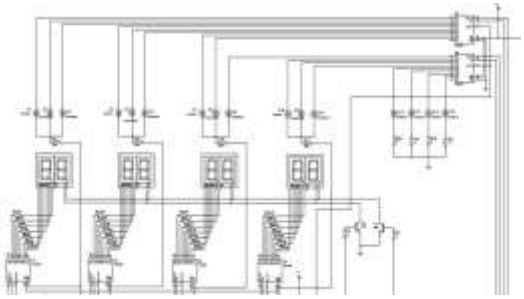


Figure 5: The Output and Display Unit

1. Shift Register Configuration: The six shift registers are divided into two functional groups: four registers dedicated to the seven segment displays and two registers dedicated to the LEDs. To ensure synchronized updates across the entire intersection, the Latch Pins (ST_CP) of all six registers are tied to a single common control line on the Arduino Mega (Pin 22).

However, the data and clock lines are separated to maximize update speed and modularity:

(i) Group A: Traffic signal control (2 registers)

- Traffic light set 1 which is controlled by register 5, connected to data pin 38 and clock pin 39.
- Traffic Light Set 2 and Emergency Lights which are controlled by register 6, connected to data pin 40 and clock pin 41.

(ii) Group B: Seven segment displays (4 registers)

- Display Unit A which is driven by register 1 via data pin 25 and clock pin 24.
- Display Unit B which is driven by register 2 via data pin 30 and clock pin 31.
- Display Unit C which is driven by register 3 via data pin 27 and clock pin 26.
- Display Unit D which is driven by register 4 via data pin 37 and clock pin 34.

This parallel configuration makes use of the Arduino Mega's extensive I/O capacity to prevent data latency, ensuring that the traffic lights and countdown timers update instantaneously without the lag often seen in long daisy-chained systems.

2. Common Cathode Multiplexing: The system uses common cathode 7-segment displays to show the remaining signal time. To drive these dual-digit modules using the registers described above, a time-division multiplexing technique is utilized:

    - Segment Control: The shift registers drive the individual segment anodes (A-G) with "Active-HIGH" logic.
    - Digit Selection and Amplification: The common cathodes of the displays are connected to NPN switching transistors (BC547). The base of the "Digit 1" and "Digit 2" transistors are connected to Pin 4 and Pin 5 via current limiting resistors.
    - Operational Logic: The software employs a high-speed switching algorithm that alternates between driving Pin 4 and Pin 5 HIGH. This saturates the respective transistor to ground the cathode, enabling the digit. This oscillation occurs rapidly enough to create the visual persistence of a continuous two-digit number while significantly reducing power consumption.

The Figure 6 shows the circuit diagram of the traffic control system.

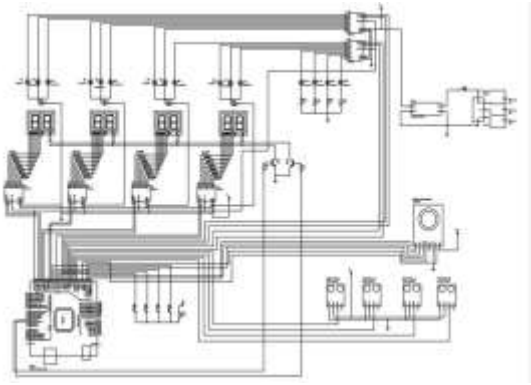


Figure 6: Circuit Diagram of the Traffic Control System

2.2 Software Design

The intelligence of the traffic control system is derived from the embedded software developed using C++ within the Arduino Integrated Development Environment (IDE). While the hardware provides the physical interface, the software is responsible for making logical decisions, managing timing sequences.

The software architecture is built upon a Finite State Machine (FSM) model which allows the system to transition dynamically between the different operational states (normal cycle, congestion cycle and emergency cycle) based on real sensor inputs. To ensure system responsiveness, the code utilizes non-blocking timing algorithms based on the millis()function, allowing the Arduino to update the seven segment displays, poll the RFID reader, and scan the infrared sensors simultaneously. The Figure 7 show the flow chat of the Emergency traffic light system.



