School of Engineering and Applied Science
Ahmedabad University
Ahmedabad - 380009

Traffic density control : Using Arduino Mega

Group-9
Report-4

Kalagee Anjaria (AU1741052)
Nidhi Golakiya (AU1741062)
Rajvee Kadchha (AU1741064)
Rajvi Patel (AU1741078)

INDEX

1. Introduction

- Motivation : This project deals with the increasing traffic problems in cities. We decided to finalize on this topic due to the following reasons :- Reducing traffic congestion, reducing long time delay, keeps track of vehicles and many more. Also we have seen that due to traffic emergency vehicles (ambulance,fire brigade) get delayed to reach on time.

- Description : We will use IR sensors to sense the limit of the maximum vehicles. The sensors will send the signal to the microcontroller. It will compare which road has the highest density of vehicles and will send the signal to open that particular road for crossing. Thus, reducing the traffic.
  We are also using sound detecting sensors. Whenever it detects sound in form of siren for any emergency vehicle(like ambulance, fire brigade). These sensors will also send signal to microcontroller to make that signal green , so that the

emergency vehicle can pass easily and takes less time.There is also a temperature sensor that can sense the temperature and display it on LCD display. These two are also connected with microcontroller .

- Final Outcome : The final demonstration of our project will be in a small model form. That side of road having the highest density of vehicles will glow a green LED. If the sound detector detects any sound that exceeds the threshold value(which is set by us in the code) then also the green LED will glow, of the road where the sensor senses the sound. The LCD will display the temperature sensed by the sensor. It will also display which side of the crossroad has the green signal glowing.So, this is how our project will work.

2. Market Survey

Our Project : In our project, we are not only focusing on controlling the traffic lights based on the density of vehicles on the road, but also focusing on giving priority to emergency vehicles like ambulance. For this, we are using a sound sensor to detect the sound of the siren so that such vehicles are given priority and they don't need to wait. In addition, we're also using a temperature sensor to sense the temperature at that time. This value of temperature will be displayed on the lcd display. All these components are interfaced to the microcontroller Arduino Mega2560.

On each road of the crossroad, two IR sensors are connected, acting as receiver and transmitter each. One sensor is interfaced at the very starting of that road, and the other one at some distance apart. Density of vehicles can be measured in this way. If on a particular road, both these sensors are detected to be high, that means the density of vehicles is maximum and traffic has reached it's limit.

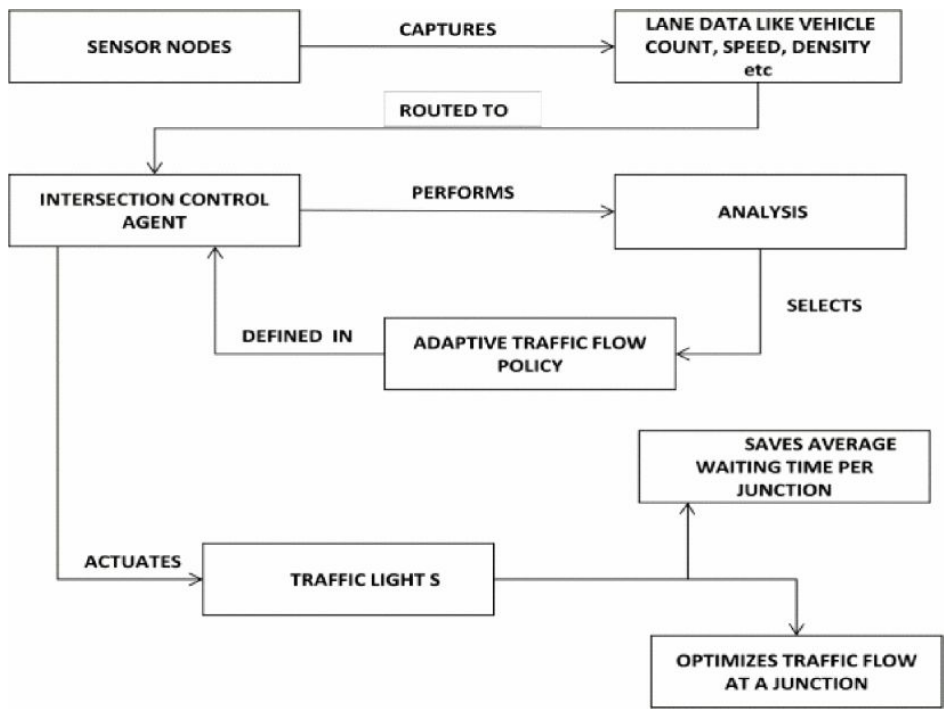
Thus, that road will be given a green signal. Priorities are assigned in this manner and according to them the assigned roads will open. Also, the sound sensor is given a higher priority than the IR sensor.

We have also created a mobile application which is connected through a bluetooth module to the microcontroller. If an exceptional emergency occurs, and we have to open some road, then we can manually do it through this app. Even though according to the priorities assigned, that particular road has a red signal, through this app we can open still open that road. So, basically, it works like an interrupt to the microcontroller.
This is the abstract of what our project contains and how it works. Now we compare our project with an IEEE report about a similar project.

IEEE Report : In this report, they have focused on WSN(Wireless Sensor Networks) for real life applications.
They developed the following algorithms: "*Maximum Intersection Utilization*" and "*Empty lane with Green Light*" are tested in a java based simulated platform called *Green Light District Simulator* (GLD). The average waiting time is the average number of cycles (time in seconds) a vehicle has to wait at the intersection during one round trip of traffic flow. Through this feature of GLD they intend to generate a graph of average waiting time vs. number of cycles

spend for the above algorithms and draw a comparison against the conventional traffic policies.

The following diagram shows the overall design structure of the proposed work :


SENSOR NODES
CAPTURES
LANE DATA LIKE VEHICLE COUNT, SPEED, DENSITY etc
ROUTED TO
INTERSECTION CONTROL AGENT
PERFORMS
ANALYSIS
SELECTS
DEFINED IN
ADAPTIVE TRAFFIC FLOW POLICY
SAVES AVERAGE WAITING TIME PER JUNCTION
ACTUATES
TRAFFIC LIGHT S
OPTIMIZES TRAFFIC FLOW AT A JUNCTION

The two algorithms proposed by them are as follows :

1) Maximum Intersection Utilization

1. Identify the roads on the intersection in a sequence numbered from 0 to 3 clockwise.
2. Start by initializing all the roads to a red state.

3. Assume any 1 of the roads to be the start, the cycle continues after 4th stage.
4. In every stage 2 roads (opposite to each other) are regulated
5. Vehicles on every lane wait for 3 cycles for their turn (as in the normal traffic flow) whereas the ones taking a left turn wait for 2 cycles each.
6. If collision is not the concern of algorithm, free left can be provided for all the stages.
7. Algorithm follows a round robin schedule giving equal time slots to each road.
8. Continue operation till a pause or stop command is issued by the user.

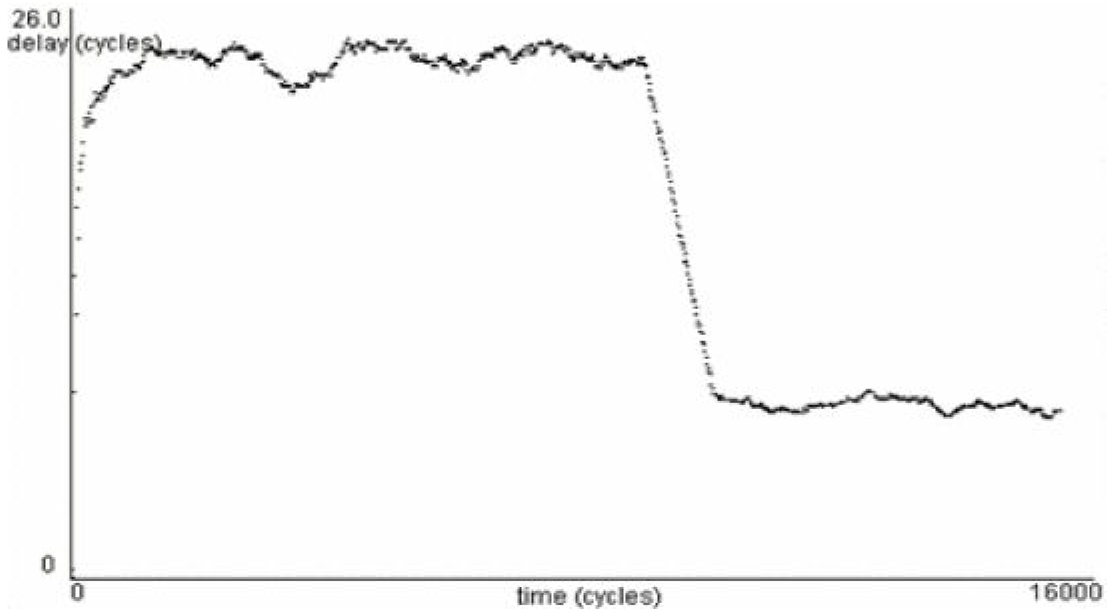
3) Empty lane with Green Light

1. Identify the roads on the intersection in a sequence numbered from 0 to 3 clockwise.
2. Start by initializing all the roads to a red state.
3. Round robin scheduling again with equal time slots of 20 seconds.
4. When there is no vehicle on 1 road (say road E) and that road is given a GREEN signal. Assign RED to all the lanes of that road (E) Compare traffic on each of the other roads (N,W & S) and assign GREEN for road with MAX traffic for biased time

(5 seconds) Check if vehicles have arrived on road E after step b If vehicles present, assign GREEN to road E else assign GREEN to the road with next MAX vehicle count for 5 seconds.

5. Repeat steps (a) through (d) till time slot (20 seconds - variable sign messages can be used for lane users) expires for that particular road.
6. Shift the turn to the next road and repeat steps 3 & 4 for the same.
7. Continue operation *till* a pause or stop command is issued by the user.

Comparison graph for the signals based on the results :

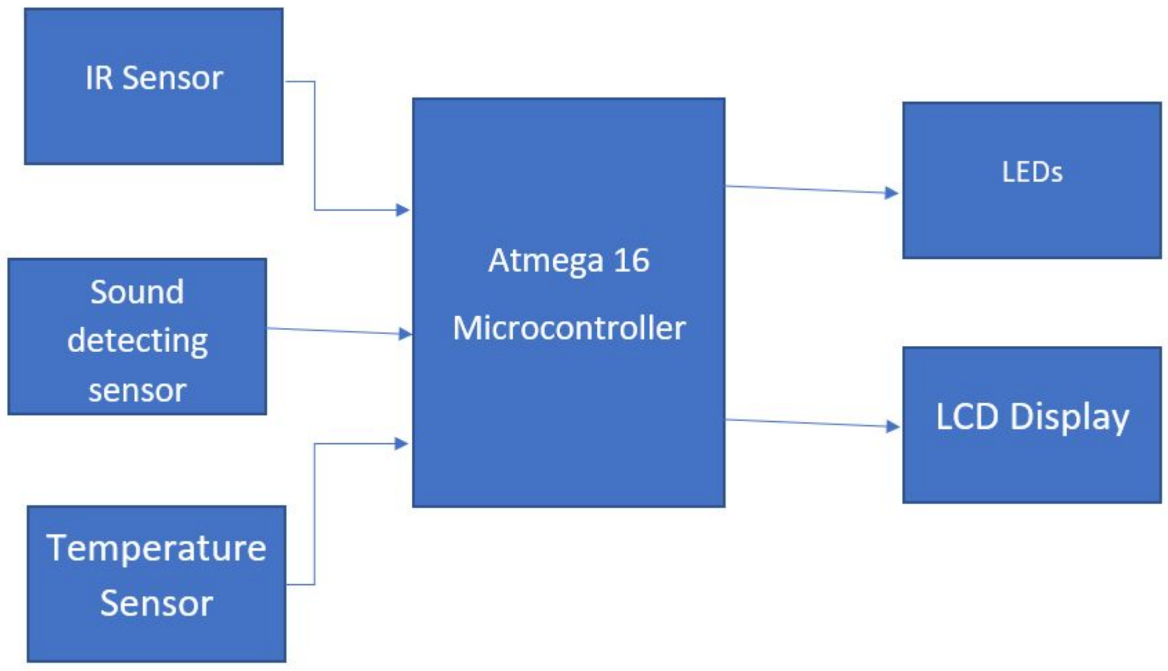


According to them, this work may also prove to be saving on fuel. Further, the Intelligent Traffic System along with other technologies like RFID, GPRS and GPS can be a potential solution for traffic control.

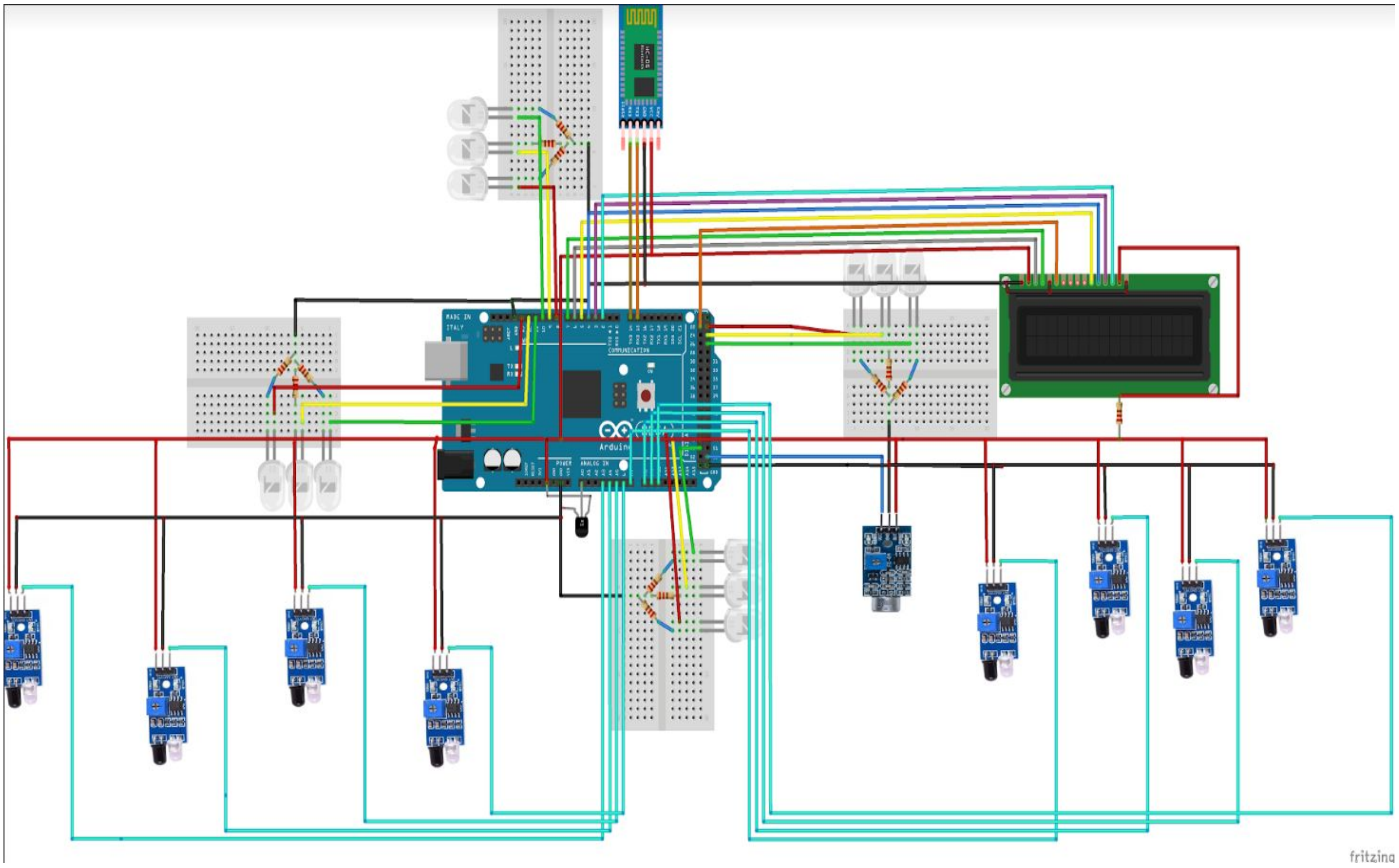
The following is the link to this report :

https://ieeexplore.ieee.org/document/6616429

3. <u>Block Diagram</u>



4.1 Circuit Diagram



4.2 Components

1) Arduino mega 2560 (x 1)

