**DENSITY BASED ARDUINO TRAFFIC LIGHT CONTROLLER**

*by*

**M. DHEERAJ SAI (19BCE1876)**

**P. SUHAS REDDY (19BCE1867)**

**AMAR DIXIT (19BCE1875)**

**K. MANOJ KUMAR (19BCE1866)**

A project report submitted to

**Dr. LUCKY AGARWAL**

**SCHOOL OF ELECTRONICS ENGINEERING**

in partial fulfilment of the requirements for the course of

**CSE2006 – MICROPROCESSOR AND INTERFACING**

in

**B. Tech. COMPUTER SCIENCE AND ENGINEERING**

****

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

**VELLORE INSTITUTE OF TECHNOLOGY**

**CHENNAI - 600127**

***June 2021***

**BONAFIDE CERTIFICATE**

This is to certify that this project report entitled “**ARDUINO TRAFFIC LIGHT CONTROLLER**” is a bonafide work of by ***M. Dheeraj Sai* (19BCE1876)*, P. Suhas Reddy* (19BCE1867), *Amar Dixit* (19BCE1875)** and ***K. Manoj Kumar* (19BCE1866)** who carried out the Project work under my supervision and guidance for the course **CSE2006-MICROPROCESSOR AND INTERFACING**.

**Dr. LUCKY AGARWAL**

Associate Professor

School of Electronics Engineering (SENSE),

VIT University, Chennai

Chennai – 600 127.

**ABSTRACT**

Congestion is a serious issue due to vehicular traffic. One of the known causes of traffic congestion is the amount of time spend waiting for the red light to change to green. The changing of traffic light is hard-coded and it is not reliant on traffic volume. There is therefore need to simulate and optimize traffic control to better accommodate density-based traffic rather than time based. This system attempts to lessen possibilities of traffic jams brought about by traffic lights to a reasonable degree. This project, a density-based traffic control system is been implemented to solve this problem. The system entails programming an Arduino using Arduino integrated development environment (IDE) to enable traffic lights give the right of access to the road by selecting the lane with the high number of cars. The traffic lights are modified to chip away at an auspicious premise until there is a signal identified by the infrared sensors. The sensor identifies an object (i.e. a vehicle, a motorcycle etc) and signals the Arduino to control the traffic lights for its individual path. Once there is no sign identified by any of the four sensors the traffic lights keep on dealing with an auspicious premise.

**Table of Contents**

|  |  |  |  |
| --- | --- | --- | --- |
| **Chapter No.** | **Title** | | **Page No.** |
|  | Abstract | | 2 |
| List of Figures | | 4 |
| List of Tables | | 4 |
| List of Abbreviations | | 4 |
| 1 | Introduction | | 5 |
| 1.1 | Objectives | 5 |
| 1.2 | Scope | 5 |
| 2 | Design/ Implementation | | 6 |
| 2.1 | Introduction | 6 |
| 2.2 | Design Approach | 6 |
| 2.3 | Proposed System | 7 |
|  | 2.3.1 Economic Feasibility | 7 |
|  | 2.3.2 Technical feasibility | 7 |
|  | 2.3.3 Operational feasibility | 8 |
| 2.4 | Overview of Software | 9 |
| 2.5 | Hardware Specification | 9 |
| 2.6 | Software Requirement | 9 |
| 2.7 | Summary | 10 |
| 3 | Result and Analysis/Testing | | 11 |
| 3.1 | Density Based Traffic Light Control System Circuit Design | 11 |
| 3.2 | Arduino Code | 12 |
| 3.3 | How to Connect the Circuit | 15 |
| 3.4 | Testing of the Circuit | 16 |
| 3.5 | Summary | 19 |
| 4 | Conclusion and future Enhancement | | 23 |
| 5 | Appendix | | 21 |
| 8 | References  Bio- Data | | 28  29 |

**List of Figures**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Title** | **Page No.** |
| 2.1 | Proposed System | 11 |
| 3.1 | Proteus Circuit | 14 |
| 3.2 | Arduino Code using Arduino IDE | 14 |
| 3.3 | Compiled Result of Code | 15 |
| 3.4 | Arduino Properties | 15 |
| 3.5 | Green light glowing in side A with high Density | 16 |
| 3.6 | Green light glowing in side B with high Density | 17 |
| 3.7 | Green light glowing in side C with high Density | 17 |
| 3.8 | Green light glowing in side D with high Density | 18 |

**List of Tables**

|  |  |  |
| --- | --- | --- |
| **Table No..** | **Title** | **Page No.** |
| 3.1 | Responsive Time | 18 |

**List of Abbreviations**

|  |  |  |
| --- | --- | --- |
| **Abbreviation** | **Title** | **Page No.** |
| IDE | Integrated Development Environment | Repeated Use |
| IR Sensor | Infra-Red Sensor | Repeated Use |
| LED | Light Emitting Diode | Repeated Use |
| AI | Artificial Intelligence | 23 |

**CHAPTER 1 -** **Introduction**

* 1. **Purpose**

One of such over-stretched infrastructure is the road, a circumstance which has resulted to increase in traffic. In spite of the fact that traffic lights have dependably been utilized for controlling the movement of traffic (pedestrian or automobile), traffic management in many real urban areas around the globe has kept on being a subject of concern. A traffic light is a signaling device which controls a traffic flow at road intersections. It consists of three basic lights which include red, yellow and green. Red signal is used to halt traffic from proceeding; yellow signal alerts vehicles for brief stop, while green signal alerts vehicles for procession in the indicated directions. Traffic congestion is a situation that happens on road networks in which vehicles travel slower than usual due to the increased physical use of vehicles (traffic) on the road at that moment. Also known as traffic jam, traffic congestions may result from roads being blocked, bad roads, accidents on the road, lack of proper traffic light system to control vehicles, inappropriate driving by road users etc. This would make the journey longer due to the slow movement of the traffic and increased queuing of vehicles. Urban communities started to make traffic tenets to minimize crashes, while traffic flags and cops were utilized to coordinate option to proceed at major urban convergences. Traffic control development in urban streets has aided easy movement and use of automobile in mega cities. Most of the major roads have effective traffic control system, which has facilitated easy flow of vehicles in urban regions. Railroads were invented serving as a temporary solution to the developing issue of street traffic control. However, this creates congestion at terminals inside urban communities. Therefore, the aim of this study is to design a density-based traffic control using Arduino in order to solve the problem of traffic congestion.

* 1. **Scope**

In the design and construction of this system all components work simultaneously. The system works just as a typical traffic light system. The uniqueness of this is in the event that the thickness of vehicles in a specific path of the road is high. At that point, the sensor in that specific path turns out to be low else it is read as a high signal. The signal from the IR (Infrared) is used by the system to control the traffic jamming of the lane. In the event that we get a low signal from any of these sensors, at that point the green LED shines to that specific way and gives a red to every other way. The Arduino IDE is programmed using C language.

