VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

## School of Computer Science and Engineering (SCOPE)

# PROJECT REPORT

| | | |
|---|---|---|
| **COURSE CODE / TITLE** | BECE204L– Microprocessors and Microcontrollers | |
| **PROGRAM / YEAR/ SEM** | B.Tech (BCE)/II Year/ FALL 2023-2024 | |
| **DATE OF SUBMISSION** | | |
| **TEAM MEMBERS DETAILS** | **22BCE1789** | **KRITHICK.S** |
| | **22BCE1477** | **ABISHEK DEVADOSS** |
| | **22BCE1640** | **R. SACHEEV KRISHANU** |
| | **22BCE1634** | **MOHAMMED ASIF** |
| **PROJECT TITLE** | **DENSITY-BASED TRAFFIC CONTROL SYSTEM** | |

| | | | |
|---|---|---|---|
| **COURSE HANDLING FACULTY** | **Dr. S. REVATHI Associate Professor, SENSE** | **REMARKS** | |
| **FACULTY SIGNATURE** | | | |

# PROJECT REPORT: DENSITY-BASED TRAFFIC CONTROL SYSTEM

## OBJECTIVE:

The main aim of this project is to develop a density-based dynamic traffic signal system. With our system, we wish to deduce an efficient method to control the congested traffic in highly connected and urbanized cities/towns. The system is devised to detect the density of automobiles on each particular lane present in a junction or 3 or more lanes. Based on the density of vehicles in the lanes, our traffic light system will automatically allow vehicles of a lane to pass, while also adjusting signal timings based on real-time traffic density at a junction.

## INTRODUCTION:

Traffic congestion is a significant issue in major cities worldwide. It is even more prevalent in the major metropolitan cities of India. Ever since the pandemic, connectivity through roads in India has increased throughout the country, at an average of 22%. However, worldwide surveys, like the Global Traffic Index showed that prime Indian metropolitan cities like Mumbai, Bangalore, Delhi and Pune were ranked at #2, #6, #8, and #16 respectively globally. This is of great concern for our ever-growing population. As a result of such traffic related problems, millions of office-goers and also students face issues with travel times.

The key reasons behind the extreme traffic scenarios are due to the growth in vehicle sales, improper road architecture and road traffic control. Existing traffic control systems usually include traffic lights

with a fixed time frame for the traffic signals to change from red to green, and from green to yellow to red. However, there still are many roads/junctions in India which operate without traffic signals, and are usually monitored sparsely by the traffic police. Such conditions contribute to road accidents all over the country.

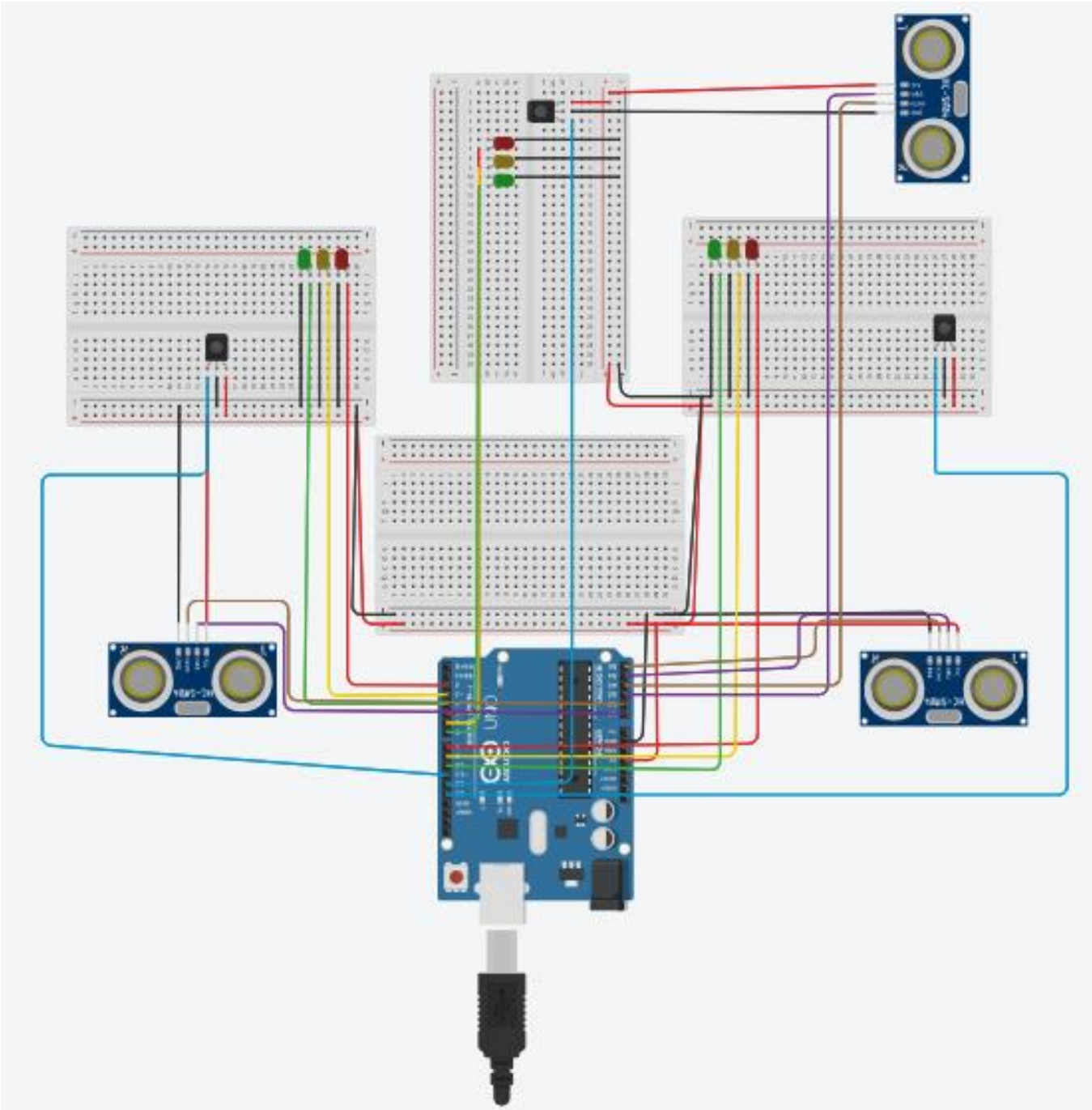
The present automatic traffic signal control systems often lack adaptability to real-time traffic conditions. Such traffic systems are set with a fixed timing after which the colour of the signal will change. This method is enough to control the flow of traffic through junctions. But is it efficient? There are no real systems which check which lanes among the junction have to be cleared to ensure smooth flow of traffic. The existing traffic lights are switched from lane to lane in a sequential manner, rather than a priority-based manner. This means that a lane, when in its turn to show the green light, will remain green even when there is very little traffic in that lane. This in turn will lead to the piling up of traffic on the other lanes, which may eventually cause a roadblock.

This is where our model is more efficient than the existing system. Our traffic system model will work by checking each lane, be it 3 or 4, for density of cars along it. The lane with more cars, or more car density, will be given more priority with respect to the other lanes. The lane with the most priority will be the lane to be cleared first. The traffic signals will turn red, yellow and green accordingly. As the first lane is getting cleared, the system must check the lane with the 2$^{nd}$ highest vehicle density, and measures will be taken automatically to clear the traffic on this lane. In short, the project addresses the limitation of fixed traffic light delays by implementing a system that dynamically adjusts signal timings based on the density of traffic at a junction.

