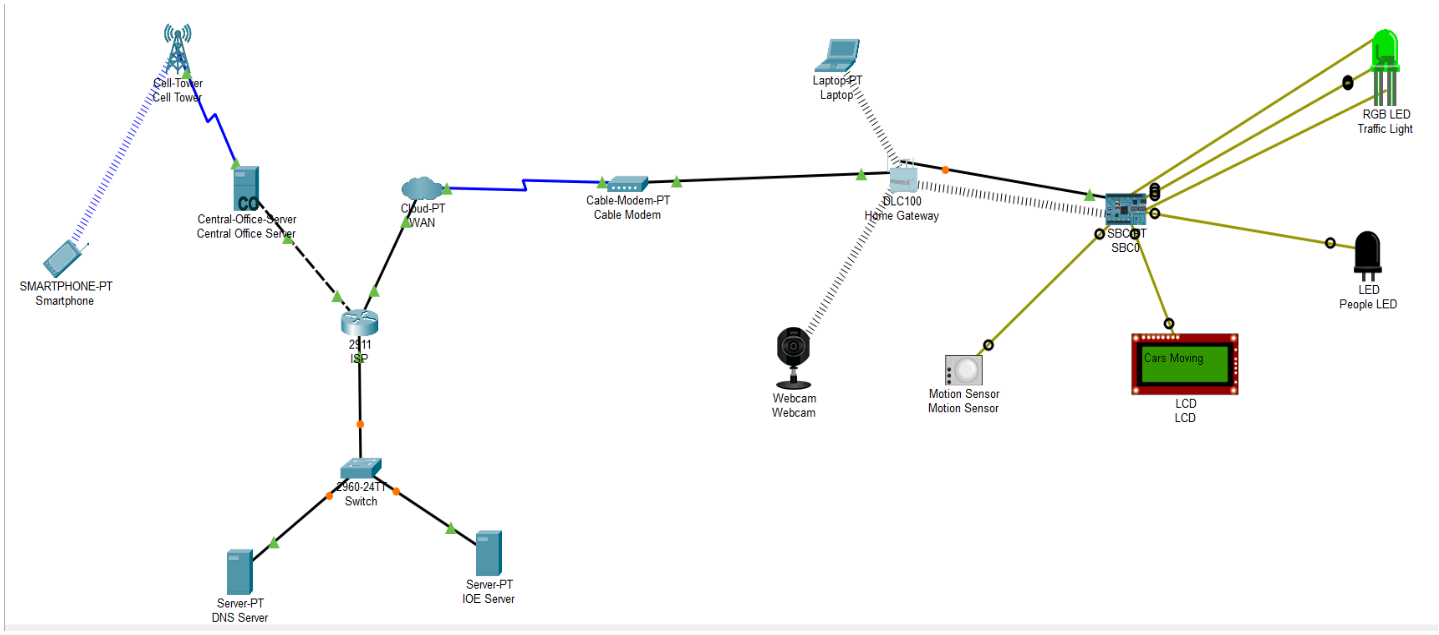
SMART TRAFFIC LIGHT SYSTEM

Introduction:

A smart traffic light is a traffic signal equipped with sensors and connectivity features that allow it to adjust its timing based on real-time traffic conditions. It can change light patterns to reduce congestion, improve traffic flow, and respond to changing traffic patterns more efficiently than traditional traffic lights. This helps manage traffic better and reduces waiting times at intersections.

Smart Traffic Light Design in Cisco Packet Tracer :



1. Edge Computing:

- **SBC0 (Single Board Computer):**

This device processes data locally and interacts directly with sensors and actuators. It's positioned at the "edge" of the network, making real-time decisions based on local data inputs.

- **Motion Sensor:**

It generates data at the edge. The motion sensor detects movements and provides data that needs to be processed quickly, which is often handled by nearby edge devices like SBCs.

- **Webcam:**

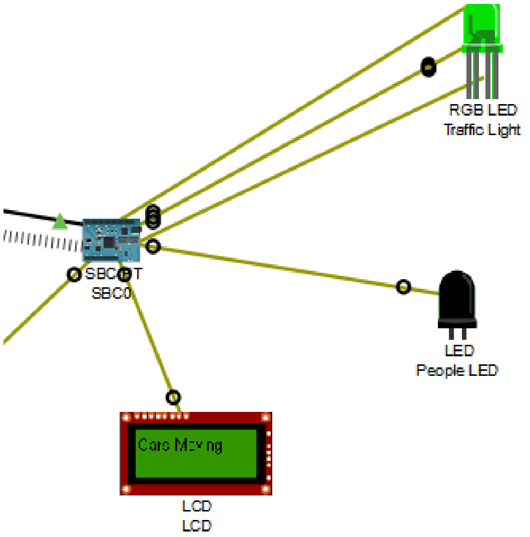
Similar to the motion sensor, the webcam captures data (video) at the edge. Edge devices process this data to perform actions such as object detection or surveillance analysis.

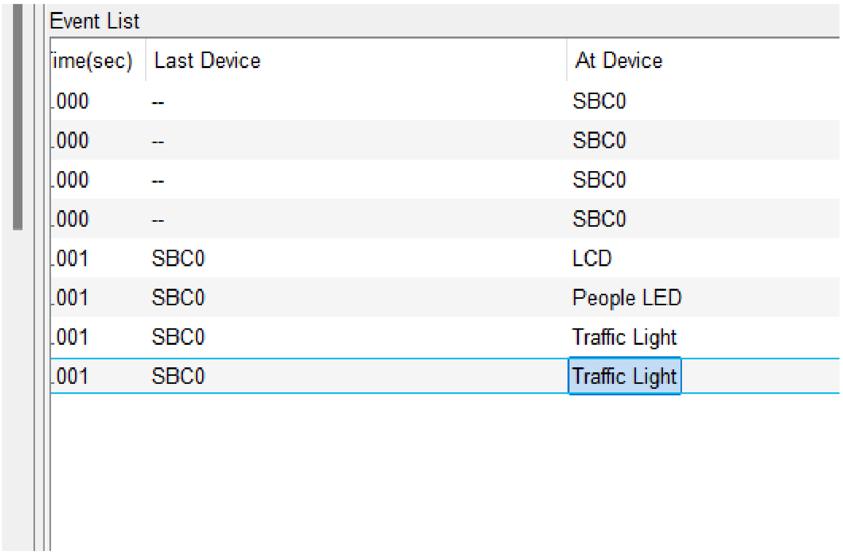
- **RGB LED Traffic Light and People LED:**

These actuators are controlled by edge devices based on processed sensor data, providing real-time responses to environmental changes.

- **LCD:**

Displays information that is processed locally, such as status messages or alerts, allowing for immediate interaction and feedback.





ime(sec)	Last Device	At Device
.000	--	SBC0
.000	--	SBC0
.000	--	SBC0
.000	--	SBC0
.001	SBC0	LCD
.001	SBC0	People LED
.001	SBC0	Traffic Light
.001	SBC0	Traffic Light

2. **<u>Cloud Computing:</u>**

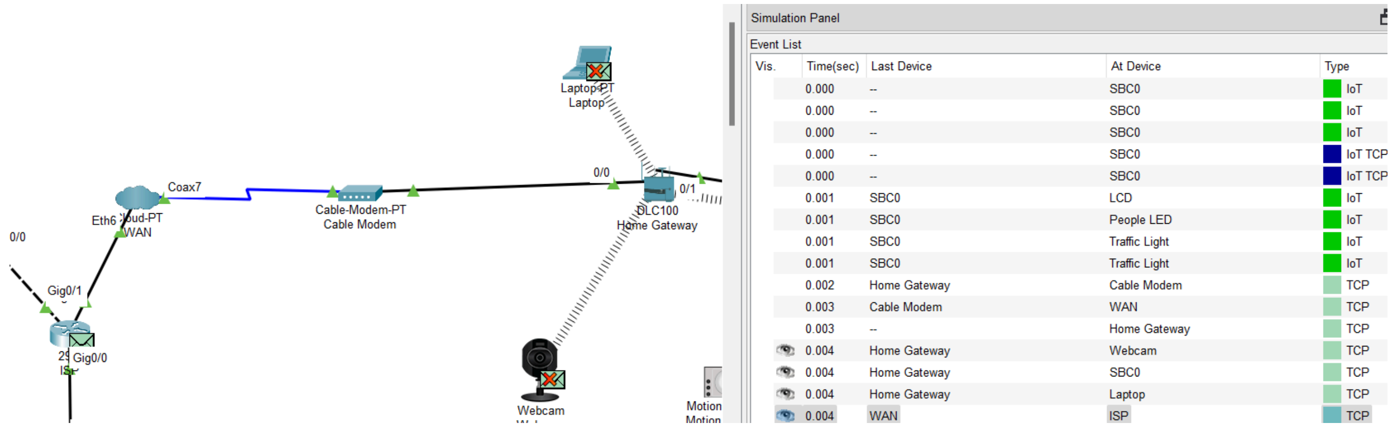
Cloud computing involves using remote servers hosted on the Internet to store, manage, and process data, rather than relying on local servers or personal devices.

- **Cloud-PT (WAN):**

Represents cloud infrastructure, providing the network link to centralized resources where large-scale data processing, storage, and analysis occur.

- **Central-Office-Server:**

Acts as an interface between local networks and cloud services, handling communications, data transfers, and possibly hosting cloud-based applications or services.



Simulation Panel

Event List

Vis.	Time(sec)	Last Device	At Device	Type
	0.000	--	SBC0	🟩 IoT
	0.000	--	SBC0	🟩 IoT
	0.000	--	SBC0	🟩 IoT
	0.000	--	SBC0	🟦 IoT TCP
	0.000	--	SBC0	🟦 IoT TCP
	0.001	SBC0	LCD	🟩 IoT
	0.001	SBC0	People LED	🟩 IoT
	0.001	SBC0	Traffic Light	🟩 IoT
	0.001	SBC0	Traffic Light	🟩 IoT
	0.002	Home Gateway	Cable Modem	🟩 TCP
	0.003	Cable Modem	WAN	🟩 TCP
	0.003	--	Home Gateway	🟩 TCP
👁	0.004	Home Gateway	Webcam	🟩 TCP
👁	0.004	Home Gateway	SBC0	🟩 TCP
👁	0.004	Home Gateway	Laptop	🟩 TCP
👁	0.004	WAN	ISP	🟦 TCP

3. **<u>Mobile Computing:</u>**

Mobile computing involves portable computing devices that allow users to access and process data while on the move, independent of a physical location.

