Vehicle Management System

A Project Review-3 Document

By

Anshuman Saboo - 17BCE0141

BTECH – COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTER ENGINEERING

VELLORE INSTITUTE OF TECHNOLOGY
VIT UNIVERSITY

VIT®
UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)
VELLORE ■ CHENNAI
www.vit.ac.in

INDEX

1.Introduction

1.1 Theoretical Background

Vehicle management system is to manage the parking of different type of vehicles at a predefined field. All vehicles should be allowed to enter in the parking field if there is a vacant parking spot. The project aims at helping in differentiating the different vehicle. It also aims to reduce the manual effort required to check if parking is available or not. The project will help in the parking of cars and trucks using sensors and see that the process of entering and exiting of vehicles go smoothly.

1.2 Motivation

The motivation of the project was that users will interact with the system directly without the use of a third party, the third party in this project was the workers who assist in managing a parking plot. We found that many times workers don’t utilise a parking plot properly or miss an empty parking space. With the help of sensors and a proper management system we can utilise any parking plot to its maximum. This can also help in easing traffic in big cities where parking plots are not easy to find.

1.3 Aim of the proposed Work

The aim of our project “Vehicle management system” is to manage the parking of different type of vehicles at a predefined field. All vehicles should be allowed to enter in the parking field if there is a vacant parking spot. The project aims at helping in differentiating the different vehicle. It also aims to reduce the manual effort required to check if parking is available or not. The project will help in the parking of cars and trucks using sensors and see that the process of entering and exiting of vehicles go smoothly.

1.4 Objective(s) of the proposed work

- Maximum use of parking plot
- Entry and exit of vehicle with the help of ultrasonic sensors
- Vehicle limitation
- Differentiating between different types of vehicle using ultrasonic sensor

2. Literature Survey

2.1 Survey of the Existing Models/Work

The survey of parking management in shopping malls was taken in consideration for our project. We studied how malls use digital boards to showcase the number of parking space available. We also studied how different Parking floors handle traffic. In developed countries like Dubai where parking is a huge business, we looked in how many places track the vehicles.

For sensors, we looked in the working of Arduino board, and how it detects any object. We also surveyed what materials that sensors can detect and how the ultrasonic waves determine how far the object is placed.

2.2 Summary/Gaps identified in the Survey

In our survey, we conclude that our project with the help of ultrasonics sensors can detect any type of vehicle. With this information, we can code for the parking of vehicles with a known and defined parking space.