**CHAPTER 2**

**Design/Implementation**

**2.1 Introduction**

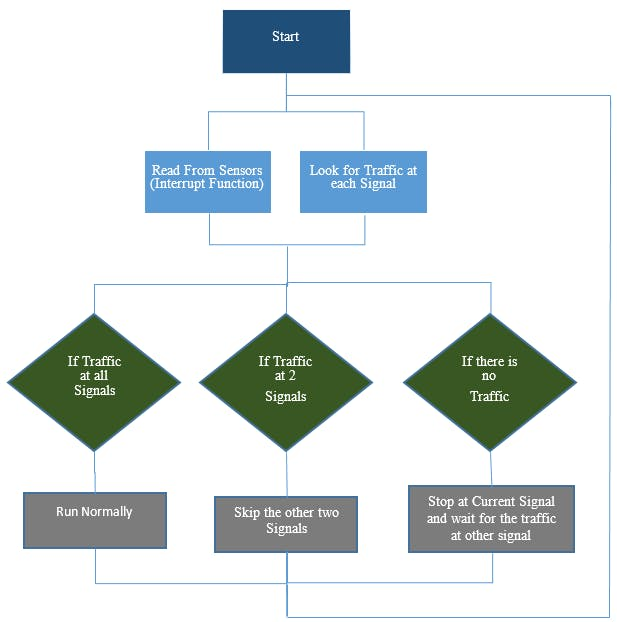
In this project, an Arduino based Traffic Light Controller system is designed. It is a simple implementation of traffic lights system but can be extended to a real time system with programmable timings, pedestrian lighting etc

**2.2 Design Approach**

In this system, we will use IR sensors to measure the traffic density. We have to arrange one IR sensor for each road; these sensors always sense the traffic on that particular road. All these sensors are interfaced to the microcontroller. Based on these sensors, controller detects the traffic and controls the traffic system. The main heart of this traffic system is microcontroller. IR sensors are connected to the PORT C (PC0, PC1, PC2, and PC3) of the microcontroller and traffic lights are connected to PORT B and PORT D. If there is a traffic on road then that particular sensor output becomes logic 0 otherwise logic 1. By receiving these IR sensor outputs, we have to write the program to control the traffic system. If you receive logic 0 from any of these sensors, we have to give the green signal to that particular path and give red signal to all other paths. Here continuously we have to monitor the IR sensors to check for the traffic.

**2.3 Proposed System**

* The traffic lights will provide instructions to the users (drivers and pedestrians) by displaying lights of standard color. The three colors used in traffic lights are Red, Yellow and Green.
* The system must be used to control the traffic lights for smooth and safe movement of traffic



**Figure 2.1 Proposed System**

**2.3.1 Economic Feasibility**

The traffic light issue is obviously a critical problem that worries citizens and governments. The influence of low efficient conventional traffic system affects the economic, health, financial, and environmental domains.

By using this smart traffic management system, we can reduce pollution and also average waiting time

**2.3.2 Technical feasibility:**

The smart light traffic control system is composed of two separate devices: the traffic master controller and the handheld portable controller. Fig. 3 shows the hardware implemented circuit of the smart traffic controller using the Proteus software. The traffic master controller is mounted with the traffic lights at the roads intersection and is responsible for the lighting transition and their timing slots. Its implemented design circuit includes: the PIC 16F877A microcontroller, the three lights (red, green, and yellow) of the traffic lights A and B associated to the roads 1 and 2, the two lights (red and green) for the traffic light R and L associated with the deflection to the right and left in the direction of roads 3 and 4, the two IR receivers to measure the traffic volume, the XBee transmitter system, and other basic components. The traffic master controller provides the duration and the schedule of the two configurations and their dedicated phases for different modes of traffic. It determines the status of the different lights by commanding the triggered switches connected to the PIC ports. The microcontroller is also connected to IR detectors whose output voltages are responsible of shifting the counter of the cars arriving at the intersection. Finally, the XBee module receives the command orders form the portable controller and calls the corresponding emergency subroutines. The portable controller commands the traffic master controller by means of XBee transceiver that communicates wirelessly with the other XBee component. A PIC 16F877A constitutes the hardware core of the portable controller. It is connected, in addition to XBee, to the buttons EA and EB that start up the emergency subroutines. An LCD screen is employed to notify the user if the mode of emergency is operating and which emergency procedure is currently running. We propose also a password of 6 digits formed by the combinations of 4 digits from 1 to 4 in order to supply the portable controller by a certain security level. The total number of arrangements is 4096 possibilities. The role of security code is to prevent unauthorized persons from accessing to the smart light system.

**2.3.3 Operational feasibility**

This section describes the calibrations and the sampling rate of various sensors used in Precision agriculture system.

* We will implement traffic light controller using smart traffic management, which means the traffic lights keeps updating based on traffic near the signal count and vehicle flowing under the traffic light per time unit and use different intervals of green/red light

**2.4 Overview of software**

* Oak Ridge National Lab is already [working](https://gcn.com/articles/2020/03/25/oak-ridge-ai-traffic-fuel-reductions.aspx) on a slice of the problem. Researchers there are using overhead cameras and roadway sensors to identify gas guzzling commercial trucks in traffic. AI and machine learning algorithms identify the least-efficient vehicles, track their path and speed and change the traffic signals ahead of the vehicles.  This eliminates much of the inefficient starting and stopping at intersections and minimizes fuel consumption.
* Creation of such a system, especially for dense urban corridors and sprawling exurbs, can greatly improve energy and sustainability impacts.
* This is critical as our transportation portfolio will continue to have a heavy reliance on gasoline-powered vehicles for some time.

**2.5 Hardware Specification**

* ATmega8 microcontroller.
* 4 X HC-SR04 ultrasonic sensors
* 4 X Red LEDs
* 4 X Green LEDs
* 4 X Yellow LEDs
* 4 X 10K ohm resistors
* Jumper cables
* Breadboards

**2.6 Software Requirements**

* PROTEUS

The Proteus Design Suite is a proprietary software tool suite used primarily for electronic design automation. The software is used mainly by electronic design engineers and technicians to create schematics and electronic prints for manufacturing printed circuit boards.

* ARDUINO

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.

This software can be used with any Arduino board.

**2.7 Summary**

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 one.

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 timer one 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.

**CHAPTER 3**

**Result and Analysis / Testing**

**3.1** **Density Based Traffic Light Control System Circuit Design**

This is the complete circuit diagram of the system. On the left you can see push buttons acting as IR sensors. Each

pair of push button is numbered according to the road they are sensing. On the right side the signals are given which

are interfaced with the port 1 of the controller .Below is the LCD interfaced with the microcontroller which will

show weather the signals are operating normally or for the dense case

This is the complete circuit diagram of the system. On the left you can see push buttons acting as IR sensors. Each

pair of push button is numbered according to the road they are sensing. On the right side the signals are given which

are interfaced with the port 1 of the controller .Below is the LCD interfaced with the microcontroller which will