2) IR sensors(x 8)
3) Sound sensor (x 1)
4) 5mm LED – Red,Green,Yellow (x 12)
5) Resistors – 1K ohms
6) Jumper wires
7) Breadboard
8) LCD Display (x 1)
9) LM35 (Temperature sensor) (x 1)
10) Bluetooth module HC05 (x 1)

4.3 Selection Criteria of components

1) IR sensors : We have selected IR sensors so that it can sense which road has the maximum density of vehicles and can open the signal for that particular road.

2) Sound sensors : Sound sensors are used in this project so that whenever it detects the sound of any emergency vehicle, we can give priority to it and immediately open that road.

3) LM35 : It is a temperature sensor which senses the temperature of the nearby atmosphere. This is used just as an additional facility so that people passing through the crossroad

can know what exactly the status of the temperature at that moment.

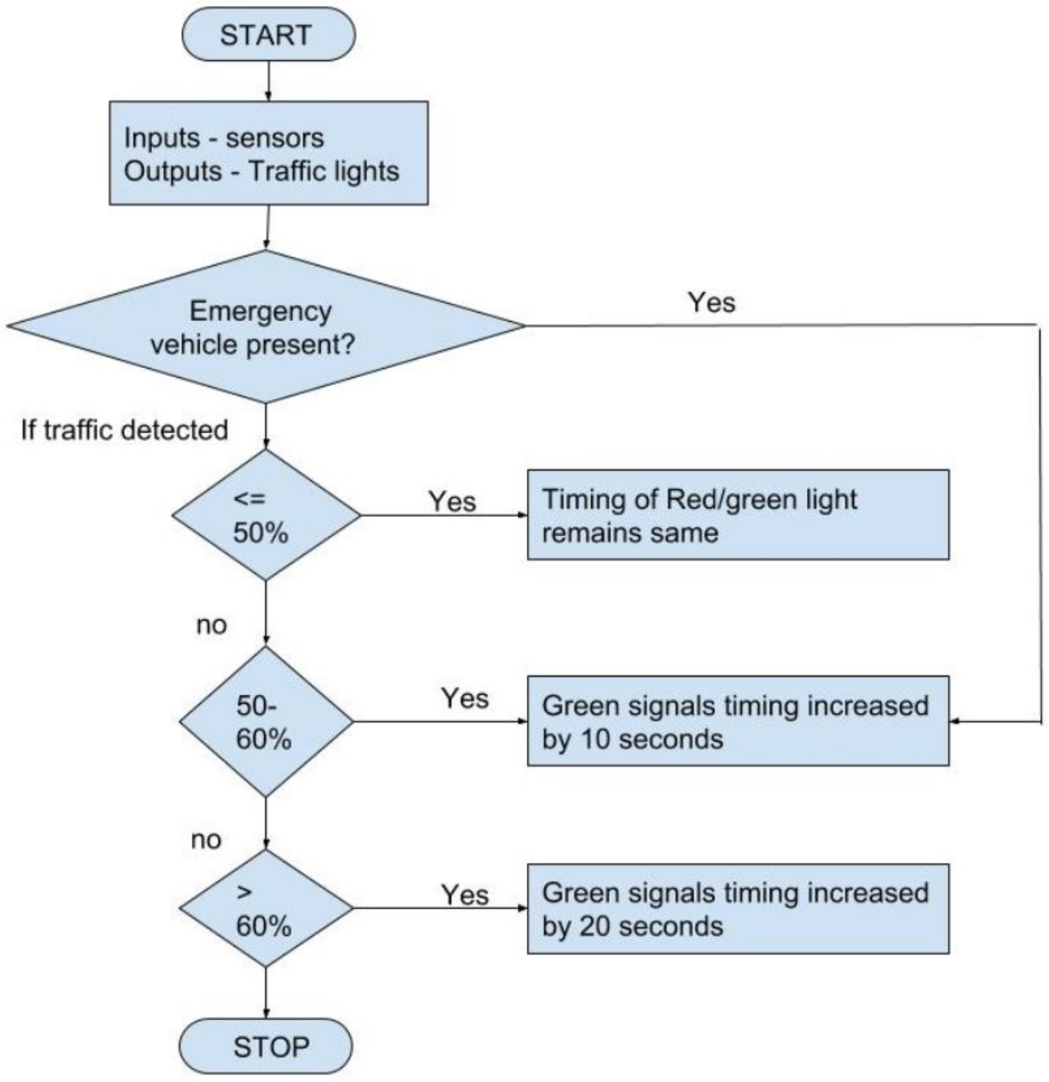
4) LCD display : This is used to display the temperature that is sensed by LM35. Also, it is used to display which side of the crossroad has the green signal, so that the corresponding vehicles can cross.

5) LEDs : LEDs are needed to represent the three types of signals i.e. red, yellow and green.

5. Arduino features

Specifications	**Arduino**	**Raspberry pi**	**Beaglebone**
Model tested	R3	Model B	Rev 5
Size	2.95”x2.10”	3.37”x2.125”	3.4”x2.1”
Processor	Atmega328	ARM11	ARM Cortex-A8
Clock speed	16MHz	700MHz	700MHz
RAM	2KB	256MB	256MB
Flash	32KB	(SD card)	4GB(microSD)
EEPROM	1KB	-	-
Input voltage	7-12 V	5 V	5 V
Min power	42mA(0.3W)	700mA(3.5W)	170mA(0.85W)
Digital GPIO	14	8	66
Analog input	6-10 bit	N/A	7-12 bit
PWM	6	-	8

6. Flowchart


START
Inputs - sensors
Outputs - Traffic lights
Emergency vehicle present?
Yes
If traffic detected
<= 50%
Yes
Timing of Red/green light remains same
no
50-60%
Yes
Green signals timing increased by 10 seconds
no
> 60%
Yes
Green signals timing increased by 20 seconds
STOP

7. <u>Sensors</u>

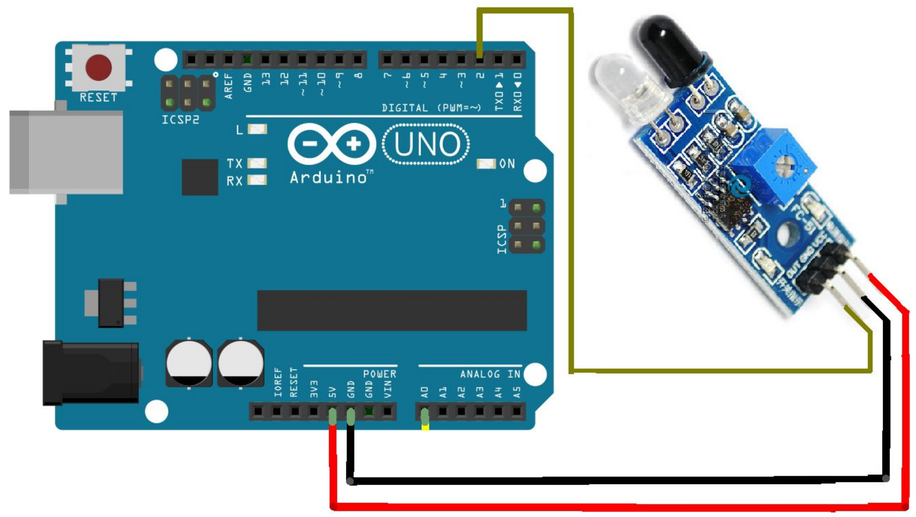
7.1 Ir sensor(Infrared Sensor)

1) Operating principles

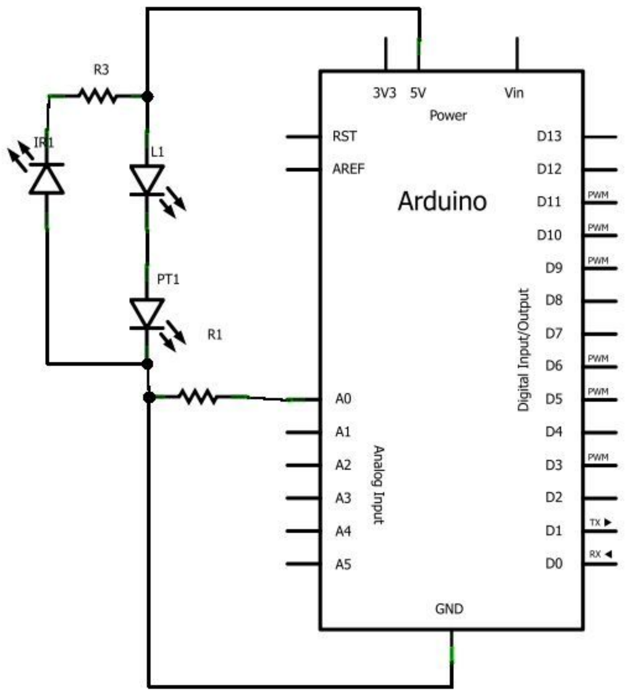
The physics behind infrared sensors is governed by three laws:

1. Planck’s radiation law: Every object at a temperature T not equal to 0 K emits radiation
2. Stephan Boltzmann Law: The total energy emitted at all wavelengths by a black body is related to the absolute temperature
3. Wein’s Displacement Law: Objects of different temperature emit spectra that peak at different wavelengths

2) Interfacing with arduino



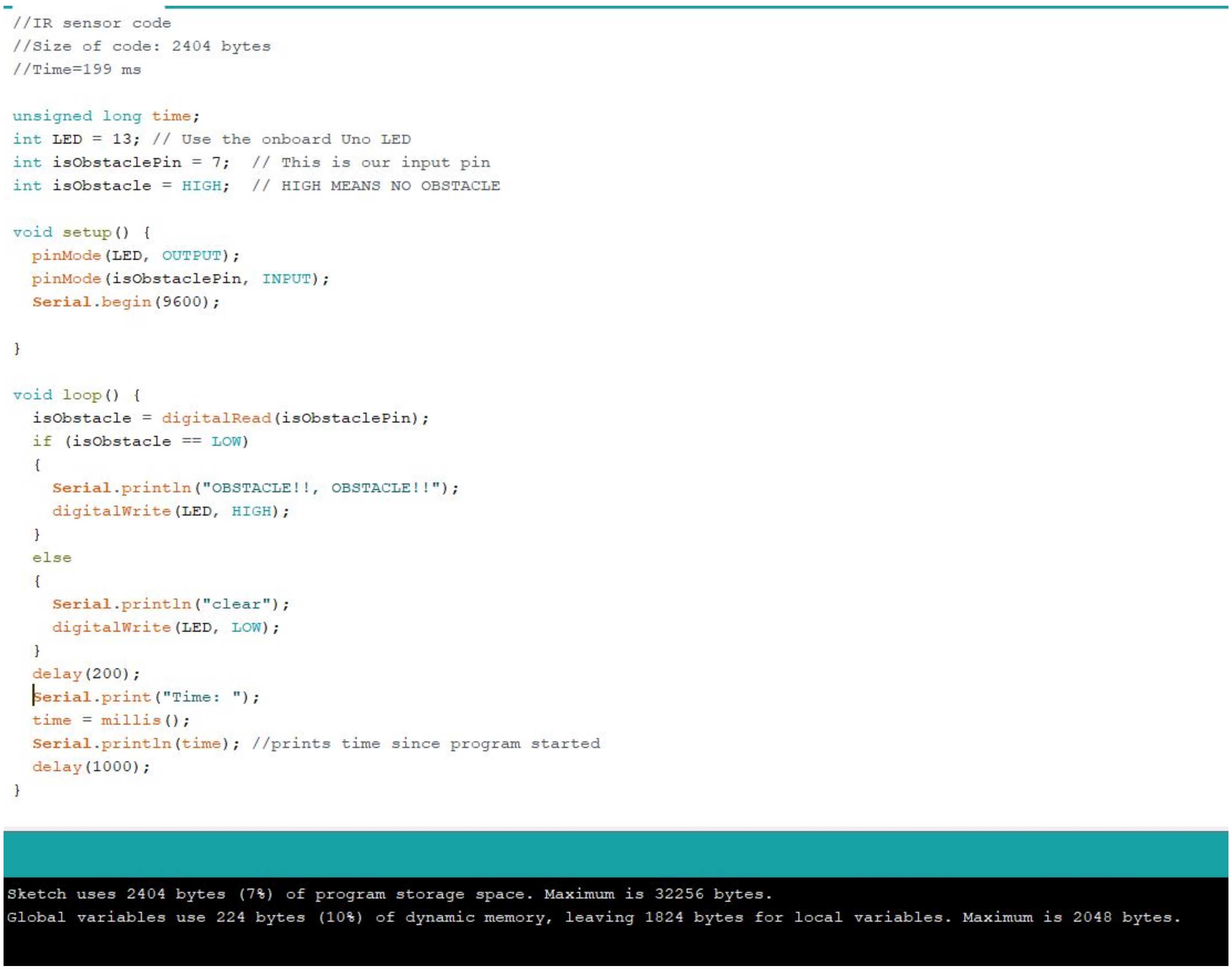
3) Pin Diagram



4) Measurement range, power ratings

- Near infrared region — 700 nm to 1400 nm — IR sensors, fiber optic
- Mid infrared region — 1400 nm to 3000 nm — Heat sensing
- Far infrared region — 3000 nm to 1 mm — Thermal imaging
  - It has a receiving range of 2 ~ 30 cm.
  - working voltage of 3 .3 V to 5 V.
  - working current of 1.5 mA.
  - operating temperature range from -25 C to 85C.