- **SMARTPHONE-PT (Smartphone):**

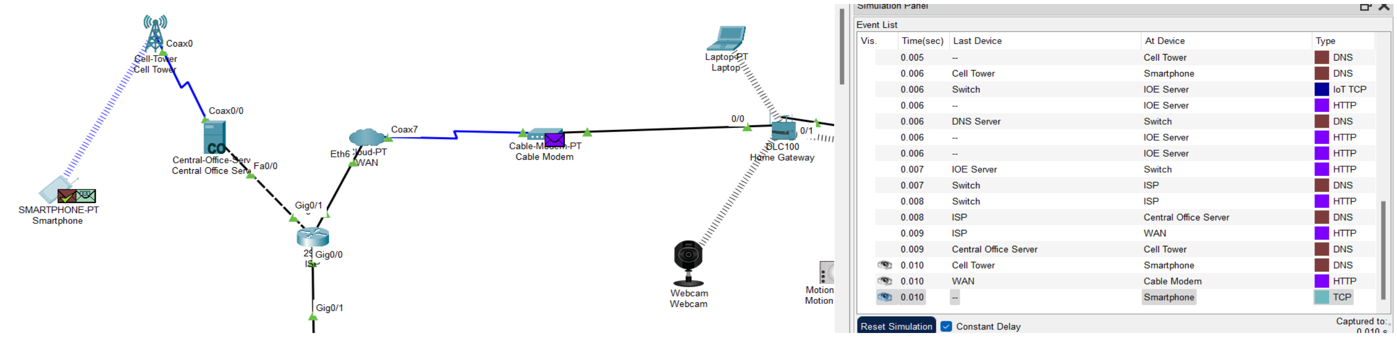
A quintessential mobile computing device, it enables users to interact with applications and services from anywhere. It can connect to edge or cloud services via cellular networks.

- **DNS (Domain Name System):**

DNS is a hierarchical naming system used to translate human-readable domain names (e.g., `www.example.com`) into IP addresses (e.g., `192.0.2.1`) that computers use to identify each other on the network. This translation allows users to access websites using easy-to-remember domain names instead of numerical IP addresses.

Working of the System:

The images below show the journey of packets, the protocols used at each layer, and how these protocols enable communication between the source and destination devices.

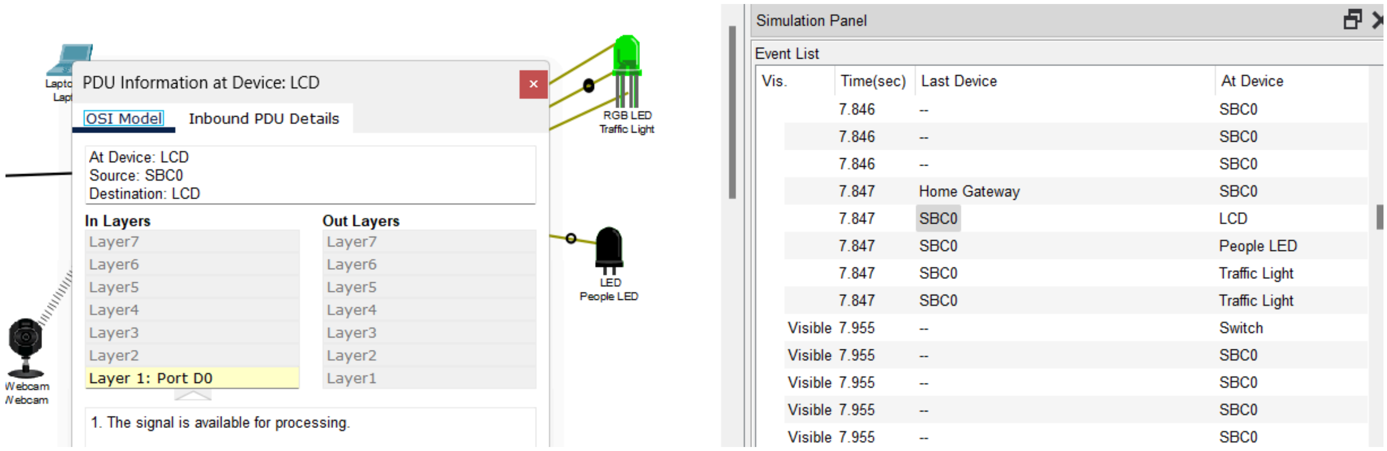


Event List

Vis.	Time(sec)	Last Device	At Device	Type
	0.005	--	Cell Tower	DNS
	0.006	Cell Tower	Smartphone	DNS
	0.006	Switch	IOE Server	IoT TCP
	0.006	--	IOE Server	HTTP
	0.006	DNS Server	Switch	DNS
	0.006	--	IOE Server	HTTP
	0.006	--	IOE Server	HTTP
	0.007	IOE Server	Switch	HTTP
	0.007	Switch	ISP	DNS
	0.008	Switch	ISP	HTTP
	0.008	ISP	Central Office Server	DNS
	0.009	ISP	WAN	HTTP
	0.009	Central Office Server	Cell Tower	DNS
👁	0.010	Cell Tower	Smartphone	DNS
👁	0.010	WAN	Cable Modem	HTTP
👁	0.010	--	Smartphone	TCP

Captured to: 0.010 s

Packets from Home Gateway to SBC0



PDU Information at Device: LCD

At Device: LCD
Source: SBC0
Destination: LCD

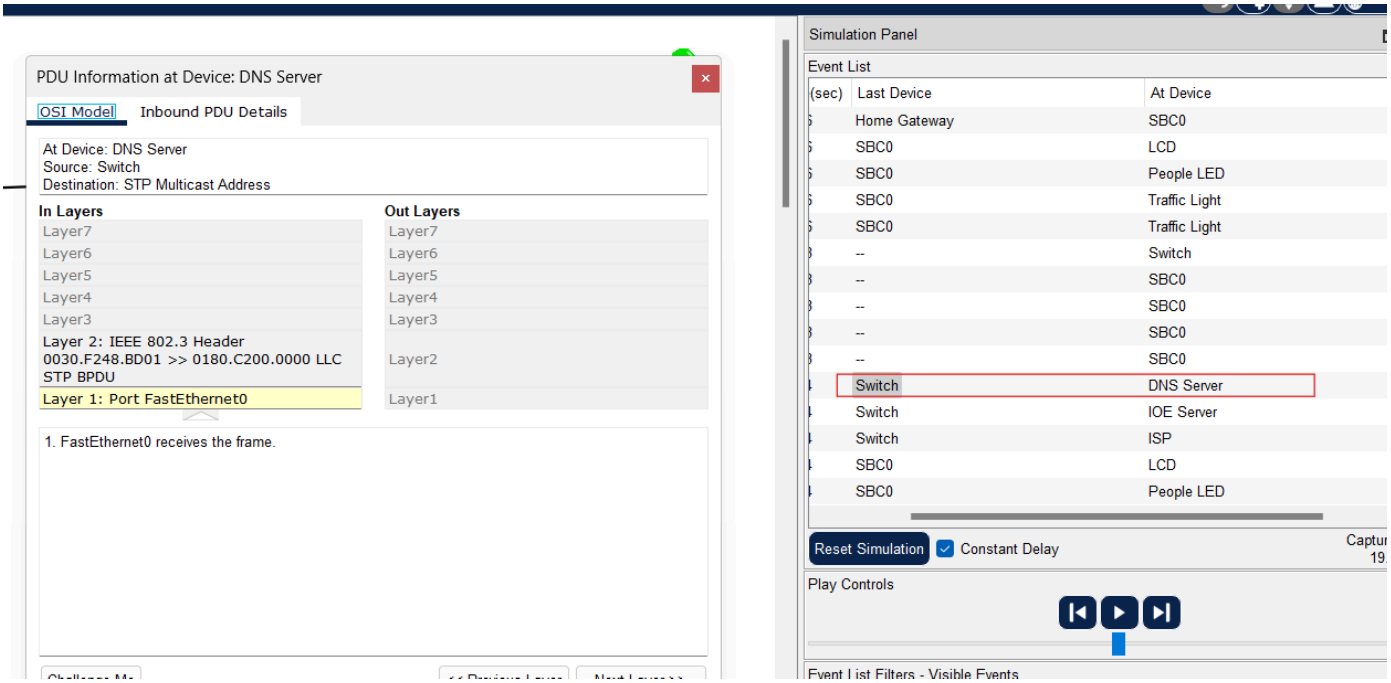
In Layers	Out Layers
Layer7	Layer7
Layer6	Layer6
Layer5	Layer5
Layer4	Layer4
Layer3	Layer3
Layer2	Layer2
Layer 1: Port D0	Layer1

1. The signal is available for processing.

Event List

Vis.	Time(sec)	Last Device	At Device
	7.846	--	SBC0
	7.846	--	SBC0
	7.846	--	SBC0
	7.847	Home Gateway	SBC0
	7.847	SBC0	LCD
	7.847	SBC0	People LED
	7.847	SBC0	Traffic Light
	7.847	SBC0	Traffic Light
Visible	7.955	--	Switch
Visible	7.955	--	SBC0
Visible	7.955	--	SBC0
Visible	7.955	--	SBC0
Visible	7.955	--	SBC0

From SBC to devices like PeopleLED, Traffic Light and LCD.



