

# **1. INTRODUCTION**

## **1.1. OVERVIEW**

The smart stick will contain sensors which will detect any obstacles or uneven surface and alert the optically disabled person. It will measure the distance between the stick and obstacle and will notify the person using a buzzer/speaker

## **1.2 EXISTING SYSTEM**

Obstacle detectors using the sensors are being used for a completely different purpose, for example in smart parking systems, reverse parking to mention a few.

## **1.3 AIM OF THE PROJECT**

The aim of the project is to make an obstacle detector stick for the visually disabled so as to make sure they don't bump into any obstacle and walk freely without getting hurt.

## **1.4 ORGANIZATION REPORT**

The organization of the report is as follows:

**Chapter 1** deals with the Introduction of the project and gives the details about the project in an abstract view.

**Chapter 2** deals with the information about Arduino Uno and the sensors that are available and the arduino Ide.

**Chapter 3** deals with the Software Requirements Specifications which is a specification of the project software and hardware requirements.

**Chapter 4** deals with the Implementation part which includes the tools and software's that are used.

**Chapter 5** deals with the Testing of the project and screenshots of the project

**Chapter 6** explains the Conclusion and further scope of the project.

## **2. TECHNOLOGIES**

### **2.1 ABOUT ARDUINO**

#### **2.1.1 INTRODUCTION**

Arduino is a prototype platform (open-source) based on an easy-to-use hardware and software. It consists of a circuit board, which can be programmed (referred to as a microcontroller) and a ready-made software called Arduino IDE (Integrated Development Environment), which is used to write and upload the computer code to the physical board.

Arduino provides a standard form factor that breaks the functions of the micro-controller into a more accessible package.

#### **2.1.2. TYPES OF ARDUINO**

Various kinds of Arduino boards are available depending on different microcontrollers used. However, all Arduino boards have one thing in common: they are programmed through the Arduino IDE. The differences are based on the number of inputs and outputs (the number of sensors, LEDs, and buttons you can use on a single board), speed, operating voltage, form factor etc. Some boards are designed to be embedded and have no programming interface (hardware), which you would need to buy separately. Some can run directly from a 3.7V battery, others need at least 5V.

Here is a list of different Arduino boards available.

## Arduino boards based on ATMEGA328 microcontroller

Board Name	Operating Volt	Clock Speed	Digital i/o	Analog Inputs	PWM	UART	Programming Interface
Arduino Uno R3	5V	16MHz	14	6	6	1	USB via ATmega16U2
Arduino Uno R3 SMD	5V	16MHz	14	6	6	1	USB via ATmega16U2
Red Board	5V	16MHz	14	6	6	1	USB via FTDI
Arduino Pro 3.3v/8MHz	3.3V	8MHz	14	6	6	1	FTDI-Compatible Header
Arduino Pro 5V/16MHz	5V	16MHz	14	6	6	1	FTDI-Compatible Header
Arduino mini 05	5V	16MHz	14	8	6	1	FTDI-Compatible Header
Arduino Pro mini 3.3v/8MHz	3.3V	8MHz	14	8	6	1	FTDI-Compatible Header
Arduino Pro mini 5v/16MHz	5V	16MHz	14	8	6	1	FTDI-Compatible Header
Arduino Ethernet	5V	16MHz	14	6	6	1	FTDI-Compatible Header
Arduino Fio	3.3V	8MHz	14	8	6	1	FTDI-Compatible Header
LilyPad Arduino 328 main board	3.3V	8MHz	14	6	6	1	FTDI-Compatible Header
LilyPad Arduino simple board	3.3V	8MHz	9	4	5	0	FTDI-Compatible Header

### Arduino boards based on ATMEGA32u4 microcontroller

Board Name	Operating Volt	Clock Speed	Digital i/o	Analog Inputs	PWM	UART	Programming Interface
Arduino Leonardo	5V	16MHz	20	12	7	1	Native USB
Pro micro 5V/16MHz	5V	16MHz	14	6	6	1	Native USB
Pro micro 3.3V/8MHz	5V	16MHz	14	6	6	1	Native USB
LilyPad Arduino USB	3.3V	8MHz	14	6	6	1	Native USB