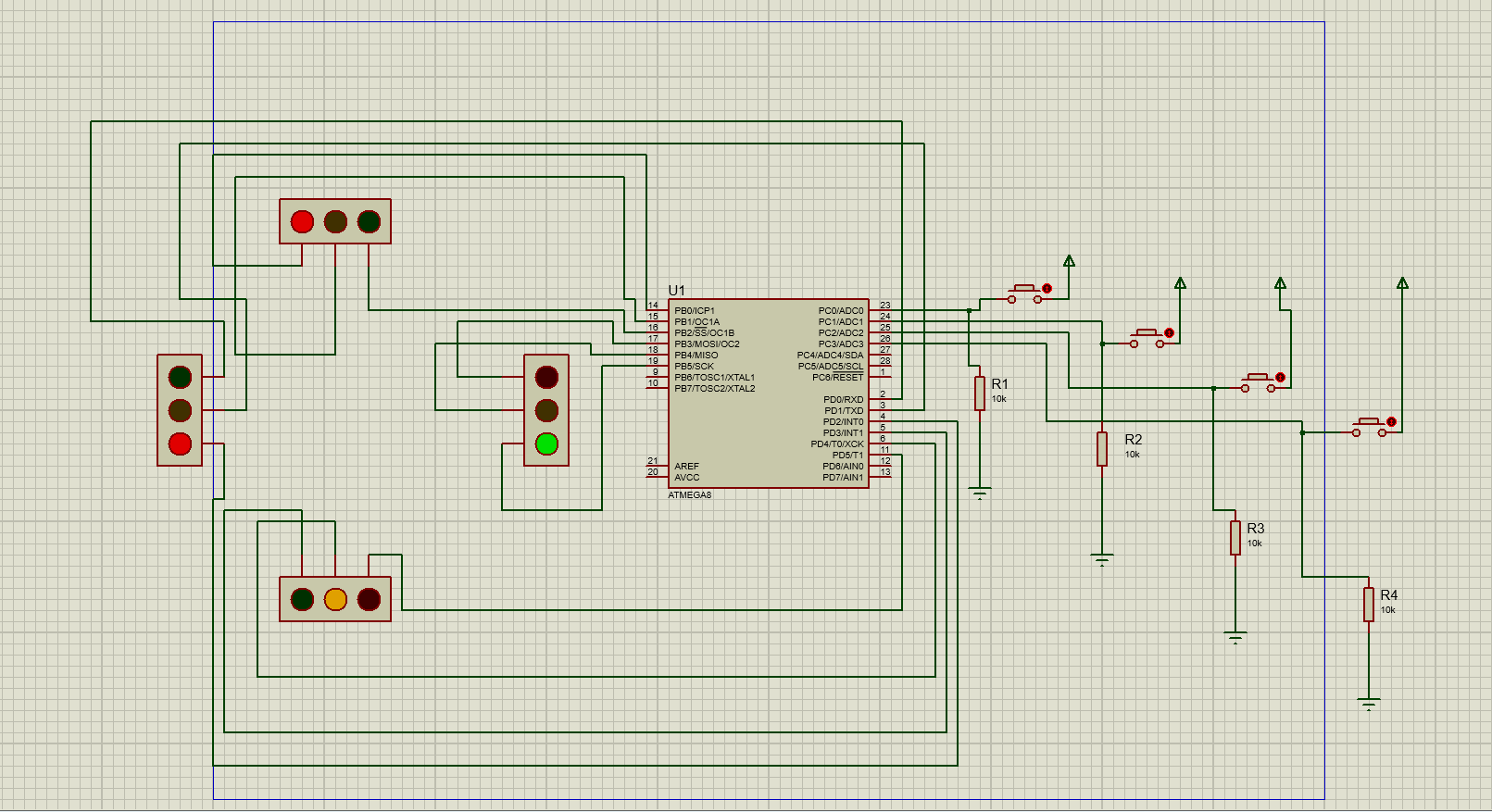
show weather the signals are operating normally or for the dense case

This circuit consists of 4 IR sensors, atmega8 microcontroller, 4 traffic lights.

IR transmitter looks like an LED. This IR transmitter always emits IR rays from it. The operating voltage of this IR transmitter is 2 to 3v. These IR (infra-red) rays are invisible to the human eye. But we can view these IR rays through camera.

IR receiver receives IR rays that are transmitted by IR transmitter. Normally IR receiver has high resistance in order of mega ohms, when it is receiving IR rays the resistance is very low. The operating voltage of IR receiver also 2 to 3V. We have to place these IR pair in such a way that when we place an obstacle in front of this IR pair, IR receiver should be able to receive the IR rays. When we give the power, the transmitted IR rays hit the object and reflect back to the IR receiver.

Instead of traffic lights, you can use LEDs (RED, GREEN, YELLOW). In normal traffic system, you have to glow the LEDs on time basis. If the traffic density is high on any particular path, then glows green LED of that particular path and glows the red LEDs for remaining paths. In normal traffic system, we allow the traffic for a time delay of 1 minute for each path. Here 10k ohm resistor is used to drop the voltage otherwise IR transmitter may get damaged.

****

**Figure 3.1 Proteus Circuit**

**3.2** **Arduino Code**

Code:

#define F\_CPU 8000000UL

#include <avr/io.h>

#include <util/delay.h>

#define R1 PB0

#define Y1 PB1

#define G1 PB2

#define R2 PB3

#define Y2 PB4

#define G2 PB5

#define R3 PD5

#define Y3 PD4

#define G3 PD3

#define R4 PD2

#define Y4 PD1

#define G4 PD0

int main(void)

{

DDRB = 0xff;

DDRD = 0xff;

DDRC = 0x00;

PORTB = 0x00;

PORTD = 0x00;

while(1)

{

if((PINC&0x01) == 0x01)

{

PORTB |= (1<<G1);

PORTB |= (1<<Y2);

PORTD |= (1<<R3);

PORTD |= (1<<R4);

}

else if((PINC&0x02) == 0x02)

{

PORTB |= (1<<R1);

PORTB |= (1<<G2);

PORTD |= (1<<Y3);

PORTD |= (1<<R4);

}

else if((PINC&0x04) == 0x04)

{

PORTB |= (1<<R1);

PORTB |= (1<<R2);

PORTD |= (1<<G3);

PORTD |= (1<<Y4);

}

else if((PINC&0x08) == 0x08)

{

PORTB |= (1<<Y1);

PORTB |= (1<<R2);

PORTD |= (1<<R3);

PORTD |= (1<<G4);

}

else

{

PORTB = 0x00;

PORTD = 0x00;

PORTB |= (1<<G1);

PORTB |= (1<<Y2);

PORTD |= (1<<R3);

PORTD |= (1<<R4);

\_delay\_us(500000);

PORTB = 0x00;

PORTD = 0x00;

PORTB |= (1<<R1);

PORTB |= (1<<G2);

PORTD |= (1<<Y3);

PORTD |= (1<<R4);

\_delay\_us(500000);

PORTB = 0x00;

PORTD = 0x00;

PORTB |= (1<<R1);

PORTB |= (1<<R2);

PORTD |= (1<<G3);

PORTD |= (1<<Y4);

\_delay\_us(500000);

PORTB = 0x00;

PORTD = 0x00;

PORTB |= (1<<Y1);

PORTB |= (1<<R2);

PORTD |= (1<<R3);

PORTD |= (1<<G4);

\_delay\_us(500000);