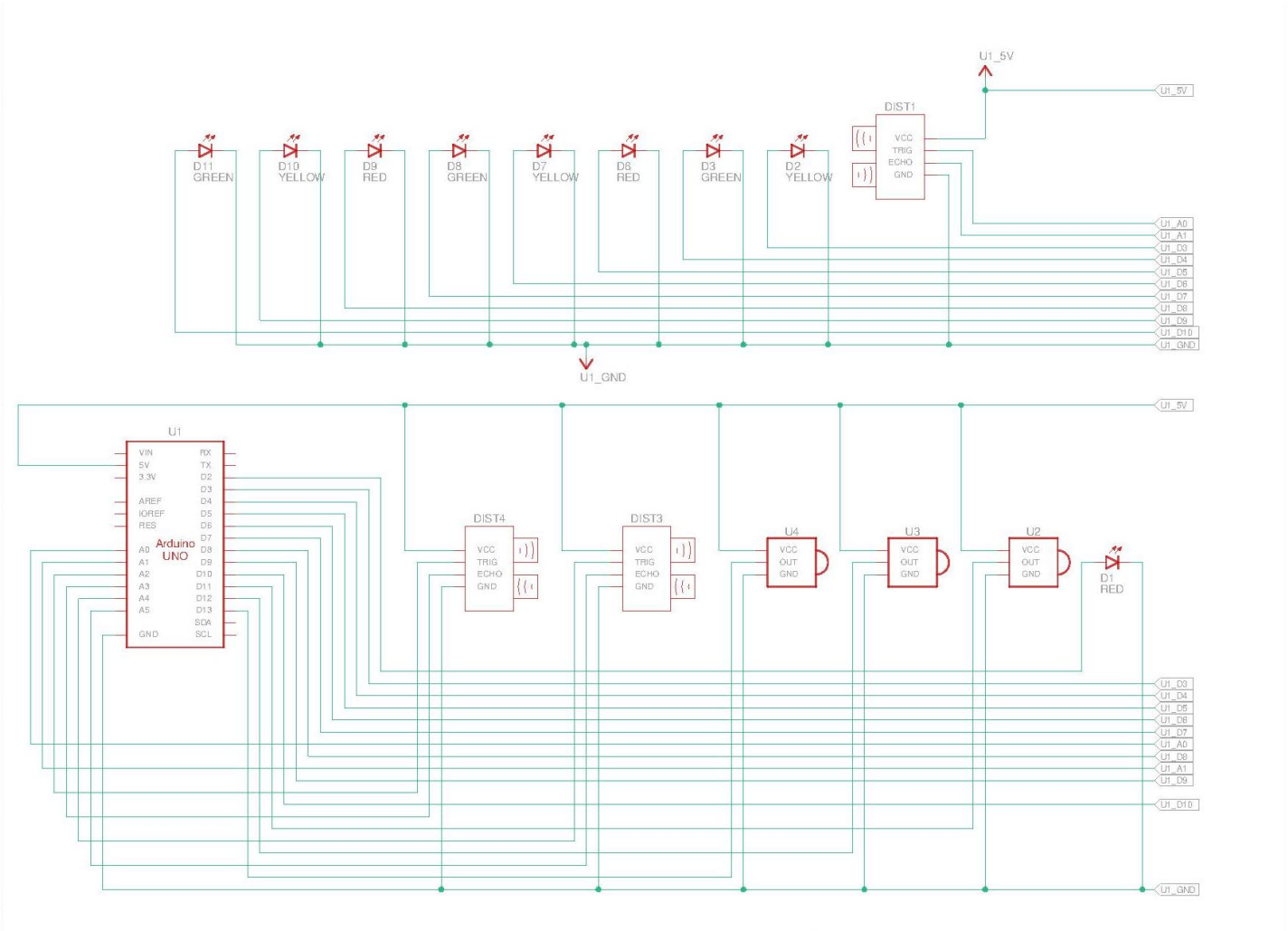
Existing models:

There are some existing systems which make use of sensors to regulate traffic lights in few cities around the world. Such systems are usually confined to only detecting the presence of vehicles on the lanes. These systems are made to detect the incoming vehicles with the usage of high-definition cameras fitted on top of the traffic light poles. Based on the data received from these cameras, the traffic signals are made to turn green. A sequential pattern follows, with the traffic signal in each lane glowing green for a fixed amount of time.

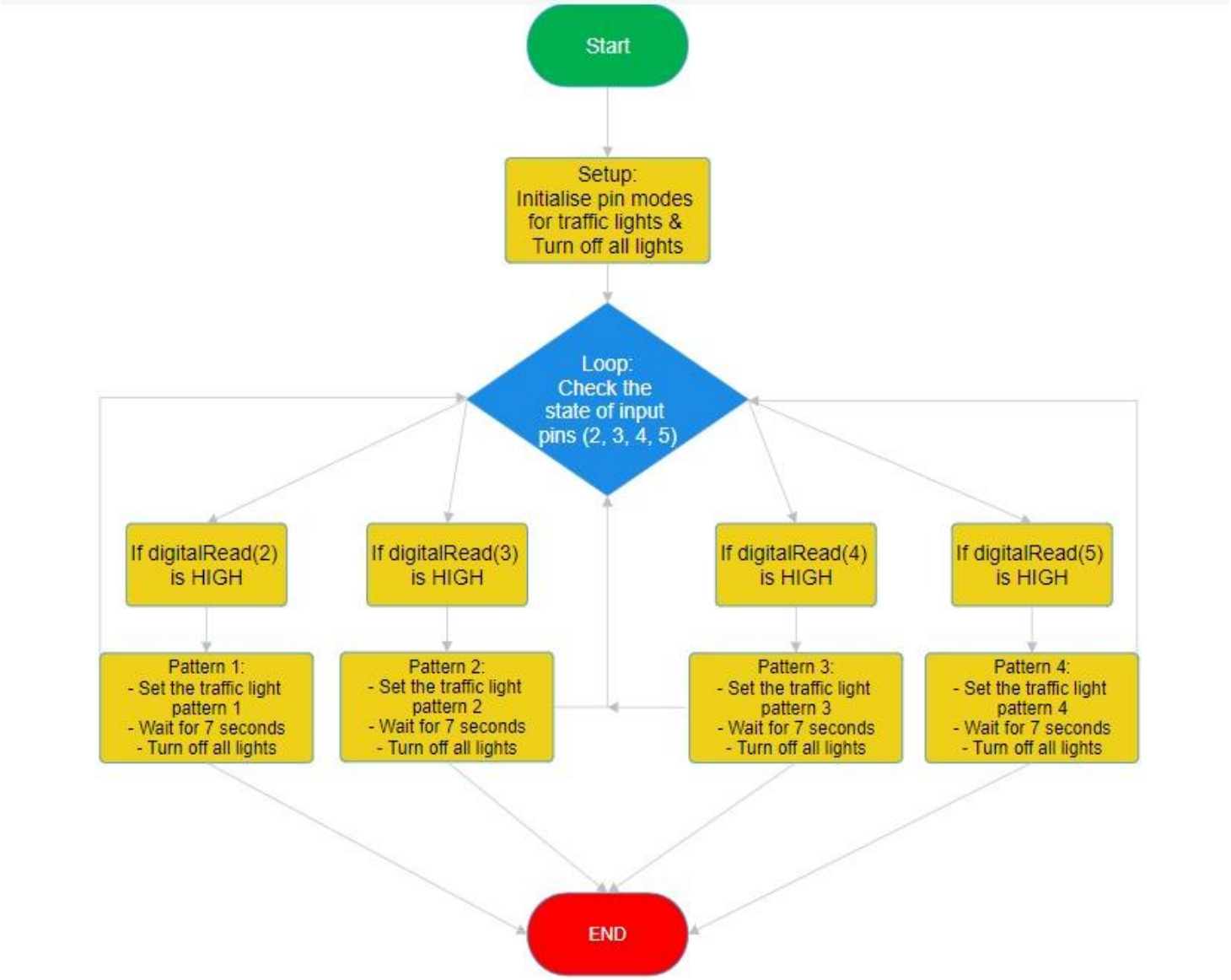
# CIRCUIT DIAGRAM:



# BLOCK DIAGRAM:



# FLOWCHART:


Start
Setup:
Initialise pin modes
for traffic lights &
Turn off all lights
Loop:
Check the
state of input
pins (2, 3, 4, 5)
If digitalRead(2)
is HIGH
If digitalRead(3)
is HIGH
If digitalRead(4)
is HIGH
If digitalRead(5)
is HIGH
Pattern 1:
- Set the traffic light
pattern 1
- Wait for 7 seconds
- Turn off all lights
Pattern 2:
- Set the traffic light
pattern 2
- Wait for 7 seconds
- Turn off all lights
Pattern 3:
- Set the traffic light
pattern 3
- Wait for 7 seconds
- Turn off all lights
Pattern 4:
- Set the traffic light
pattern 4
- Wait for 7 seconds
- Turn off all lights
END

## COMPONENTS/SOFTWARE REQUIRED:

| Component/Software | Description |
|---|---|
| Arduino Uno Board | 8-Bit Microcontroller |
| Infrared Sensors | Vehicle Detectors |
| Ultrasonic Sensors | Vehicle Density Detection |
| LEDs (Red, Yellow, Green) | Traffic Lights |
| Breadboard/Circuit Board | Basic Hardware |
| Jumper Wires | Connections |

## PROJECT DESCRIPTION:

### Components Description:

The project utilizes an Arduino Uno board, which is an 8-Bit microcontroller. It will act as the central body which controls the traffic lights. Along with the Arduino board, we use two sensors which will help us determine the presence of vehicles on the road as well as to determine the density of vehicles in a particular lane. The sensors are the infrared sensors and ultrasonic sensors. IR sensors consist of a transmitter and a receiver, using which they can detect the presence of an object in front of them. The ultrasonic sensors use a similar setup of a transmitter and a receiver. Unlike in the case of IR sensors, which transmit infrared waves, the ultrasonic sensor transmits ultrasonic sound waves. By calculating the time interval between the transmission of sound waves and the receiving of sound waves, the ultrasonic sensor can compute the distance between the sensor and the object in consideration. To simulate the traffic signals, we use simple LEDs of red, yellow and green colour to replicate a traffic signal. Using all these parts, we implement the construction of the traffic control system.

## Working:

We split the working of this traffic control system into three possible scenarios:

1. No vehicle(s)
2. Vehicle(s) on one lane only
3. Vehicle(s) on two or more lanes

The first scenario demonstrates the working of the traffic signal when there are no vehicles on any of the lanes. Such scenarios are rare in real-time, but it isn't completely non-existent. To deal with this case, we program the Arduino board to make all traffic signals on the lanes to glow red. We consider this case as the "default case". Both the IR sensor and the ultrasonic distance sensor are not used in this case.

