Traffic management system_Phase3

Phase 3: Development part 1

To build a complete traffic management system using RFID, Arduino Uno, and Arduino Mega is a complex task involving different components and hardware, so we are going with a simplified Python script for Arduino Uno and Mega collect RFID data and map traffic congestion Statistics. To fully implement this functionality, we need the necessary hardware components and libraries for the RFID connection.

Here is a basic Python script for Arduino Uno and Arduino Mega. This script reads the RFID data from the RFID sensor and plots the traffic jam data. To collect real RFID data, we need an RFID sensor and a library to communicate with.

The Arduino Uno python script is:

```
import serial

# Define the Arduino Uno's serial port.

uno_port = "COM3"  # Update this with your Uno's serial port.

try:

    uno_serial = serial.Serial(uno_port, 9600)

except serial.SerialException:

    print("Failed to open the Uno's serial port.")

    exit()

while True:

    uno_data = uno_serial.readline().strip().decode('utf-8')

    print("RFID data from Arduino Uno:", uno_data)

    # Simulate traffic congestion data (0-100, where 100 means heavy congestion).

    traffic_congestion = 50  # Change this value as needed.

    print("Traffic Congestion Level:", traffic_congestion)

    # Upload the RFID data and traffic congestion to a website or database.

    # You will need to implement the upload functionality.

# Close the serial connection when done.

uno_serial.close()
```

The Arduino Mega python script is:

```
import serial

# Define the Arduino Mega's serial port.
```

```
mega_port = "COM4"  # Update this with your Mega's serial port.
try:
    mega_serial = serial.Serial(mega_port, 9600)
except serial.SerialException:
    print("Failed to open the Mega's serial port.")
    exit()
while True:
    mega_data = mega_serial.readline().strip().decode('utf-8')
    print("RFID data from Arduino Mega:", mega_data)
    # Simulate traffic congestion data (0-100, where 100 means heavy congestion).
    traffic_congestion = 60  # Change this value as needed.
    print("Traffic Congestion Level:", traffic_congestion)
    # Upload the RFID data and traffic congestion to a website or database.
    # You will need to implement the upload functionality.
# Close the serial connection when done.
mega_serial.close()
```

This code reads RFID data and maps the traffic congestion pattern. For this function to work, we need to add the following items.

1) RFID readers and RFID tags.
2) Appropriate libraries and drivers for communication with RFID readers.
3) A web page or database that is loaded and displayed on stored data.
4) Proper Arduino code for communication with RFID readers.

To send data to a specific website, we can use Python and the ‘requests’ library to make HTTP POST requests to a web server. In this example, we will see how to modify an Arduino Uno Python script to send data to a website. We need to have a server-side script to process the data and store it in a database on the web.

Here’s how we can modify Arduino Uno python script to upload RFID and traffic congestion data to a website:

```
import serial
import requests
# Define the Arduino Uno's serial port and website URL.
uno_port = "COM3"  # Update this with your Uno's serial port.
website_url = "https://example.com/upload_traffic_data.php"  # Update with your website's URL.
try:
    uno_serial = serial.Serial(uno_port, 9600)
```

```
except serial.SerialException:
    print("Failed to open the Uno's serial port.")
    exit()
while True:
    uno_data = uno_serial.readline().strip().decode('utf-8')
    print("RFID data from Arduino Uno:", uno_data)
    # Simulate traffic congestion data (0-100, where 100 means heavy congestion).
    traffic_congestion = 50  # Change this value as needed.
    print("Traffic Congestion Level:", traffic_congestion)
    # Prepare the data to send to the website.
    data_to_send = {
        "rfid_data": uno_data,
        "traffic_congestion": str(traffic_congestion)
    }
    try:
        response = requests.post(website_url, data=data_to_send)
        if response.status_code == 200:
            print("Data uploaded successfully.")
        else:
            print("Failed to upload data. Status code:", response.status_code)
    except requests.exceptions.RequestException as e:
        print("Error:", e)
# Close the serial connection when done.
uno_serial.close()
```

In this code, we use the ‘requests.post()’ method to send RFID data and traffic congestion information to the website. The website URL must point to a server-side script (e.g., PHP) that can process the incoming data and store it in a database.

We should replace "https://example.com/upload_traffic_data.php" with the actual URL of our website's data-checking script.

In our website, we need to create a script that retrieves the data and inserts it into a database.

Below is a simple example of a PHP script that can consume data on our website

```
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $rfid_data = $_POST['rfid_data'];
    $traffic_congestion = $_POST['traffic_congestion'];
    // Insert the data into your database.
    // You'll need to set up a database connection and query accordingly.
    echo "Data received and processed successfully.";
} else {
    echo "Invalid request.";
}
```