5) C code



```
//IR sensor code
//Size of code: 2404 bytes
//Time=199 ms

unsigned long time;
int LED = 13; // Use the onboard Uno LED
int isObstaclePin = 7;  // This is our input pin
int isObstacle = HIGH;  // HIGH MEANS NO OBSTACLE

void setup() {
  pinMode(LED, OUTPUT);
  pinMode(isObstaclePin, INPUT);
  Serial.begin(9600);

}

void loop() {
  isObstacle = digitalRead(isObstaclePin);
  if (isObstacle == LOW)
  {
    Serial.println("OBSTACLE!!, OBSTACLE!!");
    digitalWrite(LED, HIGH);
  }
  else
  {
    Serial.println("clear");
    digitalWrite(LED, LOW);
  }
  delay(200);
  Serial.print("Time: ");
  time = millis();
  Serial.println(time); //prints time since program started
  delay(1000);
}
```

```
Sketch uses 2404 bytes (7%) of program storage space. Maximum is 32256 bytes.
Global variables use 224 bytes (10%) of dynamic memory, leaving 1824 bytes for local variables. Maximum is 2048 bytes.
```

Size of Code : 2404 bytes

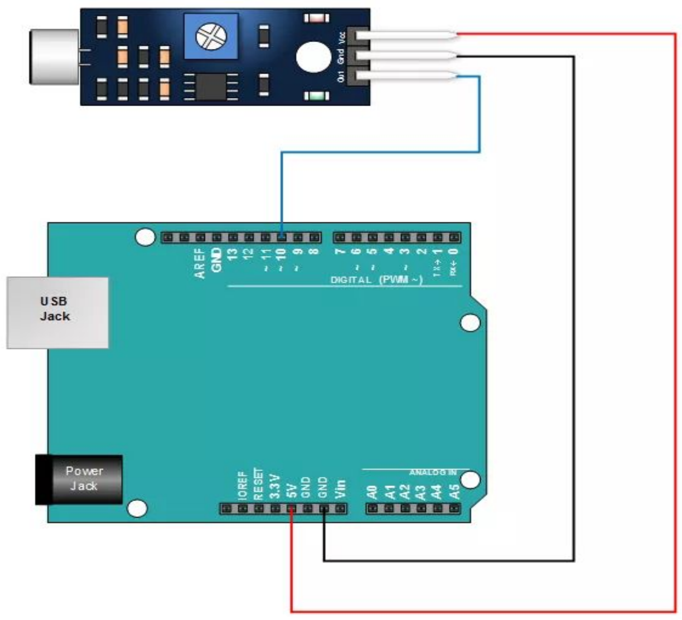
Execution time : 199 ms

7.2 Sound sensor

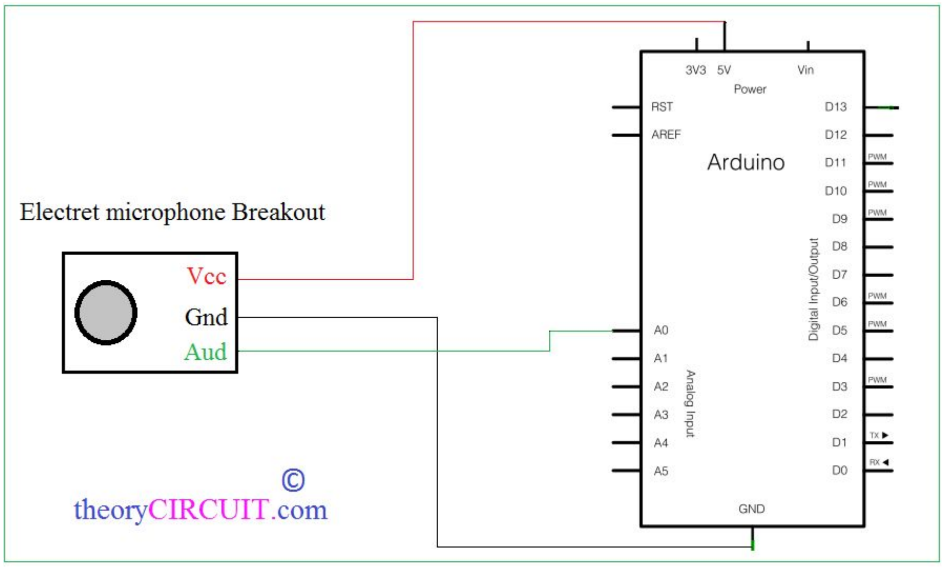
1) Operating Principles

LM386 is an audio power amplifier with features of low power consumption, adjustable voltage gain, wide voltage power supply, less requirements on peripheral components and minimum total harmonic distortion. LM386 can be applied to the consumer products with low voltage requirement. To minimize the number of the peripheral components in used, the voltage gain should be set to 20. Connecting a resistant and a capacitor externally between the Pin1 and the Pin8, it is able to configure the voltage gain to any value within the range of 0-200.

2) Interfacing with Arduino



3) Pin diagram

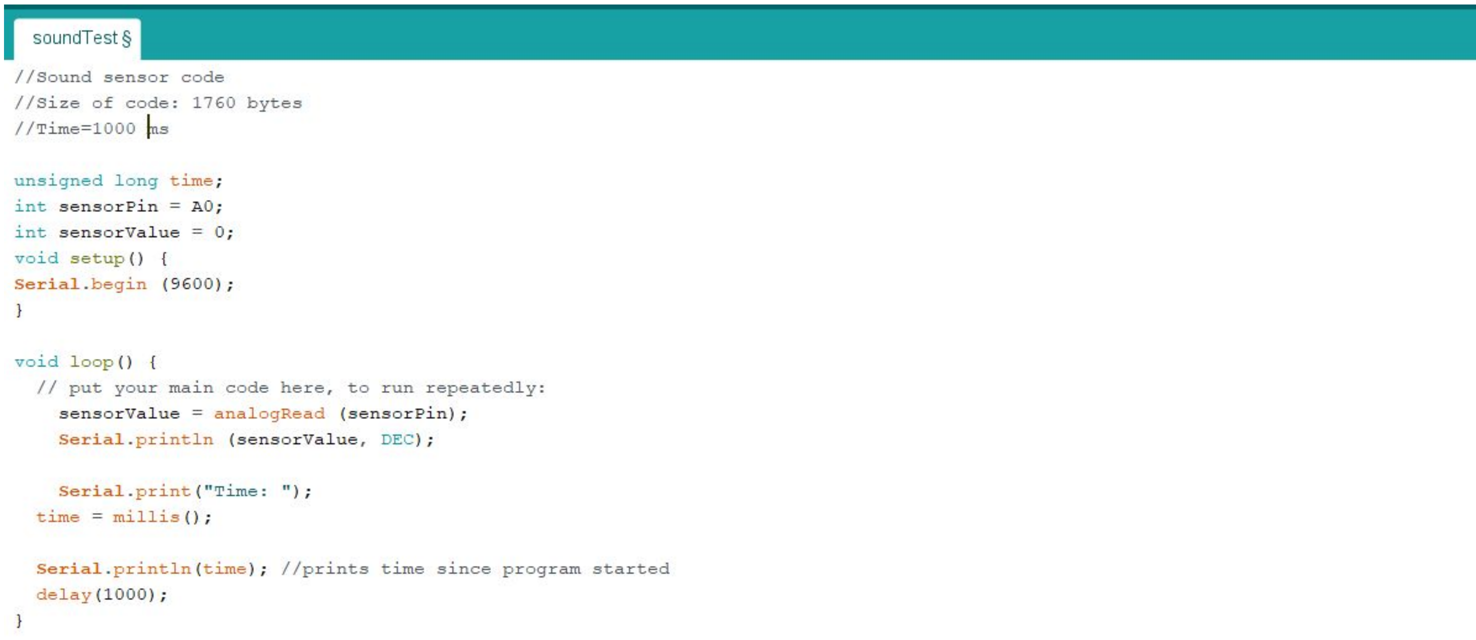


4) Dimensions, power ratings

1) Operating voltage - 3.3V-5.3V
2) Dimensions - 39.0mm*21.0mm

3) Fixing hole - size 2.0mm

5) C code



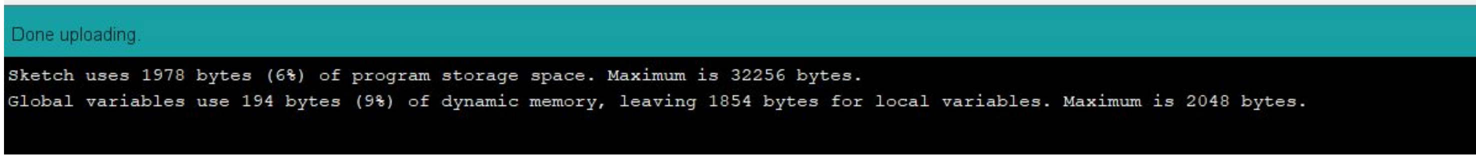
```
//Sound sensor code
//Size of code: 1760 bytes
//Time=1000 ms

unsigned long time;
int sensorPin = A0;
int sensorValue = 0;
void setup() {
Serial.begin (9600);
}

void loop() {
  // put your main code here, to run repeatedly:
    sensorValue = analogRead (sensorPin);
    Serial.println (sensorValue, DEC);

    Serial.print("Time: ");
  time = millis();

  Serial.println(time); //prints time since program started
  delay(1000);
}
```



```
Sketch uses 1978 bytes (6%) of program storage space. Maximum is 32256 bytes.
Global variables use 194 bytes (9%) of dynamic memory, leaving 1854 bytes for local variables. Maximum is 2048 bytes.
```

Size of Code : 1760 bytes

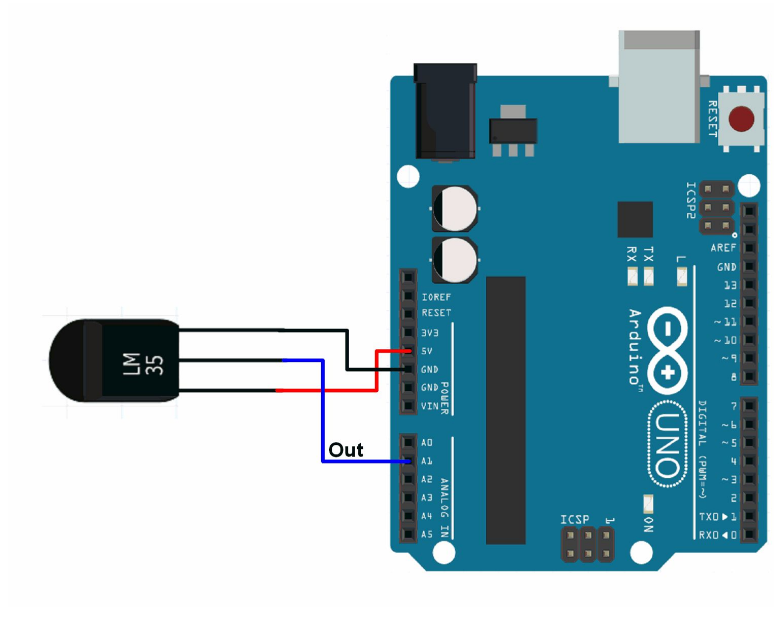
Execution time : 1000 ms

7.3 Temperature sensor (LM35)

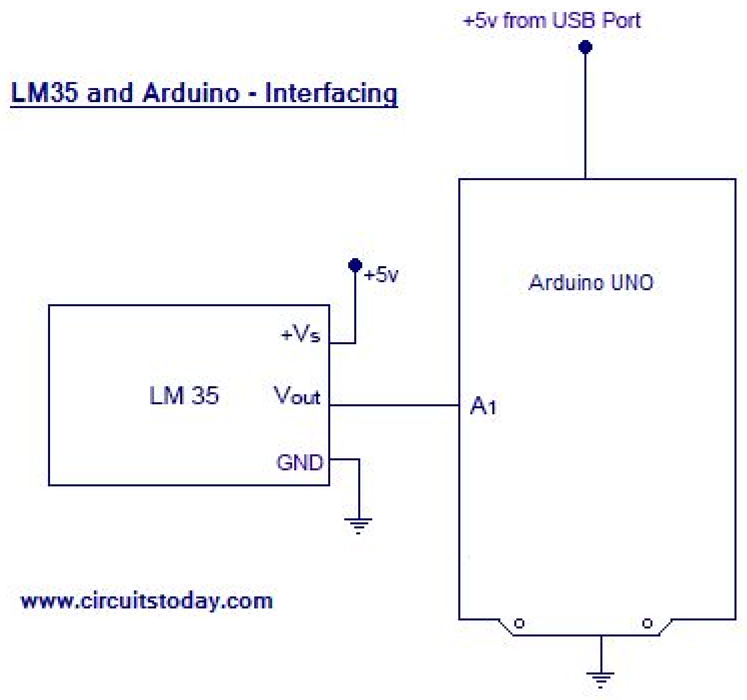
1) Operating Principles

In the features of lm35 it is given to be +10 mills volt per degree centigrade. It means that with increase in output of 10 mills volt by the sensor vout pin the temperature value increases by one. For example if the sensor is outputting 100 mills volt at vout pin the temperature in centigrade will be 10 degree centigrade. The same goes for the negative temperature reading. If the sensor is outputting -100 mills volt the temperature will be -10 degree Celsius.

