RV Educational Institutions ®
RV College of Engineering ®

Autonomous Institution
Affiliated to Visvesvaraya
Technological University,
Belagavi

Approved by AICTE,
New Delhi

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

OPERATING SYSTEMS - CS235AI

REPORT

Submitted by

ADITYA SHARMA	**1RV22CY006**
ARYAN CHATURVEDI	**1RV22CY014**
MEHAR KULKARNI	**1RV22CY039**
TANISHA AGARWAL	**1RV22CY057**

Dr. Minal Moharir
HoD of Computer Science(Cyber Security) Engineering,
RV College of Engineering®, Bengaluru, India

Computer Science and Engineering
2023-2024

INTRODUCTION:

The rise of smart cities has led to the adoption of intelligent transportation systems, including smart traffic signal management. These systems promise improved traffic flow, reduced congestion, and enhanced road safety. However, with increased connectivity comes heightened cybersecurity risks. Conducting vulnerability assessments on smart traffic systems is crucial to identify and mitigate potential threats, ensuring the reliability and security of urban transportation infrastructure. This report explores the vulnerability assessment of a prototype smart traffic signal management system, outlining discovered vulnerabilities and recommending mitigation measures to strengthen its security posture.

SYSTEM ARCHITECTURE:

The components used are:
Traffic LED Signals
Raspberry Pi (central control unit
GPON router (network access)
Attacker Device - Linux / Windows Machine
Jumper Wires
Breadboard

The Raspberry Pi acts as the brain of the system, running the necessary software for traffic signal control and communication with external devices. It interfaces with the traffic signals through GPIO (General Purpose Input/Output) pins to send control commands. The system is accessible remotely via SSH (Secure Shell) for administration and monitoring purposes. SSH allows authorized users to securely access the Raspberry Pi over the network, facilitating configuration, maintenance, and troubleshooting. Additionally, the system is connected to a GPON (Gigabit Passive Optical Network) home gateway/router for network connectivity. The GPON router serves as the entry point to the system's network and provides internet access, enabling remote access and data exchange. This architecture enables traffic monitoring and control while facilitating remote management and configuration. However, it also introduces potential vulnerabilities that need to be assessed and addressed to ensure the security and reliability of the smart traffic signal management system.

METHODOLOGY:

This section details the steps taken to assess the security of the prototype smart traffic signal system:

1. Initial Access:

The assessment began by identifying the Raspberry Pi's IP address through the GPON home gateway. This IP address was used to establish a connection via SSH, the Secure Shell protocol. We used the command **ssh up32@192.168.1.33** to connect to our Raspberry Pi over our computer network. We successfully leveraged the weak SSH Vulnerability to gain unauthorized access.