### Arduino boards based on ATMEGA2560 microcontroller

Board Name	Operating Volt	Clock Speed	Digital i/o	Analog Inputs	PWM	UART	Programming Interface
Arduino Mega 2560 R3	5V	16MHz	54	16	14	4	USB via ATmega16U2B
Mega Pro 3.3V	3.3V	8MHz	54	16	14	4	FTDI-Compatible Header
Mega Pro 5V	5V	16MHz	54	16	14	4	FTDI-Compatible Header
Mega Pro Mini 3.3V	3.3V	8MHz	54	16	14	4	FTDI-Compatible Header

### Arduino boards based on AT91SAM3X8E microcontroller

Board Name	Operating Volt	Clock Speed	Digital i/o	Analog Inputs	PWM	UART	Programming Interface
Arduino Mega 2560 R3	3.3V	84MHz	54	12	12	4	USB native

## 2.1.3 ARDUINO UNO

Arduino Uno pins description

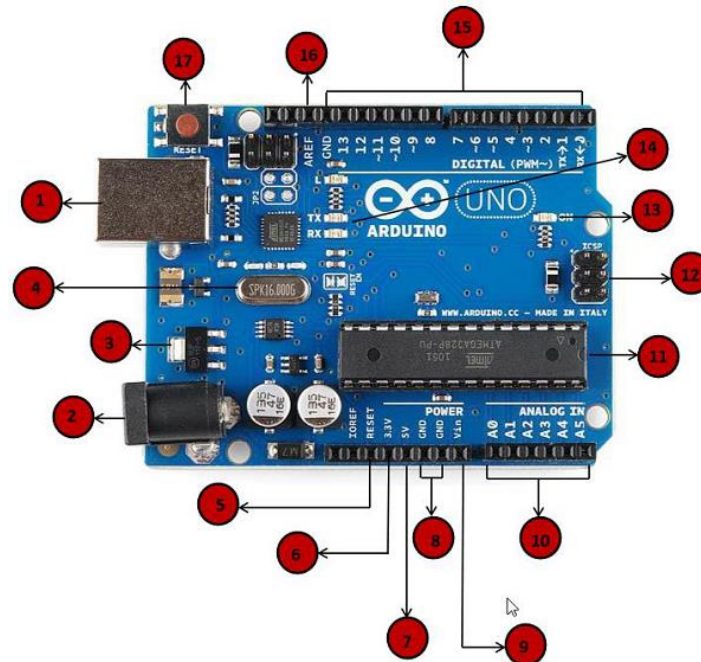










Fig. 2.1.3 Arduino Uno R3

1	<b>Power USB</b> Arduino board can be powered by using the USB cable from your computer. All you need to do is connect the USB cable to the USB connection (1).
2	<b>Power (Barrel Jack)</b> Arduino boards can be powered directly from the AC mains power supply by connecting it to the Barrel Jack (2).
3	<b>Voltage Regulator</b> The function of the voltage regulator is to control the voltage given to the Arduino board and stabilize the DC voltages used by the processor and other elements.
4	<b>Crystal Oscillator</b> The crystal oscillator helps Arduino in dealing with time issues. How does Arduino calculate time? The answer is, by using the crystal oscillator. The number printed on top of the Arduino crystal is 16.000H9H. It tells us that the frequency is 16,000,000 Hertz or 16 MHz.
5,17	<b>Arduino Reset</b> You can reset your Arduino board, i.e., start your program from the beginning. You can reset the UNO board in two ways. First, by using the reset button (17) on the board. Second, you can connect an external reset button to the Arduino pin labelled RESET (5).

