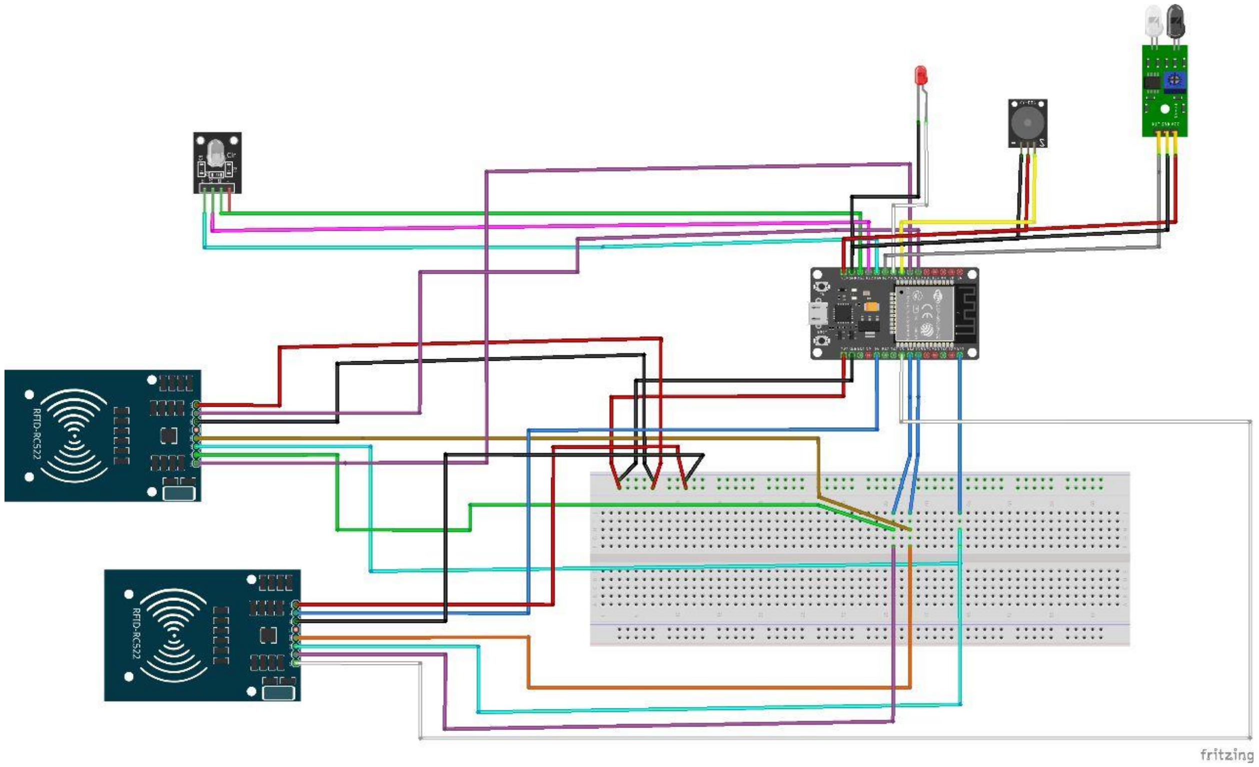
Circuit diagram part 1



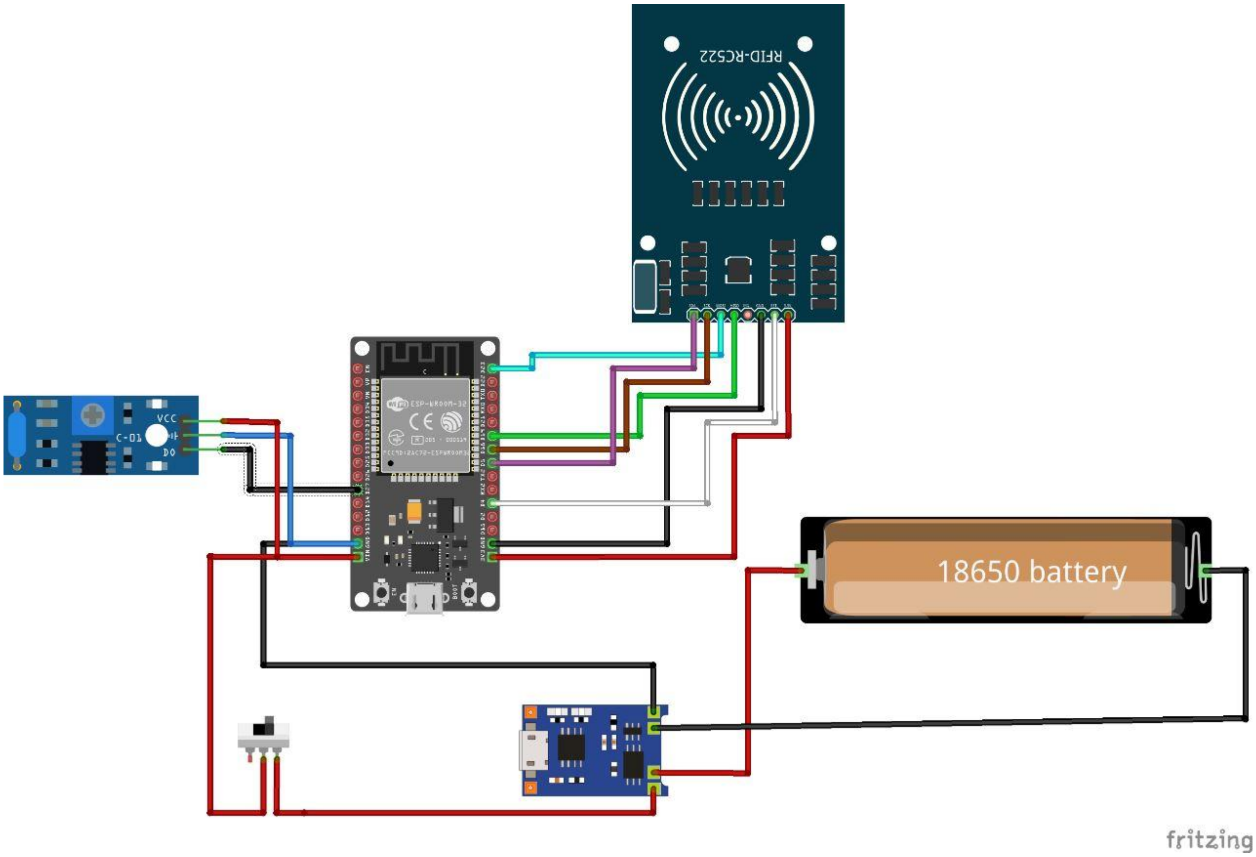
Code

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

const char* ssid = "realme8";
const char* password = "hish123456";

#define SS_PIN 5
#define RST_PIN 4
MFRC522 rfid(SS_PIN, RST_PIN);  // Instance of the class

#define SS_PIN_2 33
#define RST_PIN_2 32
MFRC522 rf2(SS_PIN_2, RST_PIN_2);  // Instance of the class

String carNumber = "";  // Car number based on UID
```

```
#define BLUE_LED_PIN 13
#define GREEN_LED_PIN 12
#define TRAFFIC_RED_LED_PIN 14

enum LEDState {
  BLUE,
  GREEN,
  RED
};

LEDState currentLEDState = BLUE;

unsigned long previousMillis = 0;
const unsigned long blueInterval = 5000;   // 5 seconds
const unsigned long greenInterval = 7000;  // 7 seconds
const unsigned long redInterval = 40000;   // 20 seconds

bool isRFIDReadEnabled = false;  // Flag to enable RFID reading when the red
light is on

#define IR_SENSOR_PIN 27      // ESP32 pin GPIO18 connected to OUT pin of
IR obstacle avoidance sensor
#define ALERT_RED_LED_PIN 26  // ESP32 pin connected to the red channel of
the RGB LED
#define BUZZER_PIN 25         // ESP32 pin connected to the buzzer

#define BUZZER_FREQ 2100  // Desired frequency of the buzzer sound

const char* violationType = "Traffic Jump";
const char* fineAmount = "1000";

void setup() {
  Serial.begin(9600);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("WiFi Connection Success!");

  SPI.begin();      // Init SPI bus
  rfid.PCD_Init();  // Init MFRC522
  rf2.PCD_Init();   // Init MFRC522

  pinMode(IR_SENSOR_PIN, INPUT);
  pinMode(ALERT_RED_LED_PIN, OUTPUT);
  pinMode(BUZZER_PIN, OUTPUT);

  pinMode(BLUE_LED_PIN, OUTPUT);
  pinMode(GREEN_LED_PIN, OUTPUT);
  pinMode(TRAFFIC_RED_LED_PIN, OUTPUT);