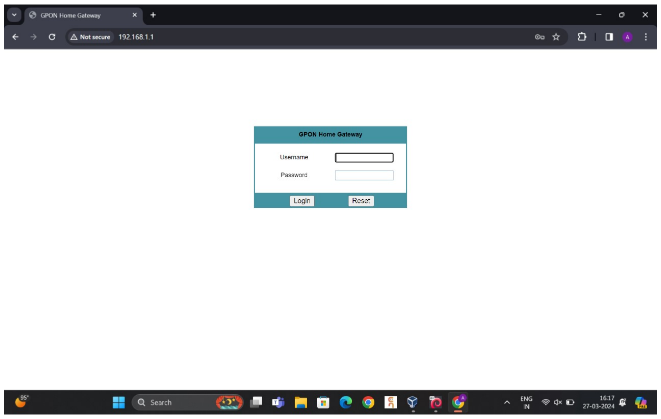
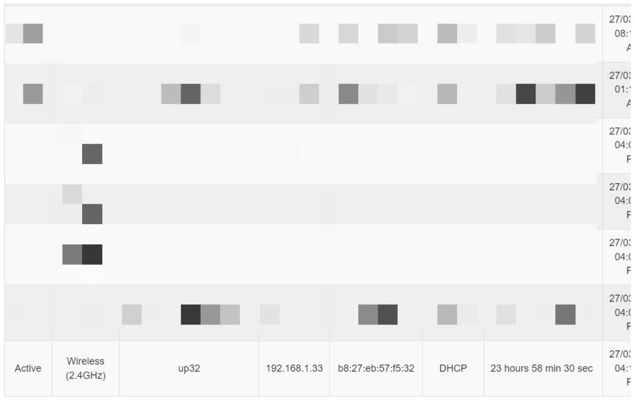
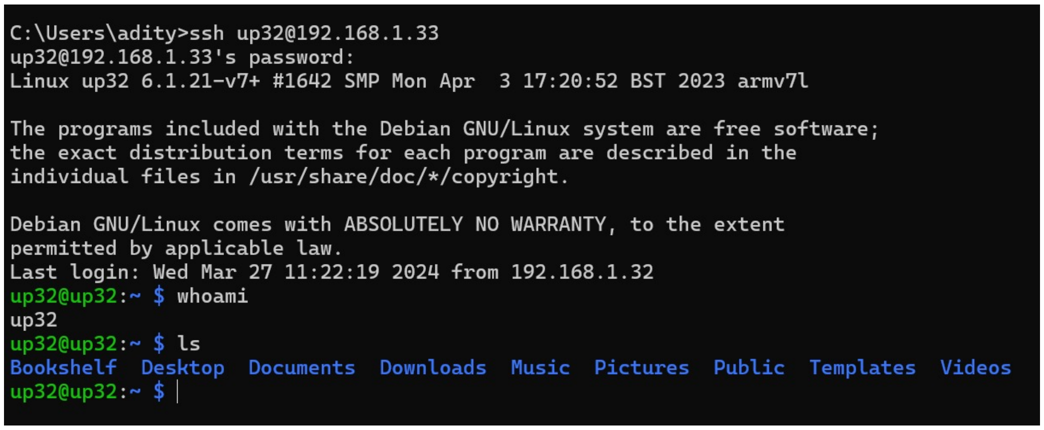


Fig1. GPON Home Gateway



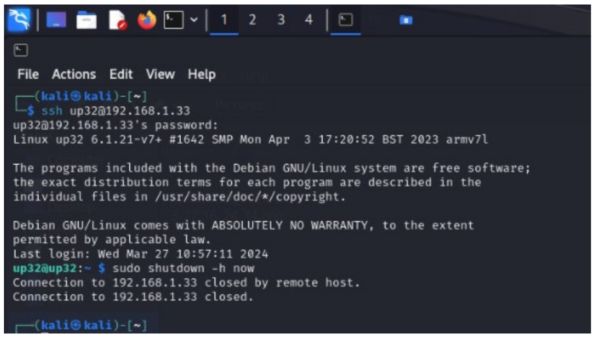
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	27/03 08: A
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	27/03 01: A
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	27/03 04: P
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	27/03 04: P
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	27/03 04: P
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	27/03 04: P
Active	Wireless (2.4GHz)	up32	192.168.1.33	b8:27:eb:57:f5:32	DHCP	23 hours 58 min 30 sec	27/03 04: P

Fig2. GPON Home Gateway>Home>List of Devices



```
C:\Users\adity>ssh up32@192.168.1.33
up32@192.168.1.33's password:
Linux up32 6.1.21-v7+ #1642 SMP Mon Apr  3 17:20:52 BST 2023 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Mar 27 11:22:19 2024 from 192.168.1.32
up32@up32:~ $ whoami
up32
up32@up32:~ $ ls
Bookshelf  Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos
up32@up32:~ $ 
```