PDU Information at Device: DNS Server

At Device: DNS Server
Source: Switch
Destination: STP Multicast Address

In Layers	Out Layers
Layer7	Layer7
Layer6	Layer6
Layer5	Layer5
Layer4	Layer4
Layer3	Layer3
Layer 2: IEEE 802.3 Header 0030.F248.BD01 >> 0180.C200.0000 LLC STP BPDU	Layer2
Layer 1: Port FastEthernet0	Layer1

1. FastEthernet0 receives the frame.

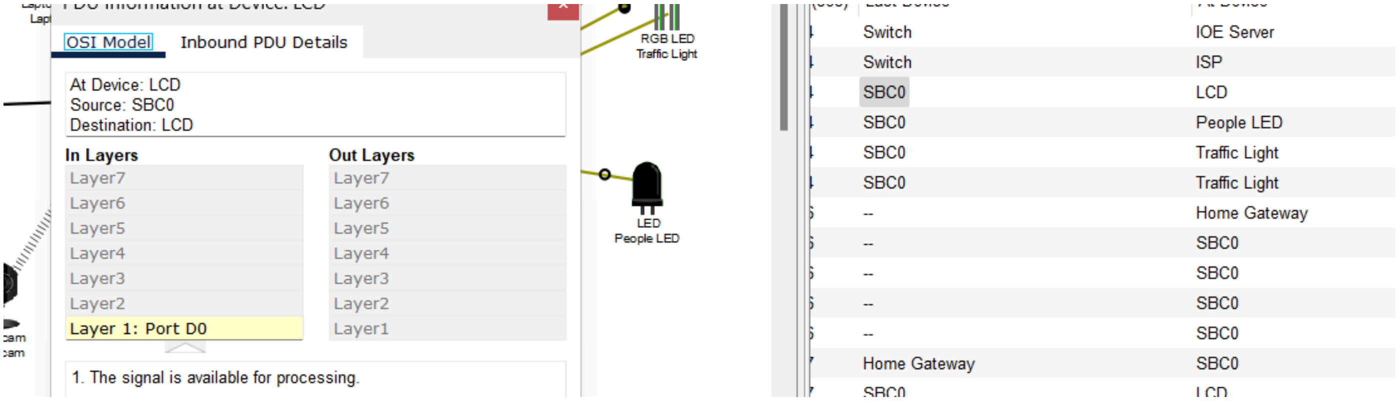
Event List

(sec)	Last Device	At Device
6	Home Gateway	SBC0
6	SBC0	LCD
6	SBC0	People LED
6	SBC0	Traffic Light
6	SBC0	Traffic Light
8	--	Switch
8	--	SBC0
8	--	SBC0
8	--	SBC0
8	--	SBC0
4	Switch	DNS Server
4	Switch	IOE Server
4	Switch	ISP
4	SBC0	LCD
4	SBC0	People LED

Captur
19.

Event List Filters - Visible Events

Packets travel from switch to DNS Server, IOE server and ISP



OSI Model | Inbound PDU Details

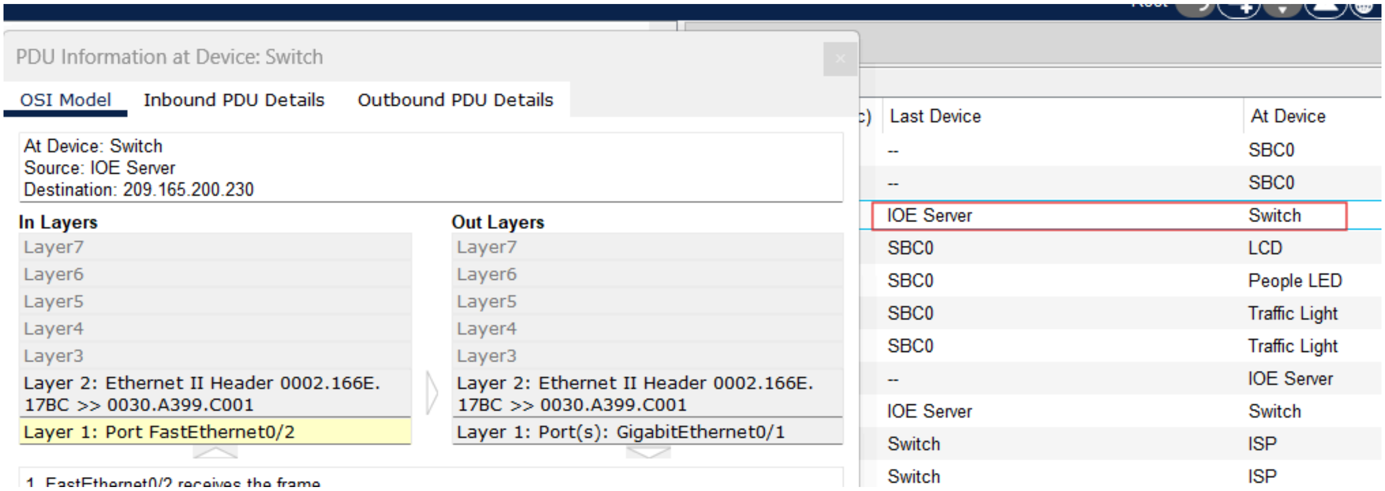
At Device: LCD
Source: SBC0
Destination: LCD

In Layers	Out Layers
Layer7	Layer7
Layer6	Layer6
Layer5	Layer5
Layer4	Layer4
Layer3	Layer3
Layer2	Layer2
Layer 1: Port D0	Layer1

1. The signal is available for processing.

Last Device	At Device
Switch	IOE Server
Switch	ISP
SBC0	LCD
SBC0	People LED
SBC0	Traffic Light
SBC0	Traffic Light
--	Home Gateway
--	SBC0
--	SBC0
--	SBC0
--	SBC0
Home Gateway	SBC0
SBC0	LCD

From SBC to devices like PeopleLED, Traffic Light and LCD.



PDU Information at Device: Switch

OSI Model | Inbound PDU Details | Outbound PDU Details

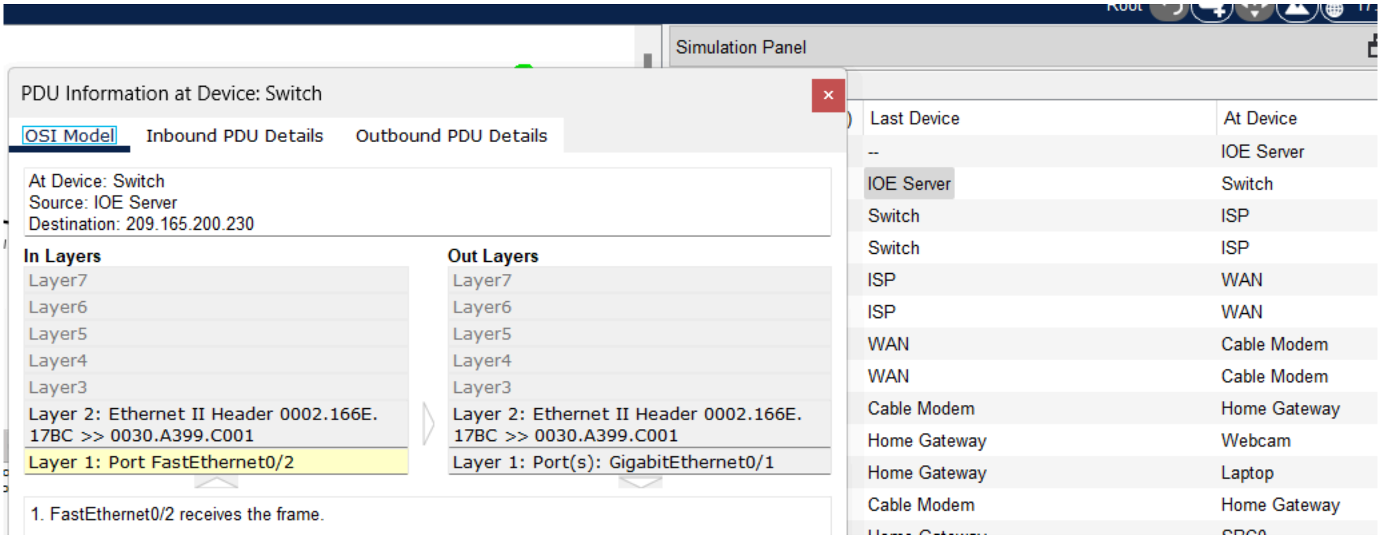
At Device: Switch
Source: IOE Server
Destination: 209.165.200.230

In Layers	Out Layers
Layer7	Layer7
Layer6	Layer6
Layer5	Layer5
Layer4	Layer4
Layer3	Layer3
Layer 2: Ethernet II Header 0002.166E.17BC >> 0030.A399.C001	Layer 2: Ethernet II Header 0002.166E.17BC >> 0030.A399.C001
Layer 1: Port FastEthernet0/2	Layer 1: Port(s): GigabitEthernet0/1

1. FastEthernet0/2 receives the frame

Last Device	At Device
--	SBC0
--	SBC0
IOE Server	Switch
SBC0	LCD
SBC0	People LED
SBC0	Traffic Light
SBC0	Traffic Light
--	IOE Server
IOE Server	Switch
Switch	ISP
Switch	ISP

Packets traverse from IOEServer to Switch and then from Switch to SBC and to PeopleLED, Traffic Light and LCD.



Simulation Panel

PDU Information at Device: Switch

OSI Model | Inbound PDU Details | Outbound PDU Details

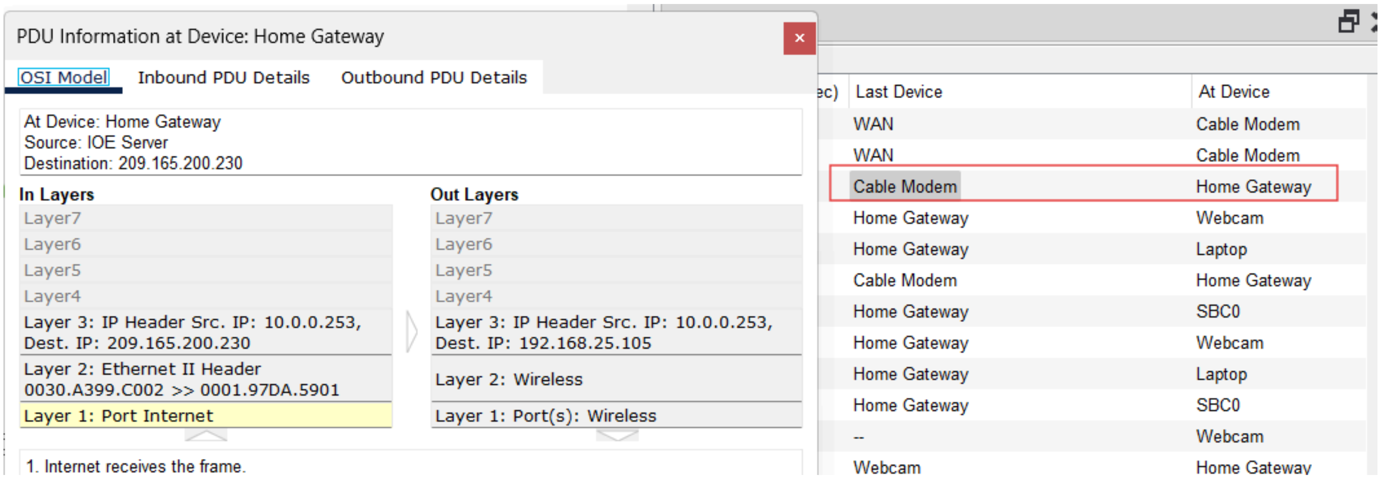
At Device: Switch
Source: IOE Server
Destination: 209.165.200.230