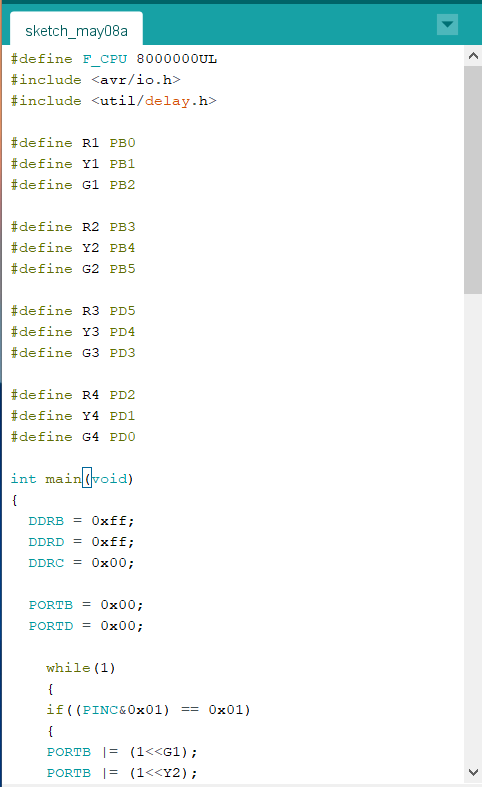
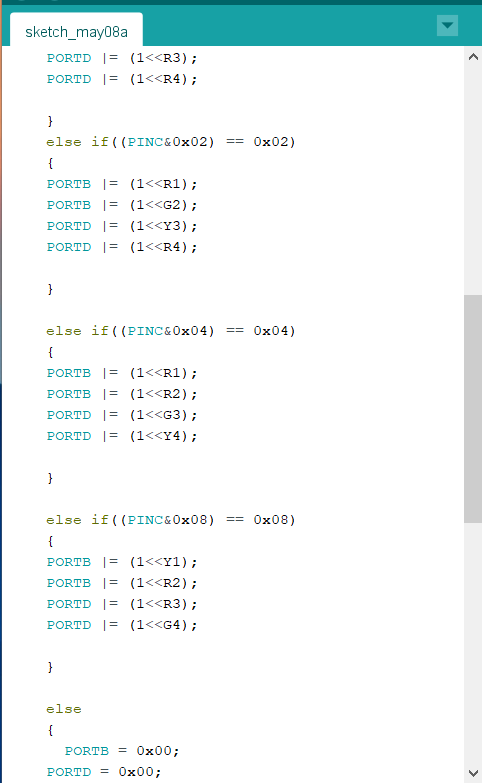
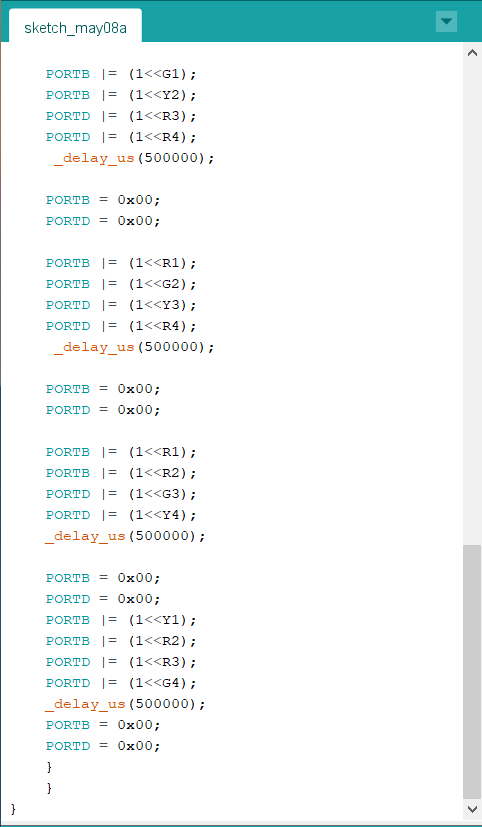
PORTB = 0x00;

PORTD = 0x00;

}

}

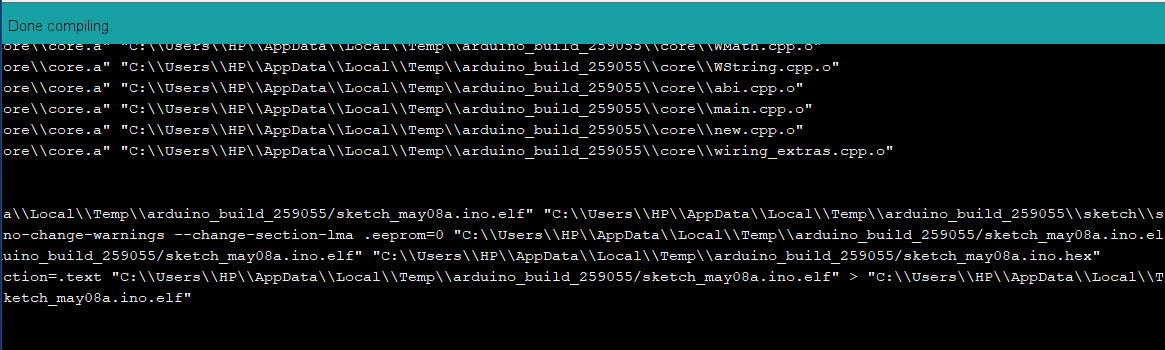
}

**Figure 3.2 Arduino Code using Arduino IDE**

**3.3** **How to Connect the Circuit**

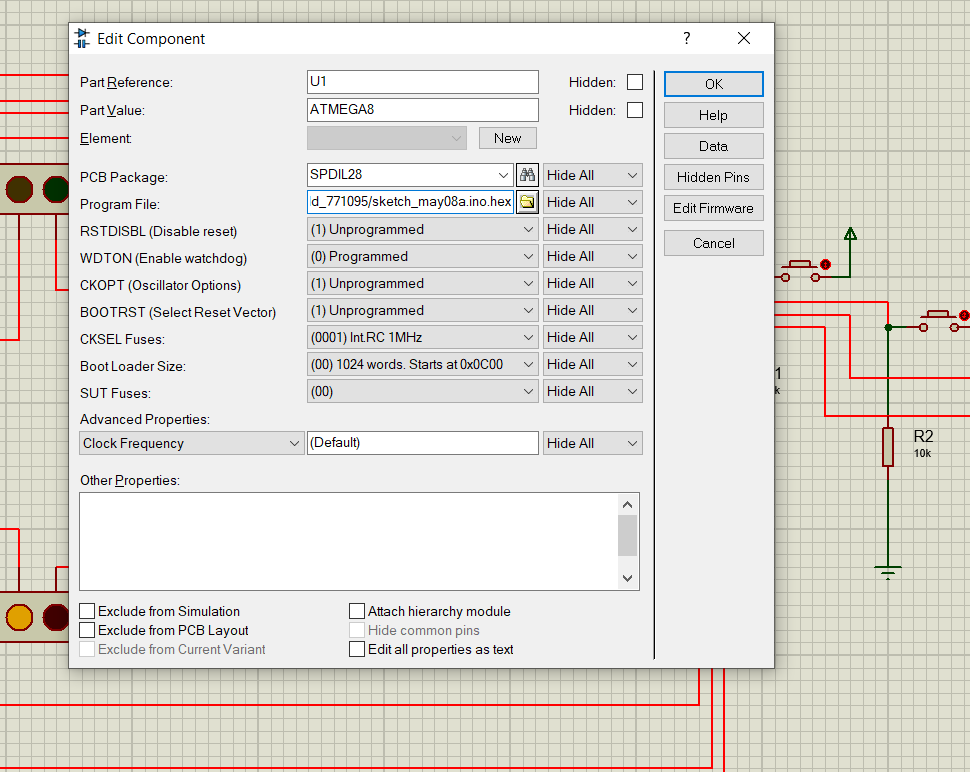
* Connect four IR sensors to PORT C.
* Connect LEDs to PORT B and PORT D.
* Arrange all this LED’s same as like traffic lights.
* Arrange one IR sensor for each road.
* Run the Arduino code in Arduino code in Arduino IDE and find the location ending with .hex shown after compiling the code.





