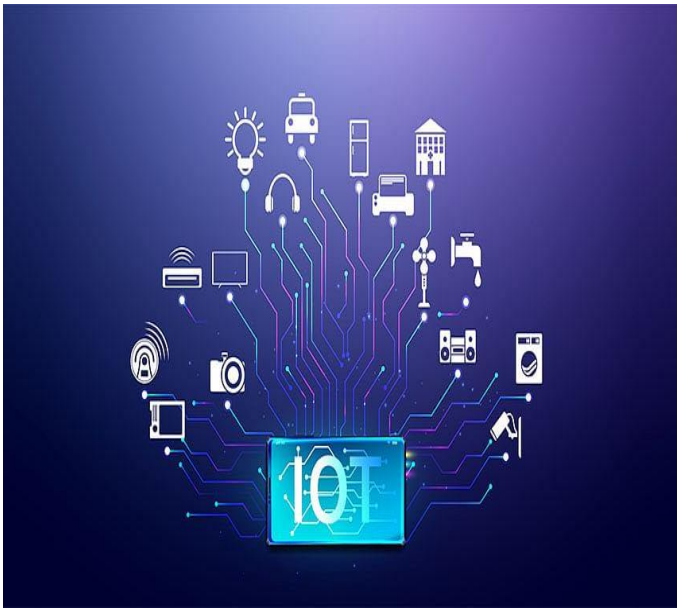
PUBLIC TRANSPORTATION OPTIMIZATION

IOT PHASE 3



SUBMITTED BY

TEAM LEADER: NITHIYA KUMAR G H

TEAM MEMBER: VIGNESH.S

TEAM MEMBER: NITHISH KUMAR M

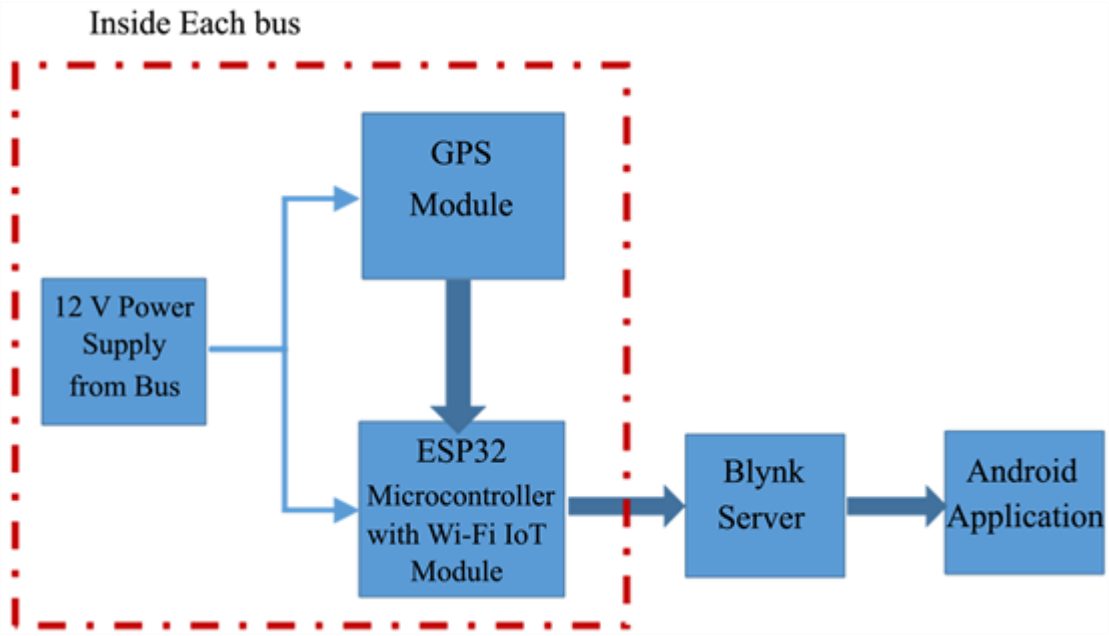
TEAM MEMBER: NAVEEN G

TEAM MEMBER: SANJAY.A

INTRODUCTION:

Public transport (also known as **public transportation**, **public transit**, **mass transit**, or simply **transit**) is a system of transport for passengers by group travel systems available for use by the general public unlike private transport, typically managed on a schedule, operated on established routes, and that charge a posted fee for each trip.

BLOCK DIAGRAM:



COMPONENTS REQUIRED:

- Arduino (e.g., Arduino Uno)
- Single Neo Pixel LED (on a strip)
- Breadboard (optional)
- Jumper wires
- Power supply
- ultrasonic sensor
- and resistor

COMPONENTS DESCRIPTION:

Certainly, here's a brief description of each of the components you've listed for your project:

ARDUINO UNO:

The Arduino Uno is a microcontroller board based on the ATmega328P. It's the brain of your project, responsible for controlling and interacting with various components. It has digital and log pins for input and output, making it suitable for a wide range of project

NEOPIXEL LED (ON A STRIP):

Neo Pixel is a brand of individually addressable RGB LEDs. A single Neo Pixel LED can display various colours, and when you have them in a strip, you can create dynamic lighting effects. These LEDs are commonly used for decorative and lighting projects.