3. Overview of the Proposed System

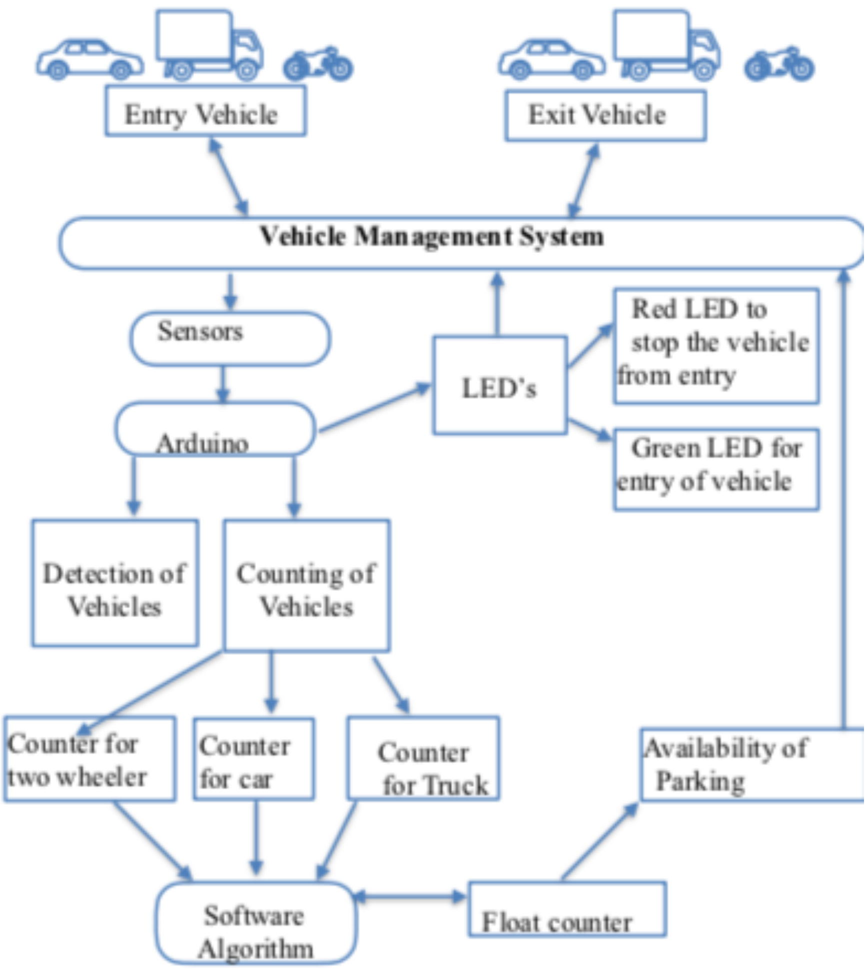
3.1 Introduction and Related Concepts

The vehicle management system detects the vehicle moments through ultrasonic sensors who feed the information to the software. The software is configured to see that all parking spots are utilised to the maximum. Each parking spot and sensors have to been carefully planned and known beforehand to the software. The software based on parking availability interacts with vehicle driver through LED lights. The vehicle management systems also helps in differentiating the different type of vehicle through the sensors. The project will use Arduino boards to gather information and feed it to the system. The code will generate with the given information and help in the parking system.

3.2 Framework, Architecture or Module for the Proposed System

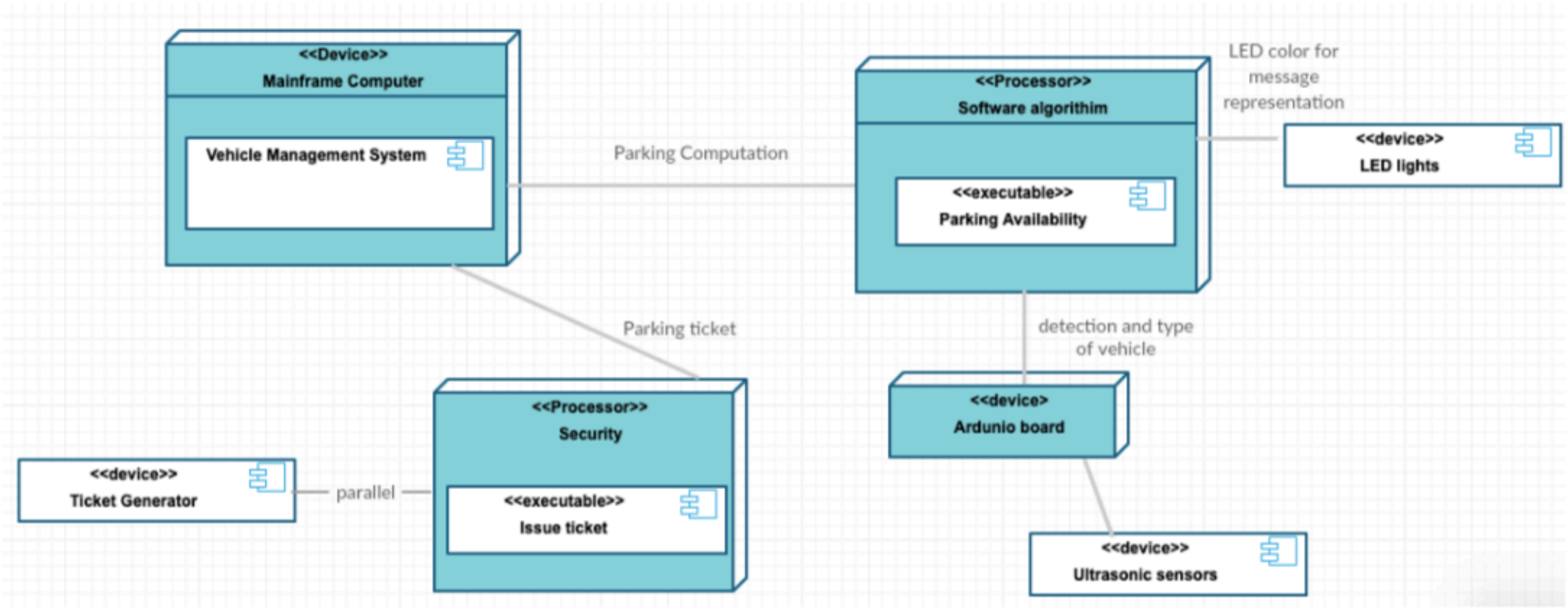
The spiral process model is best architectural strategies in the project. Arduino is an open source based prototyping platform used to sense and control physical devices. Thus for our project spiral model is most suitable as we will launch multiple prototypes by increasing the features in it. We are using the process model in our project because our project requires rapid prototyping of the product to make sure that the project and its components are working properly. After generation of every product model and its respective prototype the code is also updated and committed to Github. Spiral model make sure that the working prototype is generated for every version of the product developed. By the use of Github it is easy to fall to the code and design for a previous version thus making the project development more versatile. As and when new prototype will be launched it would be tested and after the testing a new development plan would be developed for new features. New constraints are then defined and circuit diagrams are updated according to the new needs of the development plan. Alternatives for the plan are explored and then the development of the prototype begins.

