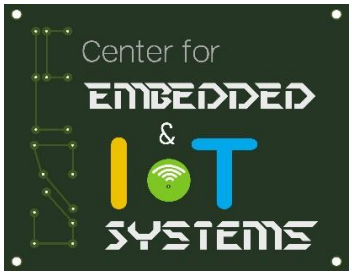
TRAFFIC DENSITY CONTROL MANAGEMENT



A project report submitted in partial fulfilment of requirement for the course

On

SMART SYSTEM DESIGN

By

E. Sai Krishna	**(2103A52014)**
E.Vinay Chandra	**(2103A52015)**
B.Shivaram	**(2103A52003)**
E. Rishitha	**(2105A21030)**

Under the guidance of

Dr. V.Malathy, Asst. Prof., Department of ECE

&

Mr. Rajeshwar Rao Arabelli

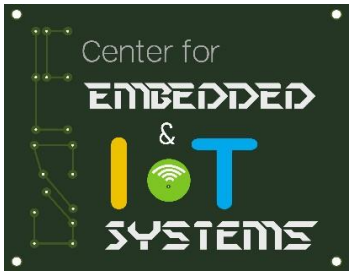
Asst. Prof. & Director, Centre for Embedded Systems and IoT,

Department of ECE

Department of Electronics and Communication Engineering

Center for Embedded Systems and Internet of things

SR UNIVERSITY



CERTIFICATE

This is to certify that the course project entitled **"Traffic Density control Management"** is the bonafide work carried out by**, Sai Krishna (2103A52014), Vinay chandra(2103A52015), Shivaram(2103A52003), Rishitha(2105A21030)** in the partial fulfilment of the requirement for the award of course **Smart System Design** during the academic year 2021-2022 under our guidance and Supervision.

Dr. V.Malathy

Asst. Prof., Department of ECE

&

Mr. Rajeshwar Rao Arabelli

Asst. Prof. & Director, Centre for Embedded Systems and IoT,

Department of ECE

Topic

Declaration

Certificate

Abstract

List of figures

ABSTRACT

Now-a-days, controlling the traffic becomes major issue because of rapid increase in automobiles and also because of large time delays between traffic lights so, in order to rectify this problem, we will go for a density based traffic lights system. This article explains you to control the traffic based on density.

This system defines another way to optimize the use of energy and also to overcome the problem of traffic jams. By sensing the number of vehicles on each side of a junction, the time for which red light of the traffic signal glows can be controlled accordingly. This project achieves this by using LEDs as the traffic lights for each side of the junction and using UR sensors for each side to sense the number of vehicles.

List of Figures

1. INTRODUCTION

Our course project is on traffic density control management. traffic congestion is a serious issue for urban cities. From city roads to highways, a lot of traffic problems occur everywhere in today's world, because of exponentially increase in the number of vehicles, the traffic management system and road capacity are not efficiently compatible with vehicles traveling on them. These frequent traffic problems like traffic jams have led to the need for an efficient traffic management system.

Principle of Project :

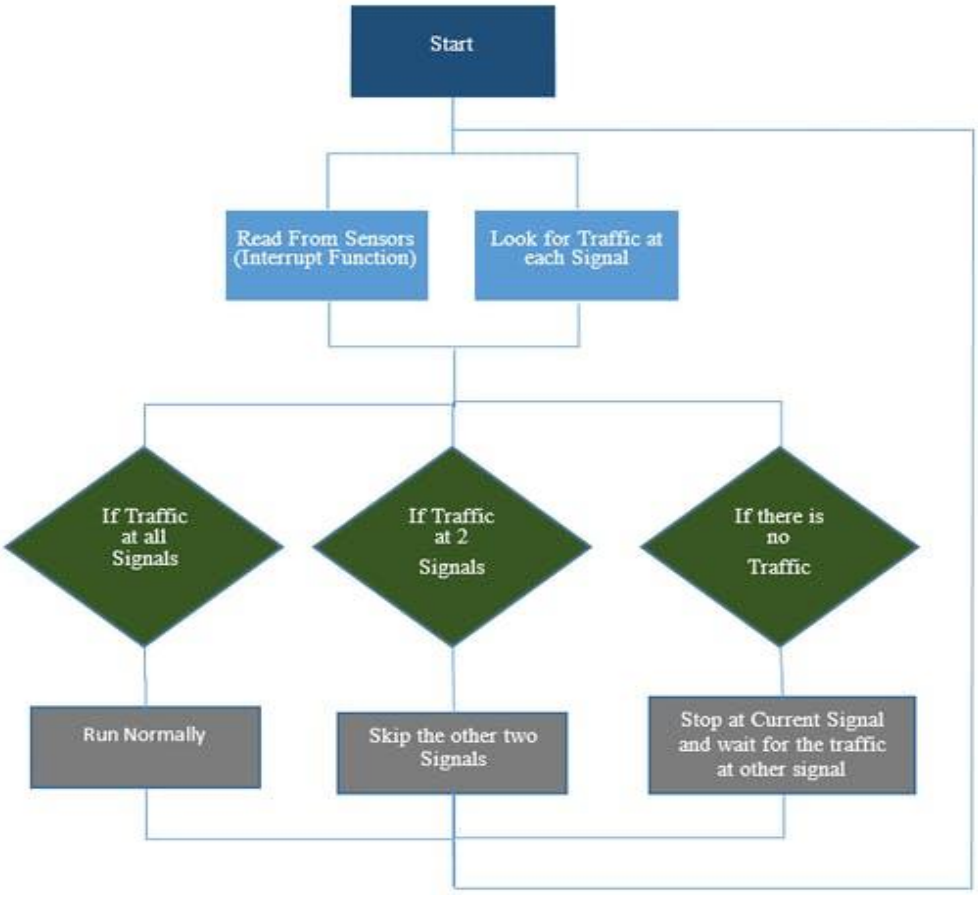
we built an Arduino Traffic Light Controller and in this post, you are going to learn about how to make an density based traffic light controller using Arduino. The main purpose of this project is, if there will be no traffic on the other signal, one shouldn't wait for that signal. The system will skip that signal and will move on the next signal automatically.

