**AUTOMATIC RAILWAY GATE CONTROL SYSTEM**



# TABLE OF CONTENT

|  |  |  |
| --- | --- | --- |
| S.NO | CONTENT | PAGE NO |
| 1 | REQUIREMENTS | 4 |
|  | 1.1 AUTOMATIC RAILWAY GATE CONTROL SYSTEM | 4 |
|  | 1.1.1 INTRODUCTION | 4 |
|  | 1.1.2 STATE OF ART | 4 |
|  | 1.1.3 4W’s and 1H | 4 |
|  | 2.1 SWOT ANALYSIS | 4 |
|  | 3.1 REQUIREMENTS | 5 |
|  | 3.1.1 HIGH LEVEL REQUIREMENTS | 5 |
|  | 3.1.2 LOW LEVEL REQUIREMENTS | 5 |
|  | 4.1 ADVANTAGES | 5 |
|  | 5.1 FUTURE SCOPE | 5 |
| 2 | ARCHITECTURE | 6 |
|  | 2.1 BLOCK DIAGRAM | 6 |
|  | 2.2 COMPONENTS | 6 |
|  | 2.3 FLOWCHART | 9 |
| 3 | IMPLEMENTATION | 10 |
| 4 | TESTPLAN & OUTPUT | 16 |
| 5 | MERITS AND DEMERITS | 16 |
|  | 5.1 MERITS | 16 |
|  | 5.2 DEMERITS | 17 |
|  | 5.3 APPLICATIONS | 17 |
| 6 | IMAGES AND VIDEOS | 17 |
| 7 | CONCLUSION | 18 |
| 8 | REFERENCES | 18 |

**1 REQUIREMENTS**

**1.1 AUTOMATIC RAILWAY GATE CONTROL SYSTEM**

**1.1.1 INTRODUCTION**

Now a days, many accidents are happening in the railway tracks. To overcome such accidents which is one of the most significant method is implemented by automatic railway gate control using arduino with automatically open and closes the gate by tracking the train which aims at reducing the accidents due to this automation in the railway industry is an important need as we are stepping into the advanced era and to reduce the risk of accidents due to human induced errors it is very important that we let these tasks be handled by these smart machines.

**1.1.2 STATE OF ART**

\* The main goal of the project is to prevent accidents occurring in the railway tracks

\* This can be implemented in real time system

\* To track or detect the movement of the train using the sensors

\* To use automatic system for the control of railway gate

\* This shall use a LCD display the distance and the gate position

\* To reduce the manual work and instead use the machine process

**1.1.3 4W's and 1H**

\* When - It can be used when people work at late night to open the railway gate

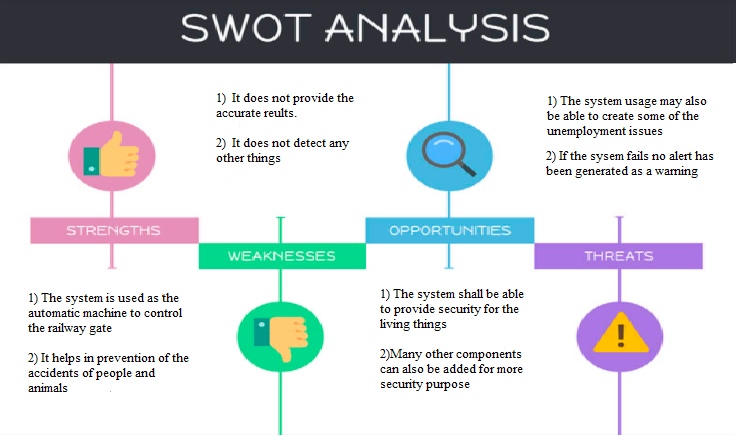
\* Where - It used in the railway track gate to prevent accidents