Figure 7 Flow chart of the traffic light system

The code programmed in the Arduino was designed to first interpret and issue a command to start counting when it receives a signal from sensor A which could either be at lane 3 or lane 4 to activate the system, which was confirmed.

Next the condition for decongestion programmed in the code which is to grant more time for the vehicle on that particular lane if and only if sensors A and B on either lane 3 or 4 senses an obstruction at the same time, which was confirmed

The last condition which is to grant emergency passage if the and only if the button was pressed for the RFID card was a read by the RFID reader which was confirmed.

2.2.1 Input Unit

The input section of the code is responsible for constant monitoring to see if the traffic conditions have changed. Inside the main loop(), the software repeatedly polls the sensor pins to update its internal variables. This process handles two distinct types of input logic: the traffic sensors and the emergency authentication.

1. Active-Low Traffic Monitoring: For the traffic density, the code reads the infrared sensors on pins 42, 43, 44, and 45. Matching the hardware section, "Active-LOW" logic is used instead of “Active-HIGH”, so instead of HIGH signal, the software checks if the pin has gone LOW.
    - If digitalRead() returns 0, the code interprets this as a vehicle being present.
    - If it returns 1, the lane is considered empty. This inverse logic is captured immediately in boolean variables (like traffic_present), allowing the rest of the program to decide if it needs to switch to congestion mode.
2. Sequential RFID Verification: The emergency input is designed with a safety interlock. The code does not just scan for cards; it waits for the lane-selection button (Pins 32, 33, 35, or 36) to be pressed first. Only when a button press is detected does the software trigger the rfid reading process.

Once a card is scanned, the verification process is strict and sequential. The code reads the card's Unique ID (UID) and compares it against the authorised list byte-by-byte, starting from the beginning (Index 0) and moving to the end (Index 3).

- First, it checks uidByte[0].
- If that matches, it checks uidByte[1], and so on.
- Access is only granted if all four bytes, from start to finish, match the stored accessUID,

(0xA3, 0xCD, 0xF3, 0x28) perfectly. This

prevents partial matches or errors from triggering the emergency lights.

2.3.2 Processing Unit

Following data acquisition, the software enters the decision-making phase. This unit is responsible for executing the control algorithms that govern state transitions, priority allocation, and data serialization for the visual output modules.

1. Finite State Machine (FSM) Implementation: To maintain precise state control, the control logic is structured around a Finite State Machine (FSM) architecture. Unlike a linear execution loop, the system operation is segmented into discrete states, tracked by the global variable condition.
    - State Management: The condition variable serves as the state pointer, identifying the current operational phase.
    - Mode Arbitration: State transitions are not solely time-dependent; they are gated by the congestion flag. This variable acts as a conditional switch, determining whether the FSM executes the standard fixed-time protocol or engages the adaptive logic derived from sensor inputs.
2. Non-Sequential Routing: To handle the uneven traffic loads, the software implements a dynamic routing logic controlled by the passto variable.
    - The processing algorithm evaluates the sensor states from the infrared sensors to identify active lanes.
    - Based on this real-time assessment, the passto variable is updated to designate the next required state. This allows the FSM to bypass inactive lanes and dynamically transition to to the active traffic demand, thereby optimizing traffic flow.
3. Data Serialization and Formatting: Before signal transmission, logical data types must be converted into hardware-compatible formats for the display drivers.
    - Integer Decomposition: The system tracks the remaining signal duration as a single integer value. To drive the dual-digit seven-segment displays, the processing unit decomposes this integer into separate "Tens" and "Units" components via string conversion and substring extraction. These components are then re-indexed for the multiplexing algorithm.
    - Bitwise Mapping: The status of each traffic LED (Red, Yellow, Green) is mapped to a specific bit position. The processor aggregates these individual states into a composite binary string (TrafficValue). This process serialises the control data, organizing it into a singular packet structure suitable for synchronous serial transmission to the 74HC595 shift registers.