	<p><b>Pins (3.3, 5, GND, Vin)</b></p> <ul style="list-style-type: none"> <li>• 3.3V (6) – Supply 3.3 output volt</li> <li>• 5V (7) – Supply 5 output volt</li> <li>• Most of the components used with Arduino board works fine with 3.3 volt and 5 volt.</li> <li>• GND (8)(Ground) – There are several GND pins on the Arduino, any of which can be used to ground your circuit.</li> <li>• Vin (9) – This pin also can be used to power the Arduino board from an external power source, like AC mains power supply.</li> </ul>
	<p><b>Analog pins</b></p> <p>The Arduino UNO board has five analog input pins A0 through A5. These pins can read the signal from an analog sensor like the humidity sensor or temperature sensor and convert it into a digital value that can be read by the microprocessor.</p>
	<p><b>Main microcontroller</b></p> <p>Each Arduino board has its own microcontroller (11). You can assume it as the brain of your board. The main IC (integrated circuit) on the Arduino is slightly different from board to board. The microcontrollers are usually of the ATMEL Company. You must know what IC your board has before loading up a new program from the Arduino IDE. This information is available on the top of the IC. For more details about the IC construction and functions, you can refer to the data sheet.</p>
	<p><b>ICSP pin</b></p> <p>Mostly, ICSP (12) is an AVR, a tiny programming header for the Arduino consisting of MOSI, MISO, SCK, RESET, VCC, and GND. It is often referred to as an SPI (Serial Peripheral Interface), which could be considered as an "expansion" of the output. Actually, you are slaving the output device to the master of the SPI bus.</p>
	<p><b>Power LED indicator</b></p> <p>This LED should light up when you plug your Arduino into a power source to indicate that your board is powered up correctly. If this light does not turn on, then there is something wrong with the connection.</p>
	<p><b>TX and RX LEDs</b></p> <p>On your board, you will find two labels: TX (transmit) and RX (receive). They appear in two places on the Arduino UNO board. First, at the digital pins 0 and 1, to indicate the pins responsible for serial communication. Second, the TX and RX led (13). The TX led flashes with different speed while sending the serial data. The speed of flashing depends on the baud rate used by the board. RX flashes during the receiving process.</p>

	<p><b>Digital I/O</b></p> <p>The Arduino UNO board has 14 digital I/O pins (15) (of which 6 provide PWM (Pulse Width Modulation) output. These pins can be configured to work as input digital pins to read logic values (0 or 1) or as digital output pins to drive different modules like LEDs, relays, etc. The pins labeled “~” can be used to generate PWM.</p>
	<p><b>AREF</b></p> <p>AREF stands for Analog Reference. It is sometimes, used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.</p>

## 2.1.4 SENSORS AND MODULES

- Humidity sensor (DHT22)
- Temperature sensor (LM35)
- Water detector sensor (Simple Water Trigger)
- PIR SENSOR
- ULTRASONIC SENSOR
- GPS
- SD Card Module

Our Main focus will be on the ultrasonic sensor as Ultrasonic sensor is the one that detects the obstacle buy sending out waves.

#### 2.1.4.1 ULTRASONIC SENSOR

The HC-SR04 ultrasonic sensor uses SONAR to determine the distance of an object just like the bats do. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package from 2 cm to 400 cm or 1" to 13 feet. The operation is not affected by sunlight or black material, although acoustically, soft materials like cloth can be difficult to detect. It comes complete with ultrasonic transmitter and receiver module. The sensor head emits an ultrasonic wave and receives the wave reflected back from the target. Ultrasonic Sensors measure the distance to the target by measuring the time between the emission and reception.

- Power Supply – +5V DC
- Quiescent Current – <2mA
- Working Current – 15mA
- Effectual Angle – <15°
- Ranging Distance – 2cm – 400 cm/1" – 13ft
- Resolution – 0.3 cm
- Measuring Angle – 30 degree