ARCHITECTURE DESIGN



3.3 Proposed System Model (ER Diagram/UML Diagram/Mathematical Modeling)

Deployment Diagram



4. Proposed System Analysis and Design

4.1 Introduction

The ultrasonic sensor has two components. Which are as follows :- Transmitter and ReceiverThe transmitter and receiver used to send the ultra sound using transmitter and receiver receives the ultrasound. This task is of high priority. Use of code to get the time and distance Stimulus/Response Sequences by using the time between sending and receiving the sound we establish the threshold for the sensor and A general use case for the entire project distance. We have to make sure that the sensor is given power from the Arduino. It requires 5V potential difference thus it has a ground and Vcc connected to the Arduino.

Giving visual signals to user—Use of LED glow
The LED's are used to notify user of various state of the machine. For instance tasks like the parking is full and the parking has space left etc. This task is of Medium priority as we can use Buzzers for this task too. Tactical use of LEDs' for informing the user Stimulus/Response Sequences This task is directly linked to the stimulus from ultrasonic. As the led gets stimulus from the ultrasonic then the code parse the input and lights the required LED. LED's shorter

4.2 Requirement Analysis

4.2.1 Functional Requirements

4.2.1.1 Product Perspective

The product will help in the services where labour wont be required. With the help of ultrasonic sensors we will detect the entry of a vehicle. These sensors feed the information to the Arduino IDE software which will enable the LED light colour depending on the situations. The product can help in the parking management where lots of labours required and can be modified for many types of vehicles and plot size.

4.2.1.2 Product Functions

- The primary function of the product is to assist the user with parking safely and accurately.
- Generating a Parking fee based on parking time
- Feeding instruction to software through sensors for generating different LED lights to interact with user.
- Sensing the entry and exit of different types of vehicles
- Whether the parking plot is filled or not

- Two wheeler and Four wheeler vehicles have their own system

4.2.1.3 User characteristics

The system is intended for use by the operator of the vehicle, therefore a user of the system must be legally allowed to operate a vehicle according to the laws applicable where they live. Users that are the drivers have to know how to operate the vehicle and safety issues. Different vehicles have different driving license which users should have in order to park their respective vehicle.

4.2.1.4 Assumption & Dependencies

The assumptions are of vehicle size and differences, like we have assumed that a truck will take the parking size of two cars. Although this system will detect failures and subsequently prevent itself from operating, it is expected that hardware components of the subsystems have their own failsafe procedures in place as well. Software will also see that any error should be handled.

4.2.1.5 Domain Requirements

The environment will be well defined parking plot with appropriate vehicle space to park. Different vehicles parking space will be defined and where to park, One truck will take the size of two four wheelers. Two wheelers will have their parking space. The Arduino and its sensors will be placed at the entry and exit gates of the parking plot. LED lights will be placed at positions where the user can see it properly. The parking plots have to be well defined as the software will need the correct amount to operate and decide the situation.

4.2.1.6 User Requirements

The primary function of the product is to assist the user with parking safely and accurate. The user needs to only know what the different LED light portray instruction. The LED light will glow based on the parking availability which gets its command from the software system. The user will not need to worry, as there will be boards present outside the entry and exit points which will educate the user on how the system and LED lights work.