In Layers	Out Layers
Layer7	Layer7
Layer6	Layer6
Layer5	Layer5
Layer4	Layer4
Layer3	Layer3
Layer 2: Ethernet II Header 0002.166E.17BC >> 0030.A399.C001	Layer 2: Ethernet II Header 0002.166E.17BC >> 0030.A399.C001
Layer 1: Port FastEthernet0/2	Layer 1: Port(s): GigabitEthernet0/1

1. FastEthernet0/2 receives the frame.

Last Device	At Device
--	IOE Server
IOE Server	Switch
Switch	ISP
Switch	ISP
ISP	WAN
ISP	WAN
WAN	Cable Modem
WAN	Cable Modem
Cable Modem	Home Gateway
Home Gateway	Webcam
Home Gateway	Laptop
Cable Modem	Home Gateway
Home Gateway	SBC0

Also Packets traverse from Switch to ISP and then to Cloud and to Cable Modem



PDU Information at Device: Home Gateway

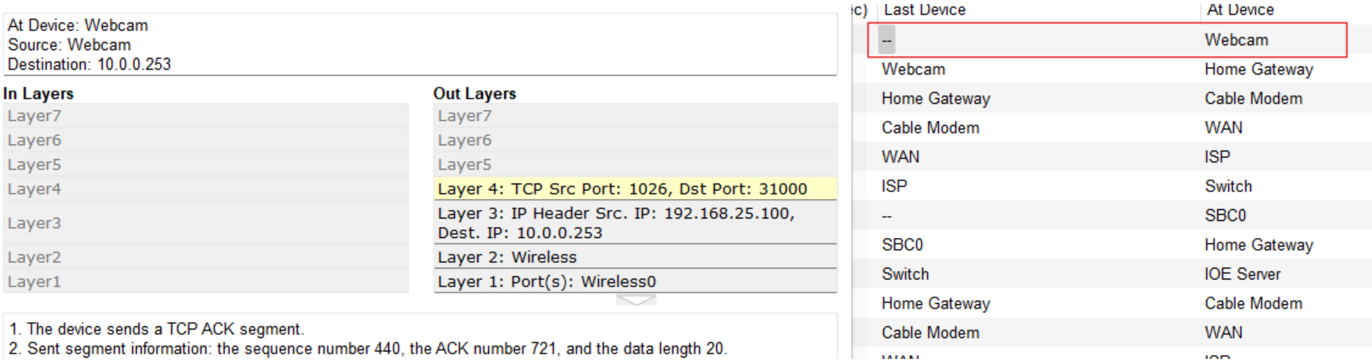
At Device: Home Gateway
Source: IOE Server
Destination: 209.165.200.230

In Layers	Out Layers
Layer7	Layer7
Layer6	Layer6
Layer5	Layer5
Layer4	Layer4
Layer 3: IP Header Src. IP: 10.0.0.253, Dest. IP: 209.165.200.230	Layer 3: IP Header Src. IP: 10.0.0.253, Dest. IP: 192.168.25.105
Layer 2: Ethernet II Header 0030.A399.C002 >> 0001.97DA.5901	Layer 2: Wireless
Layer 1: Port Internet	Layer 1: Port(s): Wireless

1. Internet receives the frame.

Last Device	At Device
WAN	Cable Modem
WAN	Cable Modem
Cable Modem	Home Gateway
Home Gateway	Webcam
Home Gateway	Laptop
Cable Modem	Home Gateway
Home Gateway	SBC0
Home Gateway	Webcam
Home Gateway	Laptop
Home Gateway	SBC0
--	Webcam
Webcam	Home Gateway

From Cable Modem the packets reach Home Gateway and then to Webcam and Laptop and SBC0.



At Device: Webcam
Source: Webcam
Destination: 10.0.0.253

In Layers	Out Layers
Layer7	Layer7
Layer6	Layer6
Layer5	Layer5
Layer4	Layer 4: TCP Src Port: 1026, Dst Port: 31000
Layer3	Layer 3: IP Header Src. IP: 192.168.25.100, Dest. IP: 10.0.0.253
Layer2	Layer 2: Wireless
Layer1	Layer 1: Port(s): Wireless0

1. The device sends a TCP ACK segment.
2. Sent segment information: the sequence number 440, the ACK number 721, and the data length 20.

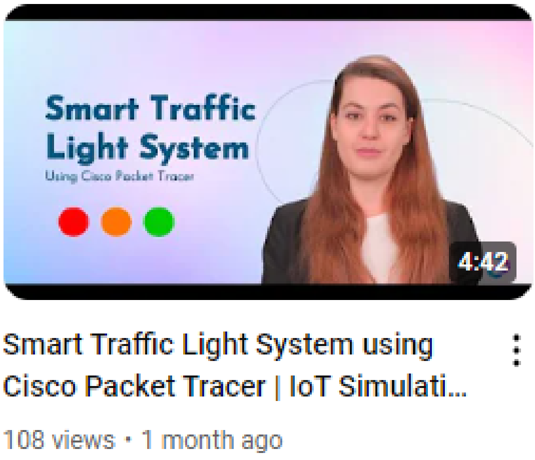
Last Device	At Device
--	Webcam
Webcam	Home Gateway
Home Gateway	Cable Modem
Cable Modem	WAN
WAN	ISP
ISP	Switch
--	SBC0
SBC0	Home Gateway
Switch	IOE Server
Home Gateway	Cable Modem
Cable Modem	WAN

The packets travel from Home Gateway to Cable Modem to Cloud and ISP and Switch.

We observe that the packets again keep traveling from IOEServer through ISP and Cloud till Home Gateway and Laptop.

To know more about the design and working of the Smart Traffic Light System in Cisco Packet Tracer, refer the video below:

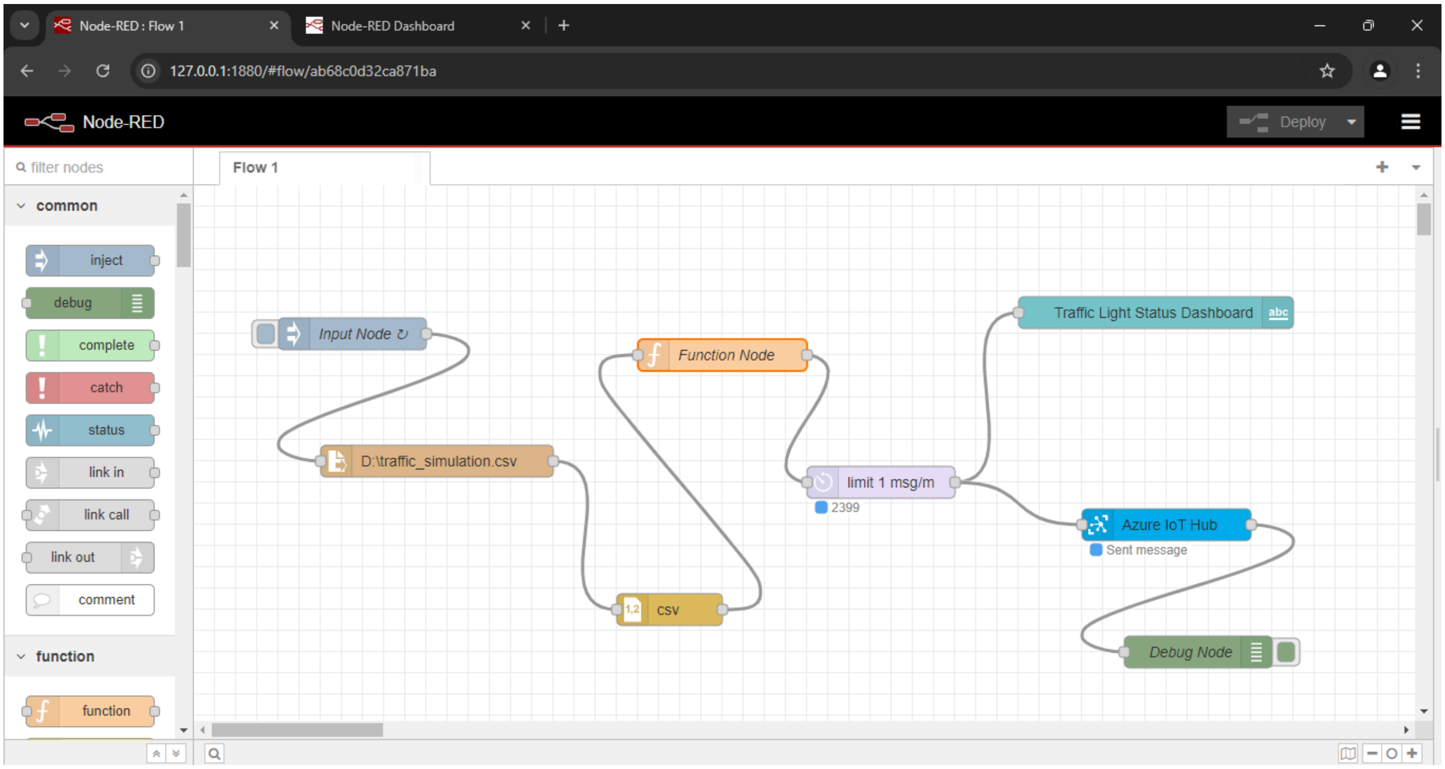
https://youtu.be/3XR6suvF3gU?feature=shared



Smart Traffic Light System using Cisco Packet Tracer | IoT Simulati...

108 views • 1 month ago

Simulation in Node-RED:



This Node-RED workflow processes traffic simulation data from a CSV file and sends it to an Azure IoT Hub, as follows:

1. **Inject Node**: Triggers the start of the flow.
2. **CSV File (D:\traffic_simulation.csv)**: Provides traffic simulation data in CSV format.
3. **CSV Node**: Converts the CSV data into a structured format.
4. **Function Node**: Likely processes or modifies the structured data for the next steps.
5. **Delay Node (limit 1 msg/m)**: Limits the flow to one message per minute to avoid overwhelming the system.
6. **Azure IoT Hub Node**: Sends the processed traffic data to the Azure IoT Hub for further use or analysis.
7. **Traffic Light Status (Text Node)**: Displays the traffic light status based on the processed data.
8. **Debug Node**: Outputs debugging information to check the workflow's performance.