The main task was to avoid use of delay so, we have to control signals which requires the use of delay function.

2. SYSTEM DESCRIPTION

Arduino is the main part of this project and it will be used to read from ultrasonic sensor HC-SR04 and calculate the distance. This distance will tell us if any vehicle is near the signal or not and according to that the traffic signals will be controlled.The main task was to avoid use of delay because we have to continuously read from the ultrasonic sensors and also at the same time, we have to control signals which requires the use of delay function. So we have used the timerone library which is used to repetitively measure a period of time in microseconds and at the end of each period, an interrupt function will be called. In this function, we will read from the sensors and in the loop function, we will control the traffic signals.

2.1 Flow chart:



This above figure explains about the traffic control management system

3. COMPONENTS REQUIRED

➢ The components you are going to require for this project are as follows:

3.1 Hardware components :

- Arduino Mega 2560
- 4 X HC-SR04 ultrasonic sensors
- 4 X Red LEDs

  4 X Green LEDs

  4 X Yellow LEDs
- 12 X 220ohm resistors
- Jumper cables
- Breadboards

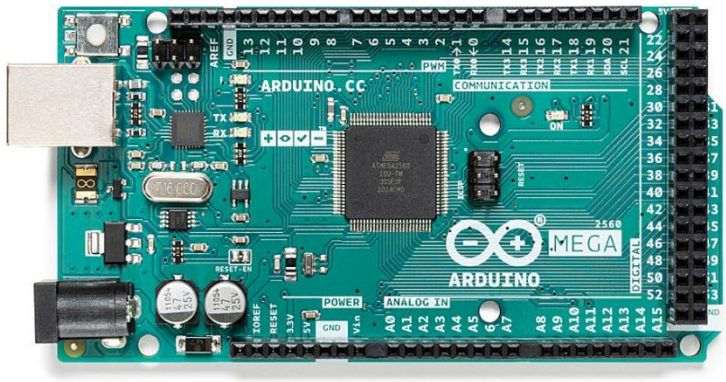
3.2 Software components :

- Arduino IDE

3.3DESCRIPTION OF COMPONENTS

Hardware Tools

1. ***Arduino Mega 2560*** **:**



The "**Arduino Mega 2560**" is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega 2560 board is compatible with most shields designed for the Uno and the former boards Duemilanove or Diecimila.

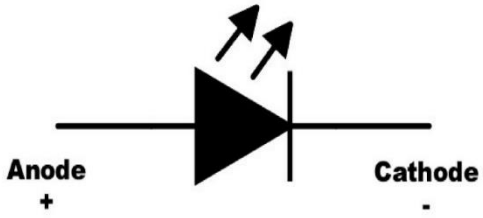
The Mega 2560 can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the GND and Vin pin headers of the POWER connector.

2. *Ultrasonic Distance Sensor - HC-SR04*

This is the**"HC-SR04 *Ultrasonic Distance Sensor"***. This economical sensor provides 2cm to 400cm of non-contact measurement functionality with a ranging accuracy that can reach up to 3mm. Each HC-SR04 module includes an ultrasonic transmitter, a receiver and a control circuit.

There are only four pins that you need to worry about on the HC-SR04: VCC (Power), Trig (Trigger), Echo (Receive), and GND (Ground). You will find this sensor very easy to set up and use for your next range-finding project!