**Figure 3.3 Compiled Result of Code**

* Copy the link address highlighted part in the above diagram and open edit properties of the Arduino and paste the copied file in the Program file section of it.

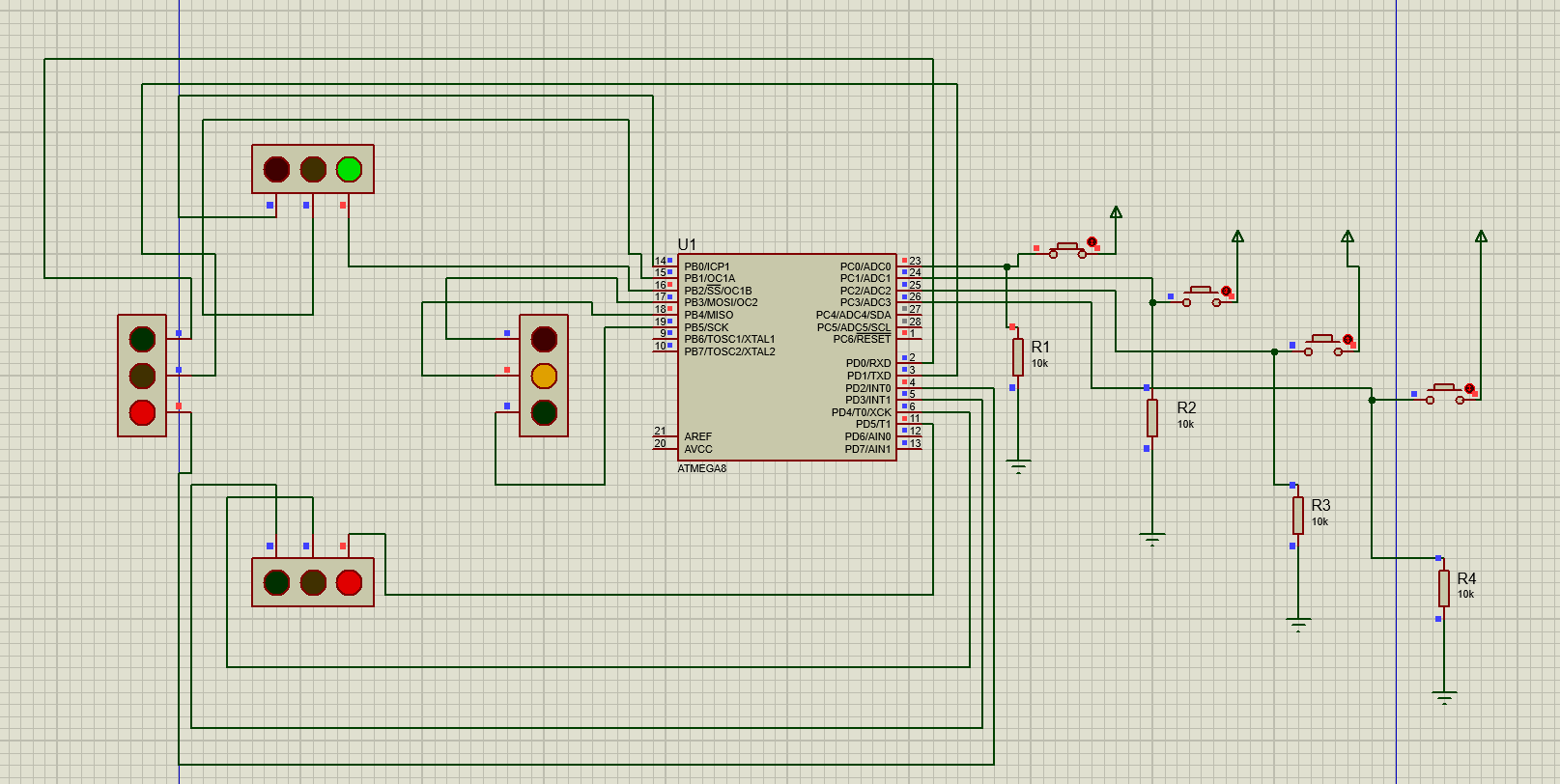


**Figure 3.4 Arduino Properties**

* Save the changes and then Run the circuit in normal mode.
* Now you can see the normal traffic system based on time basis.
* Now if we increase density/ place any obstacle in front of any IR sensor, then the system allows the traffic of that particular path by glowing GREEN light

**3.4 Testing of the Circuit**

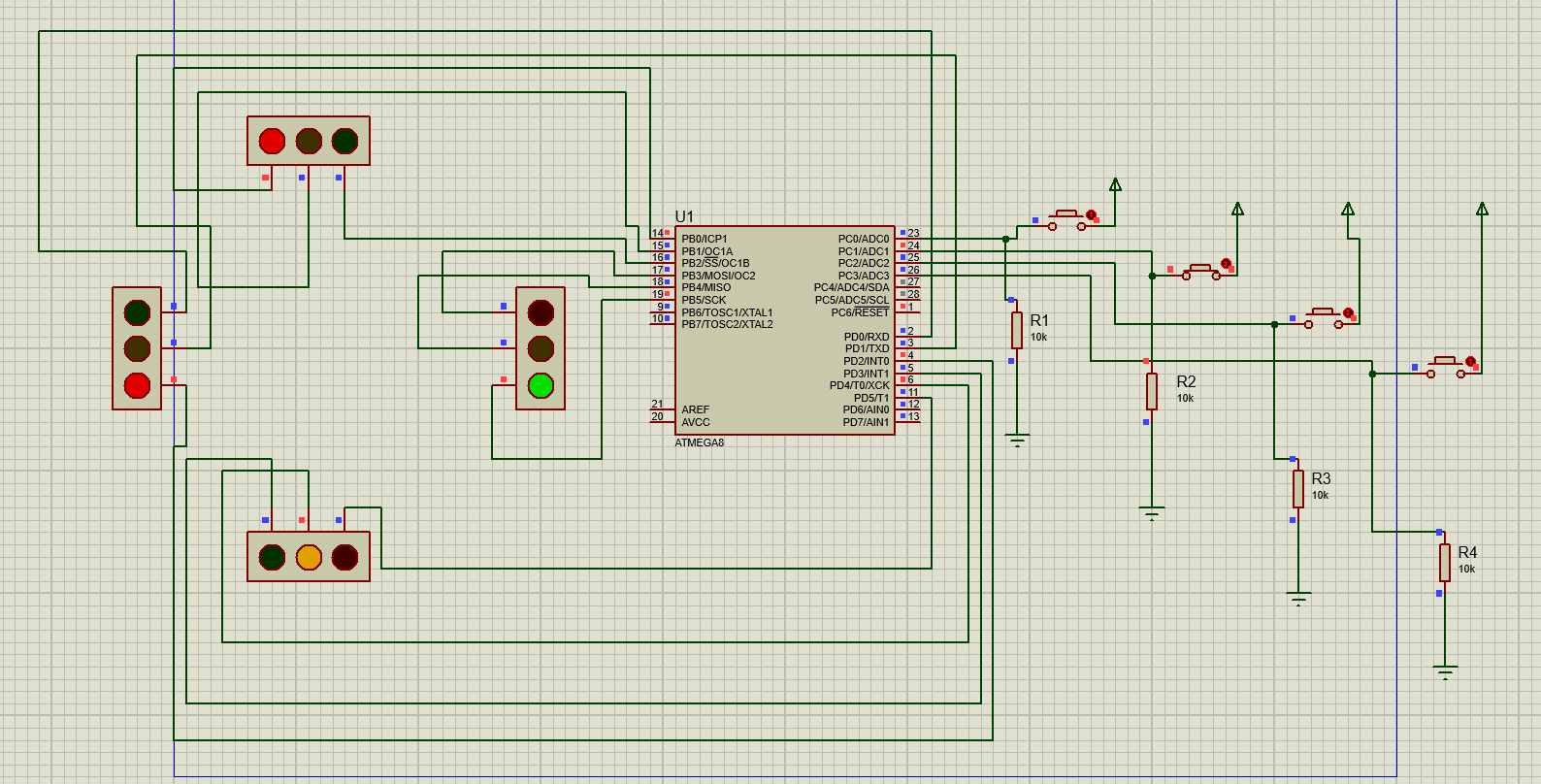
**Case 1:** Density of traffic is high in A direction at the signal then green light will be shown at A direction and remaining sides will have Red/ Yellow.

