International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

IoT-Based Traffic Monitoring and Control

K Saraswati

Govt. Degree college for Women

ABSTRACT

Understanding traffic is very important because of congestion, safety and efficiency in transportation systems. With the rapid growth of population in cities traffic congestion is often seen on roads. The movement of vehicles and people along a particular route or street, measured by speed, density, and volume. A point where the flow of traffic is restricted, causing delays, roads that are overcrowded with vehicles, causing delays. A traffic monitoring is a system which controls the traffic lights in accordance with the real time situation of traffic moving from all different directions . To solve above problem we can use IoT (Internet of things) is proposed .We can use different types of sensor to collect data and deal with traffic signal and congestion using IoT devices for better smoothly manage the traffic and safety. The goal of managing traffic is to ensure that it flows efficiently and safely, minimizing accidents and reducing delays..

Keywords— IoT sensors ,devices

Introduction

A city operates as a complex system comprising numerous interconnected subsystems, with the traffic system being one of its most critical components. Studies have highlighted its importance, with one describing it as the backbone of the global economy .Furthermore, traffic management is recognized as a key element in the concept of a smart city . As the global population continues to grow, the number of vehicles on the road also rises, leading to an inevitable increase in traffic congestion . Traffic jams not only waste valuable time but also foster a range of negative consequences, including the rise of criminal activities, such as mobile phone thefts at traffic lights, particularly in urban centers . In addition to its social impacts, traffic congestion significantly harms the environment and reduces industrial efficiency.

To the best of our knowledge, existing traffic management systems are predominantly centralized, which makes them vulnerable to network issues that can lead to system failure. Additionally, these systems often overlook the dynamic nature of traffic flow fluctuations. To address these challenges, the proposed system combines Internet of Things (IoT) to manage traffic through both local and centralized servers. Representing traffic data in a statistical format not only aids authorities in real-time traffic control and management but also provides valuable insights for future urban planning.

Internet of Things (IoT) is a network of physical devices, vehicles, sensors, and other objects embedded with software, sensors, and actuators that enable them to connect and exchange data over the internet. The key idea behind IoT is to bring the digital and physical worlds together by enabling everyday objects to collect, send, and receive data, making them "smart" and capable of making real-time decisions or performing tasks without human intervention.

COMPONENTS

The smart traffic management system consists of the following components:

1. **Sensors**: These are used to collect real-time data on traffic flow, vehicle count, and road conditions. Common sensors include cameras, radar, and inductive loop sensors.
2. **IoT Devices**: Internet of Things (IoT) devices allow for the seamless transmission of traffic data between local and centralized systems, enabling real-time monitoring and control.
3. **Local and Centralized Servers**: Local servers handle immediate traffic management on-site, while centralized servers aggregate data from multiple locations to optimize traffic patterns across the entire city.
4. **Communication Network**: A reliable communication network facilitates the exchange of data between IoT devices, sensors, local, and centralized servers.
5. **Traffic Signal Controllers**: These devices are responsible for controlling the timing of traffic lights based on real-time traffic data to ensure smooth and efficient vehicle movement.

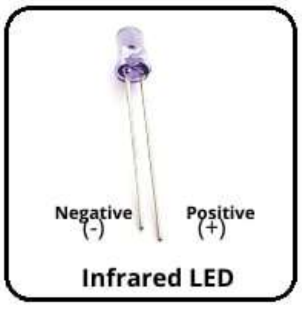
6. **Data Analytics and Visualization Tools**: These tools are used by authorities to interpret traffic data, monitor system performance, and plan for future traffic management strategies.
7. **Cloud Computing**: Cloud-based infrastructure supports large-scale data storage, processing, and remote access, enabling authorities to manage traffic data and make decisions from anywhere.
8. **User Interface (UI)**: The UI allows traffic control operators and city planners to monitor the system, make adjustments, and access insights from the data collected by the system.

These components work together to create a highly efficient and responsive smart traffic management system.

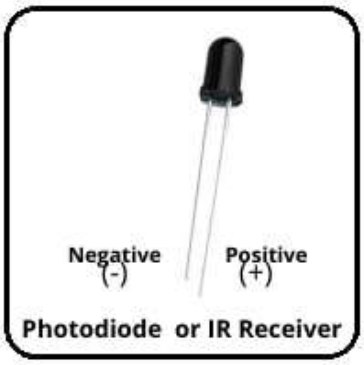
Hardware Requirement

Arduino: Arduino is a prototype platform (open source) based on easy-to-use hardware and software. It consists of a board that can be programmed (called a microcontroller) and ready-made software called Arduino IDE, which is used to write and upload computer code to the physical board. Arduino boards are capable of reading analog or digital input signals from various sensors and converting them to an output, such as activating a motor, turning LED on and off, connecting to the cloud, and performing many other actions. The board's functions are controlled by sending a series of instructions to the microcontroller on the board via Arduino IDE (this is called uploading software). Unlike most other programmable boards, Arduino does not require additional hardware (called a programmer) to upload new code to the board. A USB cable is used for this purpose. Also, Arduino IDE uses a simplified version of C++, which makes it easier to learn how to program. Finally, Arduino offers a standard form factor that puts the functions of the microcontroller in a more accessible package. Figure shows the Arduino Uno, a microcontroller board based on Atmel's ATmega328. It has 14 digital input and output pins, 6 of which can be used as PWM outputs and 6 as analog inputs. The Arduino Uno can be powered via a USB connection or with an external power supply. The Arduino Uno includes everything needed to support the microcontroller. Simply connect it to a computer with a USB cable.