4.2.2 Non Functional Requirements

(i) Use of a buzzer if an object is staying in the entrance bay for more than required time thus blocking the way

(ii) The loop halter in the code is responsible for making sure the code pauses its execution when the capacity of the lot is full

(iii) Another condition in the code would be needed when the lot is empty then the exit counter should stop working to save power

4.2.2.1 Product Requirements

4.2.2.1.1 Efficiency

The number of tasks in a given time performed will check the efficiency. In the project, we will see that the entry and exit of vehicle at the same time should not disrupt the system and the counter used in our code. That is why we use local counter in our entry and exit function.

4.2.2.1.2 Reliability

The reliability of the system is checked when there is no parking space available and the system detects at that instant. Also to check if it is reliable for entry of different type of vehicle.

4.2.2.1.3 Portability

To make sure the project can be shifted to a different Operating system and it works for several platforms. If some new obstacle comes, we will check the system operates. For example, if a parking space has become unfit for parking, we will need to see that counter does not exceed one less that limitation value.

4.2.2.1.4 Usability

The vehicle management system will make an effort to learn if different types of vehicle are introduced. Input provided can be made useful and interpret results of a program.

4.2.2.2 Organizational Requirements

4.2.2.2.1 Implementation Requirements

The product should be sure of the type of vehicle and the parking fee associated with it. The product should also be sure of the colour of LED light shown to user as that is the only communication wit user. Product should be not get confused when a vehicle enters and exits or a vehicle entering and another exiting at the same time. Product should also check that parking is utilised to its maximum. Also the product should have room for future improvement in case of any modification to type of vehicle, plot size etc.

4.2.2.2.2 Engineering Standard Requirements

The use of Male to male jumpers and male to female jumpers is vital as the Arduino and the leds and the sensors communicate using electrical signals. When the system takes the input using ultrasonic sensors it writes the data on serial monitor. The system detects when the written data goes discontinuous and then gives the signals to the LED which communicate the message to the user using LED glow i.e. Visual signals.

Certified Tester Foundation Level (CTFL) from ISTQB, the International Software Testing Qualifications Board and Certified Software Tester (CSTE) from Software Certifications (a division of the Quality Assurance Institute) are some regulation tests whose certificates can provide a boost to the products sales

4.2.3 System Requirements

4.2.3.1 H/W Requirements

The system constitutes of an arduino board powered by a 12V Battery. We code the arduino in C sharp and feed it to the arduino where it gets converted into assembly language and gets stored in its register. After this whenever the arduino is connected to a power source the assembly code starts running in a loop. Hardware required :

- Arduino UNO
- Ultrasonic sensors
- LED's
- Male to male jumpers
- Male to female jumpers

4.2.3.2 S/W Requirements

The C sharp code uses two specific functions and global variables. Each time the arduino is powered up using a battery or an adapter power supply the script first runs the setup function once which initializes the variables etc. The the loop function runs unless the power runs out We used the ARDUINO 1.8.8 software to code the 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. C Sharp is the computer language used Arduino Code Editor is the code editor.