****



**Figure 3.5 Green light glowing in side A with high Density**

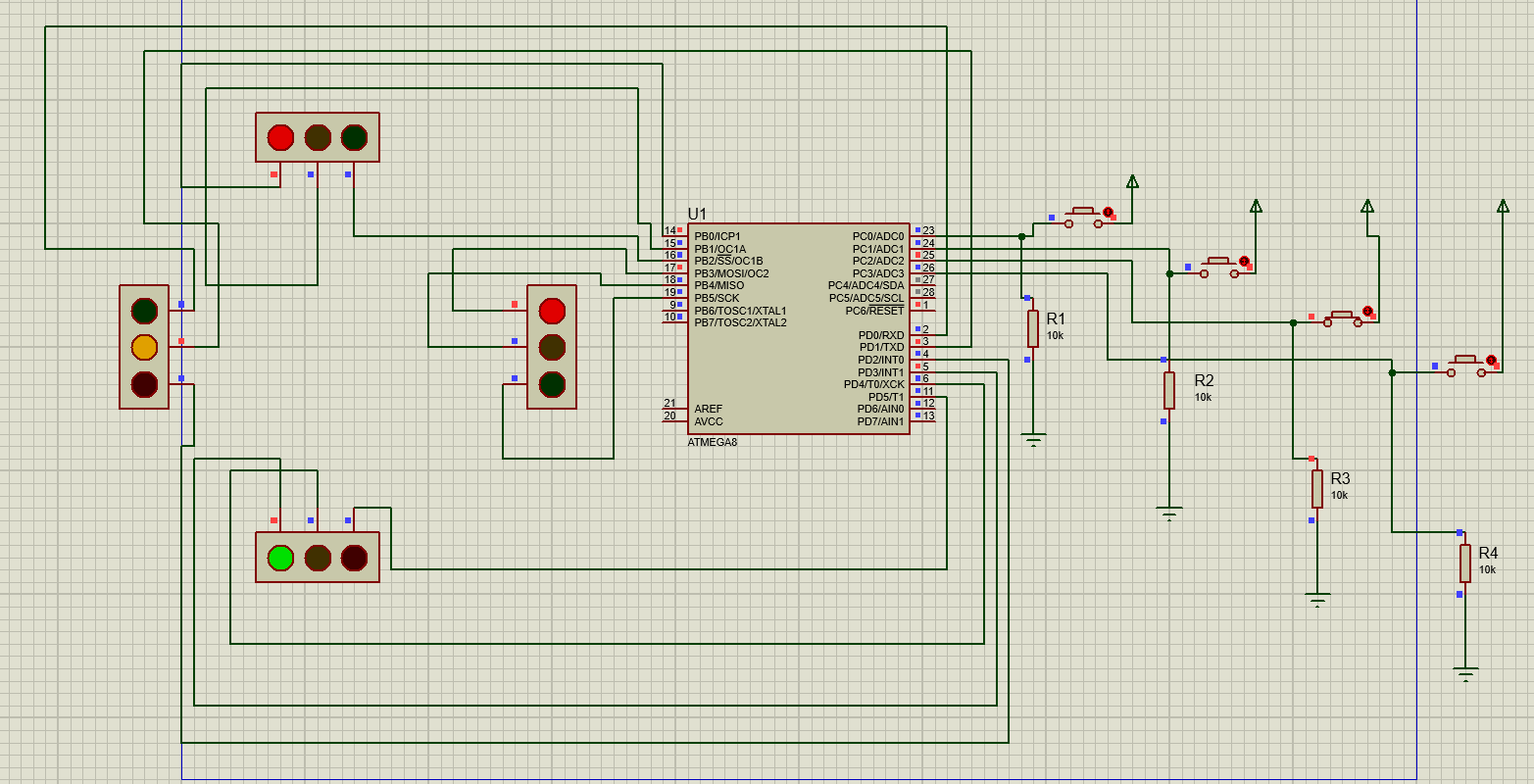
**Case 2:** Density of traffic is high in B direction at the signal then green light will be shown at B direction and remaining sides will have Red/ Yellow.