This sensor has additional control circuitry that can prevent inconsistent "bouncy" data depending on the application.



3. LED [*Light Emitting Diode*]:

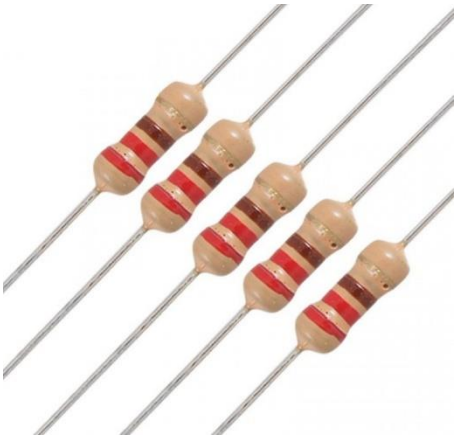
A light releasing diode is an electric component which emits light when the electric current flows through it. Further, it is a light source which is on the basis of semiconductors.

When current passes through the LED, the electrons recombine with holes and emit light in this procedure. It is a particular kind of diode which has the same type of characteristics as the p-n junction diode.

This means that an LED enables the flow of current in its forward direction whereas it blocks the flow in the reverse direction. Further, we build light-emitting diodes through the use of a weak layer of heavily doped semiconductor material.

On the basis of the semiconductor material which we use plus the amount of doping, an LED will emit a coloured light at a specific spectral wavelength when forward biased.

4.220ohm Resistor :

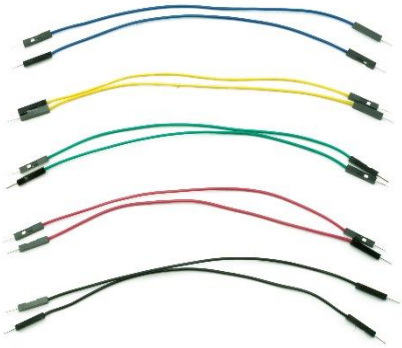


The term "***Resistor***" refers to a device that acts as a two-terminal passive electrical component that is used to limit or regulate the flow of electric current in electrical circuits. And it also allows us to introduce a controlled amount of resistance into an electrical circuit. The most important and commonly used components in an electronic circuit are resistors.

A resistor's main job is to reduce current flow and lower voltage in a specific section of the circuit. It's made up of copper wires that are wrapped around a ceramic rod and coated with insulating paint.

A 220Ω resistor has red, red, and brown stripes in that order. The last stripe represents the tolerance. Gold means ±5%.

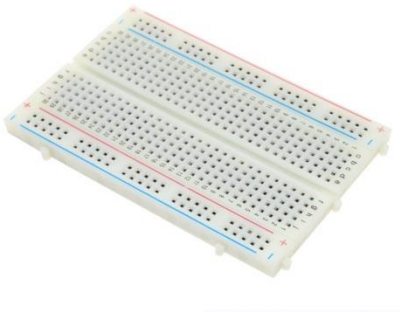
5.Jumper Cables:



A jump wire (also known as jumper, jumper wire, Dupont wire) is an electrical wire, or group of them in a cable, with a connector or pin at each end (or sometimes without them-simply "tinned"), which is normally used to interconnect the components of breadboard or other prototype.

6.Bread Board :

A "***Breadboard*** " is simply a board for prototyping or building circuits on. It allows you to place components and connections on the board to make circuits without soldering. The holes in the breadboard take care of your connections by physically holding onto parts or wires where you put them and electrically connecting them inside the board. The ease of use and speed are great for learning and quick prototyping of simple circuits. More complex circuits and high frequency circuits are less suited to breadboarding. Breadboard circuits are also not ideal for long term use like circuits built on perfboard (protoboard) or PCB (printed circuit board), but they also don't have the soldering (protoboard), or design and manufacturing costs (PCBs).