5. Implementation

5.1 Code and snapshot

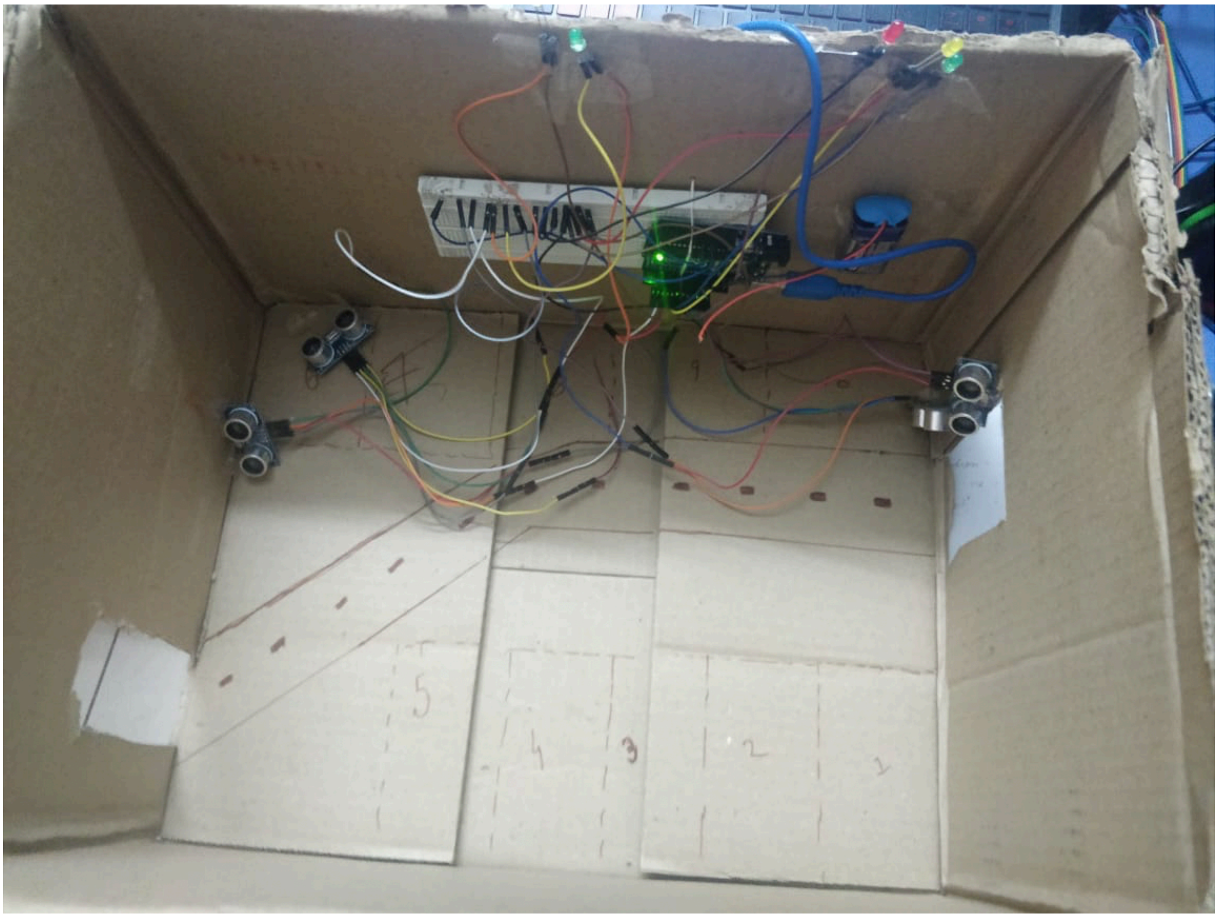
```
// Define pins for ultrasonic and LED int const trigPin = 4; //exit
int const echoPin = 5;

int const trigPin2 = 11; // entery int const echoPin2 = 10;

int const trigPin3 = 13; //entery int const echoPin3 = 12;

int const trigPin4 = 2; //exit int const echoPin4 = 3;

int const led = 6; int const led2 = 7; int const led3 = 8;

//int const led5 = 0;
```