1 IR Transmitter: IR the transmitter looks like a LED. This IR transmitter always emits IR beams. The operating voltage of this IR transmitter is 2 to 3 V. Infrared is an invisible radiant energy, electromagnetic radiation with longer wavelengths than visible light, extending from the nominal red edge of the visible spectrum at 700 nanometers (frequency 430 THz) to 1000000 nm (300 GHz) (although humans can see infrared up to at least 1050 nm in experiments). The maximum detection range of the sensors from IR is 5 m.

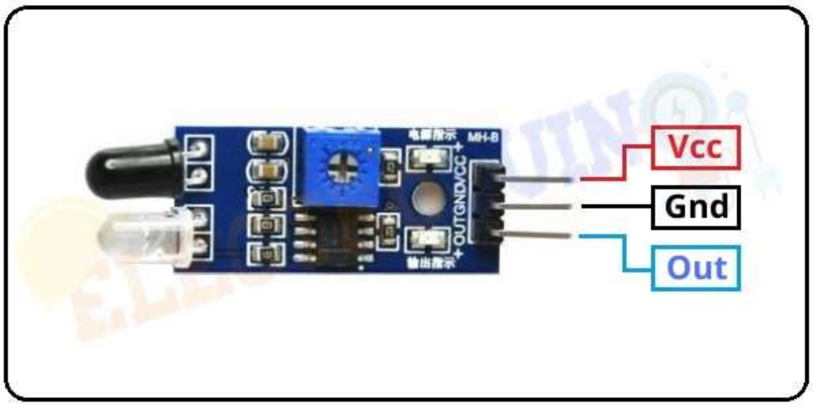


3.3.2 IR Receiver (Photodiode): A photodiode is a semiconductor device that converts light into electricity. The current is generated when photons are absorbed into the photodiode. A small amount of current is also generated when no light is present. Photodiodes may contain optical filters or built-in lenses and may have a large or small surface area. Photodiodes generally have a slower response time the larger their surface area. Photodiodes are similar to normal semiconductor diodes, except that they are either exposed to light (to detect UV or X-rays in a vacuum) or have a window or fiber optic port to allow light to reach the sensitive part of the device. Many diodes designed specifically for use as photodiodes use a PIN junction instead of a p-n junction to increase the speed of response.



How to use IR Sensor Module with Arduino

We need to connect the IR sensor with Arduino properly to read the output of the sensor. First of all, we connected the sensor **Vcc pin** to the Arduino **5v pin** and the **GND** pin is connected to the Arduino **ground (GND)** pin, to activate the IR sensor module. Then connect the sensor output pin to one of the digital pin of Arduino to read the output value from the IR sensor module.



Traffic Light: Traffic lights follow a universal color code that changes the direction given to users in the order of bright lights or three- color LEDs: Red, Green, and Yellow. Red indicator is used to stop the vehicles in the road traffic; Green indicator is used to allow the vehicles to move in the road traffic, and yellow indicator is used to alert the people in traffic lane to be ready for moving the vehicles in the road-traffic. Figure shows the colors used in traffic lights.

METHODOLOGY

The core concept behind dynamically controlling traffic light timings based on real-time traffic conditions is as follows:

- Sensors monitor and gather real-time data regarding the vehicle density on the road.
- The collected data is transmitted and stored in the cloud for further analysis.
- This traffic data is sent to the microcontroller, which processes the information to adjust the traffic signal timings for each lane accordingly.
- In the event of an emergency, the relevant data is directly sent to the microcontroller, allowing immediate changes to the traffic signals to prioritize emergency vehicles.

Programming

Arduino is an open-source electronics platform that uses **C/C++** programming to control microcontrollers. The Arduino IDE (Integrated Development Environment) allows users to write, compile, and upload C++ programs (also called "sketches") to an Arduino board, which can interact with sensors, motors, LEDs, and many other electronic components.Arduino programming is based on a simplified version of C/C++, making it easier for beginners, while still being powerful enough for advanced users.Vehicle Counting: You can modify the code to count the number of vehicles detected and display this on the Serial Monitor. You can add a vehicle counter to track how many vehicles have passed through each lane.Power Supply: Ensure that the Arduino and sensors are powered properly.Sensor Calibration: IR sensors may need to be calibrated based on your environment (e.g., adjusting the distance or sensitivity).LEDs: Ensure you are using current-limiting resistors with your LEDs to avoid burning them out. The term Arduino uses for a program. Every sketch must have two core functions:**setup()**: This function runs once when the Arduino starts or resets. It's used to initialize variables, pin modes, and start communication protocols.**loop()**: This function runs repeatedly after setup() and contains the main logic for controlling the hardware (e.g., turning an LED on/off, reading sensors). Arduino sketches are written in **C++**. They include all the key components of C/C++ programming, such as Variables **,** Functions**,** Control structures, Libraries: Reusable code packages that allow you to use advanced functionality, such as controlling motors, sensors, displays, etc.