2.2.3 Output and Display Unit

This unit translates the processed integer data into electrical signals. It is responsible for serial communication with the 74HC595 shift registers and executing the high-speed switching required for display multiplexing.

1. *Serial Communication Protocol (shiftOut):* To transmit data to the shift registers, the software utilizes the

standard Arduino shiftOut() function. However, the code implements two distinct bit-ordering protocols depending on the target device:

- 7-Segment Displays (MSBFIRST): The numerical data for the countdown timers is transmitted with the most significant bit first. This aligns with the internal wiring of the display segments (A–G).
- Traffic LED Modules (LSBFIRST): The state of the traffic lights (Red/Yellow/Green) is transmitted with the least significant bit first. This reversal ensures the binary string maps correctly to the physical LED arrangement on the circuit board.

2. *Synchronized Latching Sequence:* To prevent ghosting or flickering during data updates, the software follows a strict latching sequence.

Latch LOW: The software pulls the common latch line (Pin 22, defined as latchPin_DispA) to Logic 0. This "opens" the storage registers to accept new data.

Data Transmission: The code executes six consecutive shiftOut commands, four for the displays and two for the traffic lights—filling the registers serially.

1. Latch HIGH: Only after all data bits are sent does the software pull Pin 22 to Logic 1. This closes the latch, pushing the data to the output pins simultaneously.

3. *Software Multiplexing Algorithm (toggleDigit):* To drive the dual-digit displays using shared data lines, the software implements a time-division multiplexing algorithm controlled by a boolean variable toggleDigit. The code runs on a rapid cycle (every 5ms) to alternate between the two states:

- State A (Digit 1 - Right Side): If toggleDigit is false, the software:
- Writes Pin 4 and Pin 5 LOW (turning off all displays to prevent bleed).
- Shifts out the binary pattern for the units digit (Index A).
- Writes Pin 4 HIGH. This activates the NPN transistor for the right-side digit.
- State B (Digit 2 - Left Side): If toggleDigit is true, the software:
- Writes Pin 4 and Pin 5 LOW.
- Shifts out the binary pattern for the tens digit (Index B).
- Writes Pin 5 HIGH. This activates the NPN transistor for the left-side digit.

This oscillation occurs rapidly enough that the human eye perceives both digits as being lit simultaneously.

2.3 System Operation Algorithm

To bring the logical flow from the previous sections together, the system's core operation is summarized in the pseudocode block below. This logic block maps out the complete control loop, showing exactly how the active-low sensor checks, emergency protocols, and adaptive timing work in sequence. It serves as a simplified overview of the C++ source code, demonstrating how the Finite State Machine (FSM) runs alongside the high-speed display multiplexing.

```
START
INITIALIZE System, Sensors, and Variables
SET CycleTime = 10s
LOOP FOREVER
READ Inputs (Infrared, Buttons, RFID)
// 1. EMERGENCY OVERRIDE
IF (Button Pressed AND Authorized RFID) THEN
Set Selected Lane GREEN; Others RED
WAIT 20 seconds
CONTINUE // Skip remaining logic
END IF
```

// 2. ADAPTIVE TIMING
IF (Traffic Detected) SET CycleTime = 20s
ELSE SET CycleTime = 10s
// 3. TRAFFIC CYCLE
DECREMENT Timer
IF (Timer <= 0) THEN
Switch Green Light to Next Lane
RESET Timer = CycleTime
END IF
// 4. DISPLAY DRIVER
Shift Data to Registers & Multiplex Digits (Continuous)
END LOOP
END

2.4 Mode of Operation

Upon activation, the system transitions through a defined sequence of states, ensuring safe initialization before engaging the automatic control loops. The Figure 8 shows the visual representation of the traffic control system.



Figure 8: Visual Representation of the Traffic Control System

2.4.1 System Initialization

When the system is powered via the 12V battery source, the LM2596 buck converter energises the Arduino Mega 2560 with a regulated 5V supply. The microcontroller immediately executes the setup() routine:

- *Pin Configuration:* All control pins defined for the shift registers and sensors are set to OUTPUT or INPUT modes respectively.
- *Peripheral Startup:* The SPI bus is initialised to communicate with the mfrc522 rfid reader.
- *Safety Reset:* The system executes an initial reset() protocol. This sets the internal congestion flags to neutral (congestion = -1), loads the default timing variables (max_time_lapse = 10), and forces the traffic lights into a safe starting state (Lane T1/T2 Green, Lane T3/T4 Red).