****



**Figure 3.6 Green light glowing in side B with high Density**

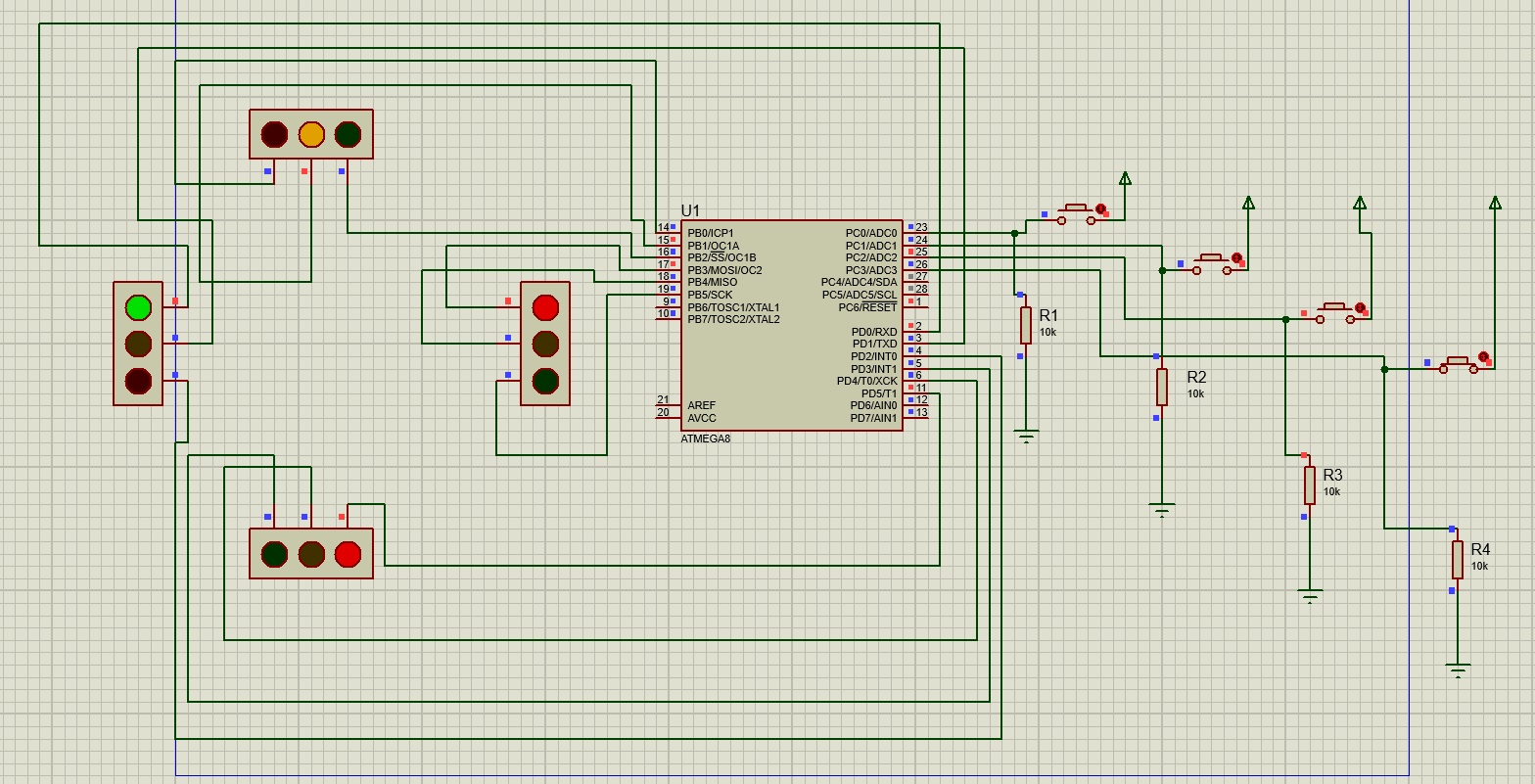
**Case 3:** Density of traffic is high in C direction at the signal then green light will be shown at C direction and remaining sides will have Red/ Yellow.

****



**Figure 3.7 Green light glowing in side C with high Density**

**Case 4:** Density of traffic is high in D direction at the signal then green light will be shown at D direction and remaining sides will have Red/ Yellow.

****



**Figure 3.8 Green light glowing in side D with high Density**

**Case 5:** If density is high in any two of the directions, then the green light will be shown for

the two directions in the order of clockwise directions.

**Responsive Time**

The response time of the sensor is the time taken for the sensor to produce an output. This is the time it takes once it detects an obstacle and sends a signal to the Arduino and changes the traffic light color for execution. The result obtained here is in agreement with the report of Table 3.1 shows the average response time and Figure 10 is the pictorial chart of the time.

|  |  |
| --- | --- |
| Number of Attempts | Time (s) |
| 1 | 0.32 |
| 2 | 0.41 |
| 3 | 0.44 |
| 4 | 0.31 |
| 5 | 0.31 |
| 6 | 0.32 |
| 7 | 0.54 |
| 8 | 0.54 |
| 9 | 0.29 |
| 10 | 0.39 |
| Mean | 0.387 |

**Table 3.1**

**3.5 Summary**

The density-based traffic control system has been designed, constructed and tested to ensure validation of its function and operations. By using this system configuration, we try to reduce the possibilities of traffic jams, caused by traffic lights. Number of passing vehicle in the fixed time slot on the road decide the density range of traffics and on the basis of vehicle density calculation, microcontroller decide the traffic light delays

**Discussion**

There are some limitations in this model

* IR sensors sometimes may absorb normal light also. As a result, traffic system works in improper way.
* IR sensors work only for fewer distances.
* We have to arrange IR sensors in accurate manner otherwise they may not detect the traffic density.

**CHAPTER 4**

**CONCLUSION AND FUTURE ENHANCEMENT**

The density-based traffic control system has been designed, constructed and tested to ensure validation of its function and operations. In this research, we have succeeded in minimizing the traffic congestions created by the fixed time-based traffic light system. The system is effective and the cost of production is very low. Future work is recommended in order to produce the device on a large scale and deploy to all roads in order to reduce traffic congestion in places like Delhi where traffic congestion has become a big issue.

**Further Enhancement:**

The Future scope includes Profiling of the traffic by storing the data and managing the traffic lights according to the

collected data. The Profiling can also be used for Traffic study and the variation in traffic density throughout the day,

week, month or a year. Further, we can optimize this system for the emergency Vehicles such as Ambulance. The

Traffic data collected can be used to locate different routes for a specific daily vehicle to avoid the congestion

problem

The Future scope includes Profiling of the traffic by storing the data and managing the traffic lights according to the

collected data. The Profiling can also be used for Traffic study and the variation in traffic density throughout the day,

week, month or a year. Further, we can optimize this system for the emergency Vehicles such as Ambulance. The

Traffic data collected can be used to locate different routes for a specific daily vehicle to avoid the congestion

problem

The Future scope includes Profiling of the traffic by storing the data and managing the traffic lights according to the

collected data. The Profiling can also be used for Traffic study and the variation in traffic density throughout the day,

week, month or a year. Further, we can optimize this system for the emergency Vehicles such as Ambulance. The

Traffic data collected can be used to locate different routes for a specific daily vehicle to avoid the congestion

problem

The Future scope includes Profiling of the traffic by storing the data and managing the traffic lights according to the collected data. The Profiling can also be used for Traffic study and the variation in traffic density throughout the day, week, month or a year. Further, we can optimize this system for the emergency Vehicles such as Ambulance. The Traffic data collected can be used to locate different routes for a specific daily vehicle to avoid the congestion problem.

In the future we can use more sophisticated Image processing Algorithms and better Accuracy models using more modules in Ai such as Traffic density, image processing, image scaling, Support vector machines, etc. And priority of vehicles can be issued for police, fire Trucks, ambulances etc.

**APPENDIX**

**Main code:**

Code:

#define F\_CPU 8000000UL

#include <avr/io.h>

#include <util/delay.h>

#define R1 PB0

#define Y1 PB1

#define G1 PB2

#define R2 PB3

#define Y2 PB4

#define G2 PB5

#define R3 PD5

#define Y3 PD4

#define G3 PD3

#define R4 PD2

#define Y4 PD1

#define G4 PD0

int main(void)

{

DDRB = 0xff;

DDRD = 0xff;

DDRC = 0x00;

PORTB = 0x00;

PORTD = 0x00;

while(1)

{

if((PINC&0x01) == 0x01)

{

PORTB |= (1<<G1);

PORTB |= (1<<Y2);

PORTD |= (1<<R3);

PORTD |= (1<<R4);

}

else if((PINC&0x02) == 0x02)

{

PORTB |= (1<<R1);

PORTB |= (1<<G2);

PORTD |= (1<<Y3);

PORTD |= (1<<R4);

}

else if((PINC&0x04) == 0x04)

{

PORTB |= (1<<R1);

PORTB |= (1<<R2);

PORTD |= (1<<G3);

PORTD |= (1<<Y4);

}

else if((PINC&0x08) == 0x08)

{

PORTB |= (1<<Y1);

PORTB |= (1<<R2);

PORTD |= (1<<R3);

PORTD |= (1<<G4);