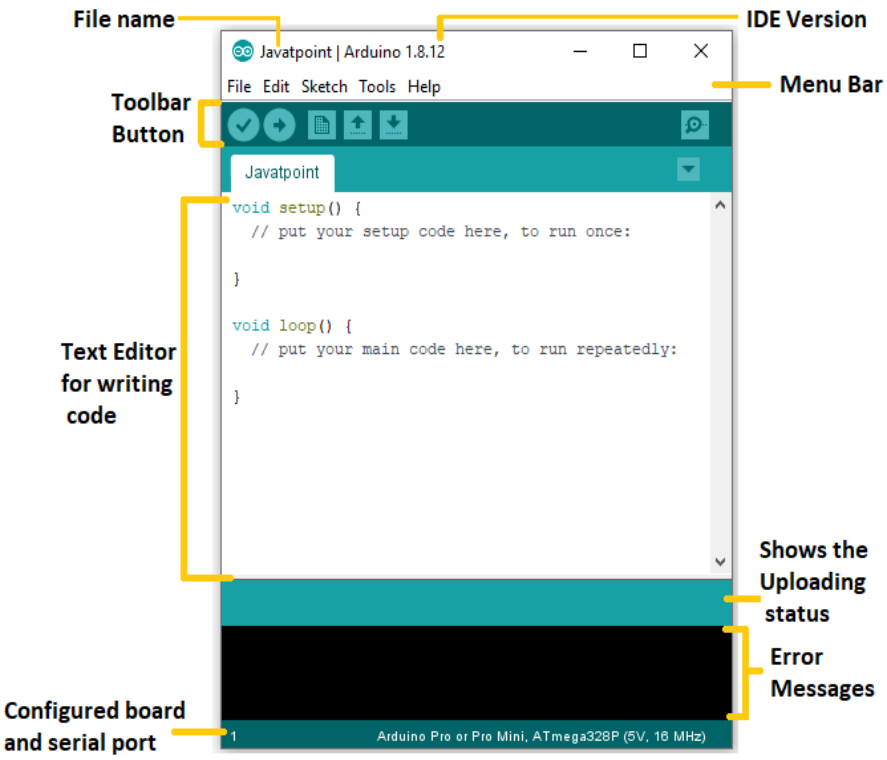


Software Tools

❖ *ARDUINO IDE* :

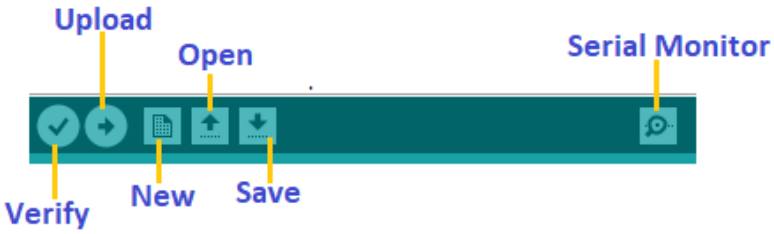
The Arduino IDE is an open-source software, which is used to write and upload code to the Arduino boards. The IDE application is suitable for different operating systems such as **Windows, Mac OS X, and Linux**. It supports the programming languages C and C++. Here, IDE stands for **Integrated Development Environment**.

The program or code written in the Arduino IDE is often called as sketching. We need to connect the Genuino and Arduino board with the IDE to upload the sketch written in the Arduino IDE software. The sketch is saved with the extension '.ino.'



Toolbar :

The icons displayed on the toolbar are **New, Open, Save, Upload,** and **Verify**. It is shown below:



Upload :

The Upload button compiles and runs our code written on the screen. It further uploads the code to the connected board. Before uploading the sketch, we need to make sure that the correct board and ports are selected.

We also need a USB connection to connect the board and the computer. Once all the above measures are done, click on the Upload button present on the toolbar.The latest Arduino boards can be reset automatically before beginning with Upload. In the older boards, we need to press the Reset button present on it. As soon as the uploading is done successfully, we can notice the blink of the Tx and Rx LED.If the uploading is failed, it will display the message in the error window.

We do not require any additional hardware to upload our sketch using the Arduino Bootloader. A **Bootloader** is defined as a small program, which is loaded in the microcontroller present on the board. The LED will blink on PIN 13.

Open :

The Open button is used to open the already created file. The selected file will be opened in the current window.

Save :

The save button is used to save the current sketch or code.

New :

It is used to create a new sketch or opens a new window.

Verify :

The Verify button is used to check the compilation error of the sketch or the written code.

Serial Monitor

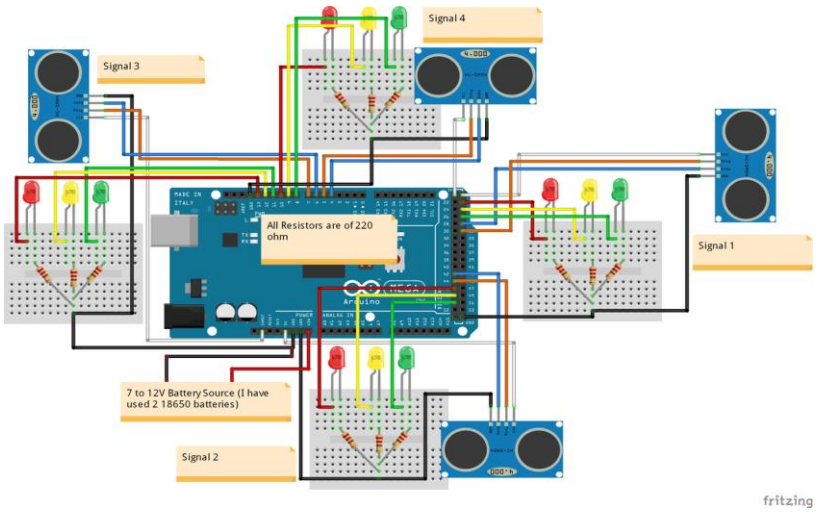
The serial monitor button is present on the right corner of the toolbar. It opens the serial monitor.