2.4.2 Standard Traffic Regulation Cycle

In the absence of heavy traffic or external interrupts, the system operates in a standard time-fixed mode. The finite state machine (FSM) cycles through using the default 10-second base interval with a specific transition logic:

Active Phase: The lanes without sensors, T1 and T2, display a Green signal while the 7segment displays show a synchronized countdown.

Preparation Phase (Yellow Phase): When the countdown reaches the calculated threshold (Time < Total/3), the system triggers a “get ready” signal for the waiting lanes. The Red light is replaced by the yellow light, indicating to stopped drivers that their Green signal is approaching.

Switching Phase: Upon reaching zero, the system seamlessly hands over priority to the waiting lanes, ensuring a conflict-free flow.

2.4.3 Adaptive Congestion Response

The system continuously monitors the four infrared sensors (Pins 42–45), on lanes T3 and T4. If a sensor detects a blockage (Active-LOW signal), the operational mode shifts dynamically:

Detection: The software checks if the blockage matches a specific "Passto" profile (e.g., Heavy traffic on Lane T4).

Locking Mechanism: To prevent erratic switching, the system sets a lock boolean to true. This ensures the current cycle finishes completely before the new timing logic is applied.

Timing Extension: For the subsequent cycle, the base time is doubled to 20 seconds. The priority lane (e.g., T4) uses this extended duration to clear the accumulated traffic, and the system maintains this locked state for the full duration, ensuring sequence complete before a new input data is processed.

2.4.4 Traffic Timing Formulation and Logic Table

To quantify the adaptive behaviour described in the previous sections, the system employs a governing mathematical model to calculate the duration of the signals for any given lane. This duration is not static; it is derived dynamically using the following general formula: Tsignal = Tbase × μstate

Where:

T_{base}: Represents the base timing unit, which shifts based on the global congestion flag.

Standard (Uncongested) mode: T_{base} = 10 seconds.

Congestion mode: T_{base} = 20 seconds.

μ_{state}: Represents the state multiplier (integer value 1, 2, or 3). This is determined by the specific passto index active at that moment.

Operational Logic: The system evaluates sensor inputs to assign a passto state (Index 1–6), which defines the active traffic scenario. Once a scenario is selected, the Finite State Machine (FSM) progresses sequentially through cases 1, 2, and 3, corresponding to the Green, Yellow, and Red signal phases. The duration for each of these active phases is calculated using the multiplier (μ) defined in tables below.

The sequence concludes at Case 4, which is a transient state designed to execute instantaneously. This final step acts as a hard reset trigger; it clears the current sequence and immediately reloads the timer with the next cycle's base value (10s or 20s) based on the updated congestion status. The Table 1,2 and 3 show the general timing and multipliers for passto 1&4, 2&5 and 3&6.

Table 1: General Timing Logic & Multipliers for Passto=1&4

Lanes	Case 1	Case 2	Case 3
T1	GREEN (T_{base} × 2)	GREEN (T_{base} × 1)	RED (T_{base} × 1)
T2	GREEN (T_{base} × 1)	RED (T_{base} × 2)	RED (T_{base} × 1)
T3	RED (T_{base} × 3)	RED (T_{base} × 2)	RED (T_{base} × 1)
T4	RED (T_{base} × 1)	GREEN (T_{base} × 2)	GREEN (T_{base} × 1)

Table 2: General Timing Logic & Multipliers for Passto=2&5

Lanes	Case 1	Case 2	Case 3
T1	GREEN (Tbase × 2)	GREEN (Tbase × 1)	RED (Tbase × 1)
T2	GREEN (Tbase × 1)	RED (Tbase × 2)	RED (Tbase × 1)
T3	RED (Tbase × 1)	GREEN (Tbase × 2)	GREEN (Tbase × 1)
T4	RED (Tbase × 3)	RED (Tbase	