}

else

{

PORTB = 0x00;

PORTD = 0x00;

PORTB |= (1<<G1);

PORTB |= (1<<Y2);

PORTD |= (1<<R3);

PORTD |= (1<<R4);

\_delay\_us(500000);

PORTB = 0x00;

PORTD = 0x00;

PORTB |= (1<<R1);

PORTB |= (1<<G2);

PORTD |= (1<<Y3);

PORTD |= (1<<R4);

\_delay\_us(500000);

PORTB = 0x00;

PORTD = 0x00;

PORTB |= (1<<R1);

PORTB |= (1<<R2);

PORTD |= (1<<G3);

PORTD |= (1<<Y4);

\_delay\_us(500000);

PORTB = 0x00;

PORTD = 0x00;

PORTB |= (1<<Y1);

PORTB |= (1<<R2);

PORTD |= (1<<R3);

PORTD |= (1<<G4);

\_delay\_us(500000);

PORTB = 0x00;

PORTD = 0x00;

}

}

}

**Other Code which we Experimented with –**

#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 = 5000;  
int yellowDelay = 2000;  
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 ravelled  
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(100000); //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);  
 }  
}

**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 LEDs 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.

#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);

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.

Timer1.attachInterrupt(softInterr);

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();  
 }  
}

‘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.

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;

**REFERENCES**

[1] Udoakah1 Y. N. and Okure I. G. (2017) Design and implementation of a density-based traffic light control with surveillance system, Nigerian Journal of Technology (NIJOTECH), 36(4): 1239 – 1248

[2] Kham N and New (2014) Implementation of modern traffic light control system”, International Journal of Scientific and Research Publications, 4(6): 1-6

[3] Traffic Control Systems Handbook (2005). Prepared for federal highway administration by Dunn Engineering Associates in association with Siemens Intelligent Transportation Systems

[4] Kell J. H. and Fullerton I. J (1998) Manual of Traffic Signal Design Institute of Transportation Engineers, Prentice-Hall, Inc., 138pp

[5] Sinhmar P (2012), "Intelligent traffic light and density control using IR sensors and

microcontroller", International Journal of Advanced Technology and Engineering Research (IJATER), 2(2): 30-35.

[6] Nwoye C. D., Usikalu M. R., Babarimisa I. O, Achuka J. A and Ayara W. A. (2017) Construction of An Automatic Power Switch using Infrared Motion Sensor, Journal of Informatics and Mathematical Sciences, 9(2): 331–337

[7] Ayara W. A, Omotosho T. V, Usikalu M. R, Singh M. S and Suparta W. (2017) Development of a solar charged laboratory bench power supply, Journal of Physics: Conference Series, 852(1): 012044

[8] Usikalu M. R, Shittu A. H and Obafemi L. N (2018) Construction of an intelligent and efficient light control system, International Journal of Mechanical and Production Engineering Research and Development (IJMPERD), 8(4): 1057-1066

[9] A. Dakhole and M. Moon (2013) Design of intelligent traffic control system based

on ARM, International Journal of Advance Research in Computer Science and Management Studies, 1(6): 76-80.

**BIODATA**



Name : M. Dheeraj Sai

Mobile Number : 8688199052

E-mail : madhalamdheeraj.sai2019@vitstudent.ac.in

Permanent Address : 18-8-25/B, Madhura Nagar, Tirupati, AP.



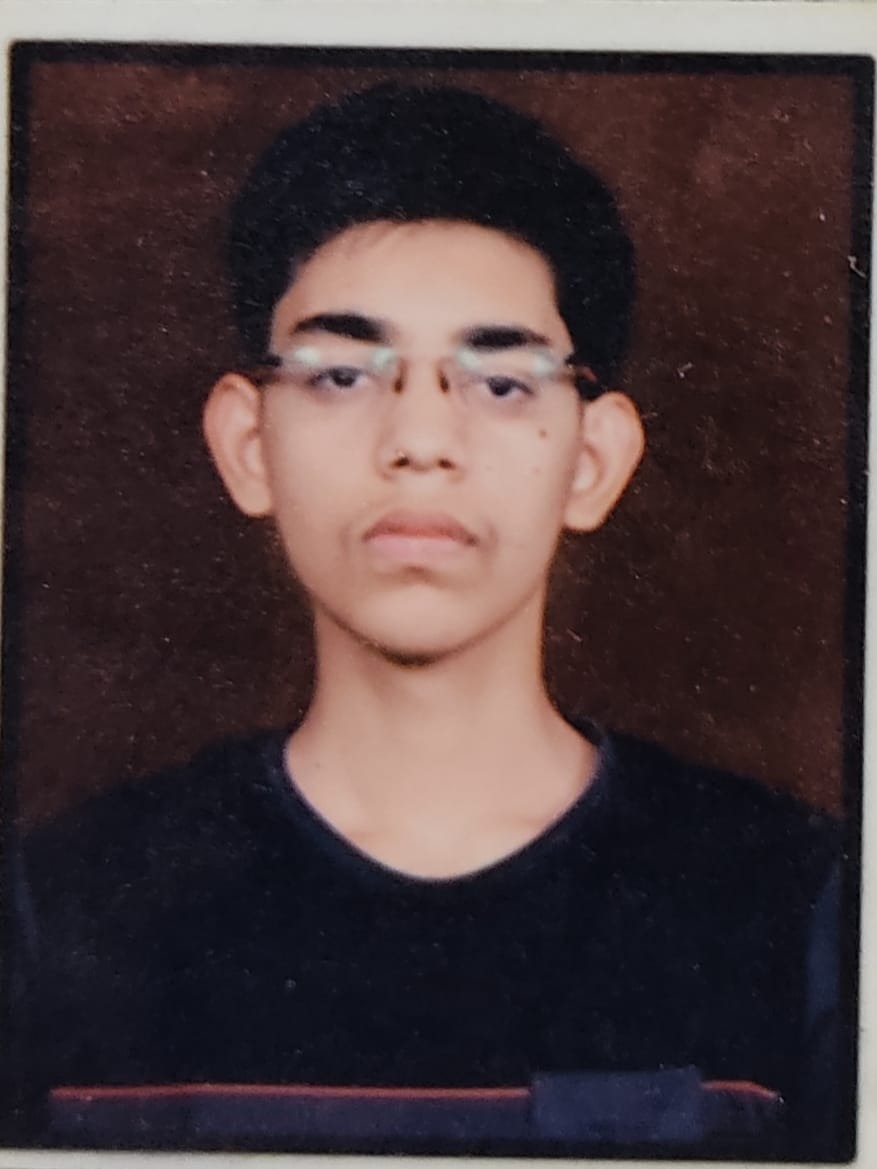
Name :k.manoj kumar

Mobile Number : 8688939744

E-mail :koppolumanoj.kumar2019@vitstudent.ac.in

Permanent Address : LR colony 4th lane sangadigunta ,Guntur,ap

Name : Amar Dixit



Mobile Number : 9369303753

E-mail : amar.dixit2019@vitstudent.ac.in

Permanent Address : B-12 O Block, Kidwai Nagar, Kanpur, UP.

Address

Name : Suhas Reddy .P



Mobile Number : 8688643773

E-mail : penubakasuhas.reddy2019@vitstudent.ac.in

Permanent Address : Sai Padham Appts,MG layout,Nellore,A.P

Address