  // Turn off all LEDs initially
  digitalWrite(BLUE_LED_PIN, LOW);
  digitalWrite(GREEN_LED_PIN, LOW);
  digitalWrite(TRAFFIC_RED_LED_PIN, LOW);

  // Set the initial LED state
  currentLEDState = BLUE;
  digitalWrite(BLUE_LED_PIN, HIGH);
}

void loop() {
  // LED control based on time intervals
  unsigned long currentMillis = millis();
  switch (currentLEDState) {
    case BLUE:
      if (currentMillis - previousMillis >= blueInterval) {
        previousMillis = currentMillis;
        currentLEDState = GREEN;
        digitalWrite(BLUE_LED_PIN, LOW);
        digitalWrite(GREEN_LED_PIN, HIGH);
      }
      break;

    case GREEN:
      if (currentMillis - previousMillis >= greenInterval) {
        previousMillis = currentMillis;
        currentLEDState = RED;
        digitalWrite(GREEN_LED_PIN, LOW);
        digitalWrite(TRAFFIC_RED_LED_PIN, HIGH);
        isRFIDReadEnabled = true;  // Enable RFID reading when the red light is on
      }
      break;

    case RED:
      if (currentMillis - previousMillis >= redInterval) {
        previousMillis = currentMillis;
        currentLEDState = BLUE;
        digitalWrite(TRAFFIC_RED_LED_PIN, LOW);
        digitalWrite(BLUE_LED_PIN, HIGH);
        isRFIDReadEnabled = false;  // Disable RFID reading when the red light is
off
      }
      break;
  }

  // Check RFID card presence and read data if enabled
  if (isRFIDReadEnabled && rfid.PICC_IsNewCardPresent() &&
rfid.PICC_ReadCardSerial()) {
    // Generate car number based on UID
    carNumber = "KA " + getUIDString(rfid.uid);

    // Print the UID and car number in the serial monitor
    // Serial.print("UID: ");
    // Serial.println(getUIDString(rfid.uid));
    Serial.print("Car Number: ");
    Serial.println(carNumber);

    HTTPClient http;
    String url =
"https://script.google.com/macros/s/AKfycbymDDgolU0jBi5D6DhZpZJwLXyTjldcq
Pp8K1saZunWOd87W4TX6EkrxBk-B-exjlX0JQ/exec?car_number=" +
urlEncode(carNumber) + "&violation_type=" + urlEncode(violationType) +
"&fine_amount=" + urlEncode(fineAmount);

    http.begin(url);
    int httpCode = http.GET();

    if (httpCode > 0) {
      String payload = http.getString();
      Serial.println(payload);
    }

    http.end();

    // Halt PICC
    rfid.PICC_HaltA();

    // Stop encryption on PCD
    rfid.PCD_StopCrypto1();
  }

  if (isRFIDReadEnabled && rf2.PICC_IsNewCardPresent() &&
rf2.PICC_ReadCardSerial()) {
    // Generate car number based on UID
    carNumber = "KA " + getUIDString(rf2.uid);

    // Print the UID and car number in the serial monitor
    // Serial.print("UID: ");
    // Serial.println(getUIDString(rf2.uid));
    Serial.print("Car Number: ");
    Serial.println(carNumber);

    HTTPClient http;
    String url =
"https://script.google.com/macros/s/AKfycbymDDgolU0jBi5D6DhZpZJwLXyTjldcq
```

```
Pp8K1saZunWOd87W4TX6EkrxBk-B-exjlX0JQ/exec?car_number=" +
urlEncode(carNumber) + "&violation_type=" + urlEncode(violationType) +
"&fine_amount=" + urlEncode(fineAmount);

    http.begin(url);
    int httpCode = http.GET();

    if (httpCode > 0) {
      String payload = http.getString();
      Serial.println(payload);
    }

    http.end();

    // Halt PICC
    rf2.PICC_HaltA();

    // Stop encryption on PCD
    rf2.PCD_StopCrypto1();
  }



  int ir_state = digitalRead(IR_SENSOR_PIN);
  // Serial.println(ir_state);

  if (ir_state == LOW) {
    // Obstacle is detected, turn on the red LED and sound the buzzer
    digitalWrite(ALERT_RED_LED_PIN, HIGH);