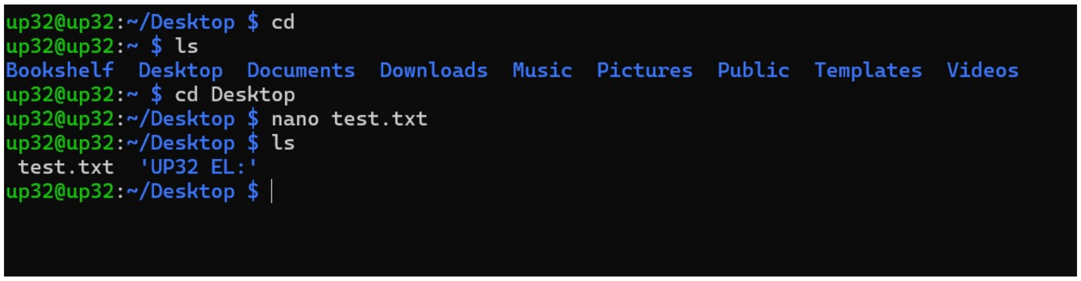
Fig3. Successful Raspberry Pi access on Terminal



```
┌──(kali㉿kali)-[~]
└─$ ssh up32@192.168.1.33
up32@192.168.1.33's password:
Linux up32 6.1.21-v7+ #1642 SMP Mon Apr  3 17:20:52 BST 2023 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Mar 27 10:57:11 2024
up32@up32:~ $ sudo shutdown -h now
Connection to 192.168.1.33 closed by remote host.
Connection to 192.168.1.33 closed.

┌──(kali㉿kali)-[~]
```

Fig4. Successful Raspberry Pi access

2. Maintaining Access and Exploring Capabilities:
To demonstrate file system manipulation capabilities, a temporary file named "test.txt" was created using the "nano" text editor. The file was then deleted using the "rm" command. This showcases the potential for unauthorized modification of critical system files.



```
up32@up32:~/Desktop $ cd
up32@up32:~ $ ls
Bookshelf  Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos
up32@up32:~ $ cd Desktop
up32@up32:~/Desktop $ nano test.txt
up32@up32:~/Desktop $ ls
 test.txt  'UP32 EL:'
up32@up32:~/Desktop $ 
```

Fig5. Addition of test.txt file from the attacking device

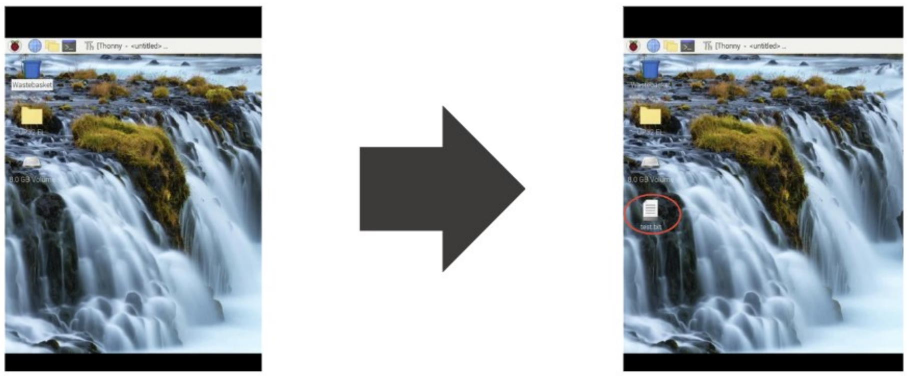
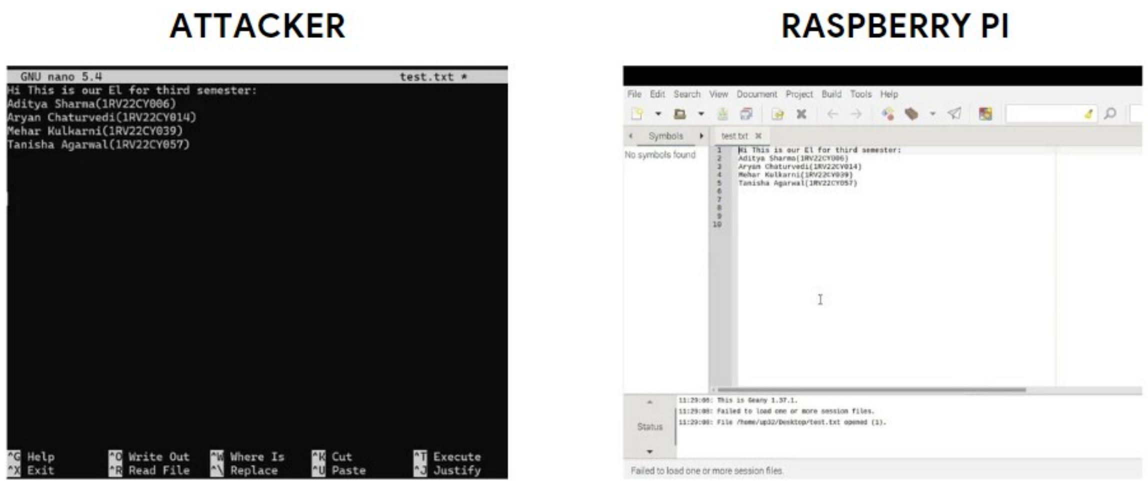


