CODE FOR SMART TRAFFIC MANAGEMENT SYSTEM:

```
#include <SPI.h>
#include <MFRC522.h>

// Traffic Light Pins [North, East, South, West][Red, Yellow, Green]
const int trafficLights[4][3] = {
  {22, 23, 24}, // North
  {25, 26, 27}, // East
  {28, 29, 30}, // South
  {31, 32, 33}  // West
};

// Blue LED Pins [North, East, South, West]
const int blueLEDs[4] = {34, 35, 36, 37};

// Ultrasonic Sensor Pins [Trig, Echo]
const int ultrasonic[4][2] = {
  {11, 12}, // North (Controls WEST traffic)
  {13, 14}, // East  (Controls NORTH traffic)
  {15, 16}, // South (Controls EAST traffic)
  {17, 18}  // West  (Controls SOUTH traffic)
};

// RFID Settings
#define RST_PIN 9
#define SS_PIN 10
MFRC522 mfrc522(SS_PIN, RST_PIN);

const int maxDetectionRange = 8; // 8cm maximum range
int currentGreenDirection = -1; // Start with no green light
```

```
const int trafficLightToSensor[4] = {1, 2, 3, 0}; // Maps traffic lights to their controlling sensors

void setup() {
  Serial.begin(9600);
  while (!Serial); // Wait for serial port to connect
  SPI.begin();
  mfrc522.PCD_Init();

  // Initialize all components
  for(int i = 0; i < 4; i++) {
    // Traffic lights
    for(int j = 0; j < 3; j++) {
      pinMode(trafficLights[i][j], OUTPUT);
      digitalWrite(trafficLights[i][j], LOW);
    }
    // Blue LEDs
    pinMode(blueLEDs[i], OUTPUT);
    digitalWrite(blueLEDs[i], LOW);
    // Ultrasonic sensors
    pinMode(ultrasonic[i][0], OUTPUT); // TRIG
    pinMode(ultrasonic[i][1], INPUT);  // ECHO
    digitalWrite(ultrasonic[i][0], LOW); // Ensure trigger starts low
  }
  // Start with all red
  for(int i = 0; i < 4; i++) {
    digitalWrite(trafficLights[i][0], HIGH);
  }
}

float getDistance(int sensor) {
  digitalWrite(ultrasonic[sensor][0], LOW);
  delayMicroseconds(2);
  digitalWrite(ultrasonic[sensor][0], HIGH);
  delayMicroseconds(10);
  digitalWrite(ultrasonic[sensor][0], LOW);

  long duration = pulseIn(ultrasonic[sensor][1], HIGH, 30000); //
Timeout after 30ms
  if (duration == 0) {
    return maxDetectionRange; // Return max range if timeout
  }
  float distance = duration * 0.034 / 2;
  return constrain(distance, 0, maxDetectionRange);
}

void setGreenLight(int direction) {
  // Yellow transition if changing from a valid direction
  if (currentGreenDirection >= 0 && currentGreenDirection < 4) {
    digitalWrite(trafficLights[currentGreenDirection][2], LOW);
    digitalWrite(trafficLights[currentGreenDirection][1], HIGH);
    delay(1000); // 1 second yellow light
  }

  // Turn all red
  for(int i = 0; i < 4; i++) {
    digitalWrite(trafficLights[i][0], HIGH);
    digitalWrite(trafficLights[i][1], LOW);
    digitalWrite(trafficLights[i][2], LOW);
  }

  // Set new green light
  if (direction >= 0 && direction < 4) {
    digitalWrite(trafficLights[direction][0], LOW);
    digitalWrite(trafficLights[direction][2], HIGH);
    currentGreenDirection = direction;
  }

  // Turn off all blue LEDs
  for(int i = 0; i < 4; i++) {
    digitalWrite(blueLEDs[i], LOW);
  }
}

void updateBlueLEDs(float densities[4]) {
  for(int i = 0; i < 4; i++) {
    if(i != currentGreenDirection && i >= 0 && i < 4) {
      digitalWrite(blueLEDs[i], densities[trafficLightToSensor[i]] < 0.25 ?
HIGH : LOW);
    }
  }
}

void loop() {
  // Read sensor distances and calculate densities
  float sensorDensity[4] = {0};
  for(int i = 0; i < 4; i++) {
    float distance = getDistance(i);
    sensorDensity[i] = (maxDetectionRange - distance) /
maxDetectionRange;

    Serial.print("Sensor ");
    Serial.print(i);
    Serial.print(" (Controls ");
    switch(i) {
      case 0: Serial.print("WEST"); break;
      case 1: Serial.print("NORTH"); break;
      case 2: Serial.print("EAST"); break;
      case 3: Serial.print("SOUTH"); break;
    }
    Serial.print("): ");
    Serial.print(distance);
    Serial.print("cm, Density: ");
    Serial.println(sensorDensity[i]);
  }

  // Calculate which traffic light should be green
  float maxDensity = 0;
  int newGreenDirection = currentGreenDirection;

  for(int dir = 0; dir < 4; dir++) {
    int controllingSensor = trafficLightToSensor[dir];
    if(controllingSensor >= 0 && controllingSensor < 4 &&
       sensorDensity[controllingSensor] > maxDensity) {
      maxDensity = sensorDensity[controllingSensor];
      newGreenDirection = dir;
    }
  }

  // Only change if we found a direction with significant density
(>12.5%)
  if(maxDensity > 0.125) {
    if(newGreenDirection != currentGreenDirection) {
      setGreenLight(newGreenDirection);
    }
    updateBlueLEDs(sensorDensity);
  }

  // RFID emergency override
  if (mfrc522.PICC_IsNewCardPresent()) {
    if (mfrc522.PICC_ReadCardSerial()) {
      String tag = "";
      for (byte i = 0; i < mfrc522.uid.size; i++) {
        tag.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? "0" : ""));
        tag.concat(String(mfrc522.uid.uidByte[i], HEX));
      }
      tag.toUpperCase();
      Serial.println("RFID detected: " + tag);

      if(tag == "AUTHORIZEDTAG") { // Removed underscore for better compatibility
        setGreenLight(0); // Force North direction
      }
      mfrc522.PICC_HaltA();
    }
  }

  delay(200);
}
```