The traffic signals shift from the "default case" when there is/are vehicles on either of the lanes. This scenario is when there is traffic on only one lane. The detection of vehicles is done by the infrared sensor. The infrared sensor is placed in-line with the lane. This is to make sure that the IR sensor can transmit and receive the infrared waves which will bounce off from the vehicles in the lane. As soon as the IR sensor detects the presence of vehicles on the lane, the Arduino board controls the traffic light of the lane in which the vehicle is detected. The signal is changed from red to green. Hence, the traffic is allowed to move through the junction. While this lane is cleared by changing the signal from red to green, the other lanes remain unaffected, i.e., they remain red, as there is no vehicle to be detected by the IR sensors. In this scenario, the ultrasonic sensor is not used.

The ultrasonic sensor is only used when there are vehicles on more than one lane. This is to determine the priority with which the system clears the traffic on the lanes. When vehicles are detected on more than one

lane by the IR sensors on each lane, the system must check for the lane with the most vehicle density or more vehicle count. The ultrasonic sensor is placed at a point away from the traffic lights where it can detect the number of vehicles on a particular lane. When there are more vehicles on a particular lane, they tend to get piled up one behind the other. The closer the vehicles are to the ultrasonic sensor, which is placed away from the traffic signals, the more crowded is the lane. By using this logic, the system determines the lane with more traffic. The ultrasonic sensors of each lane simultaneously determine the distance between the sensor and the vehicle closest to the sensor. The closer the vehicle, the more congested is the lane. Hence, the lane which gets a smaller distance value from the ultrasonic sensor is the lane which will get cleared by the control system. This lane is changed from red to green. Since all ultrasonic sensors detect simultaneously, the system obtains the priority with which the lanes will be cleared. Once any particular lane is changed from red to green, the lane with the second-highest traffic density will be changed from red to yellow. This is to indicate which lane will be cleared next. The entire cycle is carried out for as long as the infrared sensor detects vehicles on the lanes.

## Pseudocode:

**# Define constants for pin assignments**
trig1, echo1, trig2, echo2, trig3, echo3, IR1, IR2, IR3, R1, Y1, G1, R2, Y2, G2, R3, Y3, G3

**# Initialize variables for sensor readings and distances**
duration1, duration2, duration3, distance1, distance2, distance3, sensorStatus1, sensorStatus2, sensorStatus3

**# Setup function**
setup():
  Initialize serial communication
  Set pin modes for sensors and traffic lights

```
# Loop function
loop():
   AllLightsON()  # Turn on all traffic lights initially

   # Read infrared sensors
   sensorStatus1 = readSensor(IR1)
   sensorStatus2 = readSensor(IR2)
   sensorStatus3 = readSensor(IR3)

   # Read ultrasonic sensors and calculate distances
   distance1 = measureDistance(trig1, echo1)
   distance2 = measureDistance(trig2, echo2)
   distance3 = measureDistance(trig3, echo3)

   # Traffic control logic based on sensor readings and distances
   if noVehiclesDetected():
      AllLightsON()  # Keep all lights on for 7 seconds

   # Check if there's a vehicle in only one lane
   else if vehicleInOneLane():
      ActivateGreenLightForSingleLane()

   # Check if there are vehicles in any two lanes
   else if vehiclesInTwoLanes():
      ControlLightsForTwoLanes()

   # Check if there are vehicles in all three lanes
   else if vehiclesInAllLanes():
      ControlLightsForAllLanes()

# Utility functions
AllLightsON():
   Turn off all lights except the red ones

readSensor(sensor):
   return digitalRead(sensor)
```

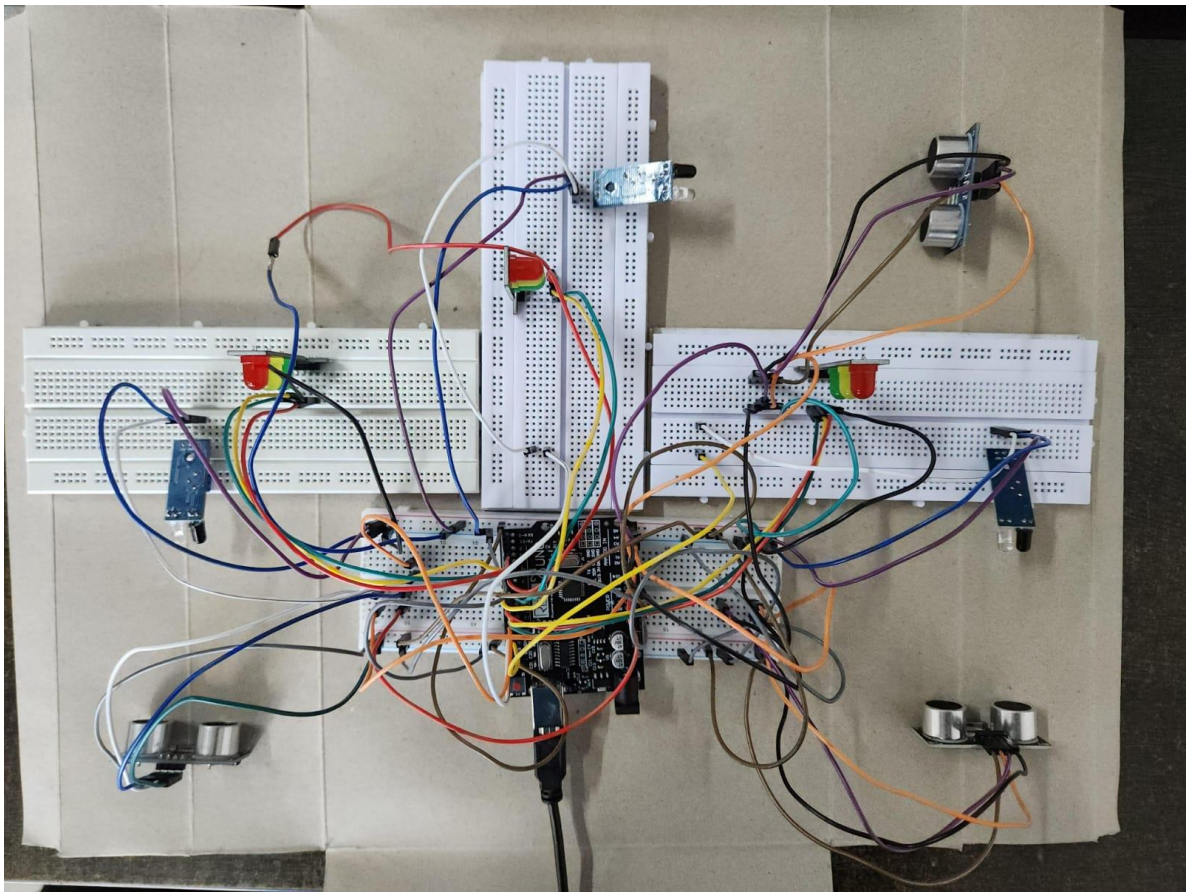
```
measureDistance(trigPin, echoPin):
   Send trigger signal to ultrasonic sensor
   Measure duration using pulseIn()
   Calculate distance based on duration
   return distance

noVehiclesDetected():
   return (sensorStatus1 == 1 and sensorStatus2 == 1 and sensorStatus3 == 1)

vehicleInOneLane():
   return ((sensorStatus1 == 0 and sensorStatus2 == 1 and sensorStatus3 == 1) or
         (sensorStatus1 == 1 and sensorStatus2 == 0 and sensorStatus3 == 1) or
         (sensorStatus1 == 1 and sensorStatus2 == 1 and sensorStatus3 == 0))

vehiclesInTwoLanes():
   return ((sensorStatus1 == 0 and sensorStatus2 == 0 and sensorStatus3 == 1) or
         (sensorStatus1 == 0 and sensorStatus2 == 1 and sensorStatus3 == 0) or
         (sensorStatus1 == 1 and sensorStatus2 == 0 and sensorStatus3 == 0))

vehiclesInAllLanes():
   return (sensorStatus1 == 0 and sensorStatus2 == 0 and sensorStatus3 == 0)

ActivateGreenLightForSingleLane():
   Activate green light for the respective lane and delay 7 seconds

ControlLightsForTwoLanes():
   Control lights for two lanes based on distances and vehicle positions with
appropriate delays

ControlLightsForAllLanes():
   Control lights for all lanes based on distances and vehicle positions with appropriate
delays
```