\* Who - Peoples, Railway-gate keeper, Gate man

\* What - Use to detect the moving train using sensors

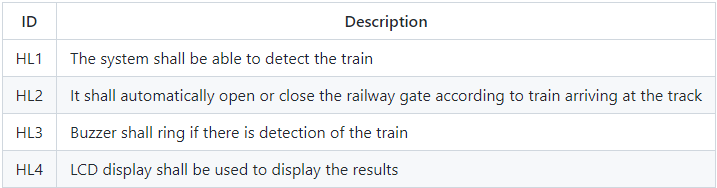
\* How - It can detect the train and automatically open or closes the railway gate and display these in LCD display

**2.1 SWOT ANALYSIS**

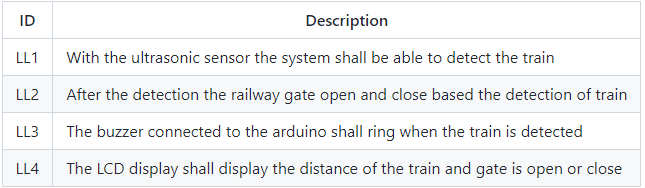


**3.1 REQUIREMENTS**

3.1.1 High Level Requirements



3.1.2 Low Level Requirements



**4.1 ADVANTAGES**

\* The main advantage of this system is the low cost implementation

\* Easy to detect the moving train

\* Helps to prevent accidents

\* Reduces human efforts in monotonous works

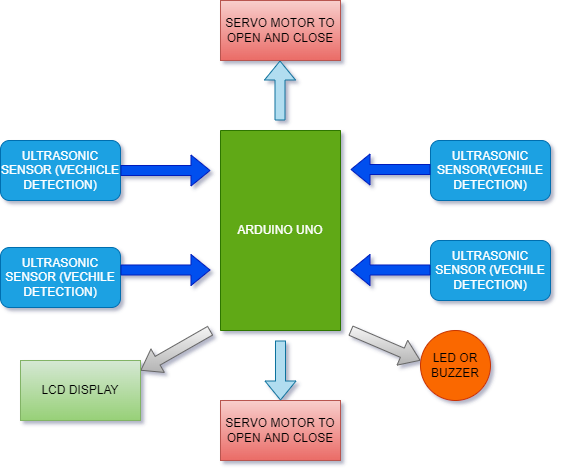
**5.1 FUTURE SCOPE**

\* Though this prototype is simple to build and highly reliable but there are some obstacles too. Rather than a train if an animal or other object is placed in front of the sensor the alarm will and the gate will be closed which is not desirable at all. There are also a scope of alerting the nearest railway station about arrival and departure of the train.

\* The problems indicated above can be overcome by adding some extra modules. Like adding the GSM module for future scope. After adding this module, upon arrival and departure of train, the GSM module will send an SMS to registered phone number for acknowledgement and safety. Also adding a pair of pressure sensor increases the chance of fault triggering of gate as well as alarm. After adding the pressure sensor, the Arduino closes the gate after receiving both signal from IR sensor as well as pressure sensor.

2 **ARCHITECTURE**

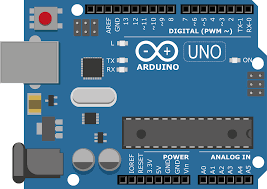
**2.1 BLOCK DIAGRAM**

****

## 2.2 COMPONENTS

**ARDUINO BOARD**

* The arduino board is the central unit of the system. The arduino uno is the microcontroller board based on the ATmega 328.



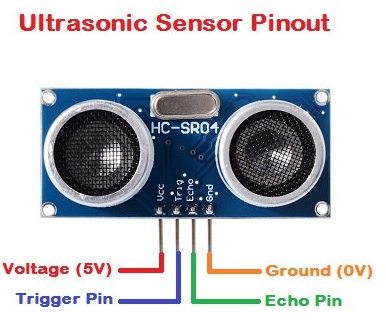
### BUZZER

* To alarm the status of the presence of train is done through buzzer.