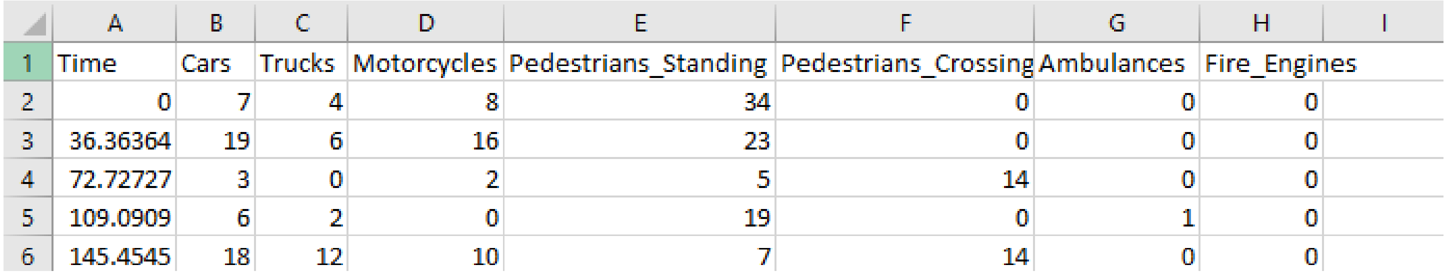
This flow simulates traffic data and sends it to a cloud service for IoT-based traffic light control.

About the dataset:

The time series dataset (traffic_simulation.csv) consists of 100 entries with the following 8 columns:

1. **Time**: Timestamps in float format.
2. **Cars**: Number of cars passing the traffic light.
3. **Trucks**: Number of trucks passing the traffic light.
4. **Motorcycles**: Number of motorcycles passing the traffic light.
5. **Pedestrians_Standing**: Number of pedestrians waiting at the crossing.
6. **Pedestrians_Crossing**: Number of pedestrians actively crossing.
7. **Ambulances**: Number of ambulances passing.
8. **Fire_Engines**: Number of fire engines passing.

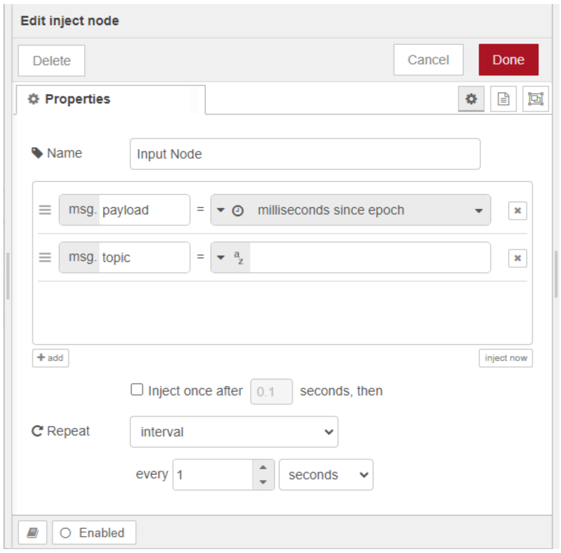
This dataset represents traffic activity and pedestrian behavior at various time intervals, also tracking emergency vehicle presence.



	A	B	C	D	E	F	G	H	I
1	Time	Cars	Trucks	Motorcycles	Pedestrians_Standing	Pedestrians_Crossing	Ambulances	Fire_Engines	
2	0	7	4	8	34	0	0	0	
3	36.36364	19	6	16	23	0	0	0	
4	72.72727	3	0	2	5	14	0	0	
5	109.0909	6	2	0	19	0	1	0	
6	145.4545	18	12	10	7	14	0	0	

- **Inject Node Configuration:**

The Inject Node serves as the flow's initiation point, either manually triggered or set to automatically activate at specific intervals. This node is likely designed to inject an initial payload, such as a timestamp or control signal, into the flow to begin data processing. It might simulate real-time traffic events or trigger the reading of traffic data from a CSV file. It plays a crucial role in setting the flow in motion, ensuring that the rest of the nodes receive the necessary data to process.



Edit inject node

Name: Input Node

msg. payload = milliseconds since epoch

msg. topic =

□ Inject once after 0.1 seconds, then

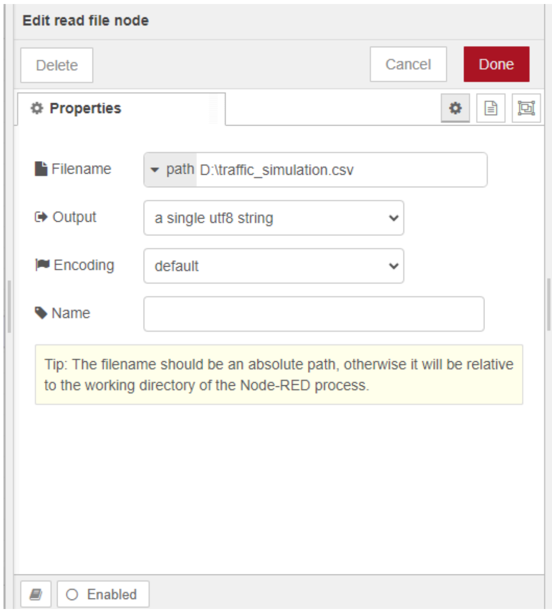
Repeat: interval

every 1 seconds

○ Enabled

- **Read File Node Configuration:**

The Read File Node in this project is responsible for reading the real-time traffic data stored in a CSV file. This CSV file contains rows of data that represent different traffic conditions, including the number of cars, trucks, motorcycles, pedestrians, ambulances, and fire engines at specific times.



Edit read file node

Filename: path D:\traffic_simulation.csv

Output: a single utf8 string

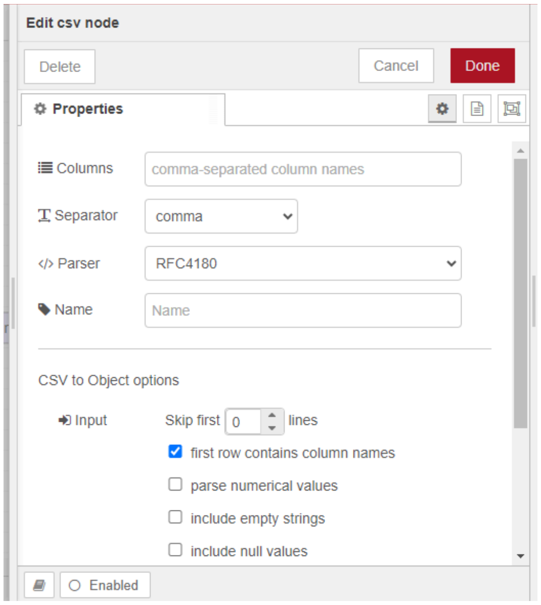
Encoding: default

Name:

Tip: The filename should be an absolute path, otherwise it will be relative to the working directory of the Node-RED process.

- **CSV Node Configuration:**

The CSV Node is responsible for reading traffic simulation data from an external file (D:/traffic_simulation.csv). It parses the CSV format and converts it into structured JSON data that can be processed further in the flow. This node is essential for simulating real-world conditions in the traffic light system, as it feeds the system with realistic vehicle movement, speed, or density data. The CSV node ensures that the system can simulate different traffic scenarios by simply reading from external datasets.



Edit csv node

Columns: comma-separated column names

Separator: comma

Parser: RFC4180

Name: Name

CSV to Object options

Input: Skip first 0 lines

☑ first row contains column names

☐ parse numerical values

☐ include empty strings

☐ include null values

- **Function Node Configuration:**

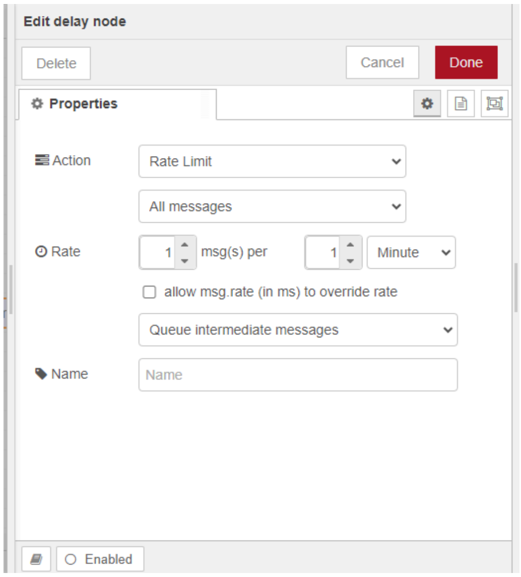
The Function Node acts as the brain of the flow, where custom JavaScript logic is executed. It processes the incoming data, transforming it based on the specific needs of the traffic light system. This node allows decision-making based on real-time inputs to control traffic flow and behavior dynamically. The code to implement this functionality is as follows:

```
let time = msg.payload.Time;
let cars = msg.payload.Cars;
let trucks = msg.payload.Trucks;
let motorcycles = msg.payload.Motorcycles;
let pedestrians_crossing = msg.payload.Pedestrians_Crossing;
let pedestrians_standing = msg.payload.Pedestrians_Standing;
let ambulances = msg.payload.Ambulances;
let fire_engines = msg.payload.Fire_Engines;
```

```
// Emergency vehicle priority
if (ambulances > 0 || fire_engines > 0) {
    msg.payload={
        "deviceId": "node-red-device",
        "key": "jTGnVCQU975D9gQY1rqtQoowub3MuWl3ClFRpaql+qc=",
        "protocol": "amqp",
        "data": { "timestamp": time, "light": "Green for Emergency Vehicles" }
    };
}
// Pedestrian standing
else if (pedestrians_standing > 20) {
     msg.payload={
        "deviceId": "node-red-device",
        "key": "jTGnVCQU975D9gQY1rqtQoowub3MuWl3ClFRpaql+qc=",
        "protocol": "amqp",
         "data": {"timestamp": time, "light": "Green for Pedestrians" }
    };
}
// Heavy traffic prioritization
else if ((cars + trucks + motorcycles) > 10) {
     msg.payload={
        "deviceId": "node-red-device",
        "key": "jTGnVCQU975D9gQY1rqtQoowub3MuWl3ClFRpaql+qc=",
        "protocol": "amqp",
         "data": {"timestamp": time, "light": "Green for Vehicles" }
    };
}

// Default case
else {
    msg.payload={
        "deviceId": "node-red-device",
        "key": "jTGnVCQU975D9gQY1rqtQoowub3MuWl3ClFRpaql+qc=",
        "protocol": "amqp",
        "data": {"timestamp": time, "light": "Yellow/Waiting" }
    };
}


return msg;
```

- **Delay Node Configuration:**

The Delay Node controls the rate at which messages flow through the system, ensuring that only one message per minute is processed. This is particularly important in IoT and real-time systems where a continuous flood of data could overwhelm downstream systems like the Azure IoT Hub or the dashboard. By limiting the message rate, the node ensures that data is processed at a manageable pace, avoiding performance bottlenecks or data overload while maintaining accurate simulation of traffic conditions.