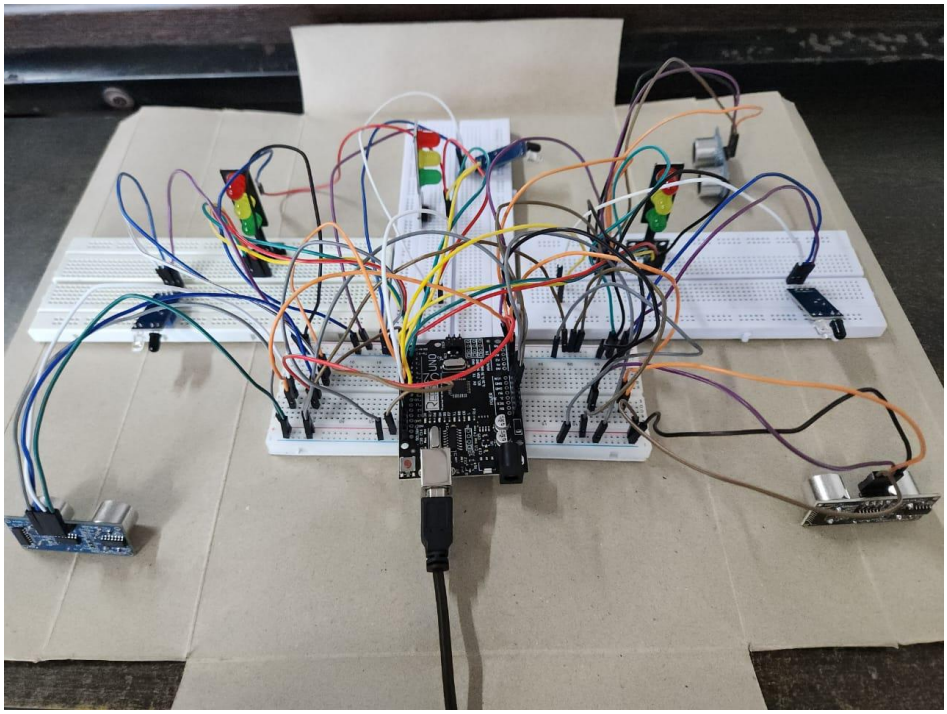
## CONCEPTS LEARNED:

Through this project, we learnt important concepts related to microcontroller programming, sensor integration, traffic flow analysis, and remote-control applications. Basic Arduino programming was studied. Writing code for the Arduino board ensured the learning of declaration of input and output pins, finding the addressing modes, the baud rate at which the microcontroller operates, basic analogue and digital pin programming, and port programming. We learnt how to incorporate the sensors like infrared sensor and ultrasonic distance sensor. We studied the patterns of traffic control which exist in the present around the world. Understanding the dynamics of traffic control systems and the importance of adaptability in real-time scenarios is a key takeaway.

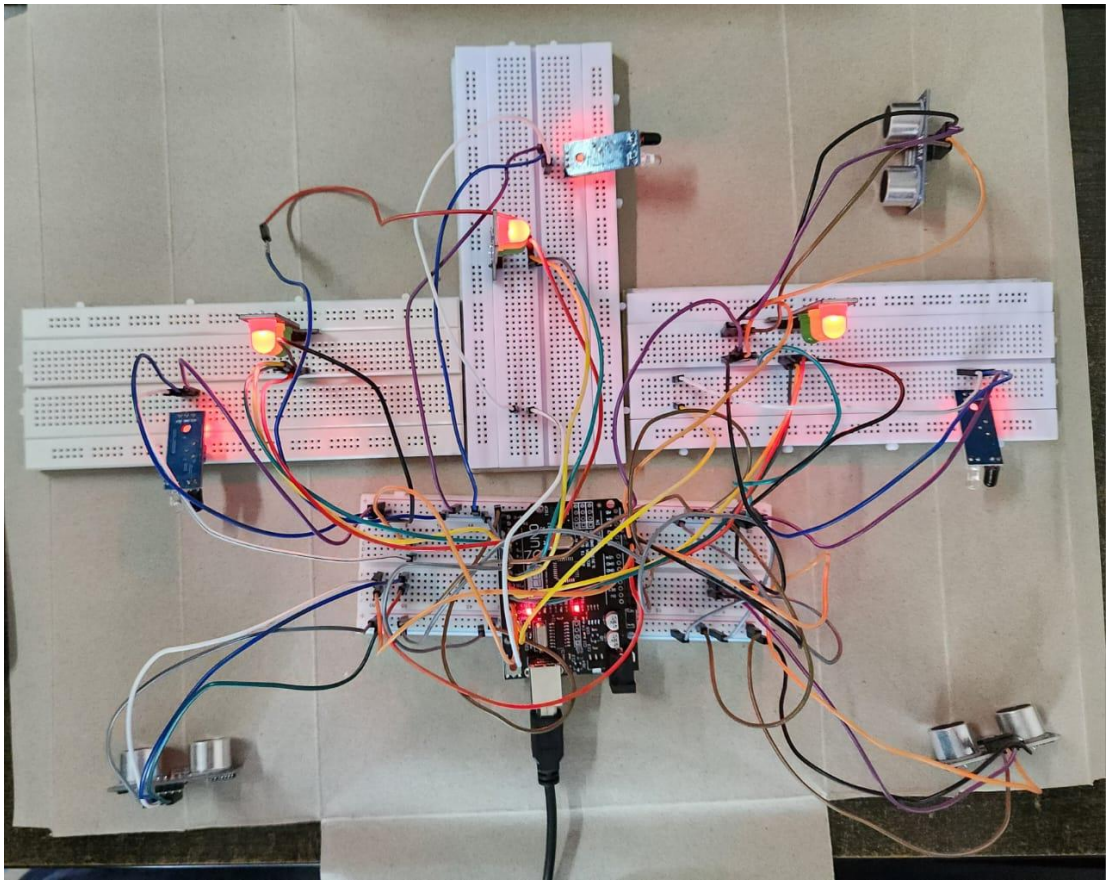
## IMPLEMENTATION:

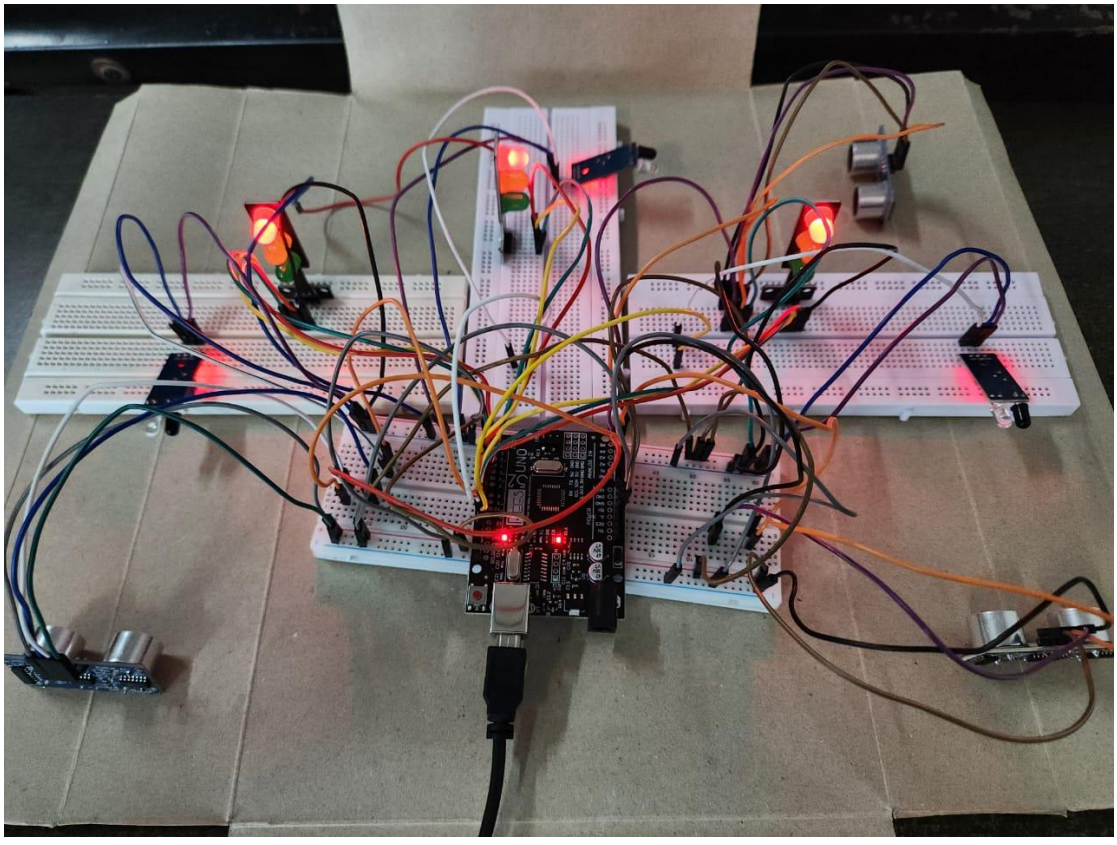
Overall hardware connection



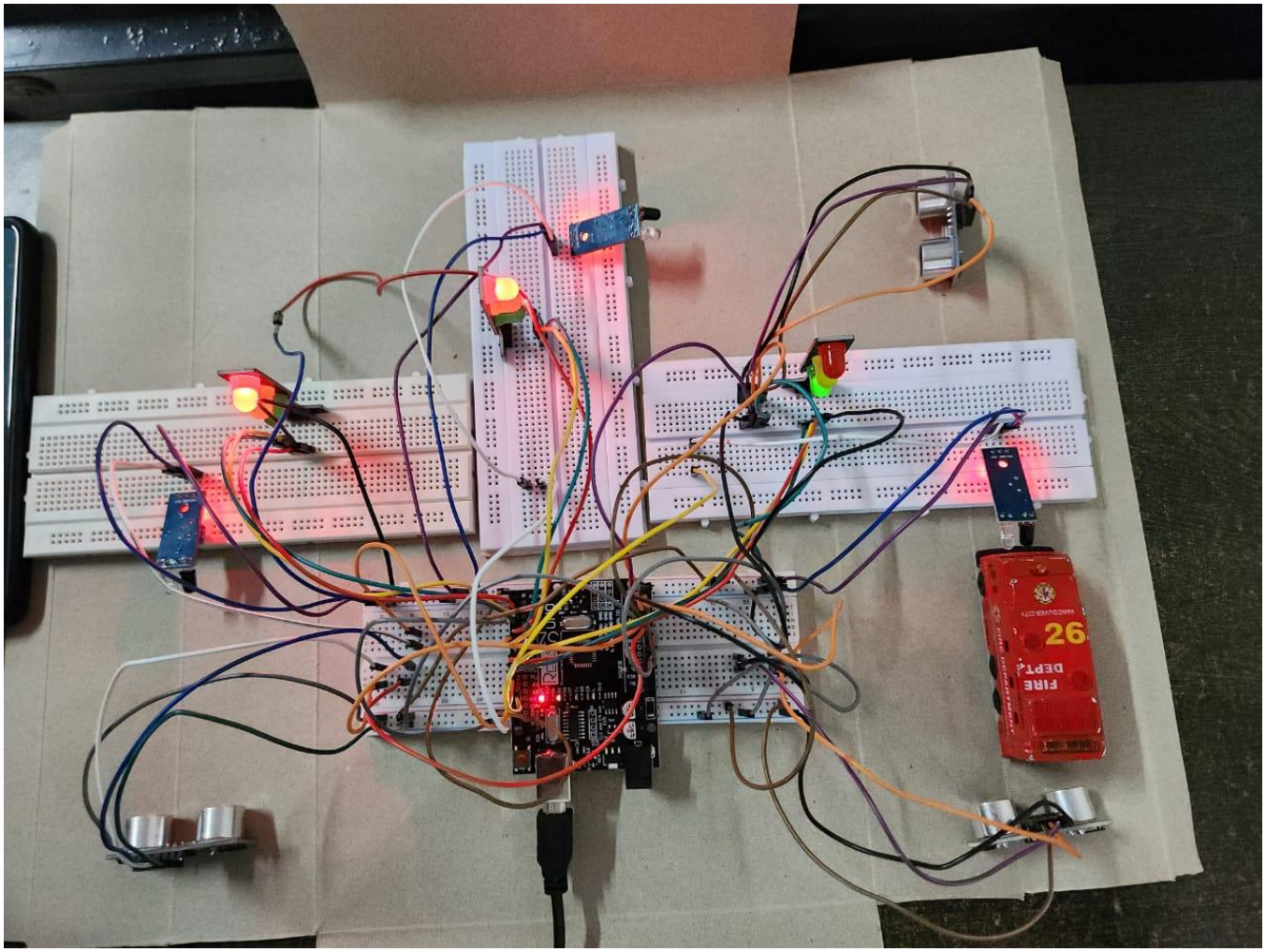


“Default Case”