4.*IMPLEMENTATION*

Ultrasonic sensor basically emits an ultrasonic wave from the trigger and it is received by the echo after deflecting an object. In order to generate a wave, we will have to set the trigger at high for 10 us which will send an 8 cycle sonic burst at 40 KHz which will hit the object and after hitting the object, the wave will be received by the echo. The echo will then tell us the time that the wave have traveled in us (micro seconds). We will then convert this time into distance travelled by using S = v*t.

LED's are connected to the Arduino through the 220 ohm resistors. It is necessary to use the resistor with the LED. The resistor limits the current flowing through the LED. If you won't use it then the LED will burn out soon. You can use the resistor of value from 100 ohm to 10k ohm with the LED. Larger the value of LED, lesser the current will pass.

4.1 Circuit Diagram :



4.2 Code Explanation :

First of all, we included the timerone library. This library is used to repetitively measure a period of time in microseconds and at the end of each period, an interrupt function will be called.

We have used this library because we want to read from the sensors and control LED's at the same time. We will have to use the delay in between the traffic signal so we can't read from the sensors continuously. Therefore we have used this library which will allow us to call a function in which we will read from the sensors continuously and in the loop function, we will control the traffic signals.

i. `#include<TimerOne.h>`

In the setup function, we have used the Timer1.initialize(microseconds) function. This must be called before you use any of the other methods of timerone library.

“Microseconds” is actually the period of time the timer takes. It is optionally to specify the timer’s period here. The default period is 1 second. Keep in mind that it breaks analogWrite() on digital pins 9 and 10.

```
Timer1.initialize(100000);
```

ii. Timer1.attachInterrupt(softInterr);

Timer1.attachInterrupt(softInterr) calls a function each time the timer period finishes. We have set the timer period at 100000 so our function will be called after 100 milli seconds.

iii. Void loop( )

In the loop function it is looking if there is any vehicles under the 5 cm distance or not. If there will be vehicle, then the function to that signal will be called.

```
void loop()
{
// If there are vehicles at signal 1
if(S1<t)
{
signal1Function();
}
// If there are vehicles at signal 2
if(S2<t)
{
signal2Function();
}
// If there are vehicles at signal 3
if(S3<t)
{
signal3Function();
}
// If there are vehicles at signal 4
if(S4<t)
{
signal4Function();
}
}
```

iv. Void softinterr( )

‘Softinterr()’ is the interrupt function and it will called after every 100 milliseconds. In this function, we have read from the ultrasonic sensors and have calculated the distance.

```
voidsoftInterr()
{
// Reading from first ultrasonic sensor
digitalWrite(triggerpin1, LOW);
delayMicroseconds(2);
digitalWrite(triggerpin1, HIGH);
delayMicroseconds(10);
digitalWrite(triggerpin1, LOW);
time = pulseIn(echopin1, HIGH);
S1= time*0.034/2;
```