× 2) RED (Tbase × 1)

Table 3: General Timing Logic & Multipliers for Passto=3&6

Lanes	Case 1	Case 2	Case 3
T1	GREEN (T_{base} × 2)	GREEN (T_{base} × 1)	RED (T_{base} × 1)
T2	GREEN (T_{base} × 1)	RED (T_{base} × 2)	RED (T_{base} × 1)
T3	RED (T_{base} × 2)	RED (T_{base} × 1)	GREEN (T_{base} × 1)
T4	RED (T_{base} × 1)	GREEN (T_{base} × 1)	RED (T_{base} × 1)

2.4.4.1 Standard (Uncongested) Mode Specifics

In the standard mode, the traffic density is low enough that vehicles do not back up to the position of the rear sensors, T3_SensPin_B and T4_SensPin_B. So,the base time (T_{base}) is set to 10 seconds. The specific signal durations for the passto=1,2,3 states are shown in the Tables 5, 6 and 7:

Table 4: Timing for Passto=1 (T4 has Traffic)

Lanes	Case 1	Case 2	Case 3
T1	GREEN (20)	GREEN (10)	RED (10)
T2	GREEN (10)	RED (20)	RED (10)
T3	RED (30)	RED (20)	RED (10)
T4	RED (10)	GREEN (20)	GREEN (10)

Table 5: Timing for Passto=2 (T3 has Traffic)

Lanes	Case 1	Case 2	Case 3
T1	GREEN (20)	GREEN (10)	RED (10)
T2	GREEN (10)	RED (20)	RED (10)
T3	RED (10)	GREEN (20)	GREEN (10)
T4	RED (30)	RED (20)	RED (10)

Table 6: Timing for Passto=3 (Both T3 & T4 have Traffic)

Lanes	Case 1	Case 2	Case 3
T1	GREEN (20)	GREEN (10)	RED (10)
T2	GREEN (10)	RED (20)	RED (10)
T3	RED (20)	RED (10)	GREEN (10)
T4	RED (10)	GREEN (10)	RED (10)

2.4.4.1 Congestion Mode Specifics

When the sensors flag a congestion event, the system shifts the base timing unit (T_{base}) to 20 seconds. This globally doubles the operational window giving priority to the congested lane. The specific signal durations for the passto=4,5,6 states are shown in the Tables 7, 8 and 9:

Table 7: Timing for Passto=4 (T4 has Traffic)

Lanes	Case 1	Case 2	Case 3
T1	GREEN (40)	GREEN (20)	RED (20)
T2	GREEN (20)	RED (40)	RED (20)
T3	RED (60)	RED (40)	RED (20)
T4	RED (20)	GREEN (40)	GREEN (20)

Table 8: Timing for Passto=5 (T3 has Traffic)

Lanes	Case 1	Case 2	Case 3
T1	GREEN (40)	GREEN (20)	RED (20)
T2	GREEN (20)	RED (40)	RED (20)
T3	RED (20)	GREEN (40)	GREEN (20)
T4	RED (60)	RED (40)	RED (20)

Table 9: Timing for Passto=6 (Both T3 & T4 have Traffic)

Lanes	Case 1	Case 2	Case 3
T1	GREEN (40)	GREEN (20)	RED (20)
T2	GREEN (20)	RED (40)	RED (20)
T3	RED (40)	RED (20)	GREEN (20)
T4	RED (20)	GREEN (20)	RED (20)

2.4.5 Emergency Override Protocol

This mode represents the highest priority interrupt in the system. It is activated only when an authorized RFID tag is scanned while a specific Lane Selection Button is held down.

- *Immediate Interrupt:* Upon authentication, the emergency() function triggers. The system ignores all current congestion flags and countdown timers.
- *Safety Clearance:* All conflicting lanes are immediately forced to Red.
- *Priority Access:* The selected lane (determined by the pressed button) is granted a fixed Green signal for a duration of 20 seconds (Emerg_time) to allow the emergency vehicle to pass .