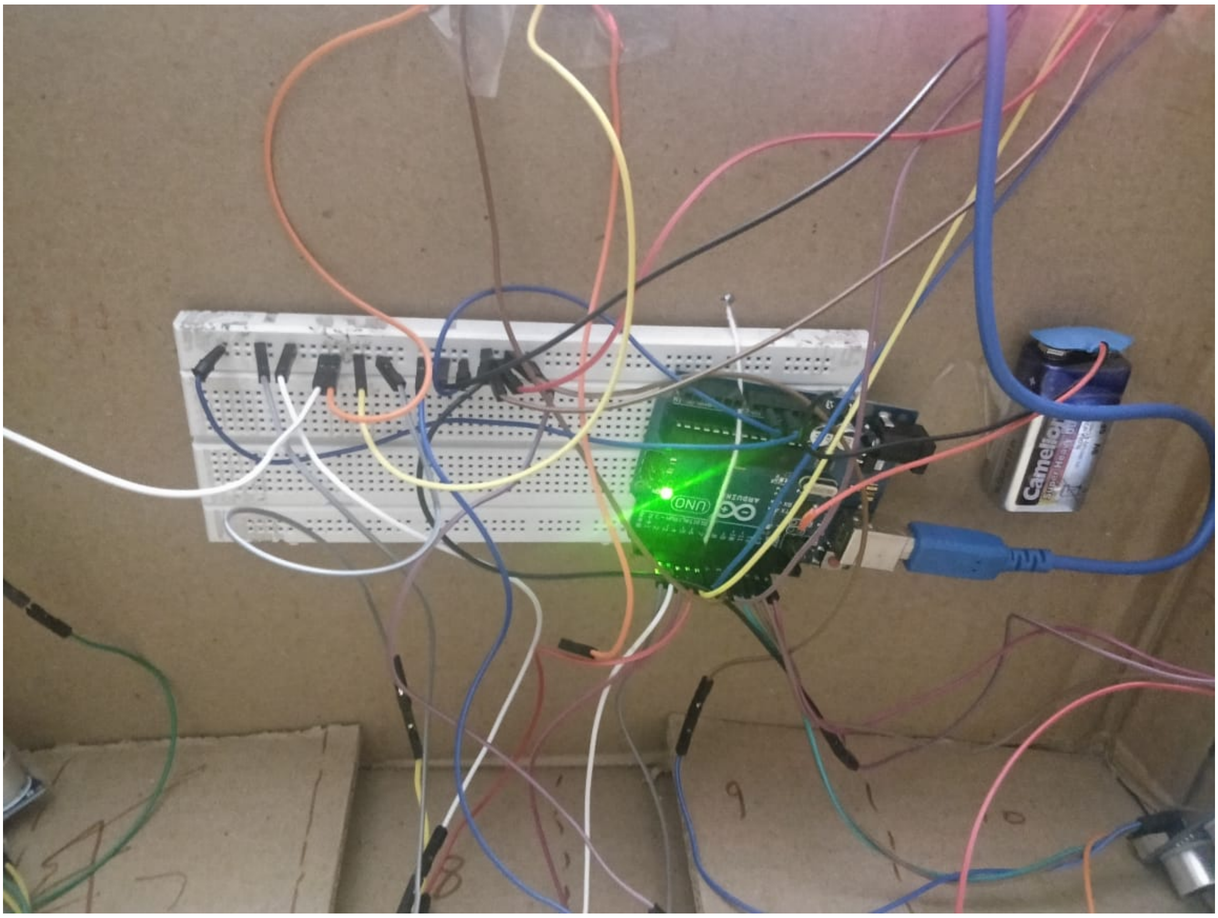
```
int const counter_led = 9; long count=0;
long count2=0;
long count3=0;

long count4=0; long tk=0;

void setup() {

Serial.begin(115200);
pinMode(trigPin, OUTPUT); // trig pin will have pulses output pinMode(echoPin, INPUT); // echo pin
should be input to get pulse width

pinMode(trigPin2, OUTPUT); // trig pin will have pulses output pinMode(echoPin2, INPUT); // echo pin
should be input to get pulse width

pinMode(trigPin3, OUTPUT); // trig pin will have pulses output pinMode(echoPin3, INPUT); // echo pin
should be input to get pulse width

pinMode(trigPin4, OUTPUT); // trig pin will have pulses output pinMode(echoPin4, INPUT); // echo pin
should be input to get pulse width

pinMode(led, OUTPUT); // led pin is output to control LED lights pinMode(led2, OUTPUT); // led pin is
output to control LED lights pinMode(led3, OUTPUT); // led pin is output to control LED lights

// pinMode(led4, OUTPUT); // led pin is output to control LED lights pinMode(led5, OUTPUT); // led
pin is output to control LED lights

pinMode(counter_led, OUTPUT); }

void loop() {

if(tk>10) digitalWrite(counter_led, HIGH);

else
digitalWrite(counter_led, LOW);

entry1(); entry2(); exit1();

// exit2(); }

void entry1() {

int duration2, distance2;

digitalWrite(trigPin2, HIGH); delay(1); digitalWrite(trigPin2, LOW);

duration2 = pulseIn(echoPin2, HIGH); distance2 = (duration2/2) / 74;
if (distance2 < 5) {

count2++; digitalWrite(led2, HIGH);

} else {

if(count2!=0) {

//increment tk value and display the count

tk++; Serial.print("\n+++++++++++++++++++"); Serial.print('\n');

Serial.print(tk);
Serial.print('\n'); Serial.print("+++++++++++++++++++++");

} count2=0;
// LED off

digitalWrite(led2, LOW); }

delay(60); }

void entry2() {

int duration3, distance3;

digitalWrite(trigPin3, HIGH); delay(1); digitalWrite(trigPin3, LOW);

duration3 = pulseIn(echoPin3, HIGH); distance3 = (duration3/2) / 74;

if (distance3 < 5) {

count3++; digitalWrite(led, HIGH);

} else {

if(count3!=0) {

//increment tk value and display the count

tk++; Serial.print("\n+++++++++++++++++++"); Serial.print('\n');
Serial.print(tk);
Serial.print('\n'); Serial.print("+++++++++++++++++++++");

} count3=0;

// LED off

digitalWrite(led, LOW); }

delay(60); }

void exit1() {

int duration, distance;

digitalWrite(trigPin, HIGH); delay(1); digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH); distance = (duration/2) / 74;

if (distance < 5) {

count++; digitalWrite(led3, HIGH);

} else {

if(count!=0) {

//increment tk value and display the count

tk--; Serial.print("\n------------------"); Serial.print('\n');
Serial.print(tk);
Serial.print('\n'); Serial.print("--------------------");

} count=0;

// LED off

digitalWrite(led3, LOW); }
```

```
delay(60);

}

void exit2() {

//

//

int duration4, distance4;

digitalWrite(trigPin4, HIGH); delay(1); digitalWrite(trigPin4, LOW);

duration4 = pulseIn(echoPin3, HIGH); distance4 = (duration4/2) / 74;

if (distance4 < 5) {

count4++; digitalWrite(led4, HIGH);

} else {

if(count4!=0) {

//increment tk value and display the count

tk--; Serial.print("\n------------------"); Serial.print('\n');
Serial.print(tk);
Serial.print('\n'); Serial.print("--------------------");

}

count4=0;

// LED off digitalWrite(led4, LOW);

} delay(60);

}
```