4.3 Code

The Arduino code for density based traffic light controller using Arduino is as follows

```
#include<TimerOne.h>
int signal1[] = {23, 25, 27};
int signal2[] = {46, 48, 50};
int signal3[] = {13, 12, 11};
int signal4[] = {10, 9, 8};
int redDelay = 1000;
int yellowDelay = 1000;
volatile int triggerpin1 = 31;
volatile int echopin1 = 29;
volatile int triggerpin2 = 44;
volatile int echopin2 = 42;
volatile int triggerpin3 = 7;
volatile int echopin3 = 6;
volatile int triggerpin4 = 5;
volatile int echopin4 = 4;
volatile long time;                 // Variable for storing the time traveled
volatile int S1, S2, S3, S4;          // Variables for storing the distance covered
int t = 5;  // distance under which it will look for vehicles.
void setup(){
  Serial.begin(115200);
  Timer1.initialize(1000000);   //Begin using the timer. This function must be called first. "microseconds" is the period of time the timer takes.
  Timer1.attachInterrupt(softInterr); //Run a function each time the timer period finishes.
  // Declaring LED pins as output
```

```
  for(int i=0; i<3; i++){
    pinMode(signal1[i], OUTPUT);
    pinMode(signal2[i], OUTPUT);
    pinMode(signal3[i], OUTPUT);
    pinMode(signal4[i], OUTPUT);
  }
  // Declaring ultrasonic sensor pins as output
  pinMode(triggerpin1, OUTPUT);
  pinMode(echopin1, INPUT);
  pinMode(triggerpin2, OUTPUT);
  pinMode(echopin2, INPUT);
  pinMode(triggerpin3, OUTPUT);
  pinMode(echopin3, INPUT);
  pinMode(triggerpin4, OUTPUT);
  pinMode(echopin4, INPUT);
}
void loop()
{
  // If there are vehicles at signal 1
  if(S1<t)
  {
    signal1Function();
  }
  // If there are vehicles at signal 2
  if(S2<t)
  {
    signal2Function();
  }
```

```
  // If there are vehicles at signal 3
  if(S3<t)
  {
    signal3Function();
  }
  // If there are vehicles at signal 4
  if(S4<t)
  {
    signal4Function();
  }
}
// This is interrupt function and it will run each time the timer period finishes. The timer period is set at 100 milli seconds.
void softInterr()
{
  // Reading from first ultrasonic sensor
  digitalWrite(triggerpin1, LOW);
  delayMicroseconds(2);
  digitalWrite(triggerpin1, HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerpin1, LOW);
  time = pulseIn(echopin1, HIGH);
  S1= time*0.034/2;
  // Reading from second ultrasonic sensor
  digitalWrite(triggerpin2, LOW);
  delayMicroseconds(2);
  digitalWrite(triggerpin2, HIGH);
```

```
delayMicroseconds(10);
digitalWrite(triggerpin2, LOW);
time = pulseIn(echopin2, HIGH);
S2= time*0.034/2;
// Reading from third ultrasonic sensor
digitalWrite(triggerpin3, LOW);
delayMicroseconds(2);
digitalWrite(triggerpin3, HIGH);
delayMicroseconds(10);
digitalWrite(triggerpin3, LOW);
time = pulseIn(echopin3, HIGH);
S3= time*0.034/2;
// Reading from fourth ultrasonic sensor
digitalWrite(triggerpin4, LOW);
delayMicroseconds(2);
digitalWrite(triggerpin4, HIGH);
delayMicroseconds(10);
digitalWrite(triggerpin4, LOW);
time = pulseIn(echopin4, HIGH);
S4= time*0.034/2;
// Print distance values on serial monitor for debugging
Serial.print("S1: ");
Serial.print(S1);
Serial.print("  S2: ");
Serial.print(S2);
Serial.print("  S3: ");
Serial.print(S3);
Serial.print("  S4: ");
```

```
  Serial.println(S4);
}
void signal1Function()
{
  Serial.println("1");
  low();
  // Make RED LED LOW and make Green HIGH for 5 seconds
  digitalWrite(signal1[0], LOW);
  digitalWrite(signal1[2], HIGH);
  delay(redDelay);
  // if there are vehicels at other signals
  if(S2<t || S3<t || S4<t)
  {
    // Make Green LED LOW and make yellow LED HIGH for 2 seconds
    digitalWrite(signal1[2], LOW);
    digitalWrite(signal1[1], HIGH);
    delay(yellowDelay);
  }
}
void signal2Function()
{
  Serial.println("2");
  low();
  digitalWrite(signal2[0], LOW);
  digitalWrite(signal2[2], HIGH);
  delay(redDelay);
  if(S1<t || S3<t || S4<t)
  {
```

```
    digitalWrite(signal2[2], LOW);
    digitalWrite(signal2[1], HIGH);
    delay(yellowDelay);
  }
}
void signal3Function()
{
  Serial.println("3");
  low();
  digitalWrite(signal3[0], LOW);
  digitalWrite(signal3[2], HIGH);
  delay(redDelay);
  if(S1<t || S2<t || S4<t)
  {
    digitalWrite(signal3[2], LOW);
    digitalWrite(signal3[1], HIGH);
    delay(yellowDelay);
  }
}
void signal4Function()
{
  Serial.println("4");
  low();
  digitalWrite(signal4[0], LOW);
  digitalWrite(signal4[2], HIGH);
  delay(redDelay);
  if(S1<t || S2<t || S3<t)
  {
```

