

IOT BASED INTELLIGENT TRAFFIC MANAGEMENT SYSTEM

AIM:

To develop an intelligent IoT-based traffic management system that utilizes RFID technology to detect and enforce traffic violations such as signal jumping and blocking pedestrian lane in real-time.

HARDWARE REQUIREMENTS:

Raspberry Pi 4B, LED's, MFRC522 RFID detector, RFID tag, resistors, jumper wires

SOFTWARE REQUIREMENTS:

Raspberry Pi OS, Python IDLE v3.7

THEORY:

RFID MODULE:

The MFRC522 is a popular RFID reader/writer module that operates at a frequency of 13.56 MHz. The module utilizes an on-board antenna to communicate with RFID tags that comply with ISO/IEC 14443 Type A and Type B protocols. The module uses a contactless 2-wire serial interface for communication, which provides control signals and reads/writes data to/from the module. The MFRC522 module can read and write data to RFID tags, making it a powerful tool for various applications such as access control, payment systems, and inventory management. The module supports multiple tag types, including MIFARE Classic, MIFARE Ultralight, and MIFARE DESFire, providing compatibility with a wide range of RFID tags.

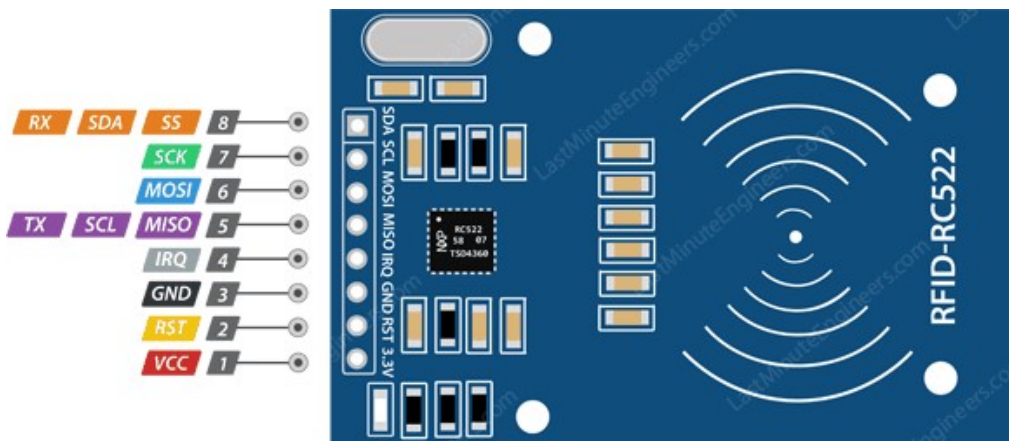


Fig 6.1: Pin Diagram of MFRC522 RFID Reader

The MFRC522 RFID reader/writer module features eight pins that are used for communication with a microcontroller or single-board computer. The pins include VCC, GND, Reset, SPI (Serial Peripheral Interface) pins (MISO, MOSI, and SCK), and Slave Select (SS) pin. The VCC pin is used to provide power to the module, typically 3.3V. The GND pin is connected to ground. The Reset pin is used to reset the module. The SPI pins (MISO, MOSI, and SCK) are used to communicate with the module using the SPI protocol. Finally, the SS pin is used to select the module for communication when multiple SPI devices are connected to the same bus. Understanding the pinout of the MFRC522 module is essential for properly connecting it to a microcontroller or single-board computer and interfacing it with RFID tags.