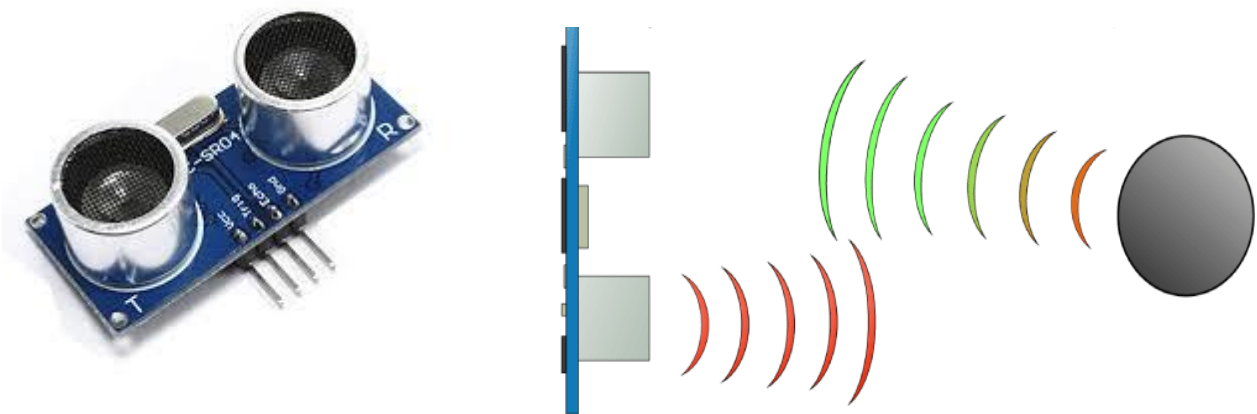


Fig.2.1.4.1 Ultrasonic sensor and its working



### 2.1.4.2 BUZZER

Piezo buzzers are used for making beeps alarms and tones. They can be used in alarm systems, for keypad feedback, or some games. Light weight, simple construction and low price make it usable in various applications



Fig.2.1.4.2 Peizo Buzzer

### 2.1.4.3 Jumper Cables and Bread Board

Jumper Cables are used to connect the different arduino components/chips/modules to the the arduino and for supply of power to the components from the arduino.

3 types of jumper cables are available

- FxF cables
- MxF/FxM cables
- MxM cables



Fig. 2.1.4.3 Jumper Cables

Breadboard is a solderless device for temporary prototype with electronics and test circuit designs. Most electronic components in electronic circuits can be interconnected by inserting their leads or terminals into the holes and then making connections through wires where appropriate.

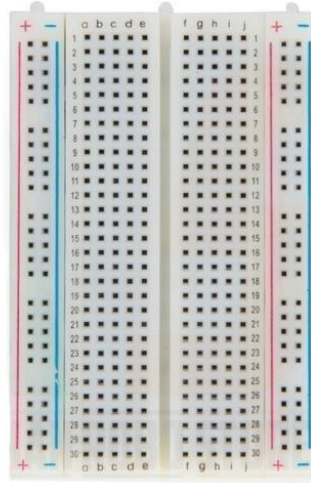


Fig.2.1.4.3 Breadboard

## 2.2 ARDUINO IDE

The Arduino IDE, which utilizes the C/C++ programming language, is a platform where the code to be loaded into the Arduino is written. This gives you access to an enormous Arduino Library that is constantly growing thanks to open-source community. The ide has been continuously upgrading through the years and new libraries are being included as new modules are being designed.

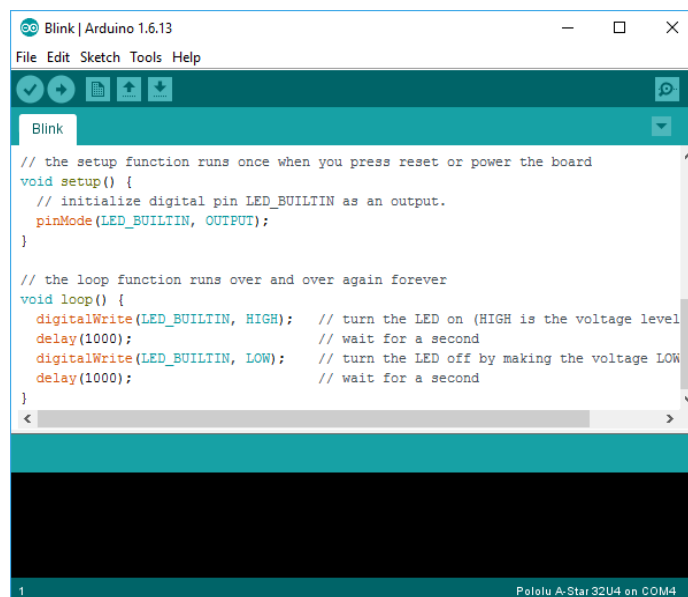


Fig. 2.2 Arduino IDE

## 2.3 FUNCTIONS

The main Functions used are

- The **setup()** function is called when a sketch starts. Use it to initialize the variables, pin modes, start using libraries, etc. The setup function will only run once, after each power up or reset of the Arduino board.
- After creating a **setup()** function, which initializes and sets the initial values, the **loop()** function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond. Use it to actively control the Arduino board.