### ULTRASONIC SENSOR (HC-SR04)

* The Ultrasonic sensor is the basic sensor that we use to determine the distance of an object. In another way we can say that it is used to measure how far away the object is located from a particular reference point. It has basically 4 pins; Trig pin, Echo pin, GND pin, VCC pin (+5V)

[](https://user-images.githubusercontent.com/98879965/157084969-4bc05b96-ac3f-4be8-8392-6dec76931333.png)

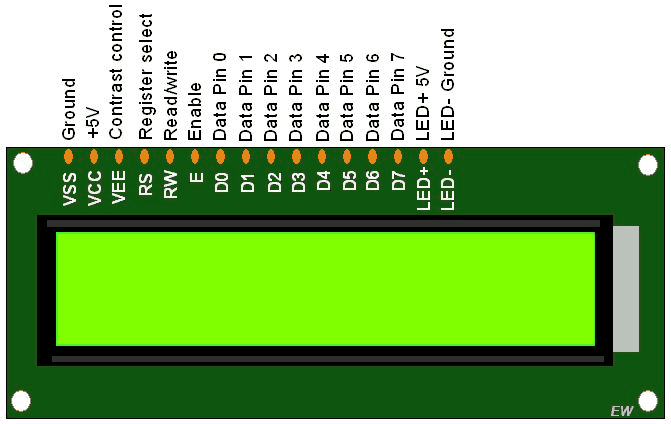
### SERVO MOTOR

* These are a special kind of motors which are employed for very specific movement at a particular angle. It is a rotary or linear actuator that permits precise control at certain angular or linear positions. Servo motor is not a single entity but a combination of a suitable motor along with a sensor which gives the feedback of the current position of the motor.

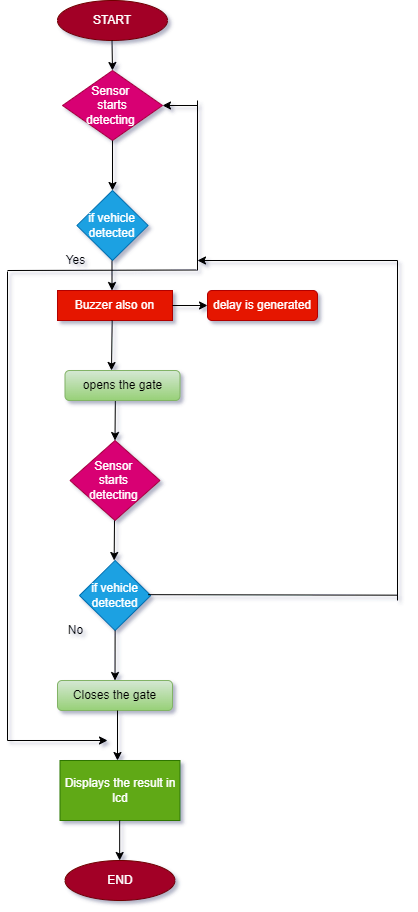
[](https://user-images.githubusercontent.com/98879965/157086122-9114cf30-6960-432a-93f4-a1702754f831.png)

### LCD DISPLAY

An LCD screen is an electronic display module that uses liquid crystal to produce a visible image.The LCD screen displays the distance between the Ultrasonic Sensor and the nearby object.It also displays the gate is open or close based on the detection

[](https://user-images.githubusercontent.com/98879965/157223990-4bebc759-a556-40dd-869e-452a4dc49437.png)

## 2.3 FLOWCHART

[](https://user-images.githubusercontent.com/98879965/157231149-6238d1ec-e29e-4799-9cb2-948bfeb2c474.png)