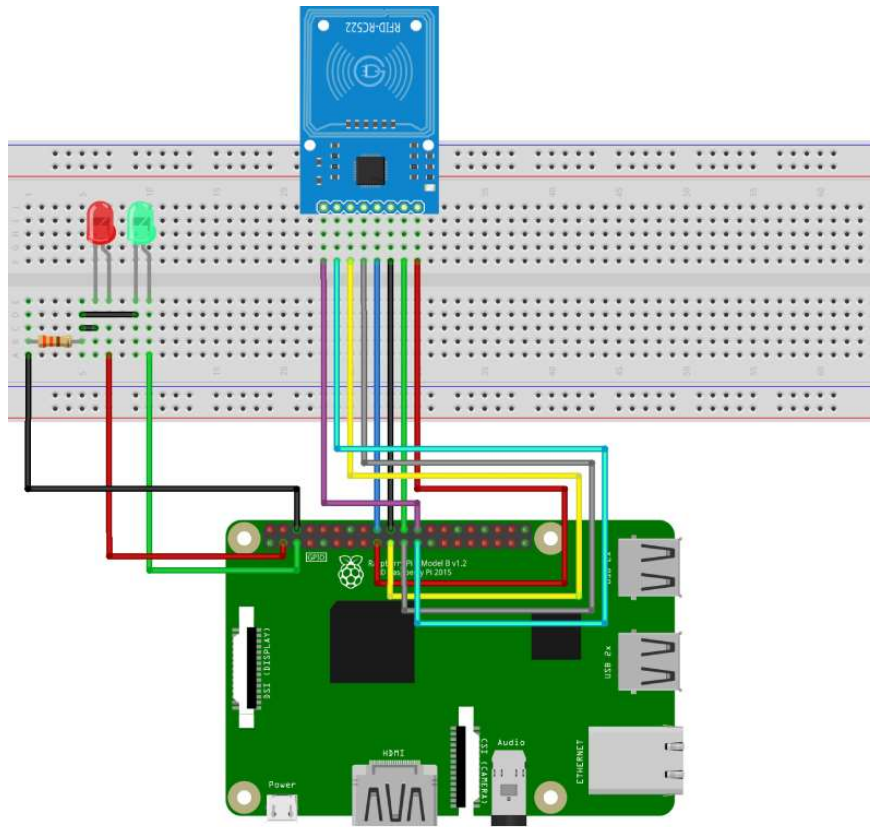


Fig 6.2: Connecting RFID and LEDs with Raspberry Pi

To connect an MFRC522 RFID reader to a Raspberry Pi 4, the module's pins must be connected to the appropriate GPIO pins on the Pi. The MFRC522 module requires 3.3V power, so the VCC pin of the module should be connected to a 3.3V pin on the Pi. The module's ground (GND) pin should be connected to a ground pin on the Pi. The module's Serial Peripheral Interface (SPI) pins (MISO, MOSI, and SCK) should be connected to the corresponding GPIO pins on the Pi. Finally, the module's Slave Select (SS) pin should be connected to a GPIO pin on the Pi that will be used as the Chip Select (CS) pin.

RFID tags, on the other hand, are small devices that store data and communicate with RFID readers. These tags typically consist of a microchip and an antenna that are embedded in a small plastic or paper-like card. When the antenna of the RFID reader is brought into close proximity to the RFID tag, the reader emits an electromagnetic field that powers the tag and allows it to transmit its data to the reader. MIFARE is a widely used RFID technology that operates at 13.56 MHz and is compatible with the MFRC522 module. MIFARE tags use a 32-bit identifier called a Unique Identifier (UID) that is assigned to each tag during manufacturing.



Fig 6.3: Example of RFID Tags

HC-SR04 ULTRASONIC SENSOR MODULE:

The HC-SR04 ultrasonic sensor is a commonly used sensor for measuring distances in projects involving robotics, automation, and object detection. It uses ultrasonic sound waves to measure distance by sending out a signal and measuring the time it takes for the signal to bounce back after hitting an object. The sensor operates in a range of 2cm to 4m, with a precision of up to 3mm. It has four pins, including two for power and ground and two for trigger and echo. When triggered, the sensor sends out an ultrasonic wave that bounces back and is received by the echo pin. The time it takes for the signal to travel to the object and back is then used to calculate the distance. The HC-SR04 is a cost-effective and reliable sensor for distance measurement applications.