BREADBOARD:

A breadboard is a handy prototyping tool for electronics projects. It allows you to build and test circuits without soldering. Components can be plugged into the holes on the board, and it provides a convenient way to connect and disconnect components during the design and testing phase.

JUMPER WIRES:

Jumper wires are used to make electrical connections between components on a breadboard or to connect components in your project. They come in various lengths and are essential for creating circuits without soldering.

POWER SUPPLY:

The power supply provides electrical power to your project. Depending on the requirements of your components, you may use batteries, a USB connection, or an external power source to ensure that all components receive the necessary voltage and current to function properly.

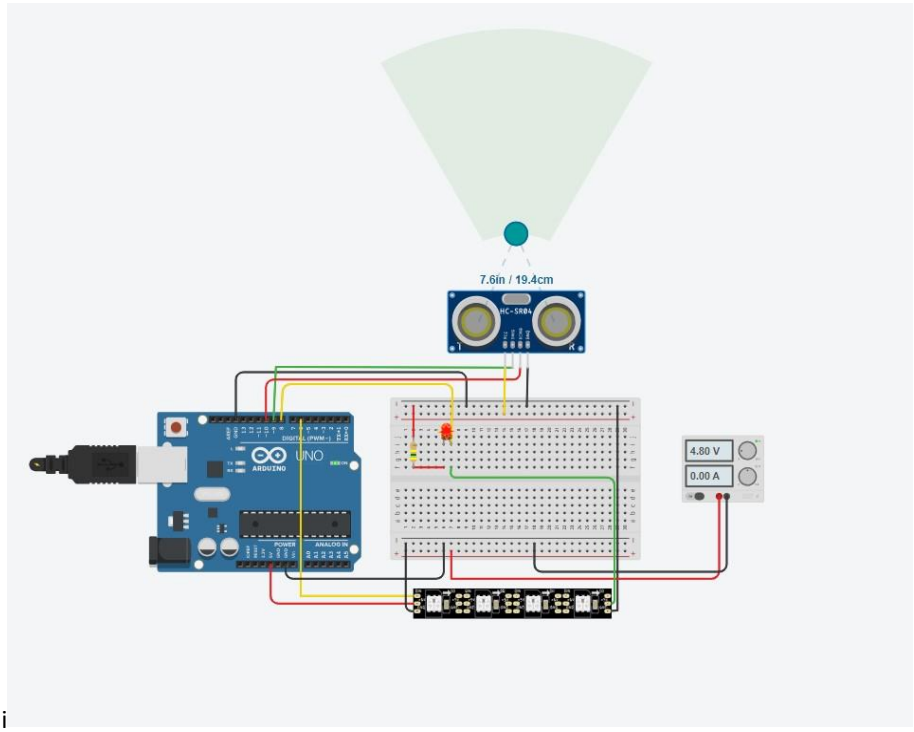
ULTRASONIC SENSOR:

An ultrasonic sensor is a device that uses ultrasonic sound waves to measure distance. It typically consists of a transmitter and a receiver. The sensor sends out a sound wave, which bounces off an object and returns to the sensor. By measuring the time it takes for the sound wave to return, the sensor can calculate the distance to the object.

RESISTOR:

A resistor is a passive electronic component used to limit the flow of electric current. It's often used to protect components from excessive current, to set a specific voltage or current level, or to act as a voltage divider in a circuit. In your project, it may be used for current limiting or other specific purposes depending on the circuit requirements.

CIRCUIT DESIGN:



i

PYTHON SCRIPT:

```
#include <Adafruit _Neo Pixel. h>


#define NEOPIXEL_PIN 6

#define NUM_PIXELS 1


Adafruit _Neo Pixel strip = Ada fruit _Neo Pixel (NUM_PIXELS, NEOPIXEL_PIN, NEO_GRB + NEO_KHZ800);

Constant int trigger Pin = 9;

constant int echo Pin = 10;
```

```
void setup () {
  Serial. begin(9600);
  strip. Begin ();
  strip. Show (); // Initialize all pixels to 'off'
  pin Mode trigger Pin, OUTPUT);
  pin Mode (echo Pin, INPUT);
}

void loop () {
  long duration;
  int distance;

  digital Write (trigger Pin, LOW);
  Delay Microseconds (2);
  Digital Write (trigger Pin, HIGH);
  Delay Microseconds (10);
  Digital Write (trigger Pin, LOW);

  duration = pulse In (echo Pin, HIGH);
  distance = duration * 0.0343 / 2;

  if (distance <= 10) {
    strip. set Pixel Colour (0, strip. Colour (255, 0, 0)); // Red colour for warning
    strip. Show ();
    Serial. Print ln ("Too close!");
  } else {
    strip. Set Pixel Colour (0, strip. Colour (0, 255, 0)); // Green colour for normal
    strip. Show ();
    Serial. Print ln ("Safe distance.");
```

```
  }

  Delay (1000);
}
```