Auto-Recovery: Once the emergency timer expires, the system automatically resets (Emerg_round = 0) and returns to the standard traffic regulation cycle.

3 RESULTS

The test results presented in the previous section confirm that the RFID-based traffic control system operates as intended across all three designed modes of operation. When the system was tested under normal traffic conditions with no congestion, the traffic lights cycled through their programmed sequences with the countdown timers displaying accurate values starting from ten seconds. This may seem like a simple achievement, but it is worth noting because many traffic lights currently in use across Nigerian cities do not have countdown displays at all. Drivers are often left guessing when the light will change, which leads to anxiety, premature acceleration, and sometimes red-light running. By providing clear countdown information, this system addresses that problem

directly.

When the system was tested under congested conditions, it behaved exactly as the design intended. The infrared sensors successfully detected when both sensors on a given lane were blocked by multiple toy cars, and this triggered the Arduino to increase the green light duration for that lane. In practical terms, what this means is that a lane with a long queue of vehicles received extra time to clear before the light changed to red. This is a significant improvement over conventional fixed-time traffic lights, which give the same green duration to a busy lane as they do to an empty lane. In real life, traffic volume varies throughout the day, with morning and evening rush hours creating long queues at major intersections. A system that cannot adapt to these changes forces drivers to wait needlessly, wasting time and fuel. The adaptive logic demonstrated in this prototype shows that a low-cost solution using simple infrared sensors can effectively address this problem.

The emergency vehicle pre-emption tests produced the most important findings of the entire study. When a lane button was pressed and the authorized RFID card was scanned, the system responded within milliseconds. If no vehicles were already moving through the intersection, the selected lane immediately turned green while all other lanes turned red, and this green light remained on for a full twenty seconds. This duration is sufficient for an ambulance to cross even a wide intersection safely, including time for acceleration and deceleration. If vehicles were already in motion when the emergency was triggered, the system showed a more sophisticated behaviour: instead of cutting the green light abruptly, it accelerated the countdown timers on the other lanes, allowing those vehicles to complete their passage before the emergency mode fully activated. This demonstrates that the designers considered real-world safety concerns, because suddenly turning a green light to red while a vehicle is already in the intersection could cause a collision.

The dual-authentication mechanism proved effective during all test trials. The system only responded when both conditions were satisfied simultaneously: a lane button pressed and a valid RFID card scanned. Pressing a button without a card produced no response. Scanning a card without pressing a button produced no response. Scanning an unauthorised card also produced no response. This is important because it prevents accidental activation by stray signals or electromagnetic interference, and it also prevents malicious actors from triggering the emergency mode without having both the correct tag and physical access to the button panel.

However, it must be acknowledged that the prototype has several limitations that would need to be addressed before the system could be deployed on real roads. The most significant limitation is the short range of both the infrared sensors and the RFID reader. The KY-032 sensors used in this prototype only detect vehicles within two to fifteen centimetres, which is perfectly adequate for a tabletop model but completely inadequate for a real intersection where vehicles are several metres away from the sensor. Similarly, the RC522 RFID reader only detects tags within three to five centimetres, meaning that an ambulance would have to stop directly

next to the reader to be detected. In a real deployment, the infrared sensors would need to be replaced with inductive loops buried in the road or with long-range laser or ultrasonic sensors. The RFID reader would need to be upgraded to a UHF model with a range of five to ten metres, and one reader would need to be installed on each traffic pole rather than using a single reader with manual lane selection buttons. The buttons themselves are another limitation, because in a real intersection an ambulance driver cannot stop to press a button at every traffic light. A practical system would use multiple readers so that the system automatically knows which lane the ambulance is approaching from.

Another limitation worth discussing is the absence of an all-red clearance period during emergency pre-emption. While the system does accelerate the countdown for moving vehicles before switching, it does not include a dedicated period where all directions show red for a few seconds. In traffic engineering, this all-red phase is a standard safety feature that allows vehicles that have already entered the intersection to clear out completely before conflicting traffic receives a green light. The absence of this feature in the current prototype means that there is a small but real risk of collision if a vehicle enters the intersection just as the emergency mode activates. Adding a three to five second all-red period before giving green to the emergency lane would eliminate this risk.