Fig6. File successfully added



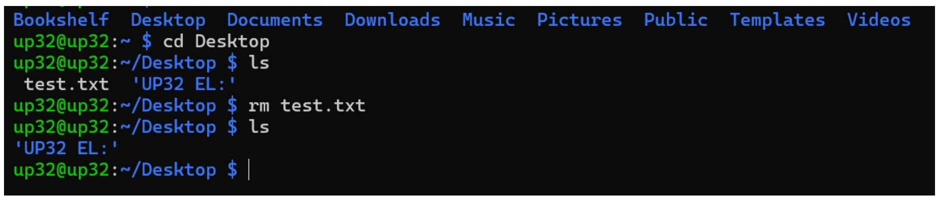
```
  GNU nano 5.4                                          test.txt *
Hi This is our El for third semester:
Aditya Sharma(1RV22CY006)
Aryan Chaturvedi(1RV22CY014)
Mehar Kulkarni(1RV22CY039)
Tanisha Agarwal(1RV22CY057)

^G Help         ^O Write Out    ^W Where Is     ^K Cut          ^T Execute
^X Exit         ^R Read File    ^\ Replace      ^U Paste        ^J Justify
```

```
test.txt
Hi This is our El for third semester:
Aditya Sharma(1RV22CY006)
Aryan Chaturvedi(1RV22CY014)
Mehar Kulkarni(1RV22CY039)
Tanisha Agarwal(1RV22CY057)
```

```
11:29:08: This is Geany 1.37.1.
11:29:08: Failed to load one or more session files.
11:29:08: File /home/up32/Desktop/test.txt opened (1).
```

Failed to load one or more session files.

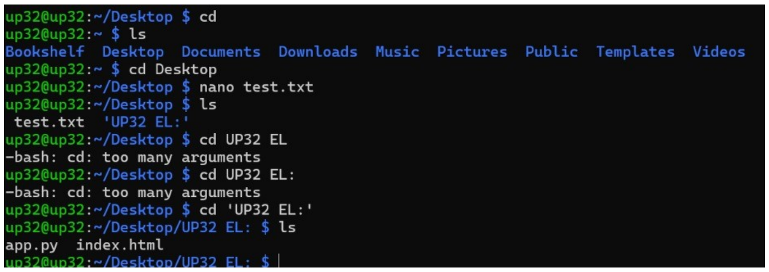
Fig7. File content



```
Bookshelf  Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos
up32@up32:~ $ cd Desktop
up32@up32:~/Desktop $ ls
 test.txt  'UP32 EL:'
up32@up32:~/Desktop $ rm test.txt
up32@up32:~/Desktop $ ls
'UP32 EL:'
up32@up32:~/Desktop $ |
```