2) Interfacing with Arduino



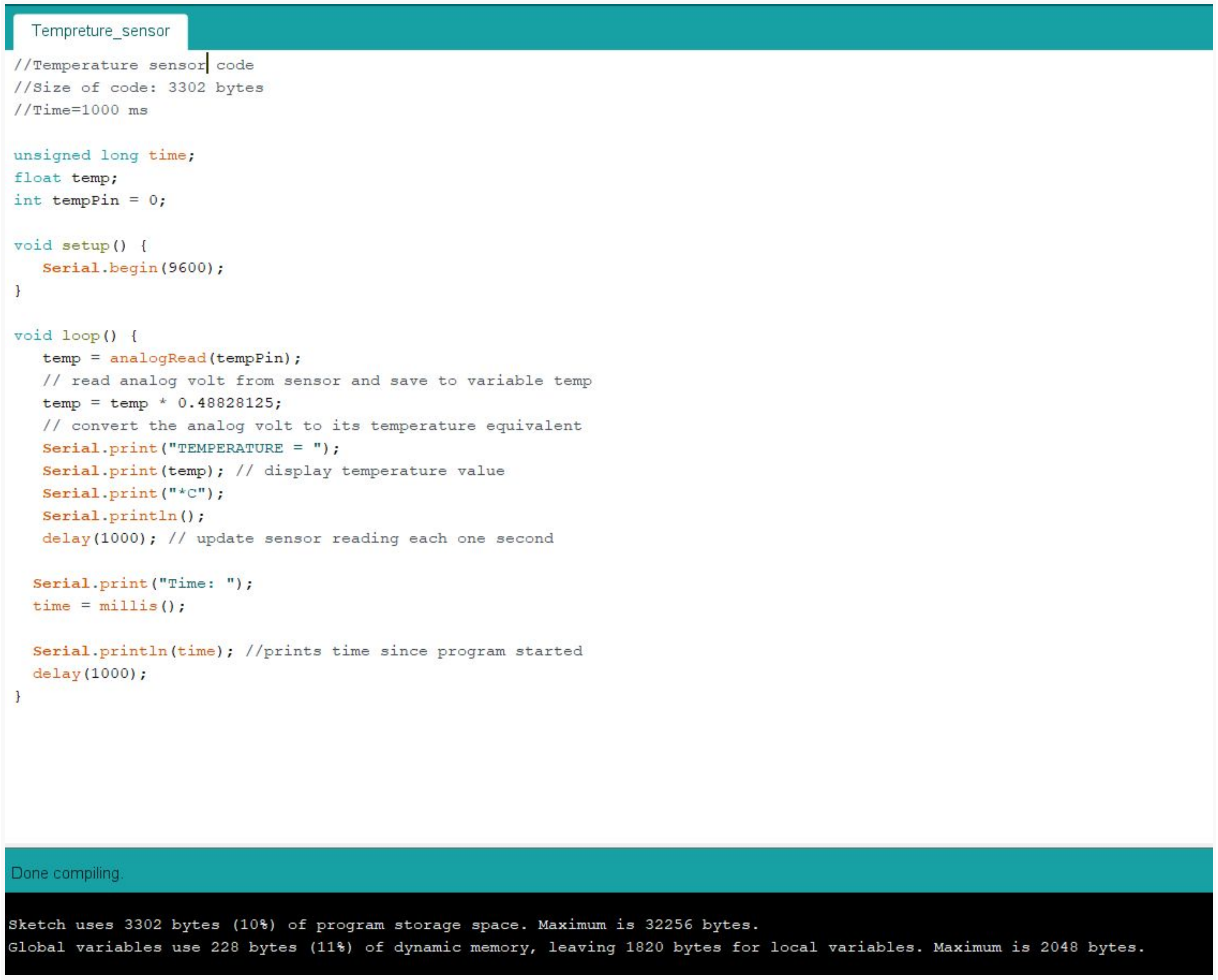
3) Pin diagram



4)Measurement range, power ratings

- Linear + 10-mV/°C Scale Factor
- 0.5°C Ensured Accuracy (at 25°C)
- Rated for Full −55°C to 150°C Range
- Suitable for Remote Applications
- Operates from 4 V to 30 V

5) C code



```
//Temperature sensor code
//Size of code: 3302 bytes
//Time=1000 ms

unsigned long time;
float temp;
int tempPin = 0;

void setup() {
   Serial.begin(9600);
}

void loop() {
   temp = analogRead(tempPin);
   // read analog volt from sensor and save to variable temp
   temp = temp * 0.48828125;
   // convert the analog volt to its temperature equivalent
   Serial.print("TEMPERATURE = ");
   Serial.print(temp); // display temperature value
   Serial.print("*C");
   Serial.println();
   delay(1000); // update sensor reading each one second

  Serial.print("Time: ");
  time = millis();

  Serial.println(time); //prints time since program started
  delay(1000);
}
```

Done compiling.

```
Sketch uses 3302 bytes (10%) of program storage space. Maximum is 32256 bytes.
Global variables use 228 bytes (11%) of dynamic memory, leaving 1820 bytes for local variables. Maximum is 2048 bytes.
```

Size os Code : 3302 bytes

Execution time : 1000 ms

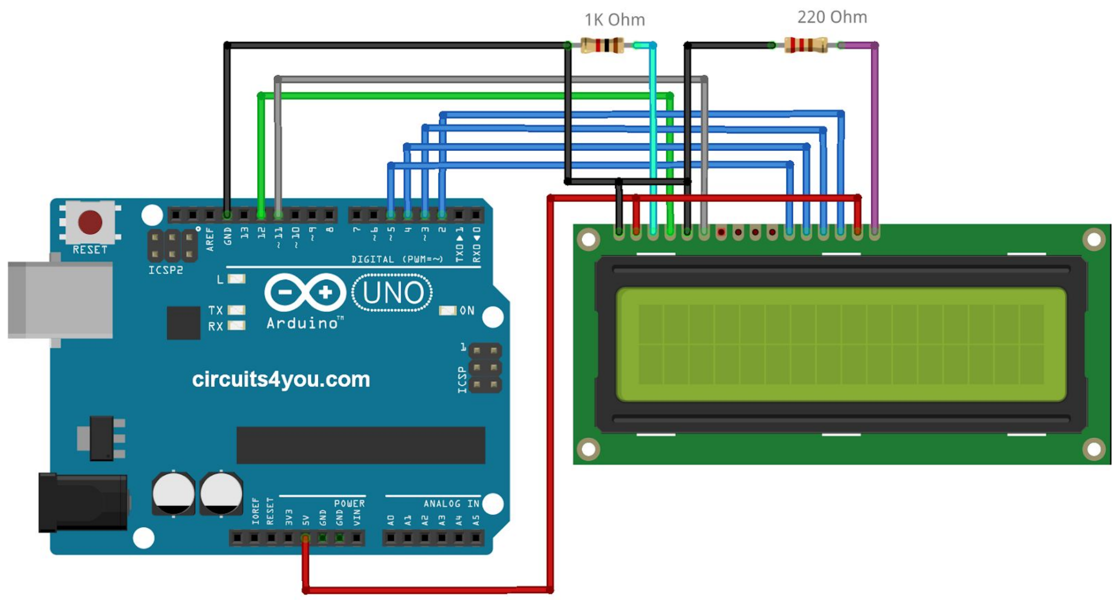
8. Displays

8.1 Lcd Displays

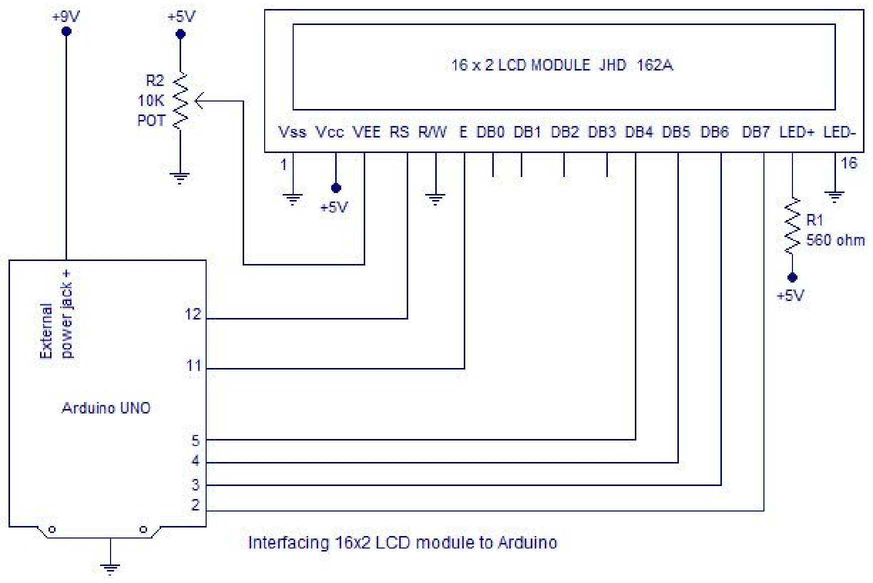
1) Operating Principles

The principle behind the LCD's is that when an electrical current is applied to the liquid crystal molecule, the molecule tends to untwist. This causes the angle of light which is passing through the molecule of the polarized glass and also cause a change in the angle of the top polarizing filter. As a result a little light is allowed to pass the polarized glass through a particular area of the LCD. Thus that particular area will become dark compared to other. The LCD works on the principle of blocking light. While constructing the LCD's, a reflected mirror is arranged at the back. An electrode plane is made of indium-tin oxide which is kept on top and a polarized glass with a polarizing film is also added on the bottom of the device. The complete region of the LCD has to be enclosed by a common electrode and above it should be the liquid crystal matter.

2) Interfacing with arduino



3) Pin diagram


+9V
+5V
R2
10K
POT
16 x 2 LCD MODULE JHD 162A
Vss Vcc VEE RS R/W E DB0 DB1 DB2 DB3 DB4 DB5 DB6 DB7 LED+ LED-
1
16
+5V
R1
560 ohm
+5V
External power jack +
12
11
Arduino UNO
5
4
3
2

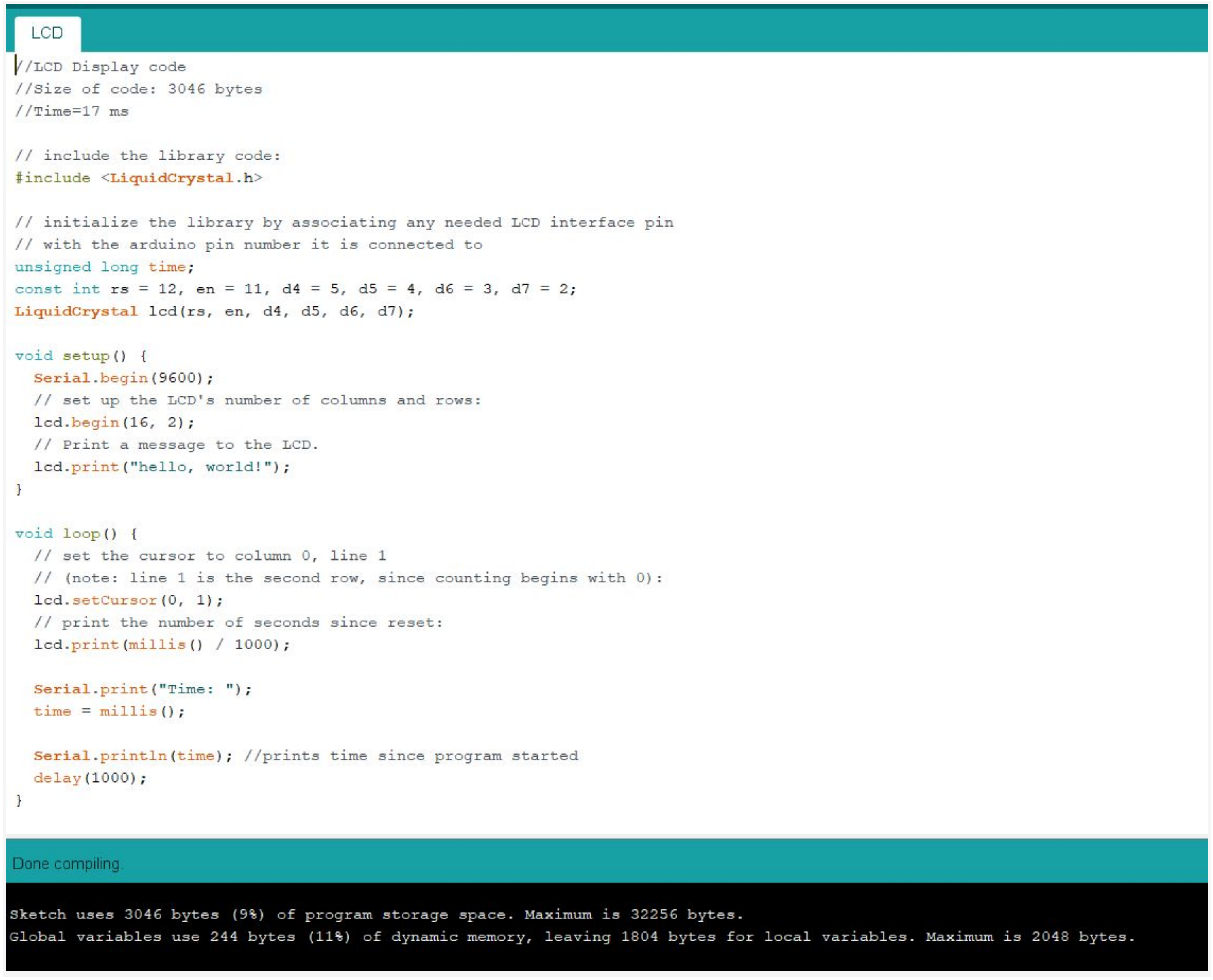
Interfacing 16x2 LCD module to Arduino

4) Physical dimensions, power ratings

5 x 8 dots with cursor

+5 V power supply

5) C code



```
//LCD Display code
//Size of code: 3046 bytes
//Time=17 ms

// include the library code:
#include <LiquidCrystal.h>

// initialize the library by associating any needed LCD interface pin
// with the arduino pin number it is connected to
unsigned long time;
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {
  Serial.begin(9600);
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("hello, world!");
}

void loop() {
  // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting begins with 0):
  lcd.setCursor(0, 1);
  // print the number of seconds since reset:
  lcd.print(millis() / 1000);

  Serial.print("Time: ");
  time = millis();

  Serial.println(time); //prints time since program started
  delay(1000);
}
```

Done compiling.

```
Sketch uses 3046 bytes (9%) of program storage space. Maximum is 32256 bytes.
Global variables use 244 bytes (11%) of dynamic memory, leaving 1804 bytes for local variables. Maximum is 2048 bytes.
```