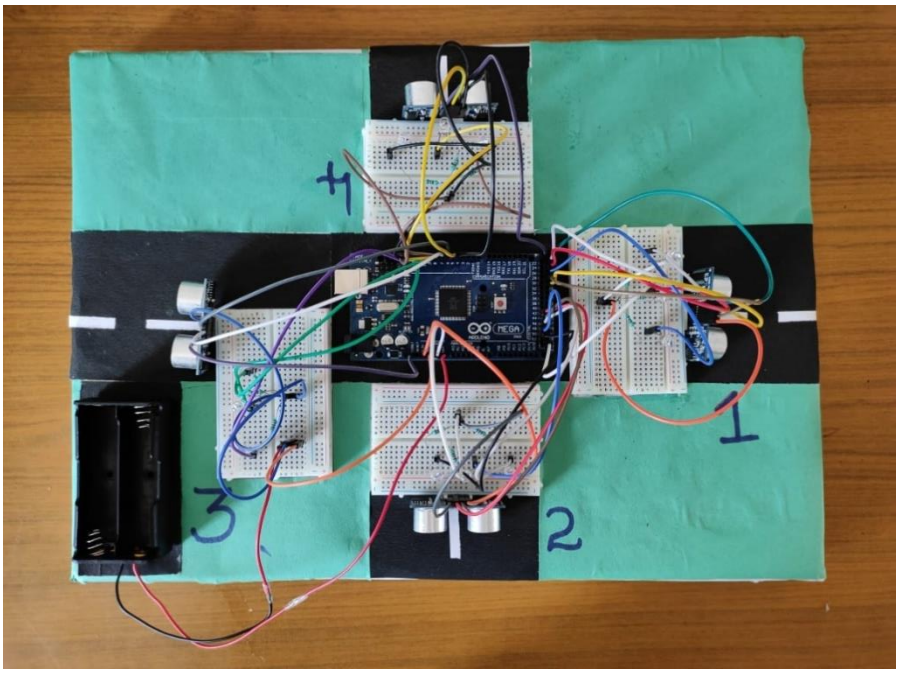
```
    digitalWrite(signal4[2], LOW);
    digitalWrite(signal4[1], HIGH);
    delay(yellowDelay);
  }
}
// Function to make all LED's LOW except RED one's.
void low()
{
  for(int i=1; i<3; i++)
  {
    digitalWrite(signal1[i], LOW);
    digitalWrite(signal2[i], LOW);
    digitalWrite(signal3[i], LOW);
    digitalWrite(signal4[i], LOW);
  }
  for(int i=0; i<1; i++)
  {
    digitalWrite(signal1[i], HIGH);
    digitalWrite(signal2[i], HIGH);
    digitalWrite(signal3[i], HIGH);
    digitalWrite(signal4[i], HIGH);
  }
}
```