    // Generate PWM signal to produce buzzer sound
    tone(BUZZER_PIN, BUZZER_FREQ);

    Serial.println("Obstacle Detected");

  } else {
    // No obstacle, turn off the red LED and silence the buzzer
    digitalWrite(ALERT_RED_LED_PIN, LOW);
    noTone(BUZZER_PIN);
  }
}

// Helper function to convert UID bytes to a string
String getUIDString(MFRC522::Uid uid) {
  String uidString = "";
  for (byte i = 0; i < uid.size; i++) {
    uidString += uid.uidByte[i] < 0x10 ? "0" : "";
    uidString += String(uid.uidByte[i], HEX);
    uidString += " ";
  }
  uidString.trim();
  return uidString;
}

String urlEncode(const String &str) {
  String encodedString = "";
  char c;
  int len = str.length();

  for (int i = 0; i < len; i++) {
    c = str.charAt(i);
    if (isAlphaNumeric(c)) {
      encodedString += c;
    } else {
      char encodedChar[4];
      sprintf(encodedChar, "%%%02X", c);
      encodedString += encodedChar;
    }
  }

  return encodedString;
}
```

Circuit diagram part 2



Code

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

const char* ssid = "realme8";
const char* password = "hish123456";

String carNumber = "";  // Car number based on UID

const char* violationType = "Crash";
const char* fineAmount = "2000";

#define SS_PIN 5
#define RST_PIN 4
MFRC522 rfid(SS_PIN, RST_PIN);  // Instance of the class
```

```
bool isRFIDReadEnabled = false;  // Flag to enable RFID reading when the red light is on

const int KNOCK_SENSOR_PIN = 27;  // GPIO pin for the knock sensor

void setup() {
  Serial.begin(9600);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("WiFi Connection Success!");

  SPI.begin();      // Init SPI bus
  rfid.PCD_Init();  // Init MFRC522
  pinMode(KNOCK_SENSOR_PIN, INPUT);
}

void loop() {

  isRFIDReadEnabled = digitalRead(KNOCK_SENSOR_PIN);
  Serial.println(isRFIDReadEnabled);

  // Add your knock sensor logic here
  // ...

  // delay(1000);  // Wait for a second before looping again


  // Check RFID card presence and read data if enabled
  if (isRFIDReadEnabled && rfid.PICC_IsNewCardPresent() && rfid.PICC_ReadCardSerial()) {
    // Generate car number based on UID
    carNumber = "KA " + getUIDString(rfid.uid);
    Serial.print("Collided Car Number: ");
    Serial.println(carNumber);

    if (carNumber!= ""){
      // Send data to Google Sheet
      sendToGoogleSheet(carNumber);
    }
    carNumber = "";

    // Halt PICC
    rfid.PICC_HaltA();

    // Stop encryption on PCD
    rfid.PCD_StopCrypto1();
  }


  isRFIDReadEnabled = false;
}

void sendToGoogleSheet(const String& carNumber) {
  HTTPClient http;
  String url =
"https://script.google.com/macros/s/AKfycbymDDgolU0jBi5D6DhZpZJwLXyTjldcq
Pp8K1saZunWOd87W4TX6EkrxBk-B-exjlX0JQ/exec?car_number=" +
urlEncode(carNumber) + "&violation_type=" + urlEncode(violationType) +
"&fine_amount=" + urlEncode(fineAmount);

  http.begin(url);
  int httpCode = http.GET();

  if (httpCode > 0) {
    String payload = http.getString();
    Serial.println(payload);
  }

  http.end();
```

```
}

// Helper function to convert UID bytes to a string
String getUIDString(MFRC522::Uid uid) {
  String uidString = "";
  for (byte i = 0; i < uid.size; i++) {
    uidString += uid.uidByte[i] < 0x10 ? "0" : "";
    uidString += String(uid.uidByte[i], HEX);
    uidString += " ";
  }
  uidString.trim();
  return uidString;
}

String urlEncode(const String& str) {
  String encodedString = "";
  char c;
  int len = str.length();

  for (int i = 0; i < len; i++) {
    c = str.charAt(i);
    if (isAlphaNumeric(c)) {
      encodedString += c;
    } else {
      char encodedChar[4];
      sprintf(encodedChar, "%%%02X", c);
      encodedString += encodedChar;
    }
  }

  return encodedString;
}
```