Size os Code : 3046 bytes

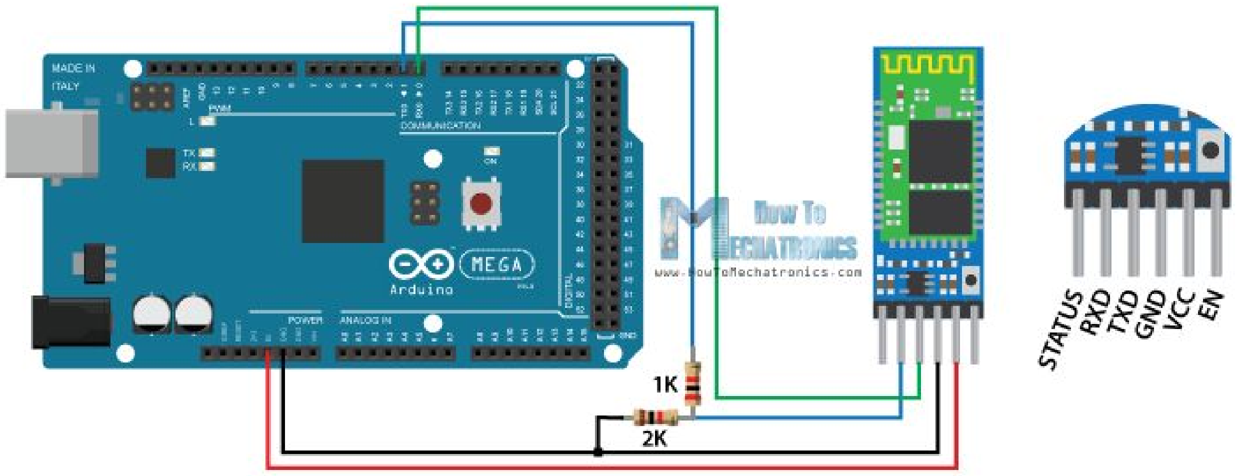
Execution time : 17 ms

9. Details of supporting tools

9.1 Technical Details

Bluetooth module

Bluetooth module circuit diagram



MIT app

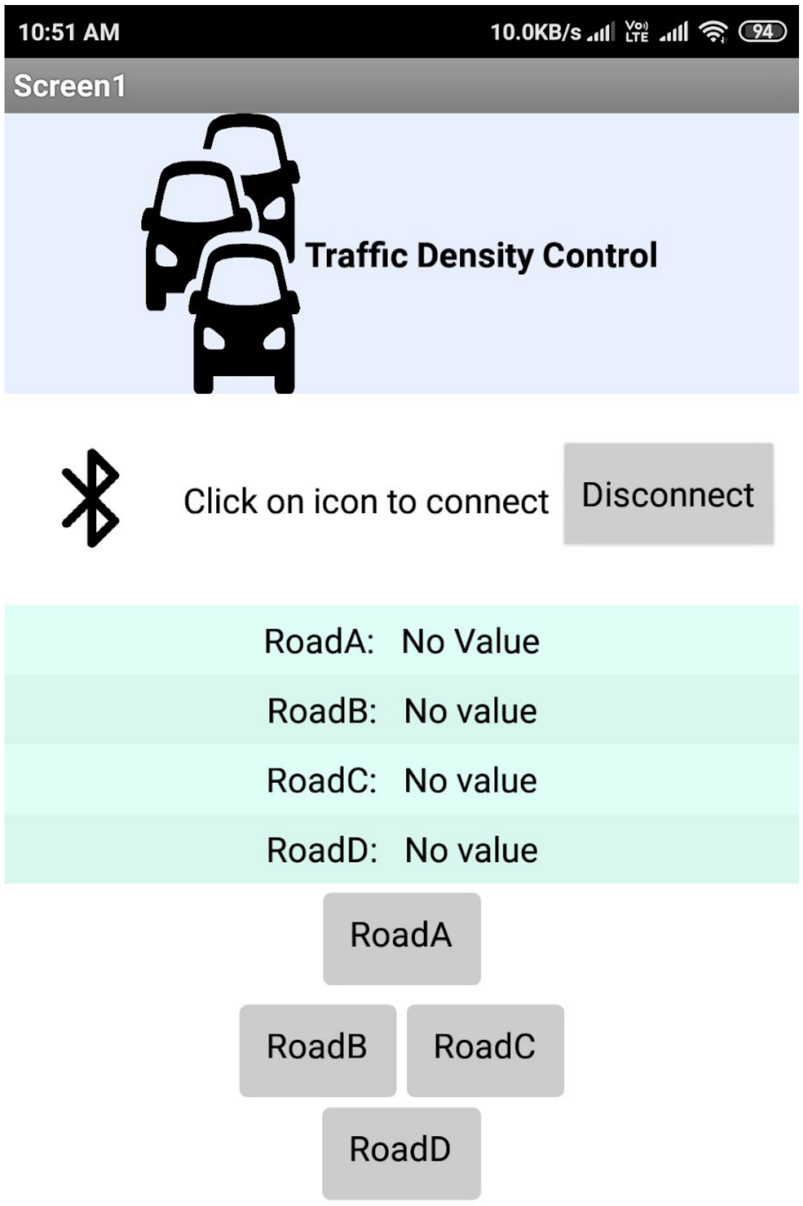
App Inventor for Android is an open-source web application originally provided by Google, and now maintained by the Massachusetts Institute of Technology (MIT).

It allows to create software applications for the Android operating system (OS). It uses a graphical interface which allows users to drag-and-drop visual objects to create an application that can run on Android devices.

9.2 Photo of app

Icon of the app

Screenshot of the app



10:51 AM 10.0KB/s 94

Screen1

Traffic Density Control

Click on icon to connect

Disconnect

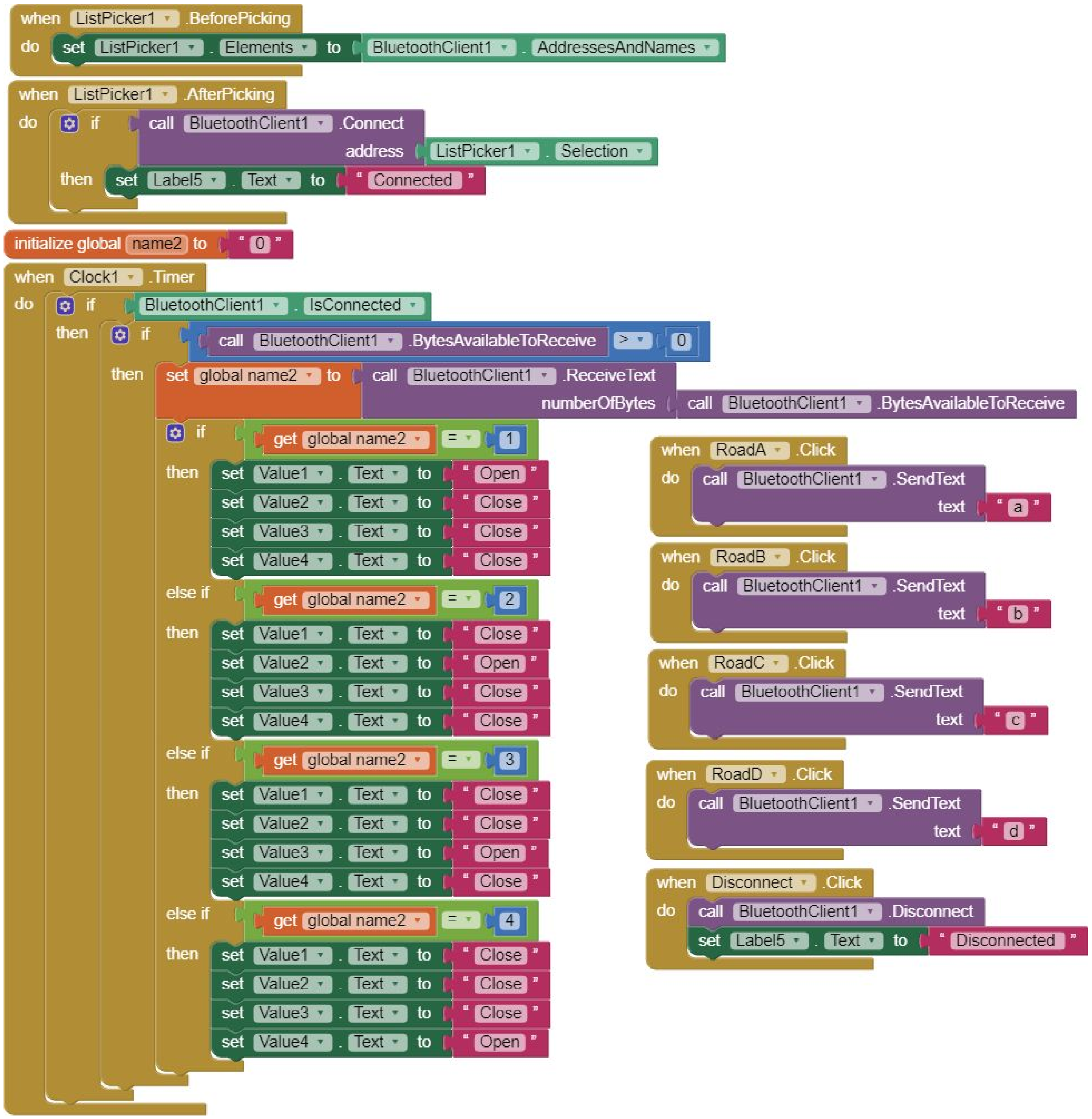
RoadA:	No Value
RoadB:	No value
RoadC:	No value
RoadD:	No value

RoadA

RoadB RoadC

RoadD

9.3 Code of the app



```
when ListPicker1 .BeforePicking
do  set ListPicker1 . Elements to  BluetoothClient1 . AddressesAndNames

when ListPicker1 .AfterPicking
do  if   call BluetoothClient1 .Connect
                                address  ListPicker1 . Selection
    then set Label5 . Text to " Connected "

initialize global name2 to " 0 "

when Clock1 .Timer
do  if   BluetoothClient1 . IsConnected
    then if   call BluetoothClient1 .BytesAvailableToReceive > 0
         then set global name2 to  call BluetoothClient1 .ReceiveText
                                        numberOfBytes  call BluetoothClient1 .BytesAvailableToReceive
              if   get global name2 = 1
              then set Value1 . Text to " Open "
                   set Value2 . Text to " Close "
                   set Value3 . Text to " Close "
                   set Value4 . Text to " Close "
              else if  get global name2 = 2
              then set Value1 . Text to " Close "
                   set Value2 . Text to " Open "
                   set Value3 . Text to " Close "
                   set Value4 . Text to " Close "
              else if  get global name2 = 3
              then set Value1 . Text to " Close "
                   set Value2 . Text to " Close "
                   set Value3 . Text to " Open "
                   set Value4 . Text to " Close "
              else if  get global name2 = 4
              then set Value1 . Text to " Close "
                   set Value2 . Text to " Close "
                   set Value3 . Text to " Close "
                   set Value4 . Text to " Open "

when RoadA .Click
do  call BluetoothClient1 .SendText
                          text " a "

when RoadB .Click
do  call BluetoothClient1 .SendText
                          text " b "

when RoadC .Click
do  call BluetoothClient1 .SendText
                          text " c "

when RoadD .Click
do  call BluetoothClient1 .SendText
                          text " d "

when Disconnect .Click
do  call BluetoothClient1 .Disconnect
    set Label5 . Text to " Disconnected "
```