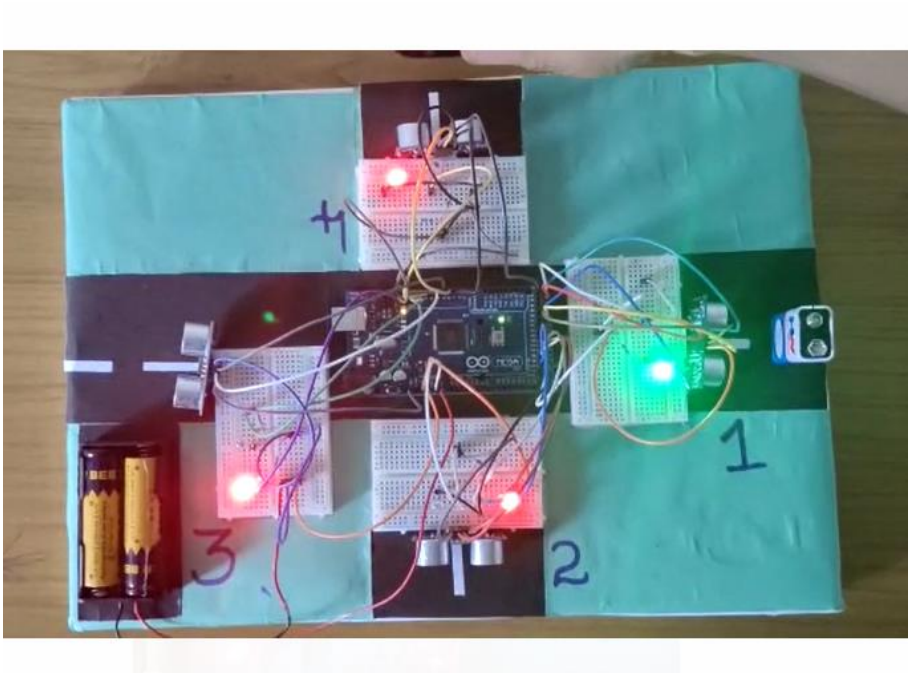
5. RESULTS

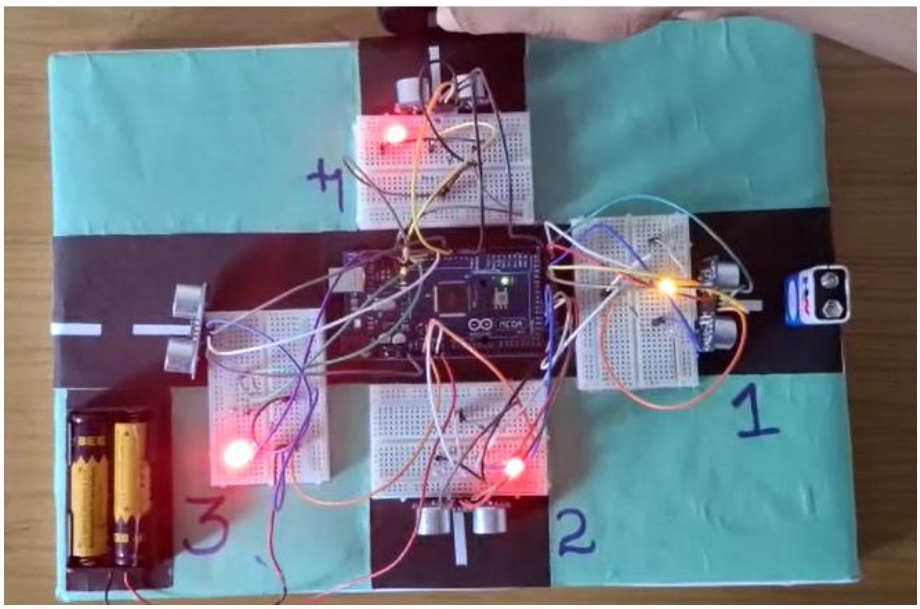
If there is traffic at all the signals, then the system will work normally by controlling the signals one by one.

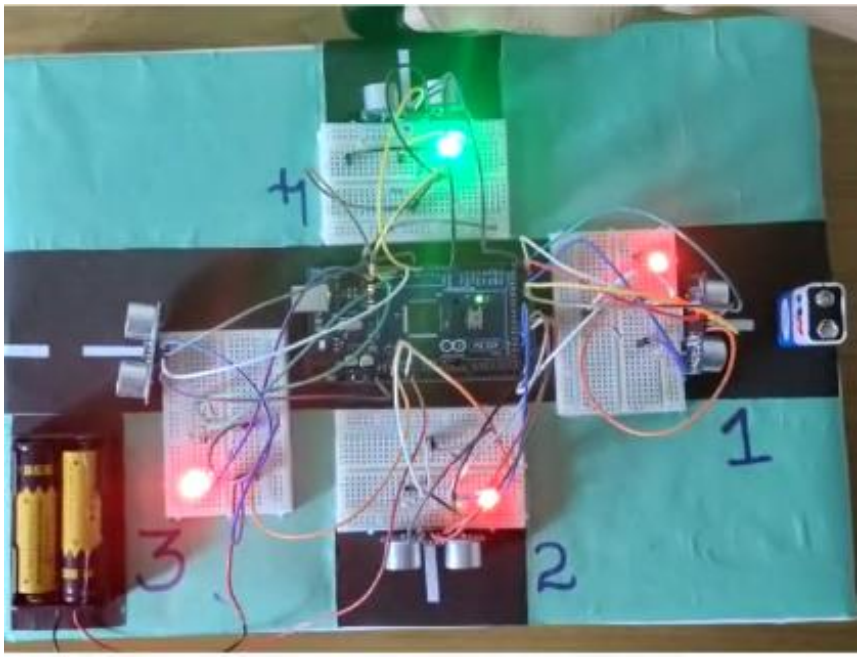
If there is no traffic near a signal, then the system will skip this signal and will move on to the next one. For example, if there is no vehicle at signal 2, 3 and currently the system is allowing vehicles at signal 1 to pass. Then after signal 1, the system will move on to signal 4 skipping signal 2 and 3.

If there is no traffic at all the 4 signals, system will stop at the current signal and will only move on the next signal if there will be traffic at any other signal.




4
1
3
2


4
1
3
2


1
2
3

6.CONCLUSION

In this project, we have implemented densitybased traffic signal system using Arduino .The hardware equipment is tested & the result is obtained . Implementation of project is in present day will effectively solve the traffic congestion which is a general problem in many modern cities all over the world. Consider a scenario of highly congested area where many vehicles such as personal transport ,public transport and emergency vehicles have to wait for long change of traffic signals at intersection points. This leads in delay of time. It is possible to propose dynamic time-based coordination of traffic, This is achieved by using UR sensors across the rod to monitor the density of vehicles blocking the road traffic. The signals from the UR receives are fed to the Arduino to follow the program with the time desired. with a slight modification in the project can be implemented in a near-by area.

7.REFERENCES

https://en.m.wikipedia.org/wiki/Arduino_Uno

https://www.theengineeringprojects.com/2019/01/introduction-to-lm35.html

https://www.elprocus.com/sensor-based-electronics-projects/