**3 IMPLEMENTATION**

**#include <LiquidCrystal.h> // includes the LiquidCrystal Library**

**#include <Servo.h> //includes the Servo library**

**LiquidCrystal lcd(1, 2, 4, 5, 6, 7); // Creates an LCD object. Parameters: (rs, enable, d4, d5, d6, d7)**

**const int trigpin1 = 13,trigpin2 = 3; // Attaches to the Trigger Pin**

**const int echopin1 = 12,echopin2 = 2 ; //Attaches to the Echo Pin**

**int buzzer=1;**

**unsigned long value1,value2,distance1,distance2; //Set long variable duration as the time between Trigger and Echo**

**Servo motorservo1; //create servo object to control a servo**

**Servo motorservo2; //create servo object to control a servo**

**int position1 = 0; //variable to store the servo position**

**void setup() {**

**lcd.begin(16, 2); // Initializes the interface to the LCD screen, and specifies the dimensions (width and height) of the display**

**pinMode(trigpin1, OUTPUT);**

**pinMode(echopin1, INPUT);**

**motorservo1.attach(13); // Attaches pin 13 to the servo object**

**lcd.begin(16, 2); // Initializes the interface to the LCD screen, and specifies the dimensions (width and height) of the display**

**pinMode(trigpin2, OUTPUT);**

**pinMode(echopin2, INPUT);**

**motorservo2.attach(13);**

**pinMode(trigpin1, OUTPUT);**

**pinMode(trigpin2, OUTPUT);**

**pinMode(buzzer, OUTPUT);**

**pinMode(echopin1, INPUT);**

**pinMode(echopin2, INPUT);**

**motorservo1.attach(8);**

**motorservo2.attach(7);**

**Serial.begin(9600);**

**}**

**void loop()**

**{**

**tone(buzzer, 450);**

**delay(500);**

**noTone(buzzer);**

**delay(500);**

**digitalWrite(trigpin1, LOW); //Send out a low pulse from Trigger Pin**

**delayMicroseconds(2); //A sharp delay of 2 microseconds before...**

**digitalWrite(trigpin1, HIGH); //...sending out a high pulse from the Trigger Pin**

**delayMicroseconds(10); //A longer delay of 10 microseconds before...**

**digitalWrite(trigpin1, LOW); //...sending out a low pulse again.**

**value1 = pulseIn(echopin1, HIGH); //duration is defined as the time at which the echopin detects a high pulse**

**distance1= (value1 \* 0.034) / 2; //distance calculation based on known speed of pulse wave and the detected time**

**digitalWrite(trigpin2, LOW); //Send out a low pulse from Trigger Pin**

**delayMicroseconds(2); //A sharp delay of 2 microseconds before...**

**digitalWrite(trigpin2, HIGH); //...sending out a high pulse from the Trigger Pin**

**delayMicroseconds(10); //A longer delay of 10 microseconds before...**

**digitalWrite(trigpin2, LOW); //...sending out a low pulse again.**

**value2 = pulseIn(echopin2, HIGH); //duration is defined as the time at which the echopin detects a high pulse**

**distance2= (value1 \* 0.034) / 2; //distance calculation based on known speed of pulse wave and the detected time**

**lcd.setCursor(0, 0); // Sets the location at which subsequent text written to the LCD will be displayed**

**lcd.print("Distance: "); // Prints string "Distance" on the LCD**

**lcd.print(distance2); // Prints the distance value from the sensor**

**delay(10);**

**lcd.setCursor(0, 1);**

**delay(10);**

**motorservo1.write(position1); //set the servo to the position - on the first run it initializes to 0**

**if (distance2 < 4.00) //which means an object is detected above the Ultrasound Sensor**

**{**

**lcd.print("Gate is opening!"); //first describe the Gate action**

**if (position1 < 90) //if the Servo is not already upright**

**{**

**delay(1500); //delay 1.5 seconds before reacting - this mimics the lagged response of toll gates in reality (for safety purposes).**