10) Complete Program

```
#include <SoftwareSerial.h>
#include<LiquidCrystal.h>

SoftwareSerial BTserial(0, 1);

const int rn=12,en=11,d4=2,d5=3,d6=4,d7=5;
LiquidCrystal lcd(rn,en,d4,d5,d6,d7);
const int inPin = 8;
#define ledA1 22
#define ledA2 23
#define ledA3 24

#define ledB1 25
#define ledB2 26
#define ledB3 27
#define ledC1 28
#define ledC2 29
#define ledC3 30
#define ledD1 31
#define ledD2 32
#define ledD3 33
int sensorPin = A9;
int sensorValue = 0;
int a1, a2, b1, b2, c1, c2, d1, d2;
int led = 35;
int state;
int msg1 = 1,msg2=2,msg3=3,msg4=4;

void setup() {
  BTserial.begin(9600);
  Serial.begin (9600);
  pinMode(led, OUTPUT);
  pinMode(ledA1, OUTPUT);
  pinMode(ledA2, OUTPUT);
  pinMode(ledA3, OUTPUT);

  pinMode(ledB1, OUTPUT);
  pinMode(ledB2, OUTPUT);
  pinMode(ledB3, OUTPUT);

  pinMode(ledC1, OUTPUT);
  pinMode(ledC2, OUTPUT);
  pinMode(ledC3, OUTPUT);

  pinMode(ledD1, OUTPUT);
  pinMode(ledD2, OUTPUT);
  pinMode(ledD3, OUTPUT);
  lcd.begin(16,2);
}

void loop() {
int value = analogRead(inPin);                    //reading pin 8
lcd.setCursor(0,1);                               //setting cursor at 0th row and 1st column of lcd
float millivolts = (value / 1024.0) * 5000;       //calculation of temperature
float celsius = millivolts / 10;                  //temperature in celsius
lcd.clear();
lcd.setCursor(0,0);                               //setting cursor at 0th row and 0th column of lcd
lcd.print(celsius);
lcd.print("C");
lcd.setCursor(0,1);                               //setting cursor at 0th row and 1st column of lcd
delay(1000);                                      //delay
```

```
  readSoundSensor();                    //calling function sound sensor which reads value  of sound sensor
  readSensor();                         //calling function readSensor which reads values of 8 ir sensor.
  if(a1==1 && a2==1){                   //High traffic on A
    roadState();                        //calling function readState
    roadAopen();                        //Opening road A
  }
  else if((b1==1 && b2==1) && (a1==0 || a2==0)){     //High traffic on B
    roadState();
    roadBopen();                        //Opening road B
  }
  else if((c1==1 && c2==1) && (a1==0 || a2==0) && (b1==0 || b2==0)){//High traffic on
C
    roadState();
    roadCopen();                        //Opening road C
  }
  else if((d1==1 && d2==1) && (a1==0 || a2==0) && (b1==0 || b2==0) && (c1==0
||c2==0)){                              //High traffic on D
    roadState();
    roadDopen();                        //Opening road D
  }
  else if((a1==1 && a2==0) && (b1==0 || b2==0) && (c1==0 ||c2==0) && (d1==0 &&
d2==0)){                                //Moderate traffic on road A and zero traffic on other road
    roadState();
    roadAopen();                        //Opening road A
  }
  else if((b1==1 && b2==0) && (a1==0 || a2==0) && (c1==0 ||c2==0) && (d1==0 &&
d2==0)){                                //Moderate traffic on road B and zero traffic on other road
    roadState();
    roadBopen();                        //Opening road B
  }
  else if((c1==1 && c2==0) && (a1==0 || a2==0) && (b1==0 ||b2==0) && (d1==0 &&
d2==0)){                                //Moderate traffic on road C and zero traffic on other road
```

```
   roadState();
    roadCopen();           //Opening road C
  }
  else if((d1==1 && d2==0) && (a1==0 || a2==0) && (b1==0 ||b2==0) && (c1==0 &&
c2==0)){                               //Moderate traffic on road D and zero traffic on other road
    roadState();
    roadDopen();                   //Opening road D
  }
  else if((a1==1 && a2==0) && (b1==1 || c1==1 || d1==1) && (b2==0 && c2==0 &&
d2==0)){                               //Moderate traffic on road A and zero or moderate traffic on other road

    roadState();
    roadAopen();                  //Opening road A
  }
  else if((a1==0 && a2==0) && (b1==1 && b2==0) && (c1==1 || d1==1) && (c2==0 &&
d2==0)){
 //Zero traffic on road A , moderate traffic on road B and moderate or zero traffic on road C and D

    roadState();
    roadBopen();                //Opening road B
  }
  else if((a1==0 && a2==0) && (b1==0 && b2==0) && (c1==1 && c2==0) && (d1==1
&& d2==0)){       //Zero traffic on road A and B , moderate traffic on road C and moderate
or zero traffic on road D

    roadCopen();             //Opening road C
  }
  else if(a1==0 && b1==0 && c1==0 && d1==0){
//Zero traffic on all the roads
    roadState();
    roadAopen();           //Opening road A
    if (a1 == 0 && b1 == 0 && c1 == 0 && d1 == 0)
    {
      roadState();
      roadBopen();                  //Opening road B

    }
    if (a1 == 0 && b1 == 0 && c1 == 0 && d1 == 0)
    {
      roadState();
      roadCopen();                 //Opening road C

    }
    if (a1 == 0 && b1 == 0 && c1 == 0 && d1 == 0)
    {
      roadState();
      roadDopen();            //Opening road D
    }
  }
 }
void readSoundSensor(){
sensorValue = analogRead (sensorPin);                   //reading value of sound sensor
if(sensorValue > 800){
//if detected sound is greater than 800 then road A will be opened.
digitalWrite(led, HIGH);
roadAopen();
}
digitalWrite(led, LOW);
}
void readSensor()
{
  a1 = analogRead(A7);
  a2 = analogRead(A6);
  b1 = analogRead(A4);
  b2 = analogRead(A5);
  c1 = analogRead(A1);                                              //reading values of ir sensor
  c2 = analogRead(A0);
  d1 = analogRead(A3);
  d2 = analogRead(A2);

if (a1 < 400) a1 = 1; else a1 = 0; if (a2 < 400) a2 = 1; else a2 = 0;      //assigning values to ir
if (b1 < 400) b1 = 1; else b1 = 0; if (b2 < 400) b2 = 1; else b2 = 0;
if (c1 < 400) c1 = 1; else c1 = 0; if (c2 < 400) c2 = 1; else c2 = 0;
if (d1 < 400) d1 = 1; else d1 = 0; if (d2 < 400) d2 = 1; else d2 = 0;

}
void roadAopen()
{
  Serial.println(msg1);                  // printing value of msg1 for passing it on bluetooth
  lcd.setCursor(0,11);                   //setting cursor on 0th row and 11th column of lcd
  lcd.print("Road A Open");              //displaying on lcd
  digitalWrite(ledA3, LOW);

  digitalWrite(ledA1, HIGH);
  digitalWrite(ledB3, HIGH);
  digitalWrite(ledC3, HIGH);
  digitalWrite(ledD3, HIGH);
  delay(5000);                           //Green led is high and red leds of other road are on for 5s
  digitalWrite(ledA1, LOW);
  digitalWrite(ledA2, HIGH);             //turning on yellow led with delay of 1 second
  delay(1000);
  digitalWrite(ledA2, LOW);
  readSoundSensor();
  readSensor();
}
```

```
void roadBopen()
{
  Serial.println(msg2);                    //printing value of msg2 for passing it on bluetooth
  lcd.setCursor(0,11);                     //setting cursor on 0th row and 11th column of lcd
  lcd.print("Road B Open");                //displaying on lcd
  digitalWrite(ledB3, LOW);

  digitalWrite(ledA3, HIGH);
  digitalWrite(ledB1, HIGH);
  digitalWrite(ledC3, HIGH);
  digitalWrite(ledD3, HIGH);
  delay(5000);                             //Green led is high and red leds of other road are on for 5s
  digitalWrite(ledB1, LOW);
  digitalWrite(ledB2, HIGH);               //turning on yellow led with delay of 1 second
  delay(1000);
  digitalWrite(ledB2, LOW);
  readSoundSensor();
  readSensor();

}

void roadCopen()
{
  Serial.println(msg3);                    // printing value of msg3 for passing it on bluetooth
  lcd.setCursor(0,11);                     //setting cursor on 0th row and 11th column of lcd
  lcd.print("Road C Open");
  digitalWrite(ledC3, LOW);

  digitalWrite(ledA3, HIGH);
  digitalWrite(ledB3, HIGH);
  digitalWrite(ledC1, HIGH);
  digitalWrite(ledD3, HIGH);
  delay(5000);                      //Green led is high and red leds of other road are on for 5s
  digitalWrite(ledC1, LOW);
  digitalWrite(ledC2, HIGH);        //turning on yellow led with delay of 1 second
  delay(1000);
  digitalWrite(ledC2, LOW);
  readSoundSensor();
  readSensor();

}

void roadDopen()
{
  Serial.println(msg4);               // printing value of msg4 for passing it on bluetooth
  lcd.setCursor(0,11);                //setting cursor on 0th row and 11th column of lcd
  lcd.print("Road D Open");
  digitalWrite(ledD3, LOW);

  digitalWrite(ledA3, HIGH);
  digitalWrite(ledB3, HIGH);
  digitalWrite(ledC3, HIGH);
  digitalWrite(ledD1, HIGH);
  delay(5000);                         //Green led is high and red leds of other road are on for 5s
  digitalWrite(ledD1, LOW);
  digitalWrite(ledD2, HIGH);          //turning on yellow led with delay of 1 second
  delay(1000);
  digitalWrite(ledD2, LOW);
 readSoundSensor();
  readSensor();

}
void roadState(){
if(Serial.available()>0){
    state= Serial.read();                          //printing values of state
  }
  if(state==97){
    state=0;
//setting value of state=0 because bluetooth stores the value of state so we are making it 0.
    roadAopen();                                   //if state is 97 then open road A
  }
  else if(state==98){
    state=0;
    roadBopen();                                   //if state is 98 then open road A
  }
  else if(state==99){
    state=0;
    roadCopen();                                   //if state is 99 then open road A
  }
  else if(state==100){
    state=0;
    roadDopen();                                   //if state is 100 then open road A
  }
}
```

Size of code : 7302 bytes

11) <u>Summary</u>

To summarize the working of our entire project, it basically works according to the priorities assigned to the different sensors and module used. We've used 8 IR sensors , 2 for each road, to detect the density of vehicles. 16 conditions are defined for the different possibilities of these 8 IRs. The priorities assigned to the roads is A, B, C ,D respectively. Thus, if any of these roads has very high traffic based upon which condition is satisfied, that road will open breaking the normal priority flow. After that, the normal flow will again start from where it had stopped.

Then, there is this app which is connected to the arduino through the bluetooth module. It is used to manually open any road in case of any external emergency. Bluetooth module is given higher priority than the IR sensors. Thus, if road A has high traffic but we press the button of road B to open from the app, the first road B will open the road A. After that again it will return to the normal priority flow.

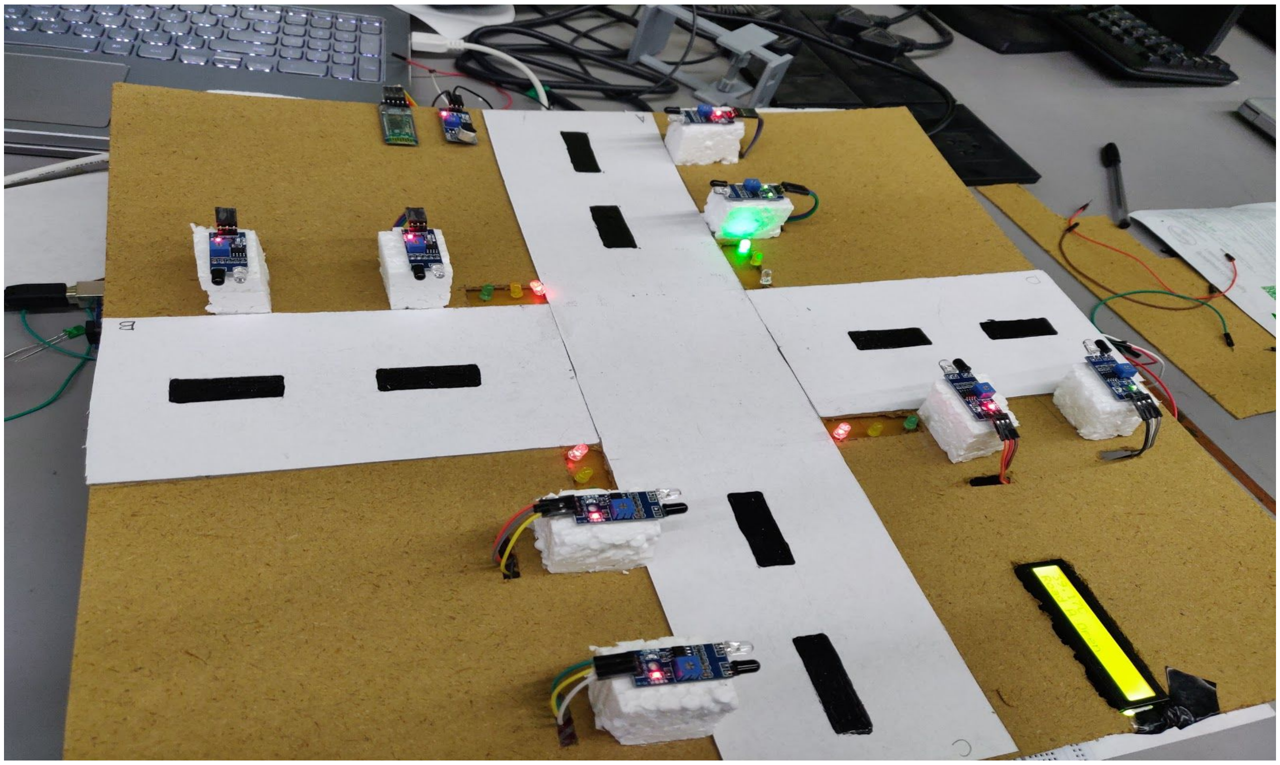
The sound sensor is given the highest priority of all. Thus, if this sensor any sound of an ambulance or so, that road connected to the sensor will open first, no matter where high traffic is detected through the IR or whatever road is pressed from the app.

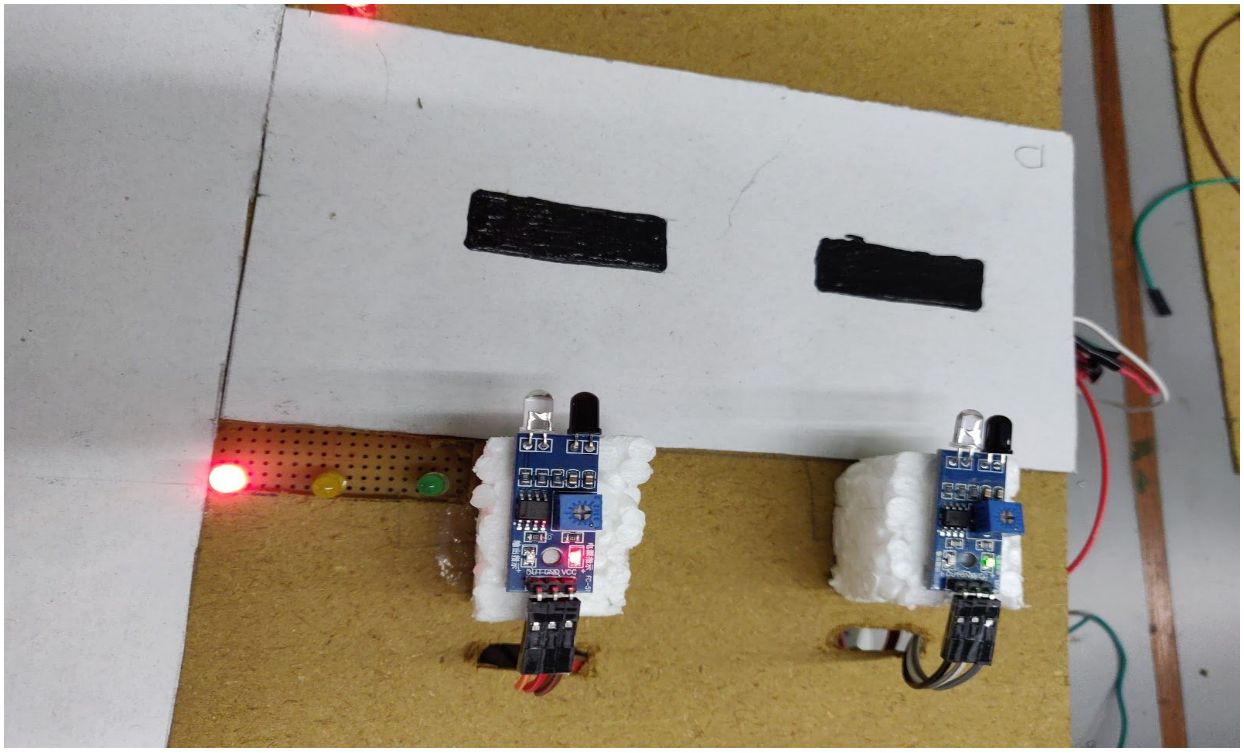
In addition, we've used a temperature sensor LM35 that will sense the temperature and display it on the LCD display. The LCD display will also display which road is open at that particular time. So, this is how our project works.