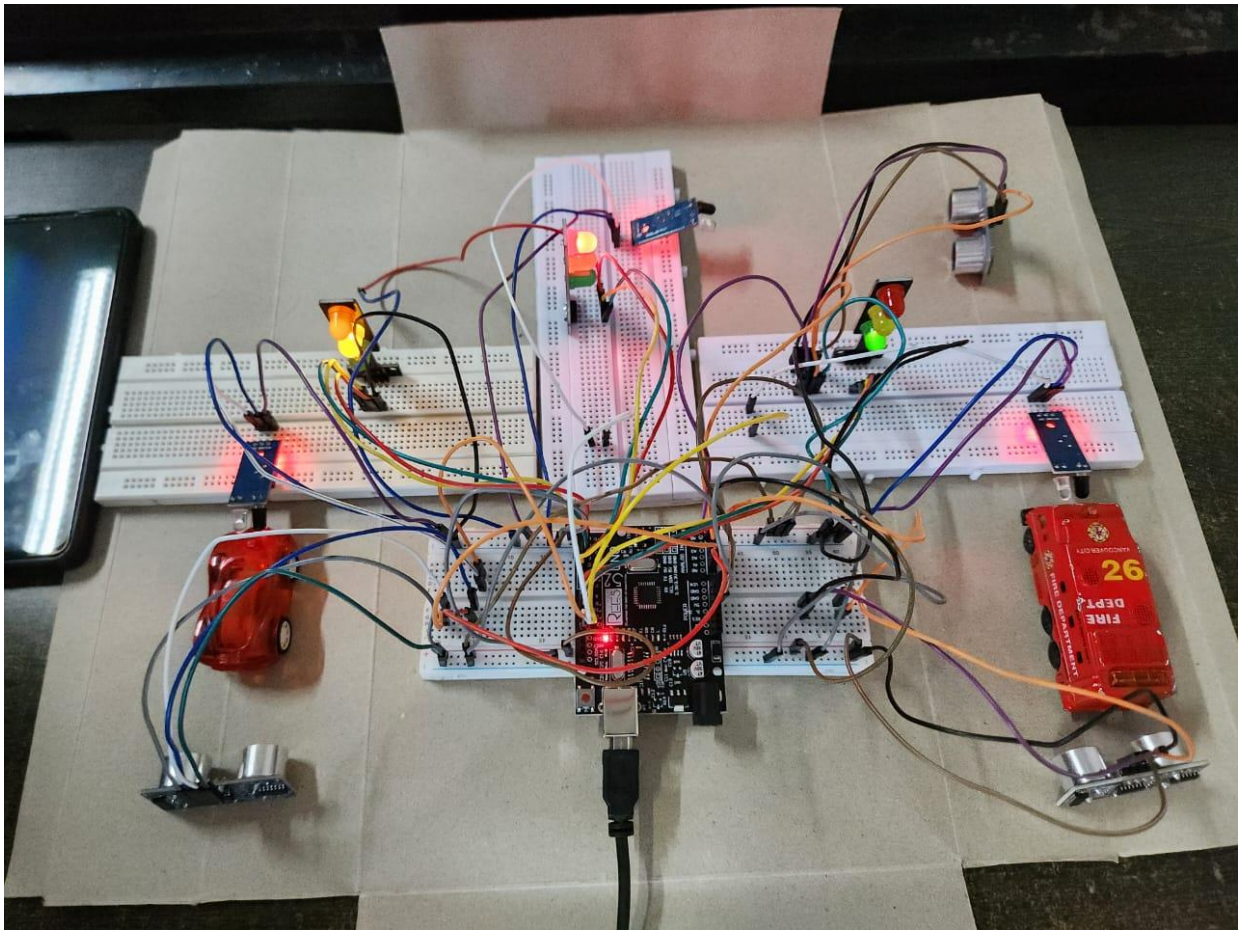


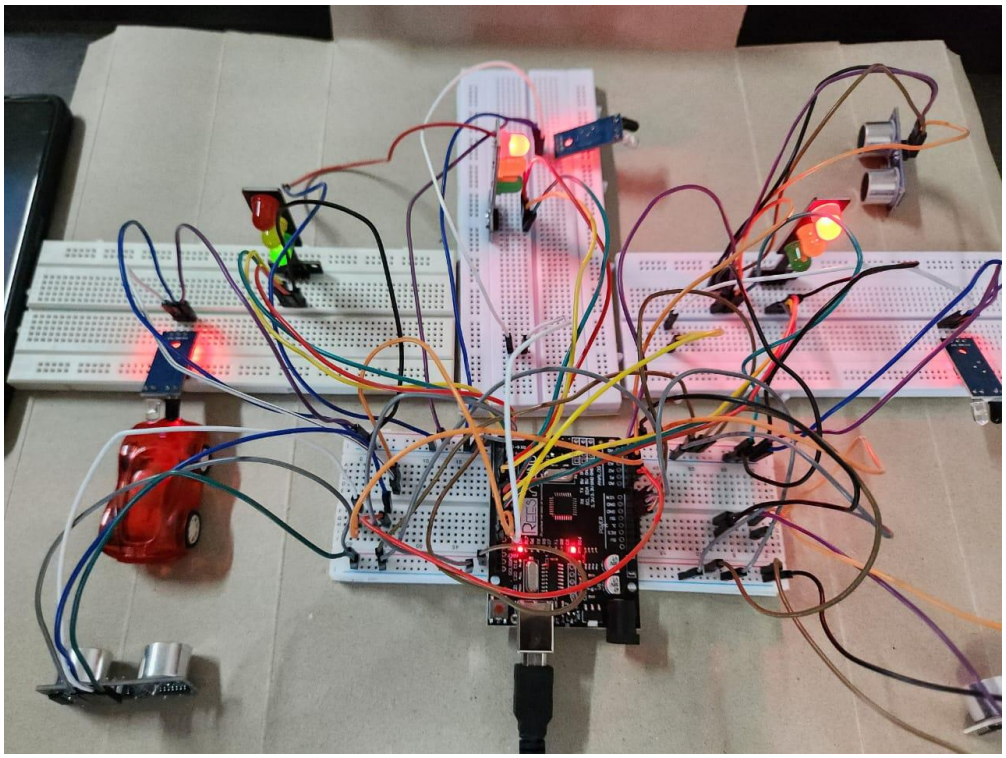


Vehicle in single lane

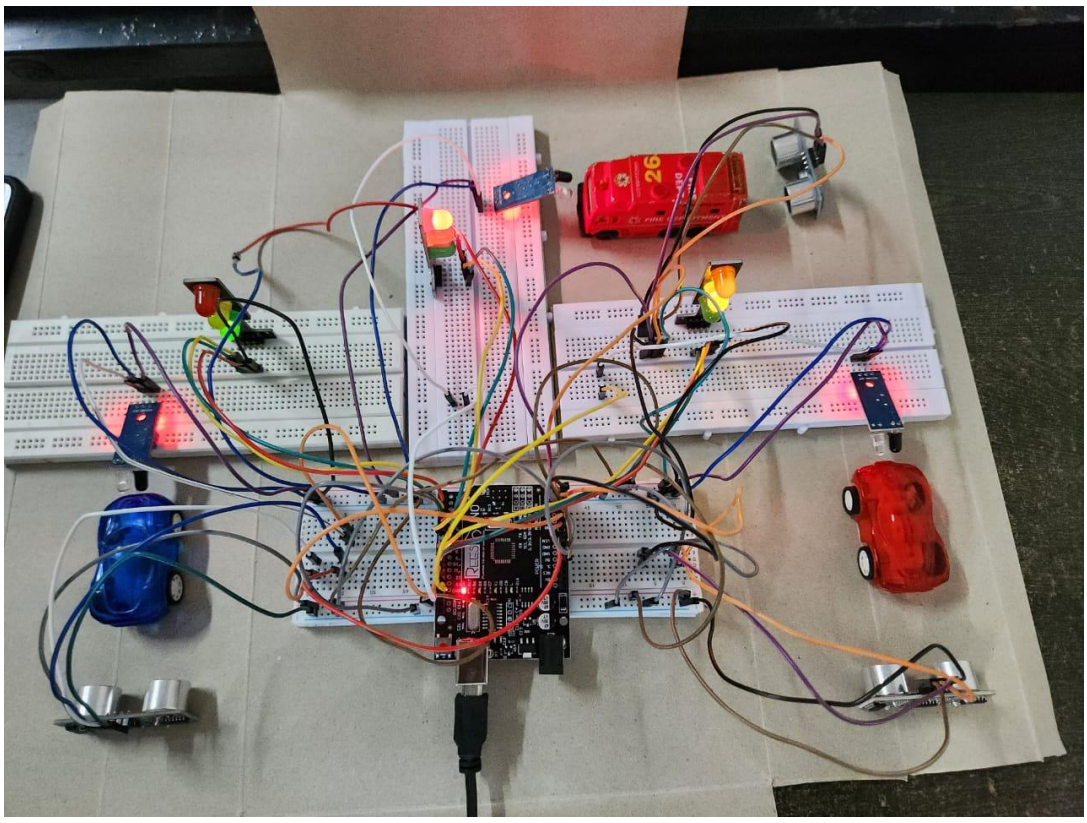


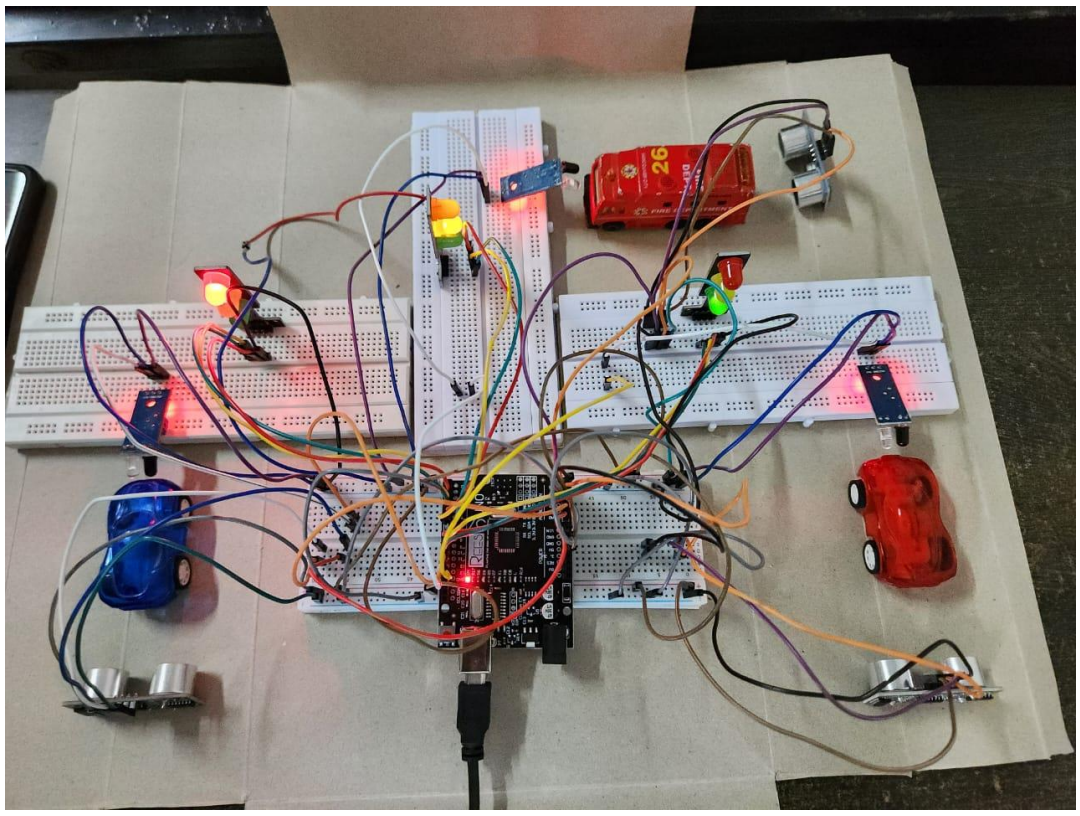
Vehicles in two lanes

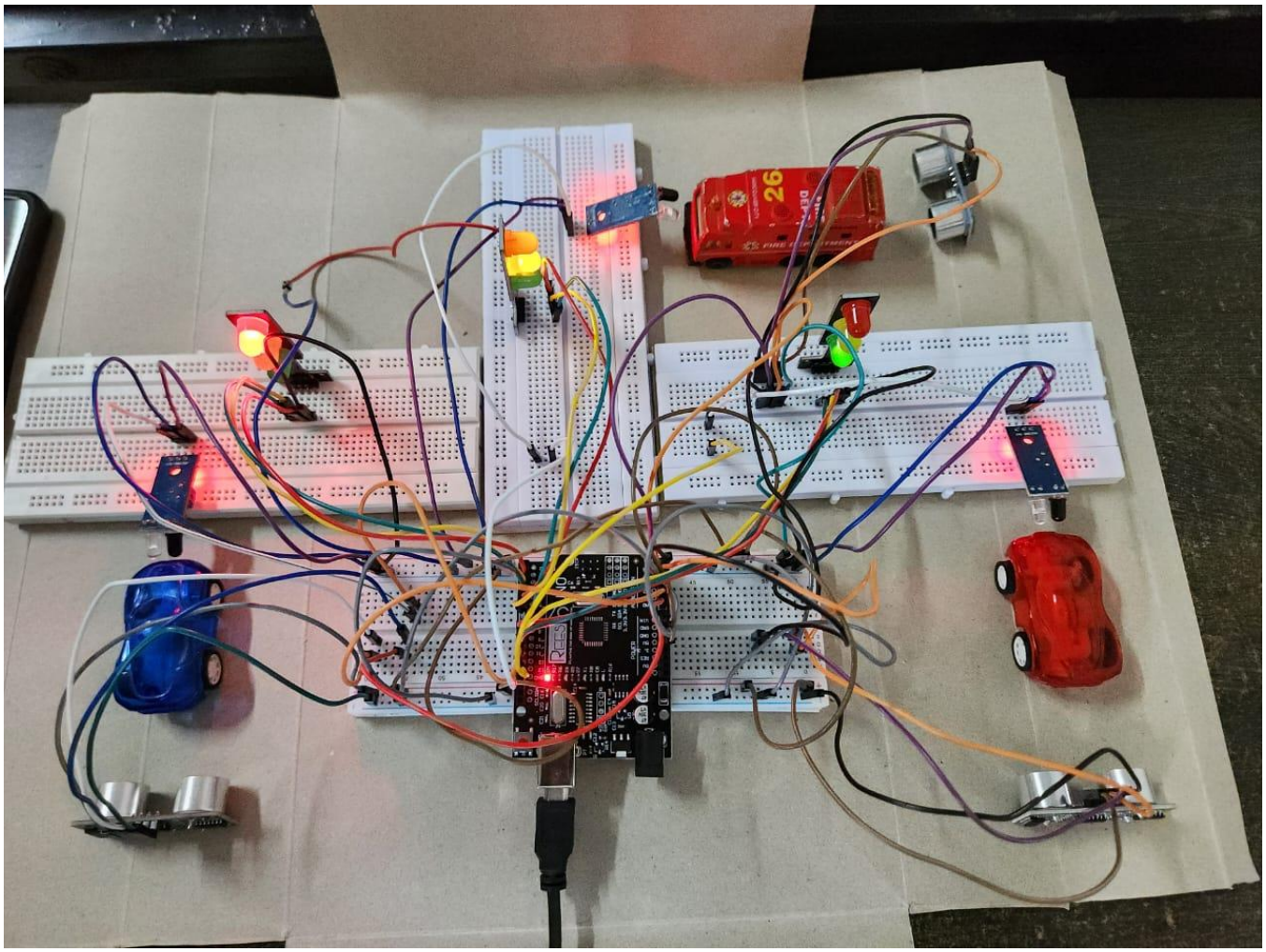


Vehicles in all lanes







## Demonstration Video Link:

https://drive.google.com/file/d/13BU0WLy2VVAmLYoFu5scEhayI3AYnVvm/view?usp=sharing