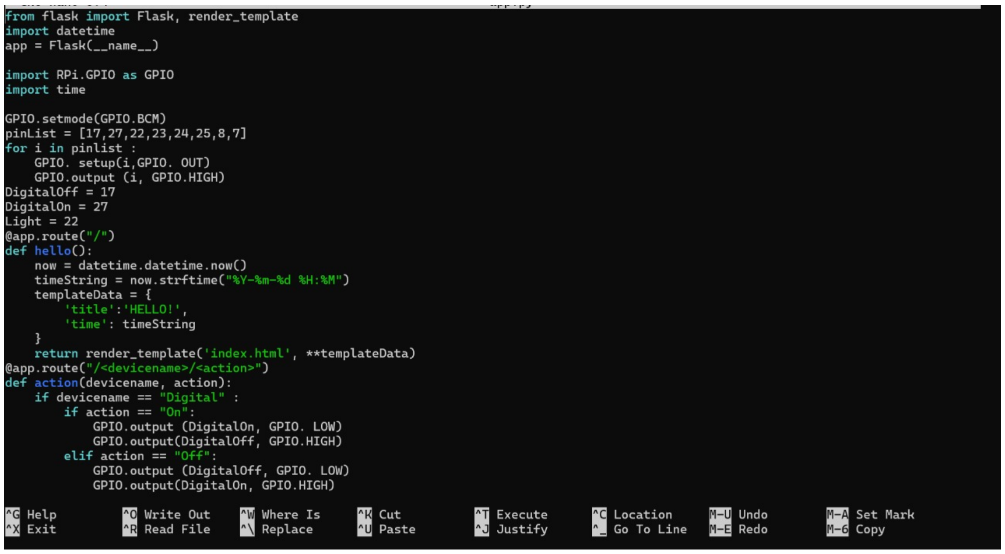
Fig8. File removed by the attacker

3. Impact Assessment - Remote Shutdown:
A significant vulnerability discovered is the ability to remotely shut down the Raspberry Pi using **sudo shutdown -h now --**. This command, with administrative privileges, allows an attacker to immediately halt the system's operation. This could lead to traffic disruptions, data loss, and potential safety hazards due to malfunctioning traffic signals.



```
up32@up32:~/Desktop $ cd
up32@up32:~ $ ls
Bookshelf  Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos
up32@up32:~ $ cd Desktop
up32@up32:~/Desktop $ nano test.txt
up32@up32:~/Desktop $ ls
 test.txt  'UP32 EL:'
up32@up32:~/Desktop $ cd UP32 EL
-bash: cd: too many arguments
up32@up32:~/Desktop $ cd UP32 EL:
-bash: cd: too many arguments
up32@up32:~/Desktop $ cd 'UP32 EL:'
up32@up32:~/Desktop/UP32 EL: $ ls
app.py  index.html
up32@up32:~/Desktop/UP32 EL: $ 
```

Fig 9. Attacker gained the access to the file where the code was stored for traffic control



```
from flask import Flask, render_template
import datetime
app = Flask(__name__)

import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
pinList = [17,27,22,23,24,25,8,7]
for i in pinlist :
    GPIO. setup(i,GPIO. OUT)
    GPIO.output (i, GPIO.HIGH)
DigitalOff = 17
DigitalOn = 27
Light = 22
@app.route("/")
def hello():
    now = datetime.datetime.now()
    timeString = now.strftime("%Y-%m-%d %H:%M")
    templateData = {
        'title':'HELLO!',
        'time': timeString
    }
    return render_template('index.html', **templateData)
@app.route("/<devicename>/<action>")
def action(devicename, action):
    if devicename == "Digital" :
        if action == "On":
            GPIO.output (DigitalOn, GPIO. LOW)
            GPIO.output(DigitalOff, GPIO.HIGH)
        elif action == "Off":
            GPIO.output (DigitalOff, GPIO. LOW)
            GPIO.output(DigitalOn, GPIO.HIGH)

^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo       M-A Set Mark
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^_ Go To Line  M-E Redo       M-6 Copy
```

Fig10. Malicious code that we injected to hack the system

<u>SYSTEM CALLS USED:</u>

Here is the list of system calls used throughout the execution.

- **SSH Connection:**
  - **Making the Connection:**
    - socket: Creates a communication channel like a virtual tunnel.
    - connect: Establishes the connection to the Raspberry Pi using the IP address.
    - gethostbyname or getaddrinfo: Translates the hostname (if used) into the numerical IP address.
  - **Authentication:**
    - open, read, write: These handle tasks like reading private keys (if used) and sending login credentials.
- **Sudo Execution:**

- **Forking and Executing:**
    - `fork`: Creates a copy of the current process, allowing sudo to run the shutdown command with elevated privileges.
    - `execve`: Replaces the current process with the shutdown program (`shutdown`).