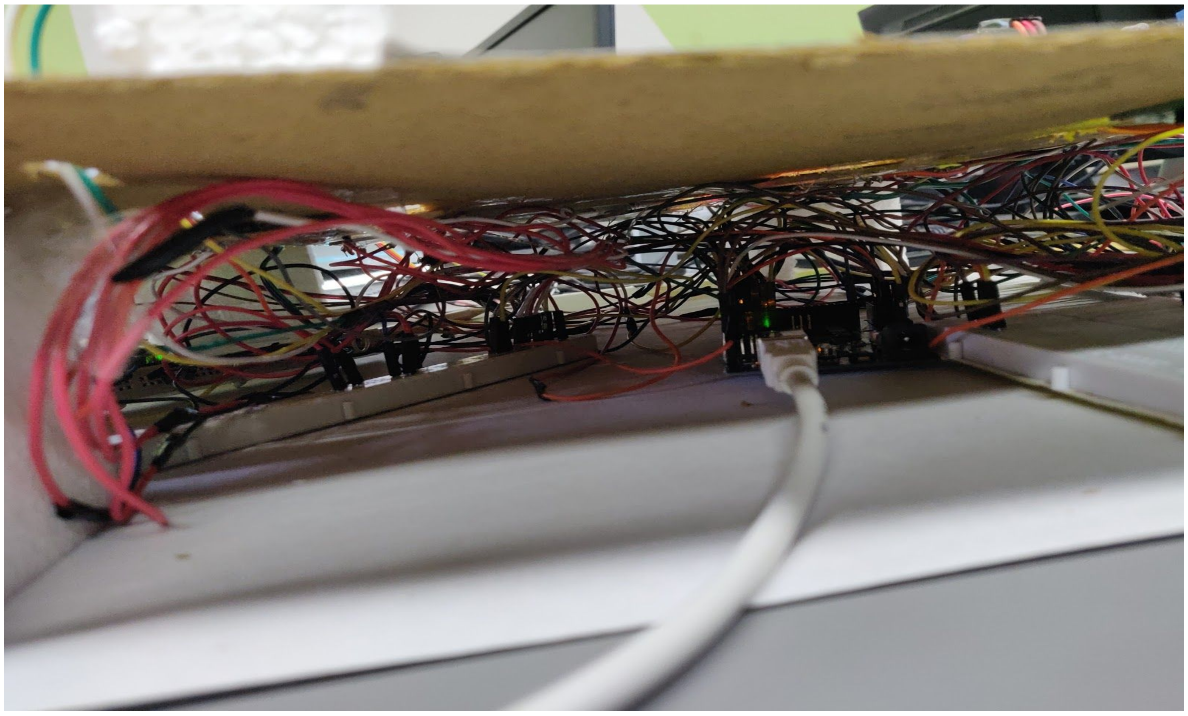
<u>Timeline</u>

	31-01-19	08-02-19	05-03-19	19-03-19	02-04-19	08-04-19
Group forming	X					
Selecting topic	X					
Deciding components		X				
Block diagram		X				
Flow chart		X				
Circuit diagram			X			
Code				X	X	
Circuit				X	X	
Working model						X

Photos of circuit




OUT GND VCC



Video of final working model

https://drive.google.com/open?id=1q6OuSU8Pqmu71qwPewc_emT4s_3fcpmL

Video of circuit

https://drive.google.com/open?id=12jU_Ws9H78y1A6H2r54pOXpWTObrDFwu

12) References

[1] Creator:Razib Shahadat.Title of video:Density Based 4 Way Traffic Signal Control System Using Arduino and IR Sensor Released date : Dec 10, 2018 [Online Video] Available : https://www.youtube.com/watch?v=CFMzBt3ta4o

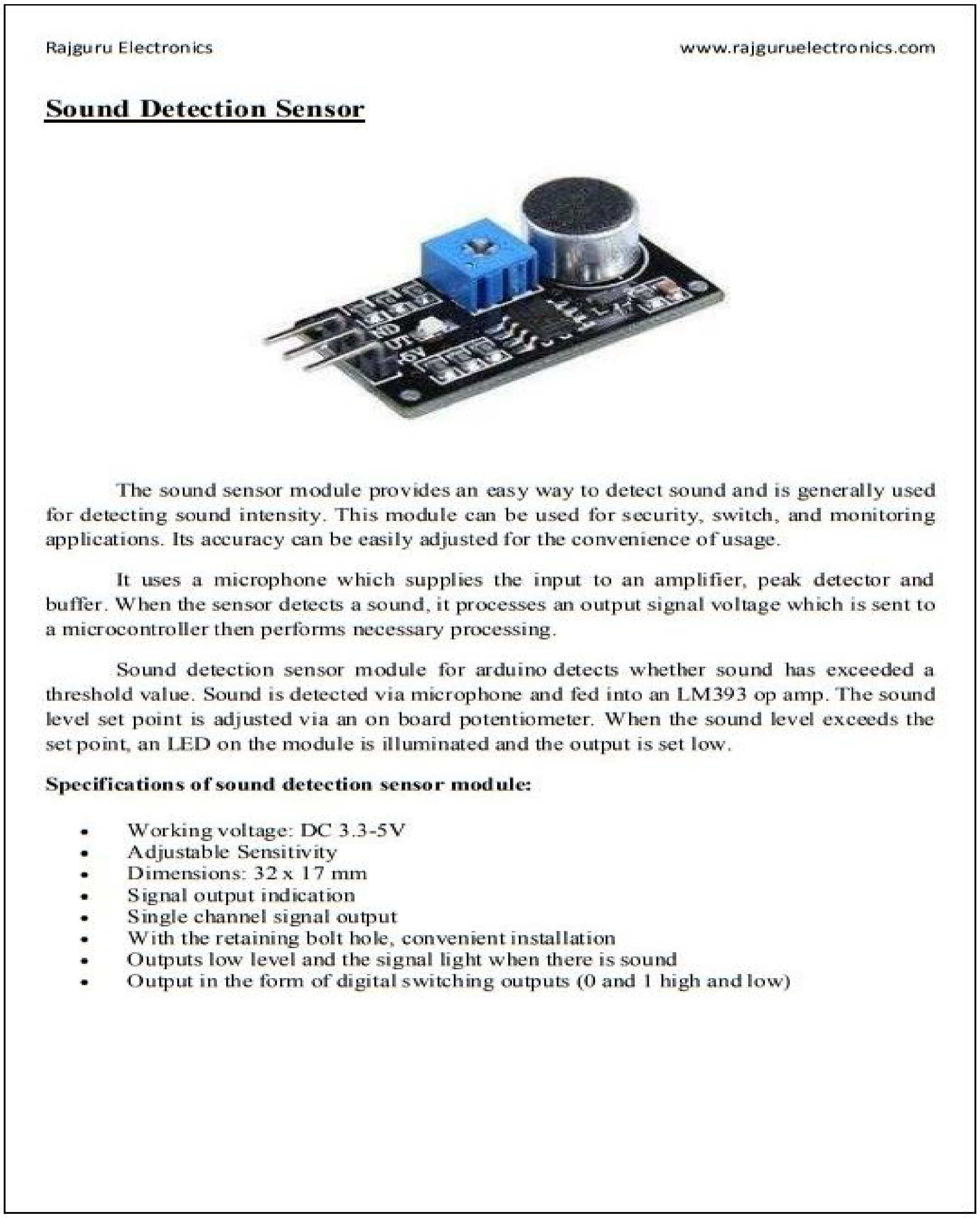
[2] Konstantin Dimitrov. Instructables. Making LCD Thermometer With Arduino and LM35/36 Autodesk.
https://www.instructables.com/id/Electronic-Thermometer-with-Arduino-UNO/

[3] Roboindia. Tutorial Portal. Sending and Receiving Data with HC-05 - MIT App Inventor
https://roboindia.com/tutorials/sending-receiving-with-hc05-mit-app-inventor

Appendix A

Datasheets

1) Sound sensor



Rajguru Electronics www.rajguruelectronics.com

Sound Detection Sensor

The sound sensor module provides an easy way to detect sound and is generally used for detecting sound intensity. This module can be used for security, switch, and monitoring applications. Its accuracy can be easily adjusted for the convenience of usage.

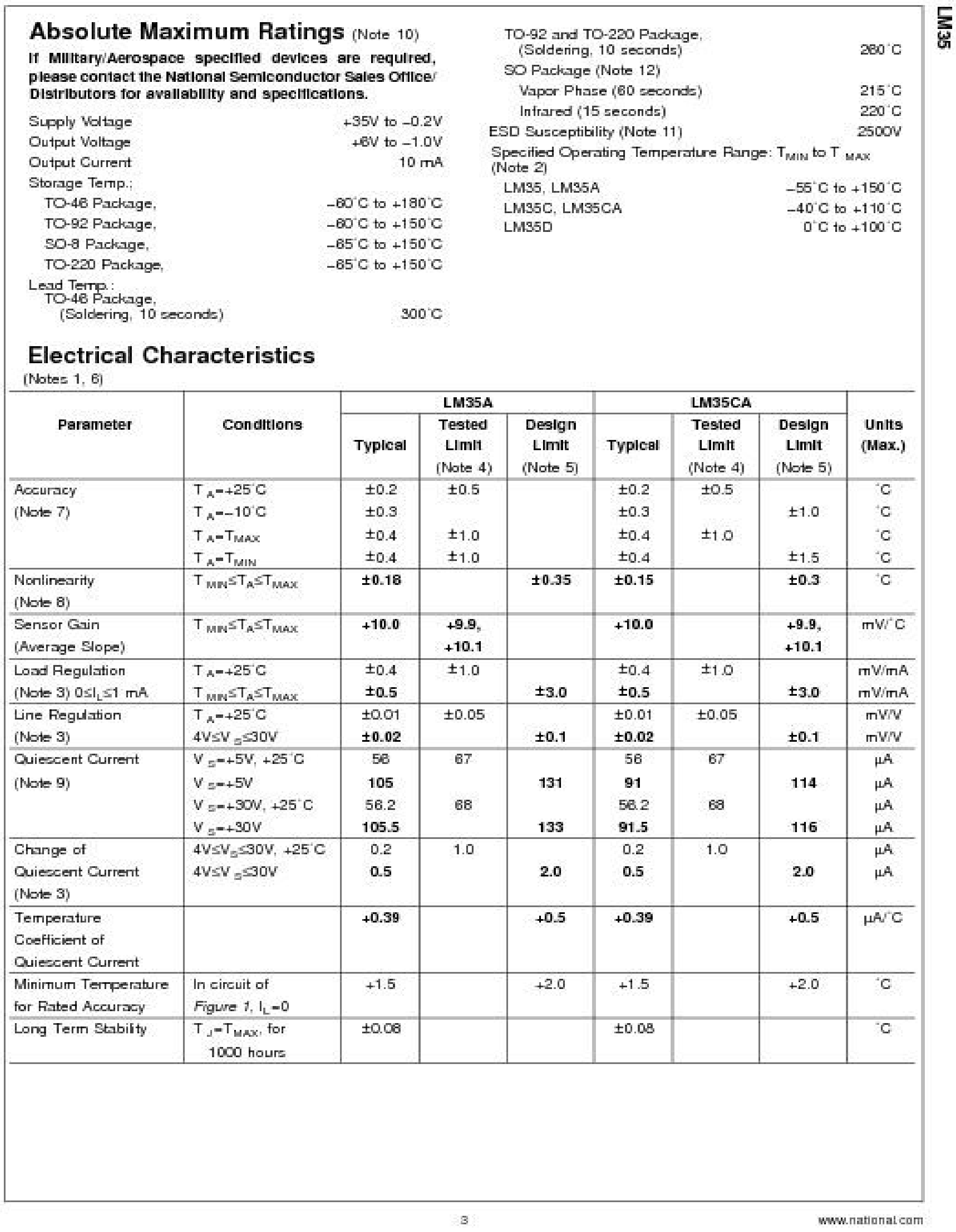
It uses a microphone which supplies the input to an amplifier, peak detector and buffer. When the sensor detects a sound, it processes an output signal voltage which is sent to a microcontroller then performs necessary processing.

Sound detection sensor module for arduino detects whether sound has exceeded a threshold value. Sound is detected via microphone and fed into an LM393 op amp. The sound level set point is adjusted via an on board potentiometer. When the sound level exceeds the set point, an LED on the module is illuminated and the output is set low.

Specifications of sound detection sensor module:

- Working voltage: DC 3.3-5V
- Adjustable Sensitivity
- Dimensions: 32 x 17 mm
- Signal output indication
- Single channel signal output
- With the retaining bolt hole, convenient installation
- Outputs low level and the signal light when there is sound
- Output in the form of digital switching outputs (0 and 1 high and low)

2) LM35



Absolute Maximum Ratings (Note 10)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage	+35V to −0.2V
Output Voltage	+6V to −1.0V
Output Current	10 mA
Storage Temp.;	
TO-46 Package,	−60°C to +180°C
TO-92 Package,	−60°C to +150°C
SO-8 Package,	−65°C to +150°C
TO-220 Package,	−65°C to +150°C
Lead Temp.:	
TO-46 Package, (Soldering, 10 seconds)	300°C
TO-92 and TO-220 Package, (Soldering, 10 seconds)	260°C
SO Package (Note 12)	
Vapor Phase (60 seconds)	215°C
Infrared (15 seconds)	220°C
ESD Susceptibility (Note 11)	2500V
Specified Operating Temperature Range: T_{MIN} to T_{MAX} (Note 2)	
LM35, LM35A	−55°C to +150°C
LM35C, LM35CA	−40°C to +110°C
LM35D	0°C to +100°C

Electrical Characteristics

(Notes 1, 6)

Parameter	Conditions	LM35A			LM35CA			Units (Max.)
		Typical	Tested Limit (Note 4)	Design Limit (Note 5)	Typical	Tested Limit (Note 4)	Design Limit (Note 5)	
Accuracy (Note 7)	T_A=+25°C	±0.2	±0.5		±0.2	±0.5		°C
	T_A=−10°C	±0.3			±0.3		±1.0	°C
	T_A=T_{MAX}	±0.4	±1.0		±0.4	±1.0		°C
	T_A=T_{MIN}	±0.4	±1.0		±0.4		±1.5	°C
Nonlinearity (Note 8)	T_{MIN}≤T_A≤T_{MAX}	**±0.18**		**±0.35**	**±0.15**		**±0.3**	°C
Sensor Gain (Average Slope)	T_{MIN}≤T_A≤T_{MAX}	**+10.0**	**+9.9, +10.1**		**+10.0**		**+9.9, +10.1**	mV/°C
Load Regulation (Note 3) 0≤I_L≤1 mA	T_A=+25°C	±0.4	±1.0		±0.4	±1.0		mV/mA
	T_{MIN}≤T_A≤T_{MAX}	**±0.5**		**±3.0**	**±0.5**		**±3.0**	mV/mA
Line Regulation (Note 3)	T_A=+25°C	±0.01	±0.05		±0.01	±0.05		mV/V
	4V≤V_S≤30V	**±0.02**		**±0.1**	**±0.02**		**±0.1**	mV/V
Quiescent Current (Note 9)	V_S=+5V, +25°C	56	67		56	67		µA
	V_S=+5V	**105**		**131**	**91**		**114**	µA
	V_S=+30V, +25°C	56.2	68		56.2	68		µA
	V_S=+30V	**105.5**		**133**	**91.5**		**116**	µA
Change of Quiescent Current (Note 3)	4V≤V_S≤30V, +25°C	0.2	1.0		0.2	1.0		µA
	4V≤V_S≤30V	**0.5**		**2.0**	**0.5**		**2.0**	µA
Temperature Coefficient of Quiescent Current		**+0.39**		**+0.5**	**+0.39**		**+0.5**	µA/°C
Minimum Temperature for Rated Accuracy	In circuit of *Figure 1*, I_L=0	+1.5		+2.0	+1.5		+2.0	°C
Long Term Stability	T_J=T_{MAX}, for 1000 hours	±0.08			±0.08			°C

3) IR Sensor

1. Descriptions

The Multipurpose Infrared Sensor is an add-on for your line follower robot and obstacle avoiding robot that gives your robot the ability to detect lines or nearby objects. The sensor works by detecting reflected light coming from its own infrared LED. By measuring the amount of reflected infrared light, it can detect light or dark (lines) or even objects directly in front of it. An onboard RED LED is used to indicate the presence of an object or detect line. Sensing range is adjustable with inbuilt variable resistor.