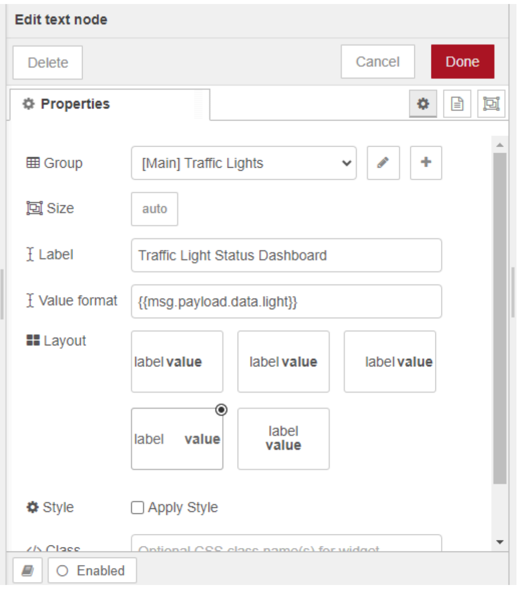


Edit delay node

Action: Rate Limit
All messages
Rate: 1 msg(s) per 1 Minute
□ allow msg.rate (in ms) to override rate
Queue intermediate messages
Name: Name

- **Text Node Configuration:**

The Text Node in Node-RED is used to display information on a dashboard, allowing users to visualize the system's status or messages in real time. The Text Node shows the output of the traffic light decision logic (e.g., "Green for Emergency Vehicles," "Green for Pedestrians," or "Yellow/Waiting") on the Node-RED dashboard. This provides a user-friendly way to monitor the current state of the traffic light system based on the CSV data.

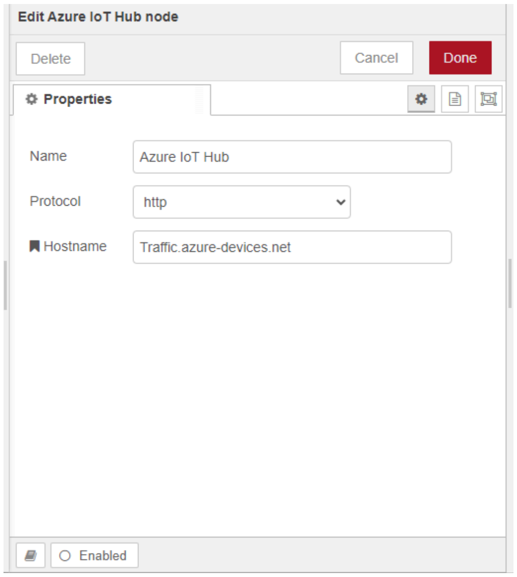


Edit text node

Group: [Main] Traffic Lights
Size: auto
Label: Traffic Light Status Dashboard
Value format: {{msg.payload.data.light}}
Layout: label value | label value | label value | label value | label value
Style: □ Apply Style
Class: Optional CSS class name(s) for widget

- **Azure Iot Hub Node Configuration:**

The Azure IoT Hub Node enables cloud integration by sending traffic data to the Azure IoT Hub platform. This node plays a vital role in expanding the system beyond the local environment by connecting it to the cloud, where data can be stored, analyzed, or used to trigger further actions. Traffic data such as vehicle count, signal times, or system alerts can be sent to the cloud for long-term analytics or for remotely controlling the system. This integration allows for a more scalable and robust smart traffic solution that benefits from the capabilities of cloud infrastructure.

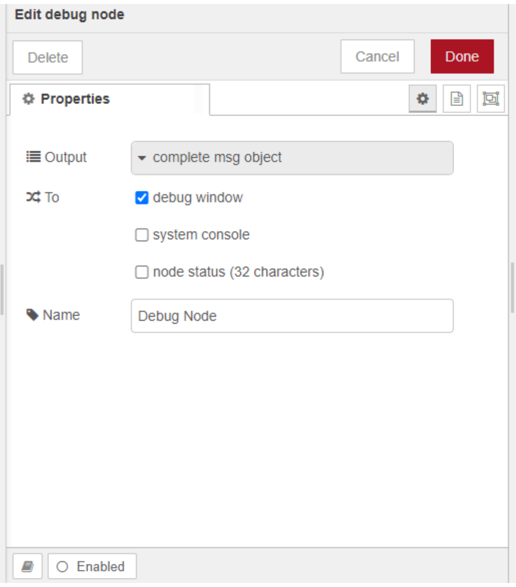


Edit Azure IoT Hub node

Name: Azure IoT Hub
Protocol: http
Hostname: Traffic.azure-devices.net

- **Debug Node Configuration:**

The Debug Node acts as a real-time monitoring tool, logging messages from various stages of the flow to Node-RED's debug pane. This node is primarily used during the development and testing phases to verify that the data is being processed correctly. It helps troubleshoot issues by displaying payloads and key data points as they pass through the flow. Tt can show the raw traffic data being processed or the final message being sent to Azure IoT Hub, allowing the developer to ensure that the flow operates as expected and identify any potential issues.



Output: complete msg object

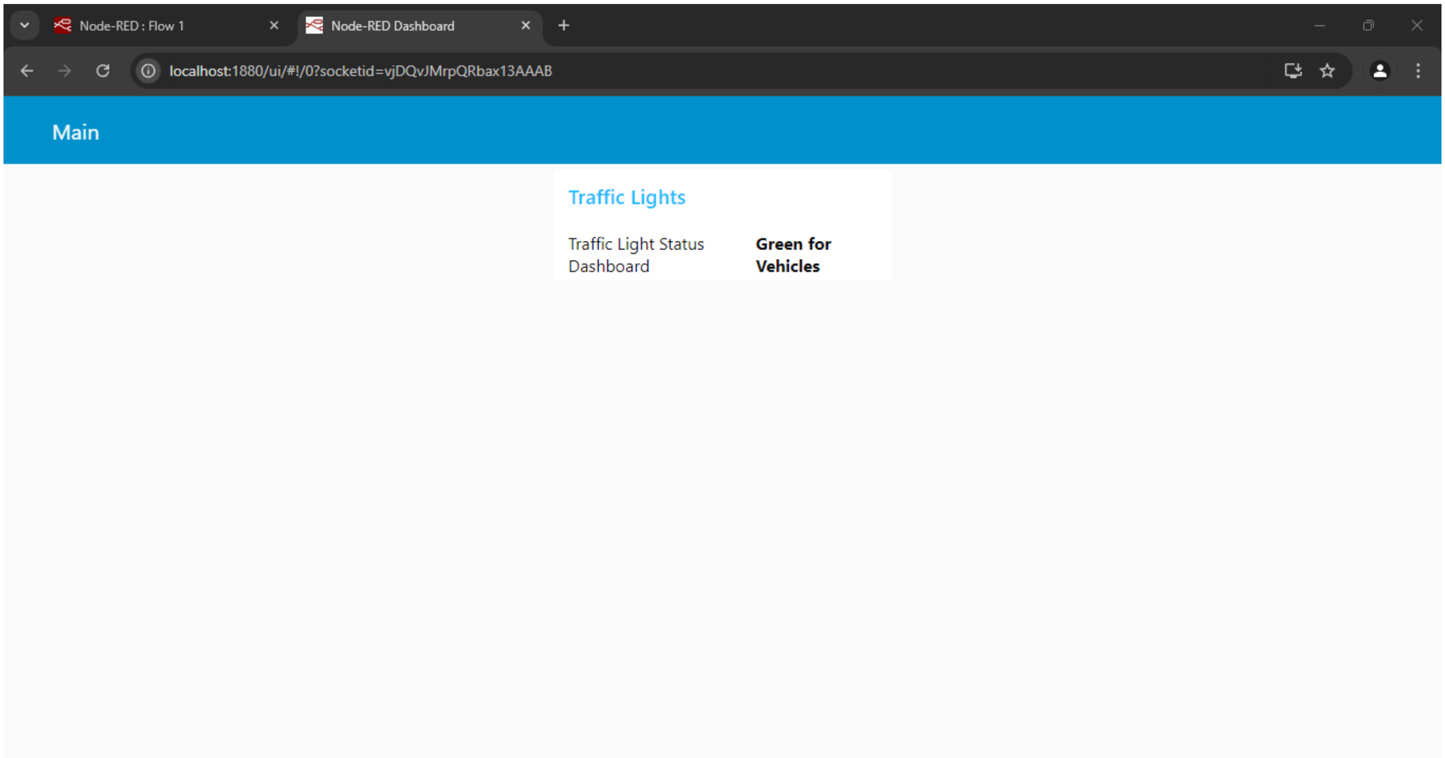
To: ☑ debug window
☐ system console
☐ node status (32 characters)

Name: Debug Node

○ Enabled

Node-RED Dashboard:

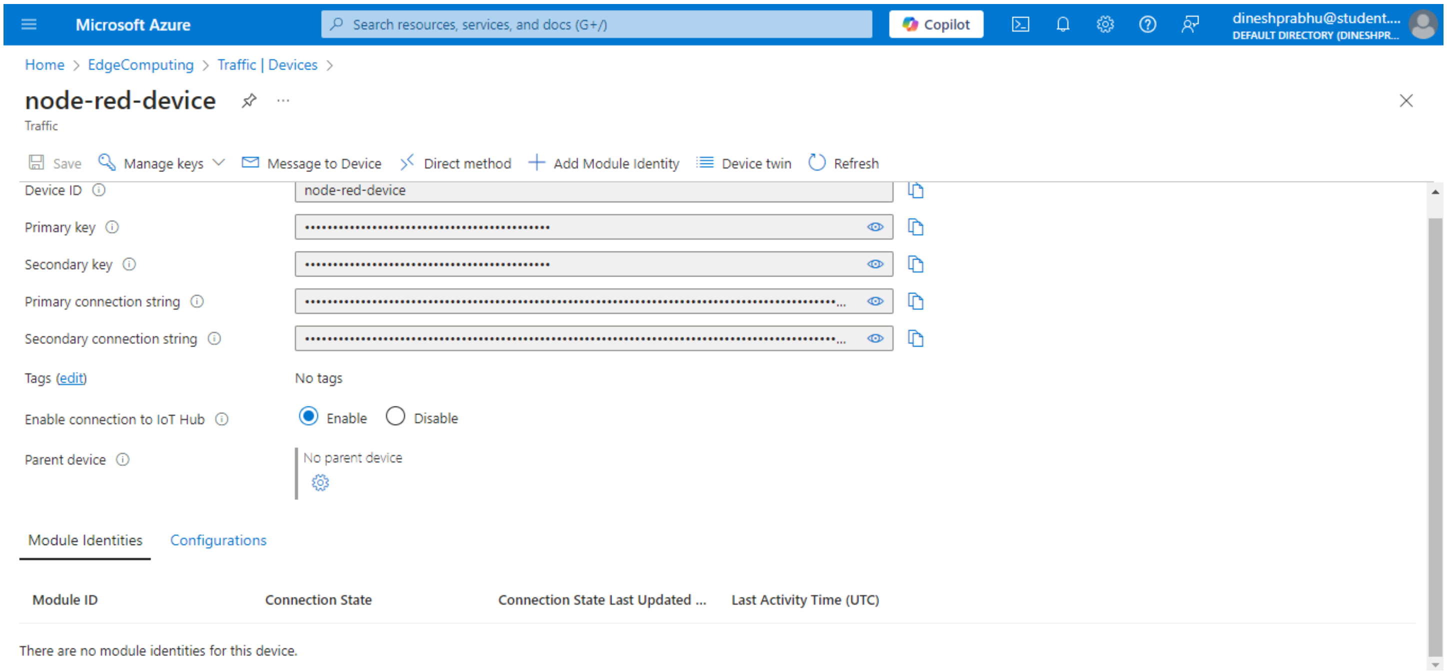
The dashboard shows the output of the traffic light decision logic (e.g., "Green for Emergency Vehicles," "Green for Pedestrians," or "Yellow/Waiting")



Traffic Light Status Dashboard: **Green for Vehicles**

Device Creation in Microsoft Azure IoT Hub:

To enable communication between the smart traffic light system in Node-RED and Azure IoT Hub, a device must be created and registered in the IoT Hub. This device acts as a representation of the physical traffic light system in the cloud, allowing data to be sent and received securely.



node-red-device

Traffic

Device ID: node-red-device

Primary key: ••

Secondary key: ••

Primary connection string: ••...

Secondary connection string: ••...

Tags (edit): No tags

Enable connection to IoT Hub: ● Enable ○ Disable

Parent device: No parent device

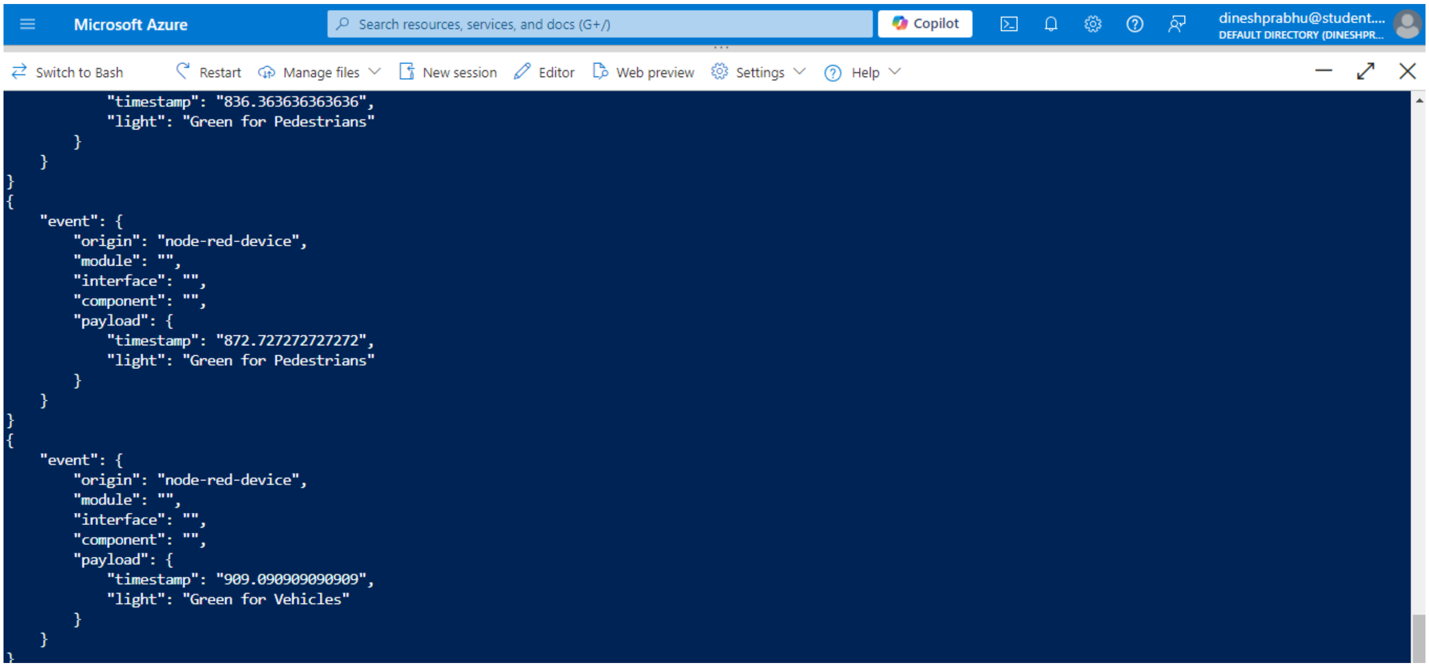
Module Identities | Configurations

Module ID	Connection State	Connection State Last Updated ...	Last Activity Time (UTC)

There are no module identities for this device.

az iot hub monitor-events -n {hub name} -d {device id} --content-type application/json

This command allows you to view real-time messages sent from your IoT device (in this case, the smart traffic light system in Node-RED) to Azure IoT Hub. It helps you see the exact data being transmitted, such as traffic light statuses.



```
            "timestamp": "836.363636363636",
            "light": "Green for Pedestrians"
        }
    }
}
{
    "event": {
        "origin": "node-red-device",
        "module": "",
        "interface": "",
        "component": "",
        "payload": {
            "timestamp": "872.727272727272",
            "light": "Green for Pedestrians"
        }
    }
}
{
    "event": {
        "origin": "node-red-device",
        "module": "",
        "interface": "",
        "component": "",
        "payload": {
            "timestamp": "909.090909090909",
            "light": "Green for Vehicles"
        }
    }
}
```

Email Notification System for Emergency Vehicles using Nodemailer and Azure Function App:

In our smart traffic light system, we have implemented a mechanism to send email notifications whenever emergency vehicles, such as ambulances and fire engines, are detected. This functionality is essential for alerting relevant parties in real time and ensuring that appropriate actions are taken, such as providing priority passage.

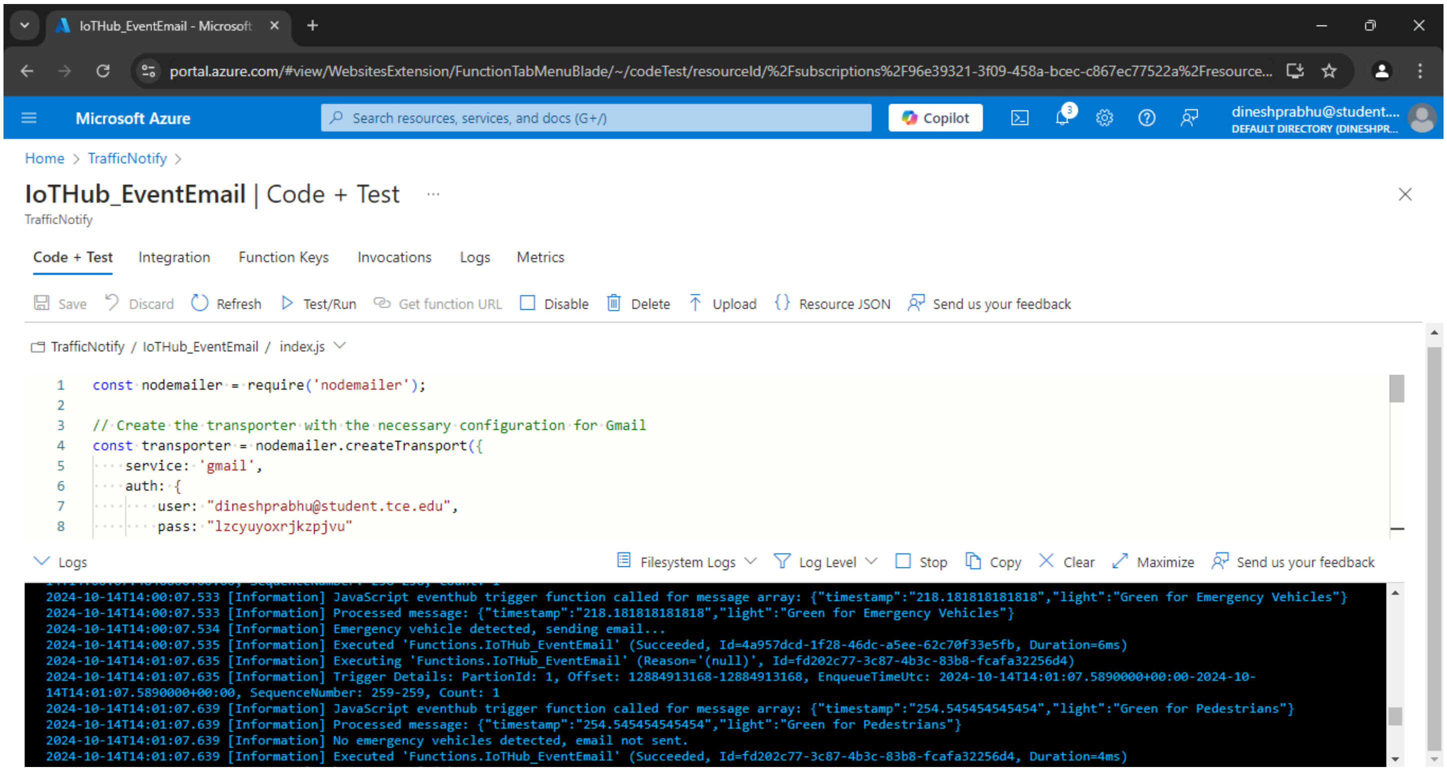
We used **Nodemailer**, a Node.js module, to handle the email-sending process. Nodemailer allows us to easily integrate email notifications into our system using a Gmail account for SMTP (Simple Mail Transfer Protocol) services. An **Azure Function App** was configured to trigger when IoT messages from the smart traffic light system are received via Azure IoT Hub.