Digital & Analog Pins: Arduino boards have digital pins for reading or writing binary values (HIGH or LOW) and analog pins for reading values in the range of 0 to 1023.

Input/Output: Arduino interacts with the physical world through I/O operations. You can read data from sensors (inputs) and control actuators like LEDs or motors (outputs).

```
const unsigned long debounceDelay = 200; // Debounce delay in milliseconds

unsigned long lastSensor1Time = 0;

unsigned long lastSensor2Time = 0;


void loop() {

  unsigned long currentTime = millis();


  // Read the sensor values

  int sensor1State = digitalRead(sensor1Pin);

  int sensor2State = digitalRead(sensor2Pin);


  // Check if vehicle is detected on the left lane

  if (sensor1State == HIGH && (currentTime - lastSensor1Time) > debounceDelay) {

    digitalWrite(led1Pin, HIGH);  // Turn on the Green LED

    Serial.println("Vehicle detected on the left lane");

    lastSensor1Time = currentTime;  // Update the time of the last detection

  } else {

    digitalWrite(led1Pin, LOW);   // Turn off the Green LED

  }


  // Check if vehicle is detected on the right lane

  if (sensor2State == HIGH && (currentTime - lastSensor2Time) > debounceDelay) {

    digitalWrite(led2Pin, HIGH);  // Turn on the Red LED

    Serial.println("Vehicle detected on the right lane");

    lastSensor2Time = currentTime;  // Update the time of the last detection
  } else {
    digitalWrite(led2Pin, LOW);   // Turn off the Red LED
  }

  delay(100); // Small delay for stability
}

int leftLaneCount = 0;
int rightLaneCount = 0;

void loop() {
  unsigned long currentTime = millis();

  // Read the sensor values
  int sensor1State = digitalRead(sensor1Pin);
  int sensor2State = digitalRead(sensor2Pin);

  // Check if vehicle is detected on the left lane
  if (sensor1State == HIGH && (currentTime - lastSensor1Time) > debounceDelay) {
    leftLaneCount++;  // Increment left lane count
    digitalWrite(led1Pin, HIGH);  // Turn on the Green LED
    Serial.print("Vehicle detected on the left lane. Total: ");
    Serial.println(leftLaneCount);
    lastSensor1Time = currentTime;  // Update the time of the last detection
  } else {
    digitalWrite(led1Pin, LOW);   // Turn off the Green LED
  }

  // Check if vehicle is detected on the right lane
  if (sensor2State == HIGH && (currentTime - lastSensor2Time) > debounceDelay) {
    rightLaneCount++;  // Increment right lane count
    digitalWrite(led2Pin, HIGH);  // Turn on the Red LED
    Serial.print("Vehicle detected on the right lane. Total: ");
    Serial.println(rightLaneCount);
    lastSensor2Time = currentTime;  // Update the time of the last detection
  } else {
    digitalWrite(led2Pin, LOW);   // Turn off the Red LED
  }
  delay(100); // Small delay for stability
}
```

Conclusion

The smart traffic management system has proven to be highly effective in reducing both waiting and travel times for passengers. Additionally, it facilitates the seamless movement of emergency vehicles, ensuring they can navigate without encountering unnecessary obstacles or delays. Implementing this system in key urban areas can significantly contribute to lowering pollution levels by optimizing traffic flow and reducing congestion. Furthermore, the proposed traffic management system is highly adaptable and reliable, making it well-suited for implementation in metropolitan cities. Its benefits in terms of efficiency, environmental impact, and safety make it a valuable solution for modernizing urban transportation infrastructure.

REFERENCES

 Z. Li, M. Shahidehpour, S. Bahramirad, and A. Khodaei, "Optimizing traffic signal settings in smart cities," *IEEE Transactions on Smart Grid*, vol. 8, no. 5, pp. 2382-2393, 2017.

 L. Singh, S. Tripathi, and H. Arora, "Time optimization for traffic signal control using genetic algorithm," *International Journal of Recent Trends in Engineering*, vol. 2, no. 2, pp. 4-6, 2009.

 S. N. Pable, A. Welekar, and T. Gaikwad-Patil, "Implementation on priority based signal management in traffic system," *International Journal of Engineering Research & Technology (IJERT)*, vol. 3, no. 5, pp. 1679-1682, 2014.

 V. Milanes, J. Villagra, J. Godoy, J. Simo, J. Pérez, and E. Onieva, "An intelligent V2I-based traffic management system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 1, pp. 49-58, 2012.