**for (position1 = 0; position1 <= 90; position1++) //gradually moves through 90 degrees**

**{**

**motorservo1.write(position1);**

**delay(10);**

**}**

**}**

**else //Servo is already upright - this accounts for when the train remains stationary before the gate**

**{**

**position1 = 90; //maintain the upright position**

**delay(10);**

**}**

**}**

**else //no object is detected above the Ultrasound Sensor**

**{**

**lcd.print("Gate is closed.");**

**if (position1 > 0) //Gate is upright and needs to close**

**{**

**delay(1500); //delay 1.5 seconds before reacting**

**for (position1 = 90; position1 >= 0; position1--) //gradually moves through 90 degrees**

**{**

**motorservo1.write(position1);**

**delay(10);**

**}**

**}**

**else //Gate is already down**

**{**

**position1 = 0; //remain down**

**delay(10);**

**}**

**}**

**motorservo2.write(position1); //set the servo to the position - on the first run it initializes to 0**

**if (distance2 < 4.00) //which means an object is detected above the Ultrasound Sensor**

**{**

**lcd.print("Gate is opening!"); //first describe the Gate action**

**if (position1 < 90) //if the Servo is not already upright**

**{**

**delay(1500); //delay 1.5 seconds before reacting - this mimics the lagged response of toll gates in reality (for safety purposes).**

**for (position1 = 0; position1 <= 90; position1++) //gradually moves through 90 degrees**

**{**

**motorservo2.write(position1);**

**delay(10);**

**}**

**}**

**else //Servo is already upright - this accounts for when the train remains stationary before the gate**

**{**

**position1 = 90; //maintain the upright position**

**delay(10);**

**}**

**}**

**else //no object is detected above the Ultrasound Sensor**

**{**

**lcd.print("Gate is closed.");**

**if (position1 > 0) //Gate is upright and needs to close**

**{**

**delay(1500); //delay 1.5 seconds before reacting**

**for (position1 = 90; position1 >= 0; position1--) //gradually moves through 90 degrees**

**{**

**motorservo2.write(position1);**

**delay(10);**

**}**

**}**

**else //Gate is already down**

**{**

**position1 = 0; //remain down**

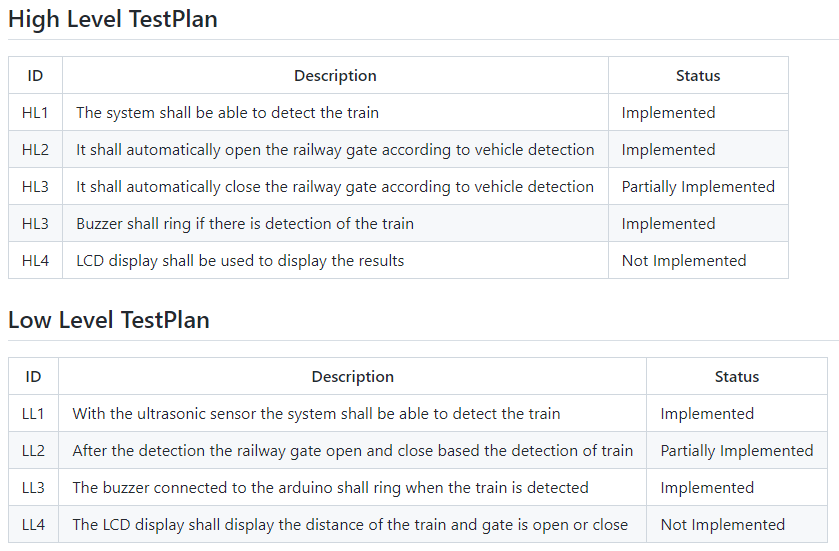
**delay(10);**

**}**

**}**

**}**

**4 TESTPLAN AND OUTPUT**

****

**5 MERITS AND DEMERITS**

## 5.1 Merits

* Prevention of accidents occurring at the railway gate
* Reliable machine, which operates the railway gate even without gatekeeper which makes it useful for operation at unmanned crossings
* Saves Labour cost
* Helpful in night time and in forest areas
* Implementation is easier
* Avoids manual errors
* The mechanism works on a simple principle