The sensor has a 3-pin header which connects to the microcontroller board or Arduino board via female to female or female to male jumper wires. A mounting hole for easily connect one or more sensor to the front or back of your robot chassis.

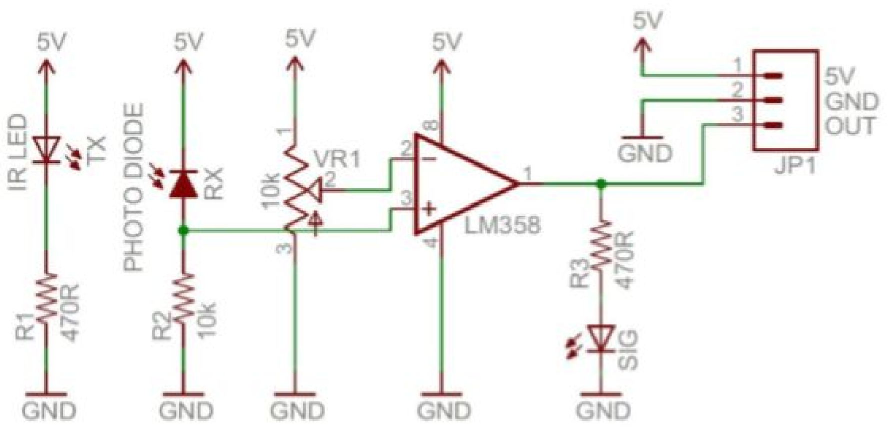
2. Features

- 5VDC operating voltage.
- I/O pins are 5V and 3.3V compliant.
- Range: Up to 20cm.
- Adjustable Sensing range.
- Built-in Ambient Light Sensor.
- 20mA supply current.
- Mounting hole.

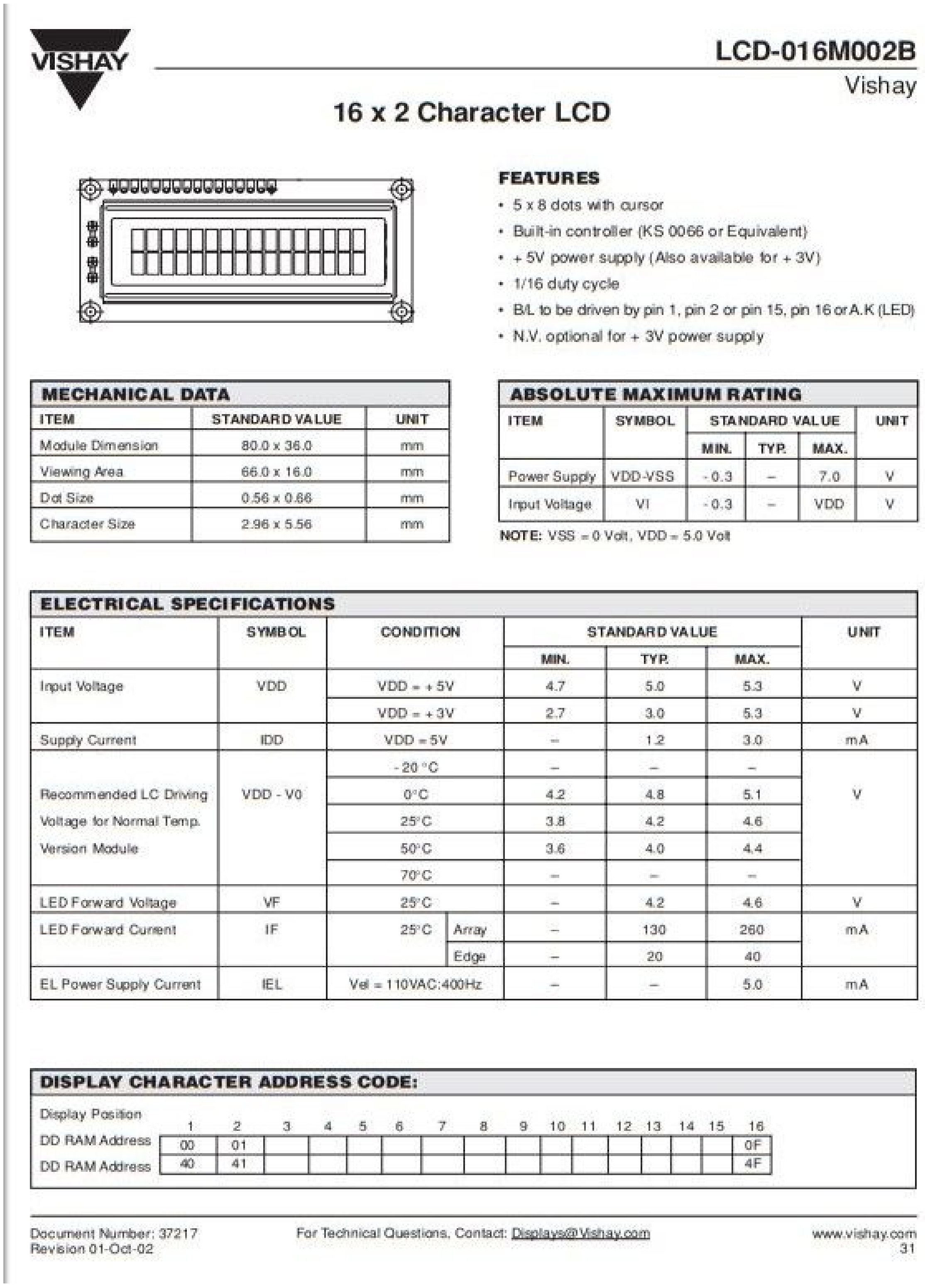
3. Specifications

- Size: 50 x 20 x 10 mm (L x B x H)
- Hole size: ϕ2.5mm

4. Schematics



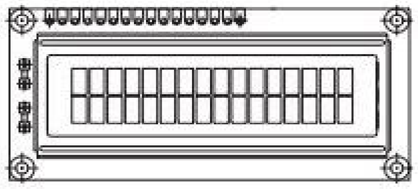
4) LCD display



VISHAY

LCD-016M002B
Vishay

16 x 2 Character LCD



FEATURES

- 5 x 8 dots with cursor
- Built-in controller (KS 0066 or Equivalent)
- + 5V power supply (Also available for + 3V)
- 1/16 duty cycle
- B/L to be driven by pin 1, pin 2 or pin 15, pin 16 or A.K (LED)
- N.V. optional for + 3V power supply

MECHANICAL DATA

ITEM	STANDARD VALUE	UNIT
Module Dimension	80.0 x 36.0	mm
Viewing Area	66.0 x 16.0	mm
Dot Size	0.56 x 0.66	mm
Character Size	2.96 x 5.56	mm

ABSOLUTE MAXIMUM RATING

ITEM	SYMBOL	STANDARD VALUE			UNIT
		MIN.	TYP.	MAX.	
Power Supply	VDD-VSS	- 0.3	–	7.0	V
Input Voltage	VI	- 0.3	–	VDD	V

NOTE: VSS = 0 Volt, VDD = 5.0 Volt

ELECTRICAL SPECIFICATIONS

ITEM	SYMBOL	CONDITION		STANDARD VALUE			UNIT
				MIN.	TYP.	MAX.	
Input Voltage	VDD	VDD = + 5V		4.7	5.0	5.3	V
		VDD = + 3V		2.7	3.0	5.3	V
Supply Current	IDD	VDD = 5V		–	1.2	3.0	mA
Recommended LC Driving Voltage for Normal Temp. Version Module	VDD - V0	- 20 °C		–	–	–	V
		0°C		4.2	4.8	5.1	
		25°C		3.8	4.2	4.6	
		50°C		3.6	4.0	4.4	
		70°C		–	–	–	
LED Forward Voltage	VF	25°C		–	4.2	4.6	V
LED Forward Current	IF	25°C	Array	–	130	260	mA
			Edge	–	20	40	
EL Power Supply Current	IEL	Vel = 110VAC:400Hz		–	–	5.0	mA

DISPLAY CHARACTER ADDRESS CODE:

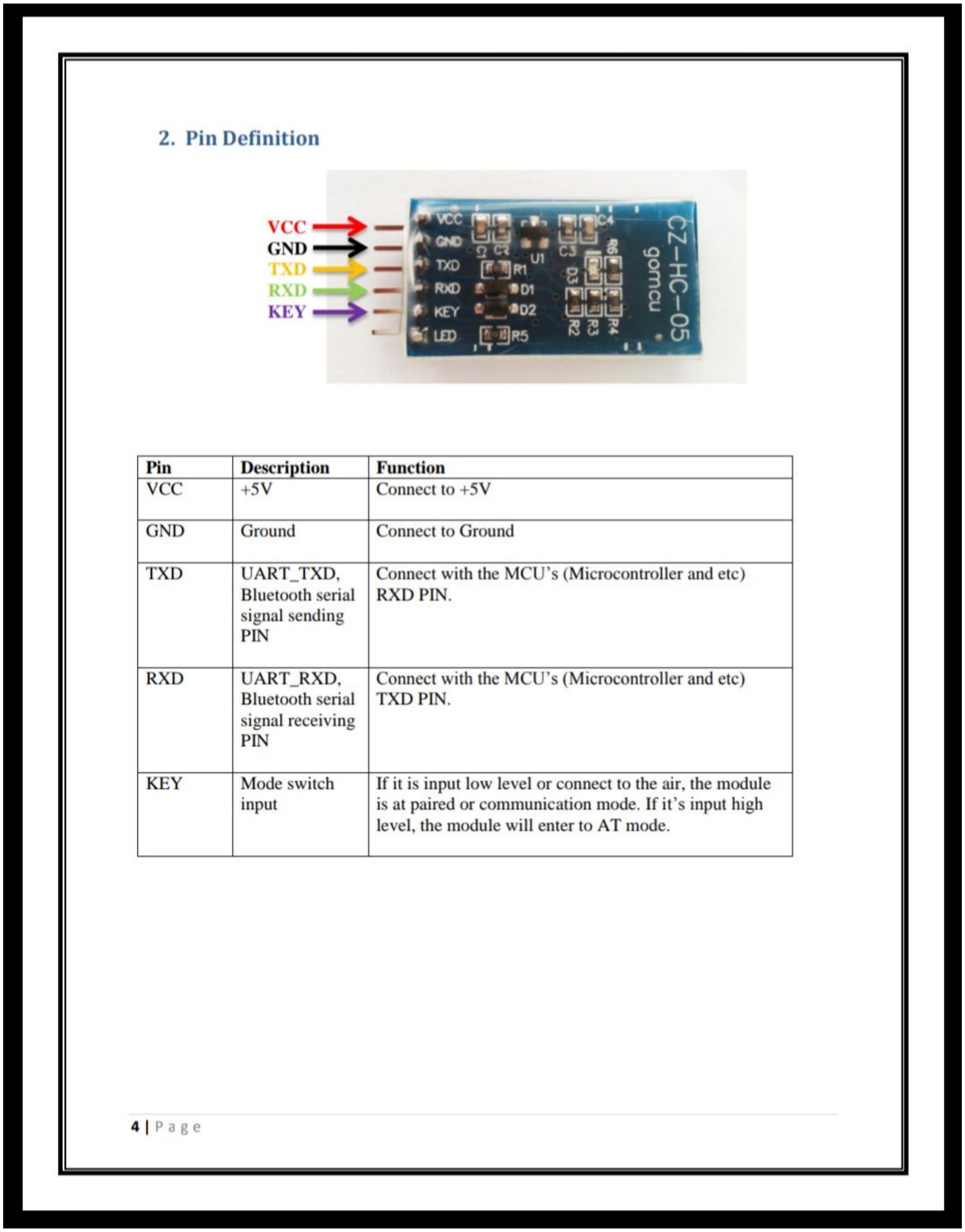
Display Position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
DD RAM Address	00	01														0F
DD RAM Address	40	41														4F

Document Number: 37217
Revision 01-Oct-02

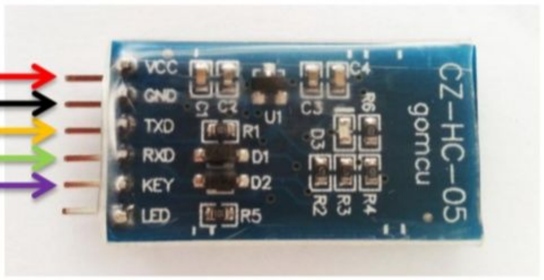
For Technical Questions, Contact: Displays@Vishay.com

www.vishay.com
31

5) Bluetooth module



2. Pin Definition



Pin	Description	Function
VCC	+5V	Connect to +5V
GND	Ground	Connect to Ground
TXD	UART_TXD, Bluetooth serial signal sending PIN	Connect with the MCU's (Microcontroller and etc) RXD PIN.
RXD	UART_RXD, Bluetooth serial signal receiving PIN	Connect with the MCU's (Microcontroller and etc) TXD PIN.
KEY	Mode switch input	If it is input low level or connect to the air, the module is at paired or communication mode. If it's input high level, the module will enter to AT mode.

Appendix B

Programming Review

C program	**Assembly language**
The code which was written in c could be easily reused on a different platform.	Assembly does not provide the portability and source code specific to a processor.
Requires lesser memory.	Requires larger memory.
Execution speed and manufacturing cost is more.	Execution speed and manufacturing cost is comparatively very less.
It is easily portable to other microcontrollers with almost no modifications.	Not easily portable with other microcontrollers.

C program	**Python**
An Imperative programming model is basically followed by C.	An object-oriented programming model is basically followed by Python.
Pointers are available in C language.	No pointers functionality is available in Python.
C is compiled direct to machine code which is executed directly by the CPU.	Python is firstly compiled to a byte-code and then it is interpreted by a large C program.
There is a limited number of built-in functions available in C.	There is a large library of built-in functions in Python.

Implementation of data structures requires its functions to be explicitly implemented.	It is easy to implement data structures in Python with built-in insert, append functions.
Declaring of variable type in C is necessary condition.	There is no need to declare a type of variable in Python.
C is a compiled language.	Python is an interpreted language.

Appendix C

Trouble shooting

1. **Error 516** : Unable to write:Broken pipe (Bluetooth)(Arduino)(MIT App Inventor)

   Possible Two Classic HC-05 mistakes.

   1) connected to pin 0 and 1 (the USB<>Serial pins).
   2) powered from the 3.3volt pin (module needs 3.6volt minimum on the VCC pin).

   Solution :
   Connected to pin 14(TX) and 15(RX) (the USB<>Serial pins).

2. **Error** : avrdude: stk500_getsync(): not in sync: resp=0x00

   Mistake : When the TX and RX pins are connected to a peripheral, such as a bluetooth module, arduino IDE cannot upload sketches, due to communication problem and gives the following error message.

   Solution : Disconnect the bluetooth module before uploading the code and connect after it is uploaded.