- **Shutdown Command:**
  - **System Halt:**
    - `reboot` or `halt`: Initiates the system shutdown process, preparing the system for a safe power off.
- **Text File Editing with Nano:**
  - **File Manipulation:**
    - `open`, `read`, `write`: These are the workhorses, enabling nano to open the file, read its content, and write any changes made.
  - **Terminal Control:**
    - `ioctl`: This system call allows nano to interact with the terminal device, managing the text editor's interface elements like displaying text and handling user input.
- **File Deletion with rm:**
  - **File Removal:**
    - `unlink`: This call removes the specified file from the file system, essentially deleting it.

RESULT:

Attacker gained the access to the file where the code was stored for the traffic light management system. The vulnerability assessment of the smart traffic signal management system prototype revealed critical security weaknesses, including default credentials, unrestricted SSH access, weak Wi-Fi security, and insufficient file permissions. These findings underscore the importance of prioritizing security measures to safeguard against potential threats and ensure the integrity and reliability of the system. By implementing the recommended mitigation strategies, stakeholders can mitigate risks and enhance the security posture of the traffic signal management system, thereby enabling safe and efficient traffic operations in the future.

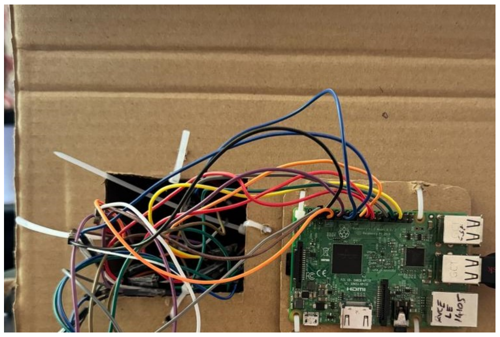


Fig11. Prototype

Fig12. System connected to Raspberry Pi shuts down

CONCLUSION:

The vulnerabilities discovered in the smart traffic signal management system prototype represent critical weaknesses that could potentially compromise the security and functionality of the entire system. These vulnerabilities, if left unaddressed, have the potential to expose the system to unauthorized access, manipulation, and disruption, posing serious risks to the safety and efficiency of traffic management operations. It is crucial to prioritize the implementation of the recommended mitigation measures to address these vulnerabilities promptly. By proactively strengthening security measures, stakeholders can fortify the system against potential threats, safeguarding its integrity and reliability. This proactive approach ensures that the smart traffic signal management system remains resilient and capable of facilitating safe and efficient traffic flow management operations, thereby enhancing overall transportation safety and efficiency.

REFERENCES:

B. E B, A. M, S. G, D. Jose and H. J. Magadum, "Intelligent Traffic Monitoring and Management System," 2023 International Conference on Computer, Electronics & Electrical Engineering & their Applications (IC2E3), Srinagar Garhwal, India, 2023, pp. 1-5, doi: 10.1109/IC2E357697.2023.10262689.

N. Jain, R. Parwanda and A. Chauhan, "Real-Time Smart Traffic Control and Simulation: An Approach for Urban Congestion Management," 2023 IEEE IAS Global Conference on Emerging Technologies (GlobConET), London, United Kingdom, 2023, pp. 1-6, doi: 10.1109/GlobConET56651.2023.10150057.

R. H. Goudar and H. N. Megha, "Next generation intelligent traffic management system and analysis for smart cities," 2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon), Bengaluru, India, 2017, pp. 999-1003, doi: 10.1109/SmartTechCon.2017.8358521.

M. M. Abo-Zahhad, "A Methodology for the Design of IoT-Based Intelligent Vehicular Management Systems in Smart Cities," 2022 10th International Japan-Africa Conference on Electronics, Communications, and Computations (JAC-ECC), Alexandria, Egypt, 2022, pp. 181-185, doi: 10.1109/JAC-ECC56395.2022.10044014.