Despite these limitations, the prototype successfully demonstrates the core concepts that matter most. The adaptive congestion logic works. The RFID-based emergency pre-emption works. The countdown displays work. The system recovers automatically after an emergency event and returns to normal operation without any manual intervention. The total material cost of approximately two hundred and eighty-eight thousand naira, or about one hundred and eighty US dollars, makes this system far more affordable than commercial alternatives. Commercial emergency vehicle pre-emption systems such as Opticom or EMTRAC typically cost thousands of dollars per intersection, putting them out of reach for most Nigerian municipalities. A system based on the design presented here, even with the necessary upgrades for real deployment, would still cost only a fraction of those commercial systems.

When compared to previous research in this area, the present work offers several advances. Researchers like Hussain and colleagues demonstrated adaptive traffic control using infrared sensors but did not include any emergency vehicle detection capability. Ali and Iqbal simulated an RFID-based pre-emption system using software but did not build and test physical hardware. Patel and colleagues developed a hybrid RFID and Wi-Fi system that required internet connectivity, which is often unreliable in developing countries. The system presented here combines adaptive congestion control and emergency pre-emption in a single offline package that has been physically constructed and tested. It also includes countdown displays, which were absent from most previous studies, and it uses shift registers to expand the limited input output pins of the Arduino, allowing control of four

full lanes of traffic.

In conclusion, the test results demonstrate that an RFID-based intelligent traffic control system is technically feasible using low-cost, readily available components. The system successfully prioritises ambulances at intersections, adapts to congested conditions, and provides clear countdown information to drivers. While the prototype requires several upgrades before it could be deployed on real roads, the underlying logic and algorithms are sound. With the addition of longer-range sensors, multiple RFID readers, an all-red safety period, and weatherproof housing, this design could become a practical and affordable solution for smart traffic management in Nigerian cities and other developing regions where budget constraints make expensive commercial systems impossible to justify. The potential impact on emergency response times and on daily commuter experience makes further development of this approach worthy. The Figure 10 and 11 shows the final prototype of the traffic light system.

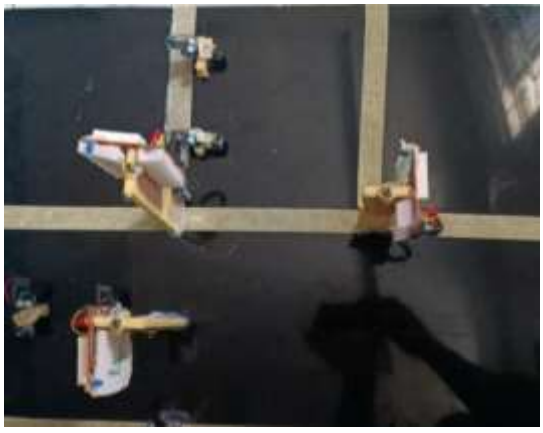


Figure 11 Final prototype of the traffic light system1

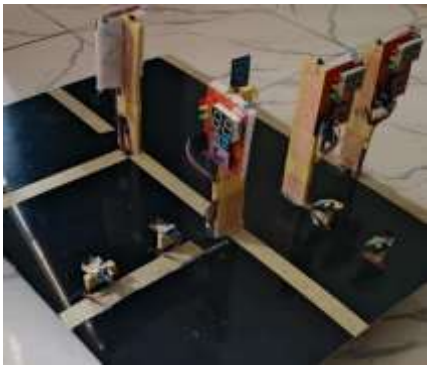


Figure 10 Final Prototype of the Traffic Light Control System2

4 CONCLUSION

This paper has presented the design, construction, and experimental validation of an intelligent traffic control system that combines adaptive congestion management and RFID based emergency vehicle pre emption. The system uses low cost, off the shelf components like Arduino Mega 2560, RC522 RFID reader, KY 032 IR sensors, 74HC595 shift registers and a seven segment displays to dynamically adjust green light durations based on real time vehicle density and to give immediate priority to authorized ambulances.

