PHASE 4 DEVELOPMENT

TRAFFIC MANAGEMENT SYSTEM

Creating a real-time Environment Management platform involves a combination of front end and backend technologies. Here's a simplified outline using C and C++ and python programming with wi-fi connection for the front end and Node.js for the back end:

PYTHON program:

```python
import time
import random

class Vehicle:
    def __init__(self, id):
        self.id = id

    def move(self):
        print(f"Vehicle {self.id} is moving")

class TrafficSignal:
    def __init__(self):
        self.state = "red"

    def change_state(self):
        if self.state == "red":
            self.state = "green"
        else:
            self.state = "red"

    def display_state(self):
        print(f"Traffic signal is {self.state}")

# Simulation
traffic_signal = TrafficSignal()

for i in range(5):
    time.sleep(1)  # Simulating time passing

    if random.random() < 0.5:
        vehicle = Vehicle(i + 1)
        vehicle.move()
```

```
    if i % 2 == 0:
        traffic_signal.change_state()

    traffic_signal.display_state()
```

C++ program:

```cpp
#include <iostream>
#include <vector>
#include <ctime>
#include <cstdlib>

class Vehicle {
public:
    enum class Type { Car, Truck, Motorcycle };

    Vehicle(Type type) : type(type) {}

    Type getType() const {
        return type;
    }

private:
    Type type;
};

class TrafficManagementSystem {
public:
    void addVehicle(const Vehicle& vehicle) {
        vehicles.push_back(vehicle);
    }

    void simulateTraffic() {
        // Simulate traffic actions here
        // Example: Print vehicle types in traffic
        std::cout << "Current traffic:\n";
        for (const auto& vehicle : vehicles) {
            switch (vehicle.getType()) {
                case Vehicle::Type::Car:
                    std::cout << "Car ";
                    break;
                case Vehicle::Type::Truck:
```

```
                std::cout << "Truck ";
                break;
            case Vehicle::Type::Motorcycle:
                std::cout << "Motorcycle ";
                break;
        }
    }
    std::cout << "\n";
  }

private:
  std::vector<Vehicle> vehicles;
};

int main() {
  srand(static_cast<unsigned>(time(0)));

  TrafficManagementSystem trafficSystem;

  // Simulate adding vehicles to traffic
  for (int i = 0; i < 10; ++i) {
    int randomType = rand() % 3; // 0: Car, 1: Truck, 2: Motorcycle
    Vehicle::Type type = static_cast<Vehicle::Type>(randomType);
    trafficSystem.addVehicle(Vehicle(type));
  }

  // Simulate traffic flow
  trafficSystem.simulateTraffic();

  return 0;
}
```

C program:

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define MAX_VEHICLES 10

typedef enum {
  Car,
```

```
    Truck,
    Motorcycle
} VehicleType;

typedef struct {
    VehicleType type;
} Vehicle;

typedef struct {
    Vehicle vehicles[MAX_VEHICLES];
    int count;
} TrafficManagementSystem;

void addVehicle(TrafficManagementSystem *trafficSystem, VehicleType type) {
    if (trafficSystem->count < MAX_VEHICLES) {
        trafficSystem->vehicles[trafficSystem->count].type = type;
        trafficSystem->count++;
    }
}

void simulateTraffic(const TrafficManagementSystem *trafficSystem) {
    printf("Current traffic:\n");
    for (int i = 0; i < trafficSystem->count; ++i) {
        switch (trafficSystem->vehicles[i].type) {
            case Car:
                printf("Car ");
                break;
            case Truck:
                printf("Truck ");
                break;
            case Motorcycle:
                printf("Motorcycle ");
                break;
        }
    }
    printf("\n");
}

int main() {
    srand((unsigned int)time(NULL));