The other functions used are

- pinMode()
- digitalWrite()
- delay()
- pulseIn()

## **3. SOFTWARE REQUIREMENT SPECIFICATION**

### **3.1 INTRODUCTION**

The requirements specification is a technical specification of requirements for the software products. It is the first step in the requirements analysis process it lists the requirements of a particular software system including functional, performance and security requirements. The requirements also provide usage scenarios from a user, an operational and an administrative perspective. The purpose of software requirements specification is to provide a detailed overview of the software project, its parameters and goals. This describes the project target audience and its user interface, hardware and software requirements. It defines how the client, team and audience see the project and its functionality.

### **3.2 PURPOSE OF THE DOCUMENT**

This software requirement specification describes all the requirements elicited for “Obstacle detector for the visually challenged” and is intended to be used by the members examining the project and implementing and verifying the application. Unless otherwise noted all requirements are of high priority and are committed.

### **3.3 USERS AND THEIR CHARACTERISTICS**

The visually impaired can use this as a walking guide instead of a guide dog or tapping their stick to detect the obstacle.

### **3.4 SOFTWARE AND HARDWARE REQUIREMENTS**

Operating System	Windows XP
Programming Languages	C,C++
Integrated Development Environment	Arduino IDE
Processor	Intel CORE i7,8 <sup>th</sup> gen.
RAM	1 GB or more
Disk Space	1GB or more

## **4. IMPLEMENTATION**

### **4.1 INTRODUCTION**

The success of the software product is determined only when it is successfully implemented according to the requirements. The analysis and the design of the proposed system provide a perfect platform to implement the idea using the specified technology in the desired environment. The implementation of our system is made user friendly.

Any software project is designed in modules and the project is said to be successfully implemented when each of the module is executed individually to obtain the expected result and, when all the modules are integrated and run together without any errors.

### **4.2 ARDUINO UNO R3**

The Arduino Uno R3 was used as a hardware for the chips/modules to function according to the code written. It is the electronic board used for implementing different modules and their code.

### **4.3 MODULES AND CONNECTORS**

The modules used for this project are Ultrasonic Sensor, piezo buzzer, breadboard and cables to interconnect the modules and the Arduino.

### **4.4 C/C++ AND ARDUINO IDE**

The code for any Arduino project is written in C/C++ using the Arduino IDE. The code is written here and loaded into the Arduino that is being used. Different libraries are used if a particular function is not available.

## **5. TESTING AND RESULTS**

### **5.1 INTRODUCTION**

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. In fact, testing is the one step in the software engineering process that could be viewed as destructive rather than constructive.

A strategy for software testing integrates software test case design methods into a well-planned series of steps that result in the successful construction of software. Testing is the set of activities that can be planned and conducted systematically. The underlying motivation of program testing is to affirm software quality with methods that can economically and effectively apply to both strategic to both large and small-scale systems.