## CHALLENGES FACED:

During the course of this project, we did indeed face difficulties. Implementation of the traffic control system logic into a code which can be implemented by the Arduino board was a challenging process as the syntax and functions of the Arduino board was unfamiliar to us. Setting up a proper layout for the project was complicated as we had to make sure we were efficient with the resources we utilized, as well as making sure that the setup was not too congested due to the large number of connecting jumper wires. Placement of both IR and ultrasonic sensors in the lanes was vital as we had to make sure the vehicles can be properly placed within the range of the sensors. Further fine-tuning of the sensitivity of IR sensors and ultrasonic sensor was highly necessary for accurate traffic density measurement.

## APPLICATIONS:

Our project's applications extend beyond regular traffic control. It can be utilized in various scenarios, including:

Emergency vehicle prioritization

Adaptive traffic control for special events

Integration with smart city systems for efficient traffic management

## CONCLUSION:

The density-based traffic control system presented in this project offers a dynamic solution to traffic congestion issues. The ability to adapt signal timings based on real-time traffic density can turn out to be a real game changer in the field of traffic management.

## PROGRAM:

```
// Define pin assignments for ultrasonic sensors, IR sensors, and traffic lights
const int trig1 = A0;
const int echo1 = A1;
const int trig2 = A2;
const int echo2 = A3;
const int trig3 = A4;
const int echo3 = A5;
const int IR1 = 11;
const int IR2 = 12;
const int IR3 = 13;
const int R1 = 2;
const int Y1 = 3;
const int G1 = 4;
const int R2 = 5;
const int Y2 = 6;
const int G2 = 7;
const int R3 = 8;
const int Y3 = 9;
const int G3 = 10;

// Define variables for sensor readings and distances
float duration1, duration2, duration3, distance1, distance2, distance3;
int sensorStatus1, sensorStatus2, sensorStatus3;

void setup() {
  Serial.begin(9600); // Initialize serial communication for debugging

  // Set pin modes for sensors and traffic lights
  pinMode(trig1, OUTPUT);
  pinMode(echo1, INPUT);
  pinMode(trig2, OUTPUT);
  pinMode(echo2, INPUT);
  pinMode(trig3, OUTPUT);
  pinMode(echo3, INPUT);
  pinMode(R1, OUTPUT);
  pinMode(Y1, OUTPUT);
  pinMode(G1, OUTPUT);
  pinMode(R2, OUTPUT);
```

```
  pinMode(Y2, OUTPUT);
  pinMode(G2, OUTPUT);
  pinMode(R3, OUTPUT);
  pinMode(Y3, OUTPUT);
  pinMode(G3, OUTPUT);
  pinMode(IR1, INPUT);
  pinMode(IR2, INPUT);
  pinMode(IR3, INPUT);
}

void loop() {
  AllLightsON(); // Turn off all traffic lights

  // Read infrared sensor status
  sensorStatus1 = digitalRead(IR1);
  sensorStatus2 = digitalRead(IR2);
  sensorStatus3 = digitalRead(IR3);
  Serial.println(sensorStatus1); // Print IR sensor readings for debugging
  Serial.println(sensorStatus2);
  Serial.println(sensorStatus3);

  // Measure distances using ultrasonic sensors
  // Lane 1
  digitalWrite(trig1, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig1, LOW);
  duration1 = pulseIn(echo1, HIGH);
  distance1 = 0.017 * duration1;

  // Lane 2
  digitalWrite(trig2, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig2, LOW);
  duration2 = pulseIn(echo2, HIGH);
  distance2 = 0.017 * duration2;

  // Lane 3
  digitalWrite(trig3, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig3, LOW);
```

```
duration3 = pulseIn(echo3, HIGH);
distance3 = 0.017 * duration3;

// Traffic control logic based on sensor readings and distances
// Check if no vehicles in any lane
if (sensorStatus1 == sensorStatus2 == sensorStatus3 == 1) {
  AllLightsON(); // Turn on all lights
  delay(7000);
}

// Check if vehicle in only one lane
if (sensorStatus1 == 0 && sensorStatus2 == 1 && sensorStatus3 == 1) {
  // Activate green light for lane 1
  digitalWrite(R1, LOW);
  digitalWrite(G1, HIGH);
  delay(7000);
}

// Check if vehicle in only one lane
if (sensorStatus1 == 1 && sensorStatus2 == 0 && sensorStatus3 == 1) {
  // Activate green light for lane 2
  digitalWrite(R2, LOW);
  digitalWrite(G2, HIGH);
  delay(7000);
}

// Check if vehicle in only one lane
if (sensorStatus1 == 1 && sensorStatus2 == 1 && sensorStatus3 == 0) {
  // Activate green light for lane 3
  digitalWrite(R3, LOW);
  digitalWrite(G3, HIGH);
  delay(7000);
}

// Check if vehicle is present in any 2 lanes
if (sensorStatus1 == 0 && sensorStatus2 == 0 && sensorStatus3 == 1) {
  if (distance2 > distance1) {
    digitalWrite(R1, LOW);
    digitalWrite(G1, HIGH);
    digitalWrite(R2, LOW);
```

```
    digitalWrite(Y2, HIGH);
    delay(7000);
    digitalWrite(G1, LOW);
    digitalWrite(R1, HIGH);
    digitalWrite(Y2, LOW);
    digitalWrite(G2, HIGH);
    delay(7000);
  }
  if (distance1 > distance2) {
    digitalWrite(R2, LOW);
    digitalWrite(G2, HIGH);
    digitalWrite(R1, LOW);
    digitalWrite(Y1, HIGH);
    delay(7000);
    digitalWrite(G2, LOW);
    digitalWrite(R2, HIGH);
    digitalWrite(Y1, LOW);
    digitalWrite(G1, HIGH);
    delay(7000);
  }
}

// Check if vehicle is present in any 2 lanes
if (sensorStatus1 == 0 && sensorStatus3 == 0 && sensorStatus2 == 1) {
  if (distance3 > distance1) {
    digitalWrite(R1, LOW);
    digitalWrite(G1, HIGH);
    digitalWrite(R3, LOW);
    digitalWrite(Y3, HIGH);
    delay(7000);
    digitalWrite(G1, LOW);
    digitalWrite(R1, HIGH);
    digitalWrite(Y3, LOW);
    digitalWrite(G3, HIGH);
    delay(7000);
  }
  if (distance1 > distance3) {
    digitalWrite(R3, LOW);
    digitalWrite(G3, HIGH);
    digitalWrite(R1, LOW);
```

```
    digitalWrite(Y1, HIGH);
    delay(7000);
    digitalWrite(G3, LOW);
    digitalWrite(R3, HIGH);
    digitalWrite(Y1, LOW);
    digitalWrite(G1, HIGH);
    delay(7000);
  }
}

// Check if vehicle is present in any 2 lanes
if (sensorStatus2 == 0 && sensorStatus3 == 0 && sensorStatus1 == 1) {
  if (distance2 > distance3) {
    digitalWrite(R3, LOW);
    digitalWrite(G3, HIGH);
    digitalWrite(R2, LOW);
    digitalWrite(Y2, HIGH);
    delay(7000);
    digitalWrite(G3, LOW);
    digitalWrite(R3, HIGH);
    digitalWrite(Y2, LOW);
    digitalWrite(G2, HIGH);
    delay(7000);
  }
  if (distance3 > distance2) {
    digitalWrite(R2, LOW);
    digitalWrite(G2, HIGH);
    digitalWrite(R3, LOW);
    digitalWrite(Y3, HIGH);
    delay(7000);
    digitalWrite(G2, LOW);
    digitalWrite(R2, HIGH);
    digitalWrite(Y3, LOW);
    digitalWrite(G3, HIGH);
    delay(7000);
  }
}

// Check if all lanes have vehicles
if (sensorStatus1 == 0 && sensorStatus2 == 0 && sensorStatus3 == 0) {
```