A physical T junction prototype was built and tested under three scenarios: normal flow, congestion, and emergency override. The results showed:

- Reliable congestion detection and automatic green time extension from 10 s to 20 s (or 40 s in multiplier cases).
- Emergency preemption with timing less than 100 milliseconds, with a 20seconds green light for the selected lane and automatic recovery.
- Accurate countdown displays with flicker free multiplexing.

- Total material cost of less than #300,000, making it affordable for developing countries.

The system directly addresses the urgent problem of delayed emergency response in Nigerian cities, offering a practical, scalable alternative to expensive commercial systems. With minor enhancements (multi lane RFID readers, solar power, and security features), this design can be deployed as a standard smart traffic controller across urban intersections.

Problems encountered are

o Complexity of the design in terms of the programming, the use of many variables to set up the system functionality, which made it complex

o It involves the use of lots of wires which made the wiring very complex there by adding to it's complexity.

o The use of push button for the emergency condition to be activated on each lane due to cutting down of cost

o Delay in shipping of the material because they were not readily available

o The pins are not firmly fixed to the Arduino board, for this reasons we sometimes encounter partial contacts where some of the components may function as expected.

REFERENCES

Akande, A. T. (2011). "Excessive speed as a Vital Human Factor in Road Traffic Accident" [Online]. Available: https://www.inigerian.com/excessive-speed-as-a-vital-human-factor-in-road-traffic-accident/. [Accessed: 25-July- 2020].

Benjamin, J. (2018). "How electronic throttle control (ETC) works" [Online]. Available: https://www.liveabout.com/how-electronic-throttle-control-works-4158200. [Accessed: 11 – August – 2020].

Daniel, A., Albert, P. and Mike, P. (2007). "*RFID A guide to radio frequency identification*", New Jersey: Wiley and Sons, Inc, pp. 5, 6.

Devikiruba, B. (2014). "Vehicle Speed Control System Using GSM/GPRS". *International Journal of Scientific and Technology Research*, 3(1): 157-162.

Eswaramoorthy, P. and Arunkumar M. (2014). "Intelligent vehicle control based on identification of road signs by solar powered RFID transponders", *International Journal of Research in Engineering and Technology*, 3(1): 85-89.

Fatema, T. H. M., Mehedi, H. A., Sukanta, D. and Moinuddin, S. (2015). "Automatic vehicle speed reduction system using RF Technology and accident prevention system", Bangladesh University of Engineering and Technology, Dhaka.

Federal Road Safety Corps (2012). "Road Traffic Crashes (RTC) Data", Federal Road Safety Corps, Abuja.

Federal Road Safety Corps (2014). "Flying on Nigeria roads", Federal Road Safety Corps, Abuja.

Gopal, P. G., Shruti, V. G., Irshad, A. Z. and Sagar, P. R. (2014). "Review of speed control and automatic braking system", *International Journal of Engineering Research and Technology*, 3(2): 474-476.

Gopinath, G. and Arikesh, A. (2013). "Development of Three Step Charging H-Bridge based Sine wave Inverter", *International Journal of Advanced Trends in Computer Science and Engineering*, 2(2):149-153.

Jamie, P. D. (2020). "How electronic throttle control systems works" [Online]. Available: https://auto.howstuffworks.com/car-driving-safety/safety-regulatory-devices/electronic-throttle-control-systems.htm. [Accessed: 11 – August – 2020].

Nandkishor P. J. and Ajay P. T. (2012). "Speed control of DC motor using analog PWM Technique". *International Journal of Engineering Research and Technology*, 1(9):1-5.

Payal, D. (2020). "Lack of sign boards, faulty design of speed breakers causes accidents too" [Online]. Available: https://timesofindia.indiatimes.com/city/ludhiana/lack-of-sign-boards-faulty-design-of-speed-breakers-cause-accidents-too/articleshow/76737880.cms. [Accessed: 4-August-2020].