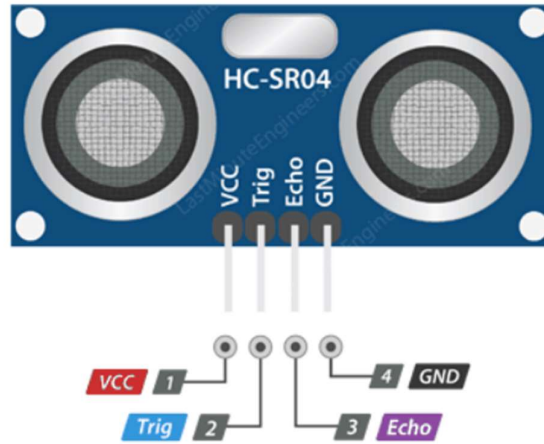


Fig 6.4: HC-SR04 Ultrasonic Sensor Pinout Details

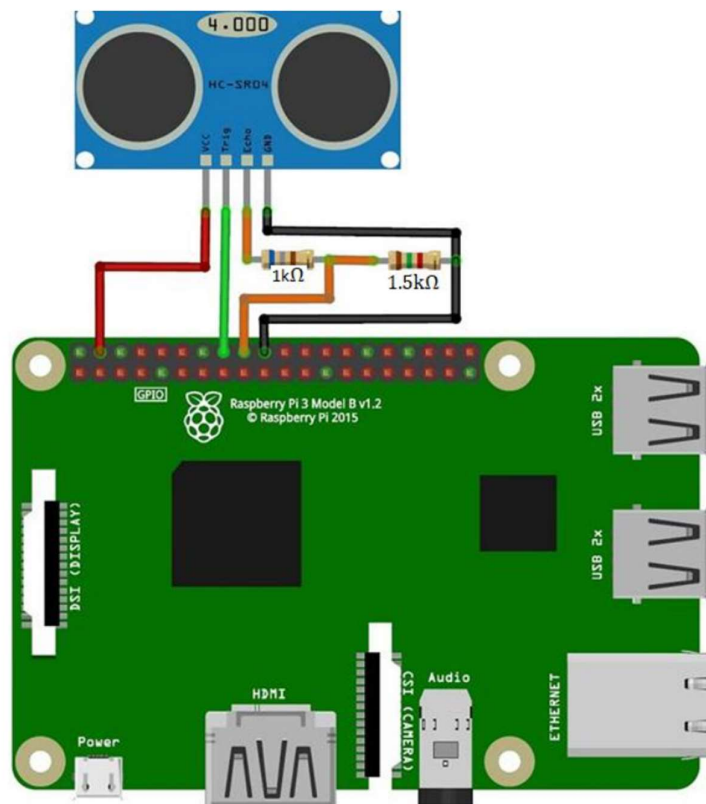


Fig 6.5: Connecting HC-SR04 Ultrasonic Sensor with Raspberry Pi

SCHEMATIC DIAGRAM:

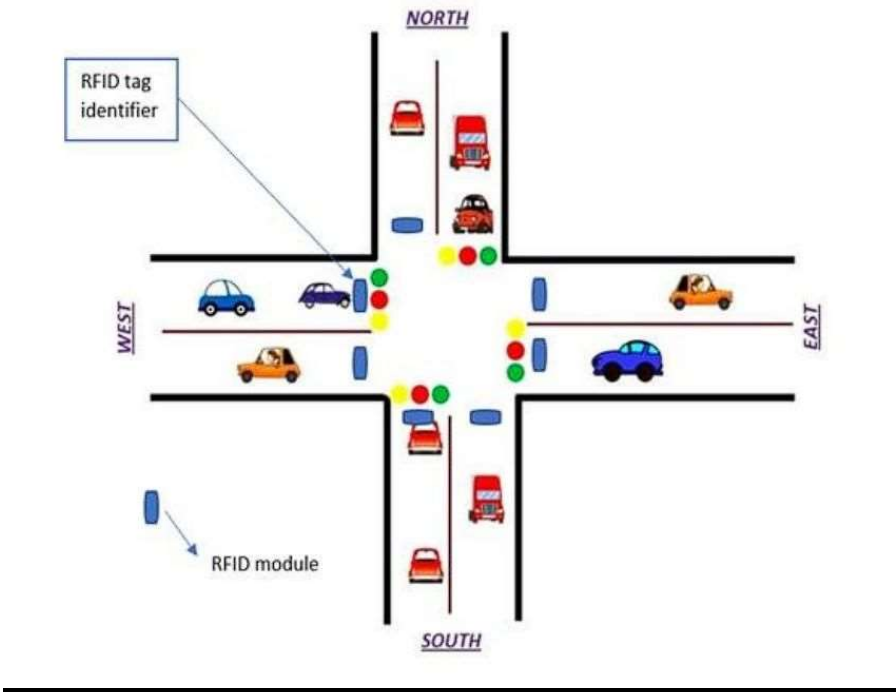


Fig 6.6: schematic of working model

FLOWCHART:

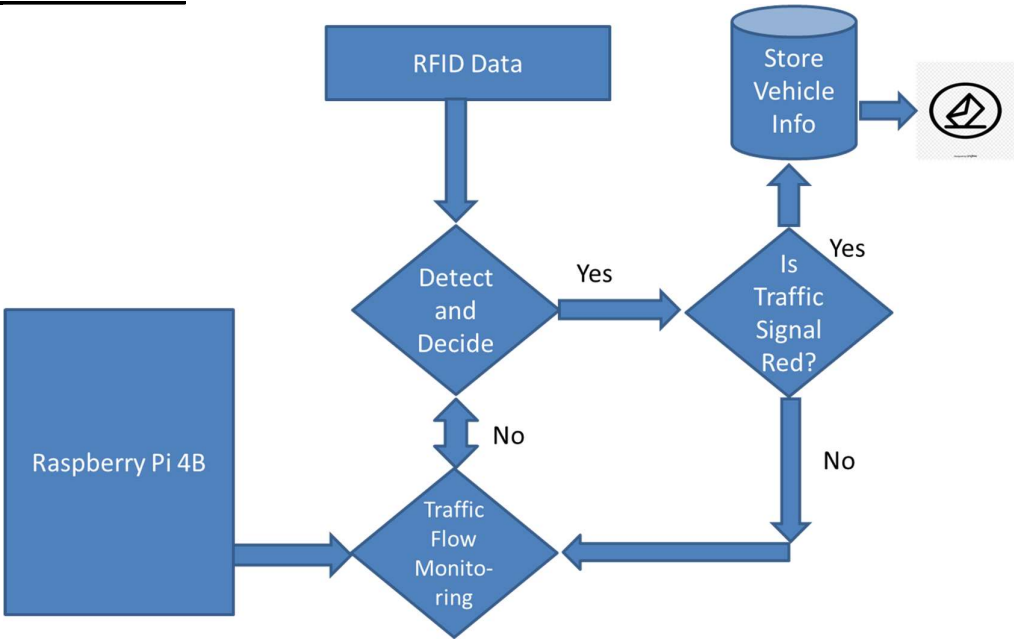


Fig 6.7: Flowchart of working model

CODE:

```
import RPi.GPIO as GPIO
import time
from mfrc522 import SimpleMFRC522
import os
import requests
import pandas as pd
from twilio.rest import Client

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)

# dataframe for storing uid values
rfid_data=pd.DataFrame(columns=['Vehicle Number','Offence'])

Trig = 40
Echo = 38
buzzer_pin = 29

GPIO.setmode(GPIO.BOARD)
GPIO.setup(Trig, GPIO.OUT)
GPIO.setup(Echo, GPIO.IN)
GPIO.setup(buzzer_pin, GPIO.OUT)

# Function to activate the buzzer
def activate_buzzer():
    GPIO.output(buzzer_pin, GPIO.HIGH)

# Function to deactivate the buzzer
def deactivate_buzzer():
    GPIO.output(buzzer_pin, GPIO.LOW)

try:
    while True:

        # RFID reading loop
        reader = SimpleMFRC522()
        id, text = reader.read()
        print(id, ":", text)

        # Get distance from sensor
        print("Initialising sensor")
        GPIO.output(Trig, False)
        time.sleep(2E-6)
        GPIO.output(Trig, True)
```

```

time.sleep(10E-6)
GPIO.output(Trig, False)
while GPIO.input(Echo) == 0:
    pulse_start = time.time()
while GPIO.input(Echo) == 1:
    pulse_end = time.time()
pulse_duration = pulse_end - pulse_start
distance = pulse_duration * 17150
distance = round(distance, 2)
print("Distance : ",distance)
violation=""
#Violation : Pedestrian lane cutting
if distance > 17 and distance < 24:
    print("Violation : Pedestrian lane cutting")
    violation="Pedestrian lane cutting"
    print("Vehicle number:",text)
    # Set environment variables for your credentials
    account_sid = "AC5b4ff006ca933bcd1bdfac9a7d31338b"
    auth_token = "7d95e1ed0ab0bdc5fd41290255fd098b"
    client = Client(account_sid, auth_token)
    message = client.messages.create(
        body="You have been fined Rs.500 for pedestrian lane cutting by
vehicle number: {} ".format(text),
        from_="+14406353615",
        to="+919682643053")

    print(message.sid)
    # Activate the buzzer for 1 second
    activate_buzzer()
    time.sleep(2)

    # Deactivate the buzzer for 1 second
    deactivate_buzzer()
    time.sleep(10)

# Violation : Red signal cutting and running
elif distance < 17:
    print("Violation : Red signal cutting and running")
    violation="Red signal cutting and running"
    print("Vehicle number:",text)
    # Set environment variables for your credentials
    account_sid = "AC5b4ff006ca933bcd1bdfac9a7d31338b"
    auth_token = "7d95e1ed0ab0bdc5fd41290255fd098b"
    client = Client(account_sid, auth_token)
    message = client.messages.create(
        body="You have been fined Rs.2000 for red signal cutting and
running",
        from_="+14406353615",
        to="+919682643053"
    )

```

```

        print(message.sid)
    elif distance > 24:
        violation = "none"

    x = {'Vehicle Number':text,'Offence':violation}
    rfid_data = pd.concat([rfid_data,pd.DataFrame(x, index=[-
1])),ignore_index=True)
    rfid_data.to_csv('rfid_data.csv',index=False)
    csv_file=pd.read_csv("rfid_data.csv")
    html_output = csv_file.to_html(index=False)
    html_file = r"/home/aditikannan/Desktop/mini-project/rfid_data.html"
    with open(html_file,'w') as f:
        f.write(html_output)
        print("success")
        time.sleep(1)

except KeyboardInterrupt:
    GPIO.cleanup()

```

CONNECTION DIAGRAM:

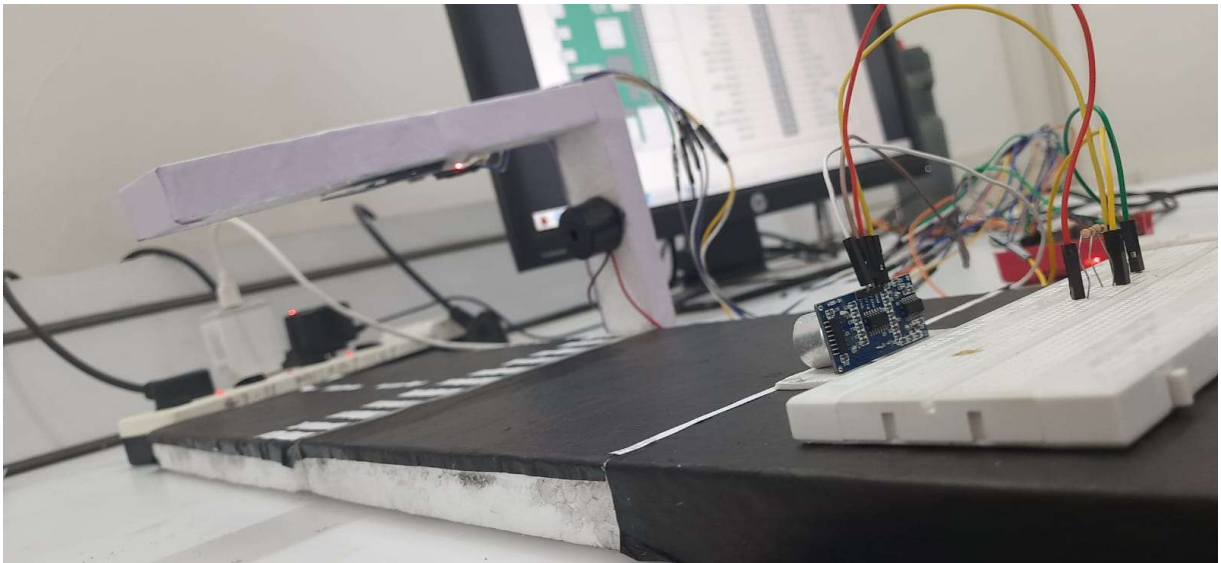


Fig 6.8: Side View of working model



Fig 6.9: Top View of working model

OUTPUT:

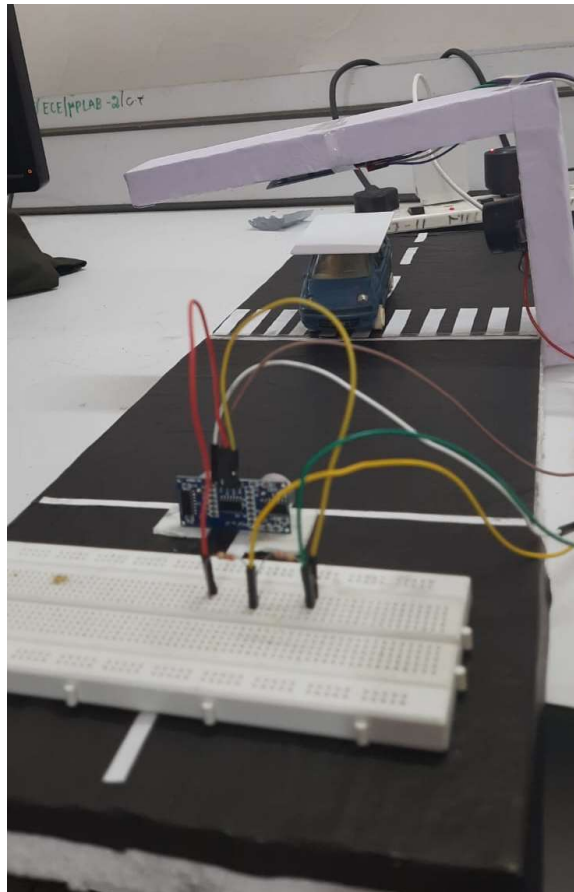


Fig 6.10: Pedestrian lane obstruction

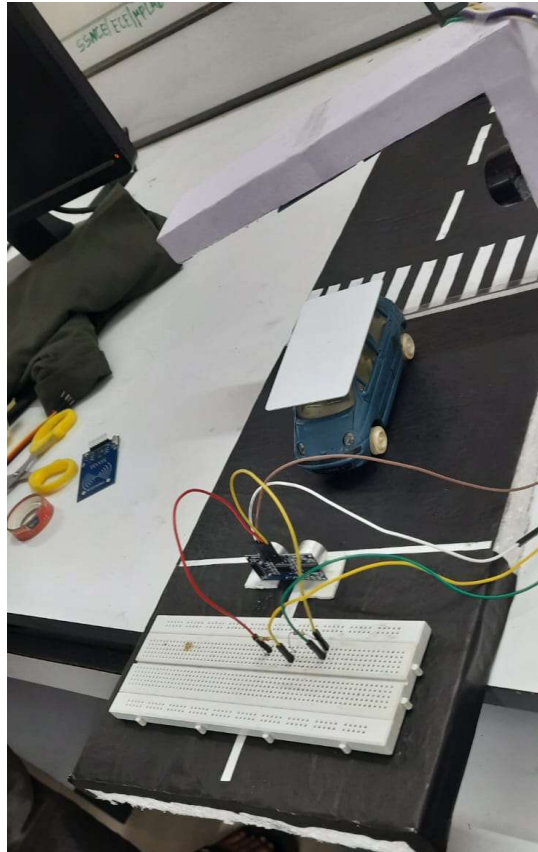


Fig 6.11: Signal jumping

11:44 AM

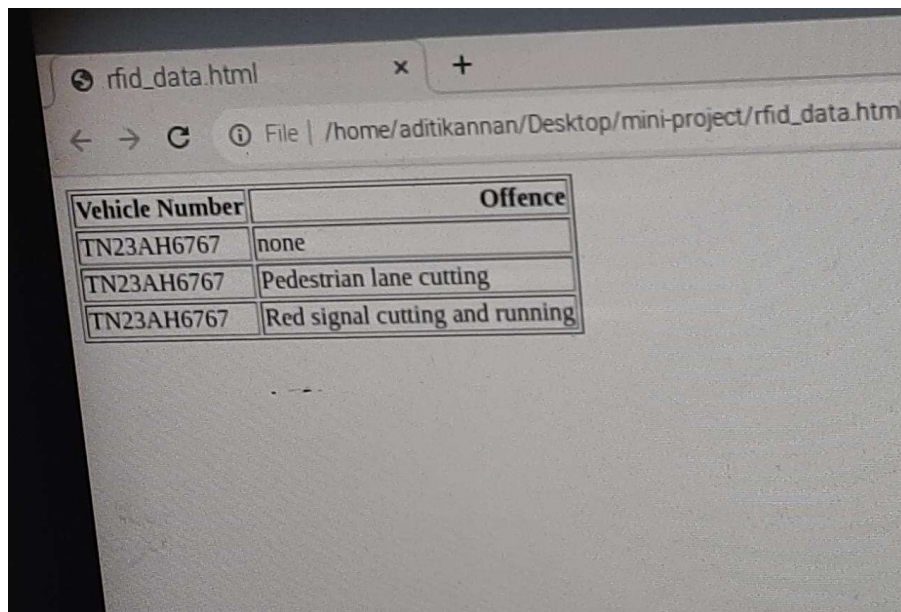
Sent from your Twilio trial account - You have been fined Rs.500 for pedestrian lane cutting

Sent from your Twilio trial account - You have been fined Rs.2000 for red signal cutting and running

Fig 6.12: Message sent to violator

```
Support
aditikannan@raspberrypi: ~/Desktop/mini-project
File Edit Tabs Help
Violation : Pedestrian lane cutting
Vehicle number: TN23AH6767
SMbdaac98fb8597cfd7e265c9f72f43ac7
success
^C(mini) aditikannan@raspberrypi:~/Desktop/mini-project
497430179498 : TN23AH6767
Initialising sensor
Distance : 27.81
success
497430179498 : TN23AH6767
Initialising sensor
Distance : 17.15
Violation : Pedestrian lane cutting
Vehicle number: TN23AH6767
SM02d4b86c88335136ab6b54f897290829
success
497430179498 : TN23AH6767
Initialising sensor
Distance : 7.77
Violation : Red signal cutting and running
Vehicle number: TN23AH6767
SMad1c12c9e943a338b71d56f599b997be
success
^C(mini) aditikannan@raspberrypi:~/Desktop/mini-project
```

Fig 6.13: Result displayed on system



The screenshot shows a web browser window with the address bar displaying `/home/aditikannan/Desktop/mini-project/rfid_data.html`. The main content area contains a table with two columns: 'Vehicle Number' and 'Offence'.

Vehicle Number	Offence
TN23AH6767	none
TN23AH6767	Pedestrian lane cutting
TN23AH6767	Red signal cutting and running

Fig 6.14: Database update

INFERENCE :

The intelligent IoT-based traffic management system using RFID technology and Raspberry Pi has the potential to significantly improve traffic safety by accurately detecting traffic violations in real-time. The system's ability to capture and store violator information in a database or cloud platform and issue SMS alerts to violators helps enforce traffic rules and prevent further violations. The project demonstrates the capability of IoT technologies to address real-world problems and provides a foundation for further research and development in the field of traffic management and safety. With further improvements, this system can be implemented in cities and urban areas to reduce traffic violations and promote safer roads.

RESULT:

Thus, an intelligent IoT-based traffic management system that utilizes RFID technology to detect and enforce traffic violations such as signal jumping and blocking pedestrian lane in real-time was developed.