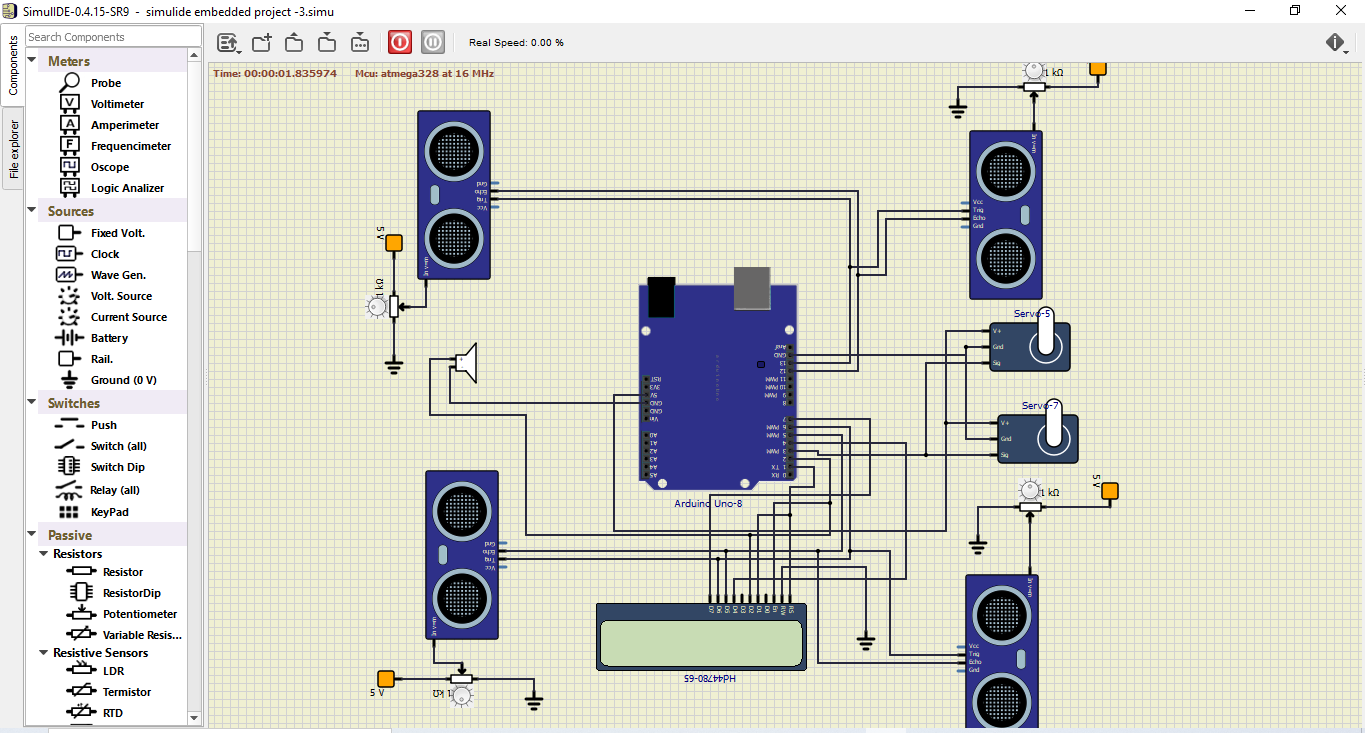
## 5.2 Demerits

* The Arduino board is a delicate device so it has to be handled carefully
* Sometimes the sensor may fail to detect
* Accuracy may be a problem

## 5.3 Applications

* Railways
* Industries
* Schools and College Junctions

**6 IMAGES AND VIDEO**

****

**7 CONCLUSION**

The project is an approach to provide the security to the RAILWAY transportation. This is also developed for the reduction of human effort. This project is an approach to the modern transportation system which provides machine automation and helps in accident prevention.

**8 REFERENCES**

**https://www.electricaltechnology.org/2020/04**