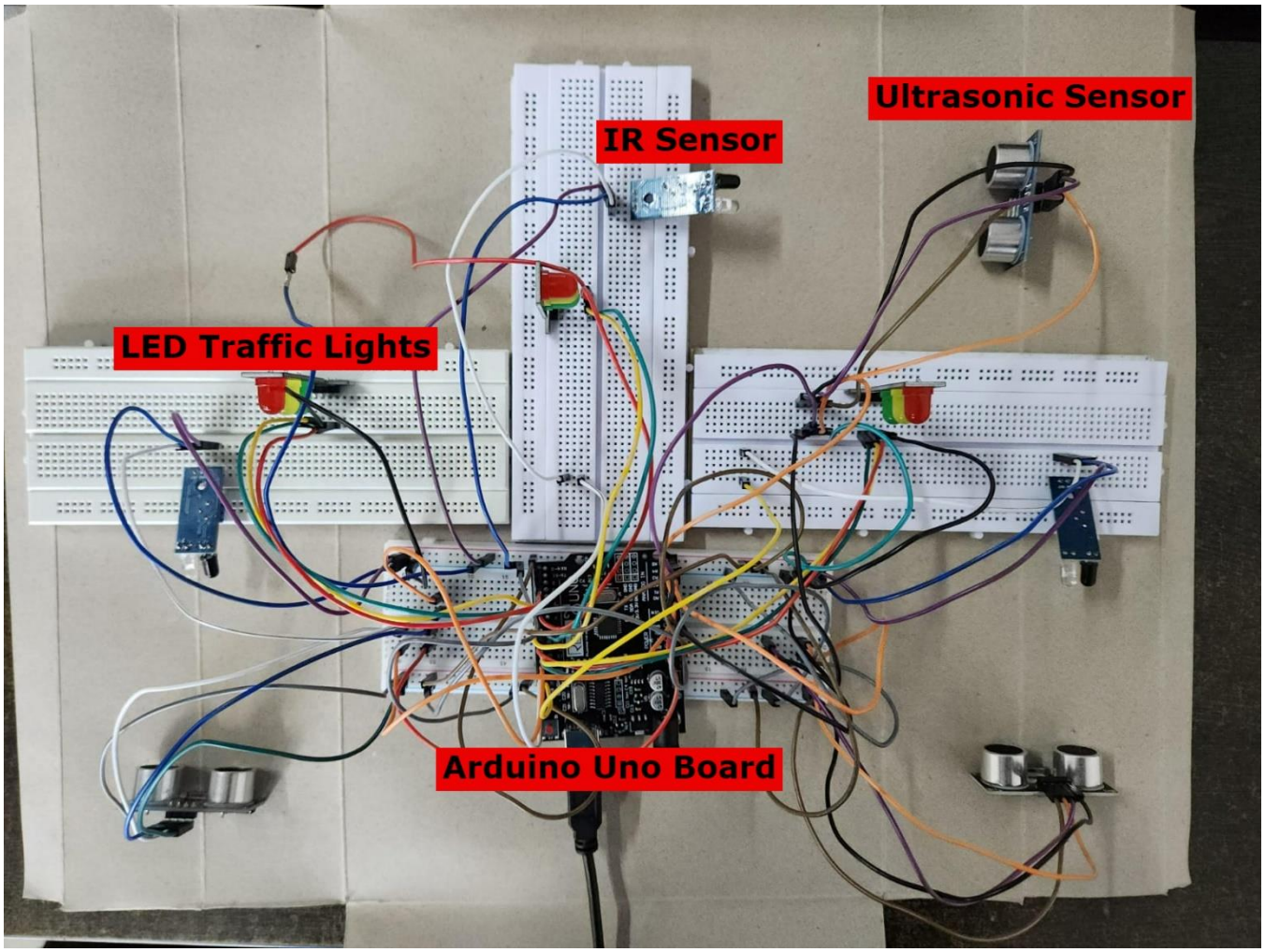
```
  if (distance1 >= distance2 && distance2 >= distance3) {
    digitalWrite(R3, LOW);
    digitalWrite(G3, HIGH);
    digitalWrite(R2, LOW);
    digitalWrite(Y2, HIGH);
    delay(7000);
    digitalWrite(G3, LOW);
    digitalWrite(R3, HIGH);
    digitalWrite(Y2, LOW);
    digitalWrite(G2, HIGH);
    digitalWrite(R1, LOW);
    digitalWrite(Y1, HIGH);
    delay(7000);
    digitalWrite(G2, LOW);
    digitalWrite(R2, HIGH);
    digitalWrite(Y1, LOW);
    digitalWrite(G1, HIGH);
    delay(7000);
  }
// Check if all lanes have vehicles
if (sensorStatus1 == 0 && sensorStatus2 == 0 && sensorStatus3 == 0) {
  if (distance1 >= distance2 && distance2 >= distance3) {
    digitalWrite(R3, LOW);
    digitalWrite(G3, HIGH);
    digitalWrite(R2, LOW);
    digitalWrite(Y2, HIGH);
    delay(7000);
    digitalWrite(G3, LOW);
    digitalWrite(R3, HIGH);
    digitalWrite(Y2, LOW);
    digitalWrite(G2, HIGH);
    digitalWrite(R1, LOW);
    digitalWrite(Y1, HIGH);
    delay(7000);
    digitalWrite(G2, LOW);
    digitalWrite(R2, HIGH);
    digitalWrite(Y1, LOW);
    digitalWrite(G1, HIGH);
    delay(7000);
  }
```

```
if (distance3 >= distance2 && distance2 >= distance1) {
  digitalWrite(R1, LOW);
  digitalWrite(G1, HIGH);
  digitalWrite(R2, LOW);
  digitalWrite(Y2, HIGH);
  delay(7000);
  digitalWrite(G1, LOW);
  digitalWrite(R1, HIGH);
  digitalWrite(Y2, LOW);
  digitalWrite(G2, HIGH);
  digitalWrite(R3, LOW);
  digitalWrite(Y3, HIGH);
  delay(7000);
  digitalWrite(G2, LOW);
  digitalWrite(R2, HIGH);
  digitalWrite(Y3, LOW);
  digitalWrite(G3, HIGH);
  delay(7000);
}
if (distance2 >= distance3 && distance3 > distance1) {
  digitalWrite(R1, LOW);
  digitalWrite(G1, HIGH);
  digitalWrite(R3, LOW);
  digitalWrite(Y3, HIGH);
  delay(7000);
  digitalWrite(G1, LOW);
  digitalWrite(R1, HIGH);
  digitalWrite(Y3, LOW);
  digitalWrite(G3, HIGH);
  digitalWrite(R2, LOW);
  digitalWrite(Y2, HIGH);
  delay(7000);
  digitalWrite(G3, LOW);
  digitalWrite(R3, HIGH);
  digitalWrite(Y2, LOW);
  digitalWrite(G2, HIGH);
  delay(7000);
}
if (distance2 >= distance1 && distance1 >= distance3) {
  digitalWrite(R3, LOW);
```

```
  digitalWrite(G3, HIGH);
  digitalWrite(R1, LOW);
  digitalWrite(Y1, HIGH);
  delay(7000);
  digitalWrite(G3, LOW);
  digitalWrite(R3, HIGH);
  digitalWrite(Y1, LOW);
  digitalWrite(G1, HIGH);
  digitalWrite(R2, LOW);
  digitalWrite(Y2, HIGH);
  delay(7000);
  digitalWrite(G1, LOW);
  digitalWrite(R1, HIGH);
  digitalWrite(Y2, LOW);
  digitalWrite(G2, HIGH);
  delay(7000);
}
if (distance3 >= distance1 && distance1 >= distance2) {
  digitalWrite(R2, LOW);
  digitalWrite(G2, HIGH);
  digitalWrite(R1, LOW);
  digitalWrite(Y1, HIGH);
  delay(7000);
  digitalWrite(G2, LOW);
  digitalWrite(R2, HIGH);
  digitalWrite(Y1, LOW);
  digitalWrite(G1, HIGH);
  digitalWrite(R3, LOW);
  digitalWrite(Y3, HIGH);
  delay(7000);
  digitalWrite(G1, LOW);
  digitalWrite(R1, HIGH);
  digitalWrite(Y3, LOW);
  digitalWrite(G3, HIGH);
  delay(7000);
}
if (distance1 >= distance3 && distance3 >= distance2) {
  digitalWrite(R2, LOW);
  digitalWrite(G2, HIGH);
  digitalWrite(R3, LOW);
```

```
      digitalWrite(Y3, HIGH);
      delay(7000);
      digitalWrite(G2, LOW);
      digitalWrite(R2, HIGH);
      digitalWrite(Y3, LOW);
      digitalWrite(G3, HIGH);
      digitalWrite(R1, LOW);
      digitalWrite(Y1, HIGH);
      delay(7000);
      digitalWrite(G3, LOW);
      digitalWrite(R3, HIGH);
      digitalWrite(Y1, LOW);
      digitalWrite(G1, HIGH);
      delay(7000);
    }
  }
}

void AllLightsON() {
  // Turn off all lights except red
  digitalWrite(R1, HIGH);
  digitalWrite(R2, HIGH);
  digitalWrite(R3, HIGH);
  digitalWrite(G1, LOW);
  digitalWrite(G2, LOW);
  digitalWrite(G3, LOW);
  digitalWrite(Y1, LOW);
  digitalWrite(Y2, LOW);
  digitalWrite(Y3, LOW);
}
```

## HARDWARE:


Ultrasonic Sensor
IR Sensor
LED Traffic Lights
Arduino Uno Board