    TrafficManagementSystem trafficSystem;
    trafficSystem.count = 0;
```

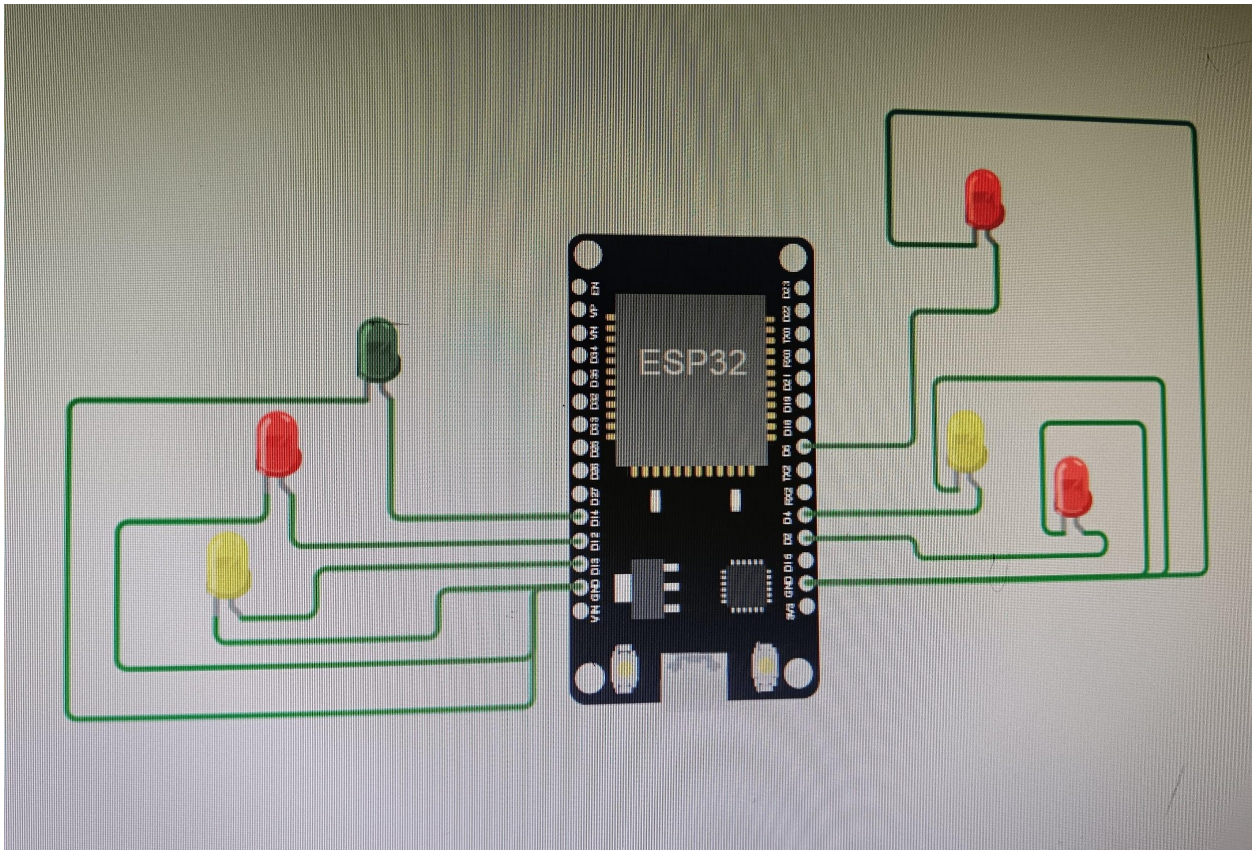
```
    // Simulate adding vehicles to traffic
    for (int i = 0; i < 10; ++i) {
        VehicleType randomType = (VehicleType)(rand() % 3); // 0: Car, 1: Truck, 2:
Motorcycle
        addVehicle(&trafficSystem, randomType);
    }

    // Simulate traffic flow
    simulateTraffic(&trafficSystem);

    return 0;
}
```

MICROPROCESSOR PROGRAM:

```
import machine
import time
# Define GPIO pins for the traffic lights
red1_pin = machine.Pin(D12, machine.Pin.OUT)  # Replace with your GPIO pin numbers
yellow1_pin = machine.Pin(D13, machine.Pin.OUT)
green1_pin = machine.Pin(D14, machine.Pin.OUT)

red2_pin = machine.Pin(D2, machine.Pin.OUT)  # Replace with your GPIO pin numbers
yellow2_pin = machine.Pin(D4, machine.Pin.OUT)
green2_pin = machine.Pin(D5, machine.Pin.OUT)

# Function to control the traffic lights
def set_traffic_lights(state1, state2):
    red1_pin.value(state1[0])
    yellow1_pin.value(state1[1])
    green1_pin.value(state1[2])

    red2_pin.value(state2[0])
    yellow2_pin.value(state2[1])
    green2_pin.value(state2[2])
import machine
import time

# Define GPIO pins for the traffic lights
red1_pin = machine.Pin(D12, machine.Pin.OUT)  # Replace with your GPIO pin numbers
yellow1_pin = machine.Pin(D13, machine.Pin.OUT)
green1_pin = machine.Pin(D14, machine.Pin.OUT)

red2_pin = machine.Pin(D2, machine.Pin.OUT)  # Replace with your GPIO pin numbers
```

```
yellow2_pin = machine.Pin(D4, machine.Pin.OUT)
green2_pin = machine.Pin(D5, machine.Pin.OUT)
# Define traffic light states
RED = (1, 0, 0)
YELLOW = (0, 1, 0)
GREEN = (0, 0, 1)

# Initial traffic light states
state1 = RED
state2 = GREEN

while True:
    set_traffic_lights(state1, state2)
    time.sleep(5)  # Red light on for 5 seconds

    set_traffic_lights(YELLOW, YELLOW)  # Both lights yellow for 2 seconds
    time.sleep(2)

    set_traffic_lights(RED, GREEN)  # Light 1 green, light 2 red for 5 seconds
    time.sleep(5)

    set_traffic_lights(YELLOW, YELLOW)  # Both lights yellow for 2 seconds
    time.sleep(2)

    set_traffic_lights(GREEN, RED)  # Light 1 red, light 2 green for 5 seconds
    time.sleep(5)
```

Result:



Conclusion:

In conclusion, an Traffic management System using the Internet of Things (IoT) represents a transformative and highly valuable technology for addressing a wide range of environmental challenges. This system harnesses the power of interconnected sensors, devices, and data analytics to collect, manage, and analyze environmental data in real-time.