### **5.2 TESTING OBJECTIVES**

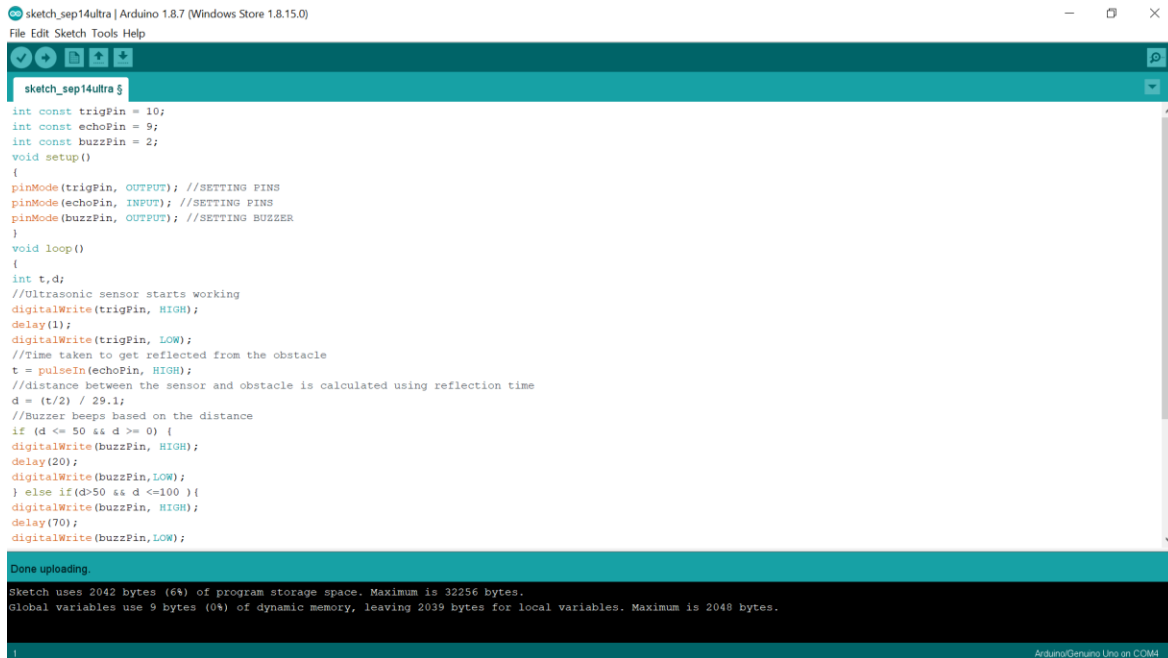
The main objective of performance testing is designed to test whether the app's display is as expected and whether the app is functioning properly or not.

As the test results are gathered and evaluated they begin to give a qualitative indication of the reliability of the app. If proper output is not obtained, the overall quality of the app is questioned. If, on the other hand, all the results which are not successful, are encountered, and are easily modifiable, then the following conclusion can be made: The tests are inadequate as the requirements mentioned are not compatible. The testing includes:

- Checking whether the information is displayed or not.
- Checking whether all the links between each button in the app works or is misdirected.
- Verifying if all the pictures are displayed and none of the files are corrupted.