The IoT messages are processed to detect if the traffic light is set to **"Green for Emergency Vehicles"**, indicating that an emergency vehicle is present. If this condition is met, an email notification is sent with the relevant information about the detected emergency vehicle.

- **Nodemailer**: A library used to send emails. It is configured to use a Gmail account for sending notifications.
- **Azure Function App**: A serverless compute service that runs the email notification logic when a message is received from Azure IoT Hub.
- **IoT Message Parsing**: Messages from the smart traffic light system contain information such as the traffic light status. The system checks for the string **"Emergency Vehicles"** in the message's light field, indicating the presence of an emergency vehicle.

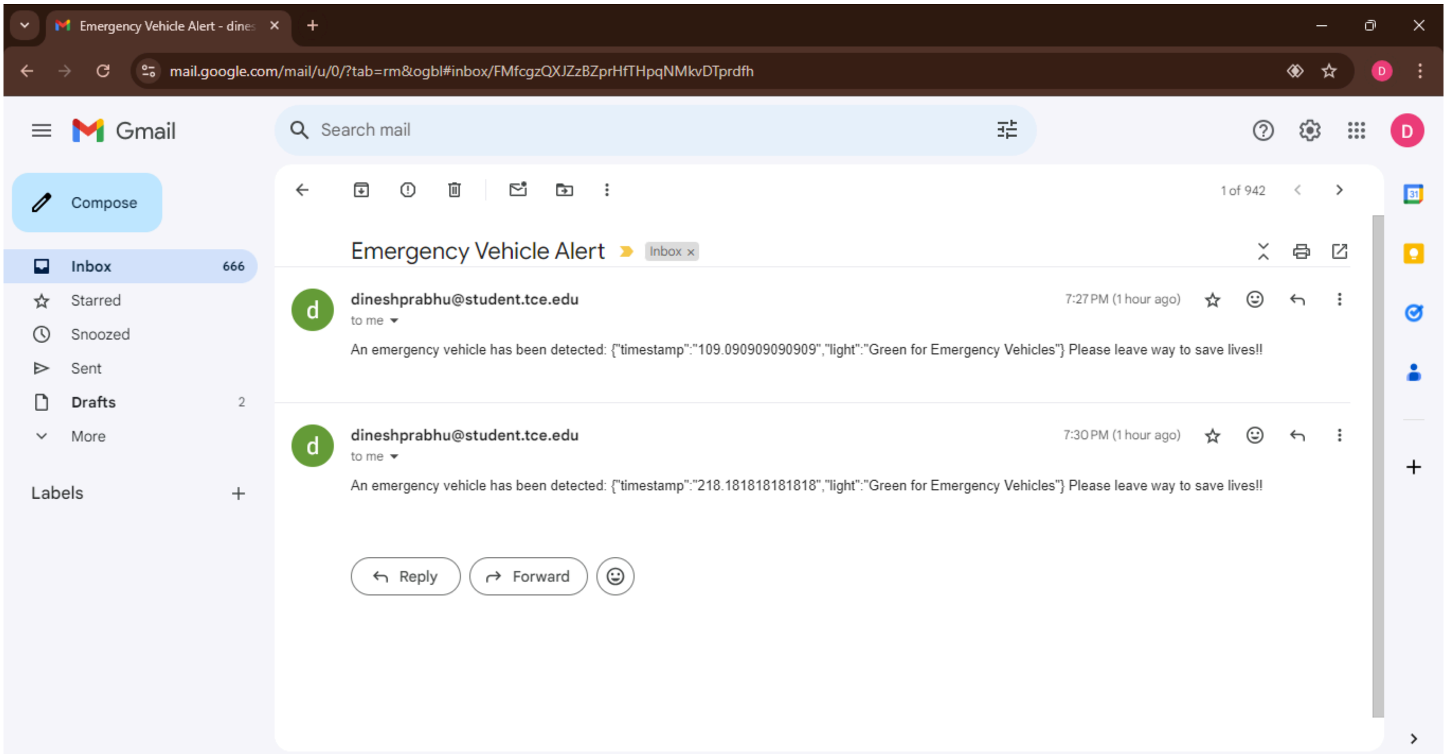
The code for achieving this functionality is as follows:

```
const nodemailer = require('nodemailer');

// Create the transporter with the necessary configuration for Gmail
const transporter = nodemailer.createTransport({
    service: 'gmail',
    auth: {
        user: "dineshprabhu@student.tce.edu",
        pass: "lzcyuyoxrjkzpjvu"
    }
});

module.exports = function (context, IoTHubMessages) {
      context.log(`JavaScript eventhub trigger function called for message array:
${IoTHubMessages}`);

    // Loop through each message received from IoT Hub
    IoTHubMessages.forEach(message => {
        // Parse the message assuming it is a JSON string
        const parsedMessage = JSON.parse(message);
        context.log(`Processed message: ${JSON.stringify(parsedMessage)}`);

        // Check if the message contains information about emergency vehicles
                if  (parsedMessage.light  &&  parsedMessage.light.includes('Emergency
Vehicles')) {
            context.log('Emergency vehicle detected, sending email...');

            // Set up email options
            const mailOptions = {
                to: 'dineshprabhu482@gmail.com',
                subject: 'Emergency Vehicle Alert',
                                 text:  `An  emergency  vehicle  has  been  detected:
${JSON.stringify(parsedMessage)} Please leave way to save lives!!`
            };

            // Send the email
            transporter.sendMail(mailOptions, (error, info) => {
                if (error) {
                    context.log('Error sending email: ', error);
                } else {
                    context.log('Email sent: ' + info.response);
                }
            });
        } else {
            context.log('No emergency vehicles detected, email not sent.');
        }
    });

    context.done();
};
```


IoTHub_EventEmail - Microsoft
portal.azure.com/#view/WebsitesExtension/FunctionTabMenuBlade/~/codeTest/resourceId/%2Fsubscriptions%2F96e39321-3f09-458a-bcec-c867ec77522a%2Fresource...
Microsoft Azure
Search resources, services, and docs (G+/)
Copilot
dineshprabhu@student....
DEFAULT DIRECTORY (DINESHPR...
Home > TrafficNotify >
IoTHub_EventEmail | Code + Test
TrafficNotify
Code + Test
Integration
Function Keys
Invocations
Logs
Metrics
Save
Discard
Refresh
Test/Run
Get function URL
Disable
Delete
Upload
Resource JSON
Send us your feedback
TrafficNotify / IoTHub_EventEmail / index.js
Logs
Filesystem Logs
Log Level
Stop
Copy
Clear
Maximize
Send us your feedback

```
const nodemailer = require('nodemailer');

// Create the transporter with the necessary configuration for Gmail
const transporter = nodemailer.createTransport({
    service: 'gmail',
    auth: {
        user: "dineshprabhu@student.tce.edu",
        pass: "lzcyuyoxrjkzpjvu"
```

```
2024-10-14T14:00:07.533 [Information] JavaScript eventhub trigger function called for message array: {"timestamp":"218.181818181818","light":"Green for Emergency Vehicles"}
2024-10-14T14:00:07.533 [Information] Processed message: {"timestamp":"218.181818181818","light":"Green for Emergency Vehicles"}
2024-10-14T14:00:07.534 [Information] Emergency vehicle detected, sending email...
2024-10-14T14:00:07.535 [Information] Executed 'Functions.IoTHub_EventEmail' (Succeeded, Id=4a957dcd-1f28-46dc-a5ee-62c70f33e5fb, Duration=6ms)
2024-10-14T14:01:07.635 [Information] Executing 'Functions.IoTHub_EventEmail' (Reason='(null)', Id=fd202c77-3c87-4b3c-83b8-fcafa32256d4)
2024-10-14T14:01:07.635 [Information] Trigger Details: PartionId: 1, Offset: 12884913168-12884913168, EnqueueTimeUtc: 2024-10-14T14:01:07.5890000+00:00-2024-10-
14T14:01:07.5890000+00:00, SequenceNumber: 259-259, Count: 1
2024-10-14T14:01:07.639 [Information] JavaScript eventhub trigger function called for message array: {"timestamp":"254.545454545454","light":"Green for Pedestrians"}
2024-10-14T14:01:07.639 [Information] Processed message: {"timestamp":"254.545454545454","light":"Green for Pedestrians"}
2024-10-14T14:01:07.639 [Information] No emergency vehicles detected, email not sent.
2024-10-14T14:01:07.639 [Information] Executed 'Functions.IoTHub_EventEmail' (Succeeded, Id=fd202c77-3c87-4b3c-83b8-fcafa32256d4, Duration=4ms)
```


Emergency Vehicle Alert - dine
mail.google.com/mail/u/0/?tab=rm&ogbl#inbox/FMfcgzQXJZzBZprHfTHpqNMkvDTprdfh
Gmail
Search mail
Compose
Inbox 666
Starred
Snoozed
Sent
Drafts 2
More
Labels
1 of 942
Inbox ×
Reply
Forward

Emergency Vehicle Alert

dineshprabhu@student.tce.edu 7:27 PM (1 hour ago)
to me

An emergency vehicle has been detected: {"timestamp":"109.090909090909","light":"Green for Emergency Vehicles"} Please leave way to save lives!!

dineshprabhu@student.tce.edu 7:30 PM (1 hour ago)
to me

An emergency vehicle has been detected: {"timestamp":"218.181818181818","light":"Green for Emergency Vehicles"} Please leave way to save lives!!

Conclusion:

In this smart traffic light system, traffic data flows through a well-orchestrated Node-RED setup, where real-time traffic conditions are simulated using a CSV file and processed by custom logic in a Function Node. Once processed, this data is visualized on a dashboard for monitoring traffic light statuses. To expand its functionality, the data is also sent to Azure IoT Hub via the Azure IoT Hub Node. In the cloud, this data is further processed by an Azure Function App, which applies decision-making algorithms. The function app monitors unusual traffic patterns, identify congestion, or detect hardware failures. Based on predefined conditions, the function app triggers an email notification alert, informing administrators or city officials in real-time. This seamless integration of Node-RED, Azure IoT Hub, and Azure Function App not only enables real-time traffic management but also ensures timely interventions, making the system highly responsive and scalable.

Team 4
Dinesh Prabhu S (21C026)
Ishvarya S (21C037)
Soubarnika M S (21C097)
Vishvak M K C (21C117)
Dhanalakshmi O T M (21C120)