### **5.3 OUTPUT SCREEN**

## 5.4 CODE

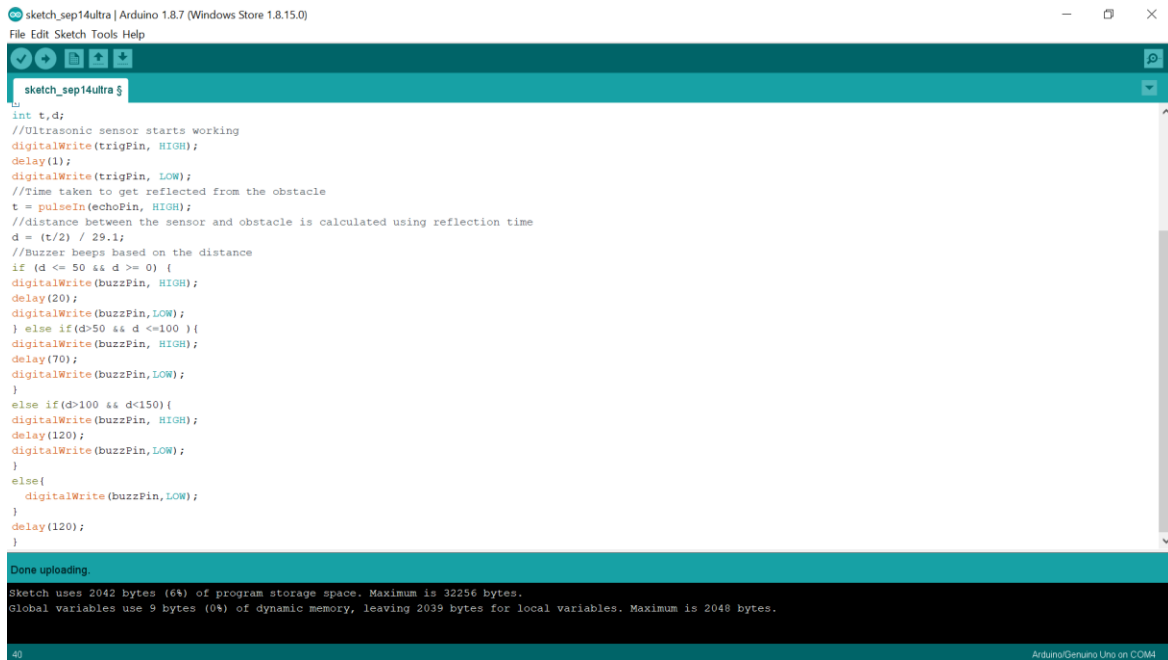


```
sketch_sep14ultra $
int const trigPin = 10;
int const echoPin = 9;
int const buzzPin = 2;
void setup()
{
  pinMode(trigPin, OUTPUT); //SETTING PINS
  pinMode(echoPin, INPUT); //SETTING PINS
  pinMode(buzzPin, OUTPUT); //SETTING BUZZER
}
void loop()
{
  int t,d;
  //Ultrasonic sensor starts working
  digitalWrite(trigPin, HIGH);
  delay(1);
  digitalWrite(trigPin, LOW);
  //Time taken to get reflected from the obstacle
  t = pulseIn(echoPin, HIGH);
  //distance between the sensor and obstacle is calculated using reflection time
  d = (t/2) / 29.1;
  //Buzzer beeps based on the distance
  if (d <= 50 && d >= 0) {
    digitalWrite(buzzPin, HIGH);
    delay(20);
    digitalWrite(buzzPin, LOW);
  } else if (d > 50 && d <= 100) {
    digitalWrite(buzzPin, HIGH);
    delay(70);
    digitalWrite(buzzPin, LOW);
  }
}
```

Done uploading.

Sketch uses 2042 bytes (6%) of program storage space. Maximum is 32256 bytes.  
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes for local variables. Maximum is 2048 bytes.

1 Arduino/Genuino Uno on COM4



```
sketch_sep14ultra $
int t,d;
//Ultrasonic sensor starts working
digitalWrite(trigPin, HIGH);
delay(1);
digitalWrite(trigPin, LOW);
//Time taken to get reflected from the obstacle
t = pulseIn(echoPin, HIGH);
//distance between the sensor and obstacle is calculated using reflection time
d = (t/2) / 29.1;
//Buzzer beeps based on the distance
if (d <= 50 && d >= 0) {
  digitalWrite(buzzPin, HIGH);
  delay(20);
  digitalWrite(buzzPin, LOW);
} else if (d > 50 && d <= 100) {
  digitalWrite(buzzPin, HIGH);
  delay(70);
  digitalWrite(buzzPin, LOW);
}
else if (d > 100 && d <= 150) {
  digitalWrite(buzzPin, HIGH);
  delay(120);
  digitalWrite(buzzPin, LOW);
}
else {
  digitalWrite(buzzPin, LOW);
}
delay(120);
}
```

Done uploading.

Sketch uses 2042 bytes (6%) of program storage space. Maximum is 32256 bytes.  
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes for local variables. Maximum is 2048 bytes.

40 Arduino/Genuino Uno on COM4

## **6. CONCLUSION AND FUTURE SCOPE**

Our website “Obstacle Detector for the visually challenged” is designed in such a way that future modifications can be done easily. The following conclusion can be deduced from the development of our project.

- Automation of the entire system improves the efficiency.
- It provides a friendly graphical user interface which proves to be better than the existing system.
- Saves time and money for the users.
- Updating of information is easier.
- The system has adequate scope for modification in future.

The enhancement is, we will develop online services. It can be made like a buy and sell book store app if linked with database management systems and acquire a domain to make it available to a wide range of audience. More information about books and genres can also be added to make the website more effective.



## **REFERENCES**

### **Books:**

1. “Let US C by Yashwanth kanetkar”
2. “C++” by Balaguruswamy

### **Online References:**

1. Tutorials Point : Arduino Uno
2. [www.arduino.cc](http://www.arduino.cc)