Snapshot:

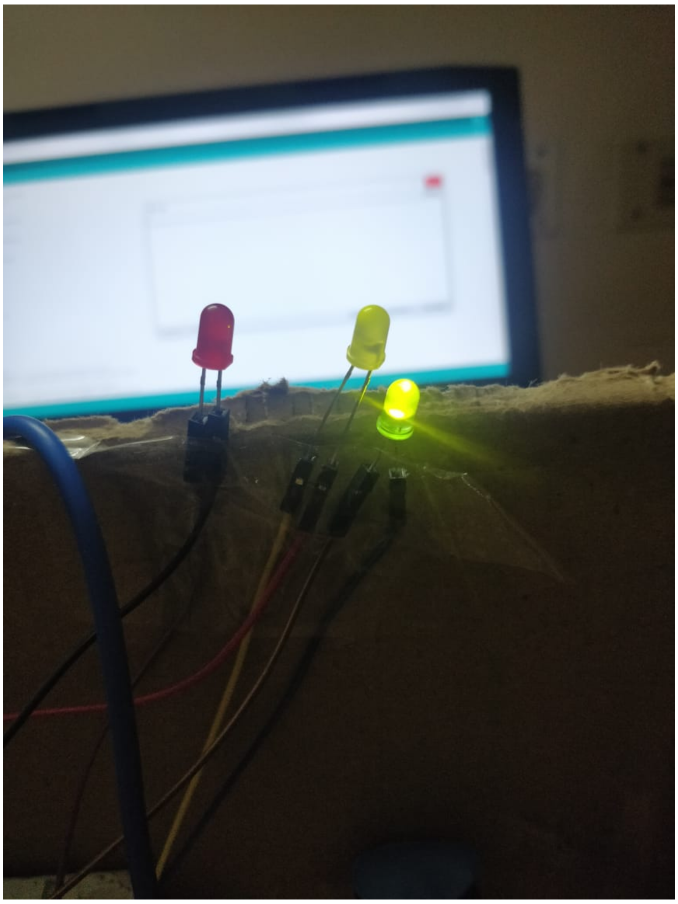
Setup of the vehicle management system:





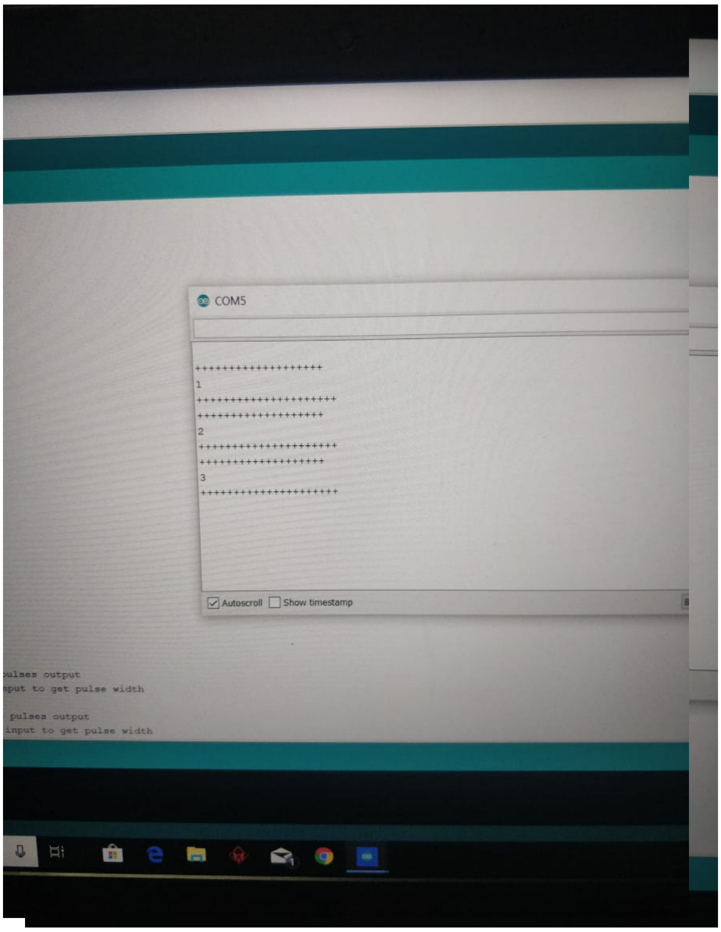
Entry of car:

Detection of entry of car will be confirmed by the glowing of Green LED at the entry gate



Counter decreases and the number of vehicles is decreased and counted. Here it is **one** parking space for car
Entry of Truck: (Here in the picture wallet is assumed as truck)
Truck has height greater than car. So we have taken a long object here.

Detection of entry of truck will be confirmed by the glowing of Green LED and Yellow LED simultaneously at the entry gate
Counter decreases and the number of vehicles is decreased and counted. Here it is **two** parking space for truck

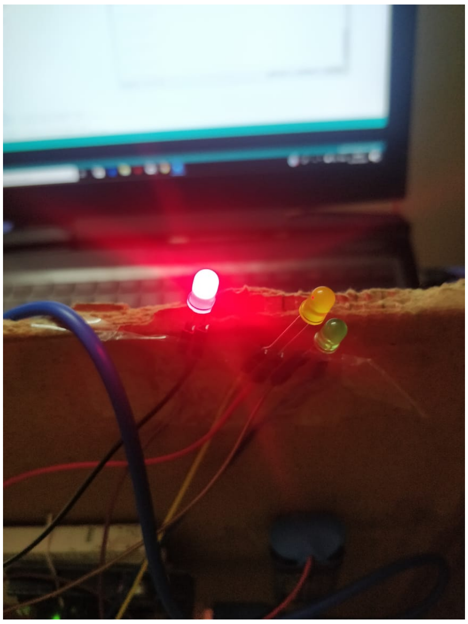


```
++++++++++++++++++++
1
++++++++++++++++++++++
++++++++++++++++++++
2
++++++++++++++++++++++
++++++++++++++++++++
3
++++++++++++++++++++++
```

☑ Autoscroll □ Show timestamp

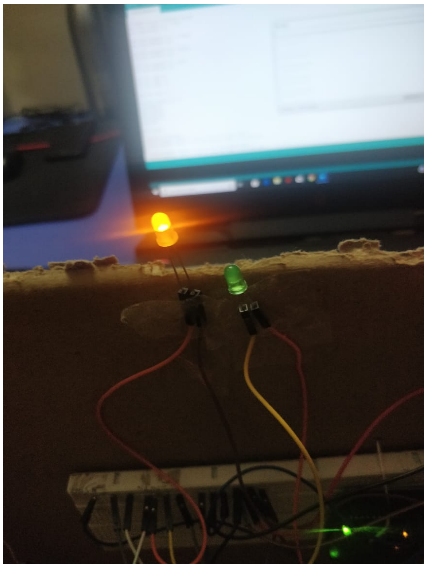
```
ulses output
nput to get pulse width

 pulses output
 input to get pulse width
```

Parking limitation: We have assumed that there are 10 parking space. If all parking space are full then red LED will glow at the entry gate

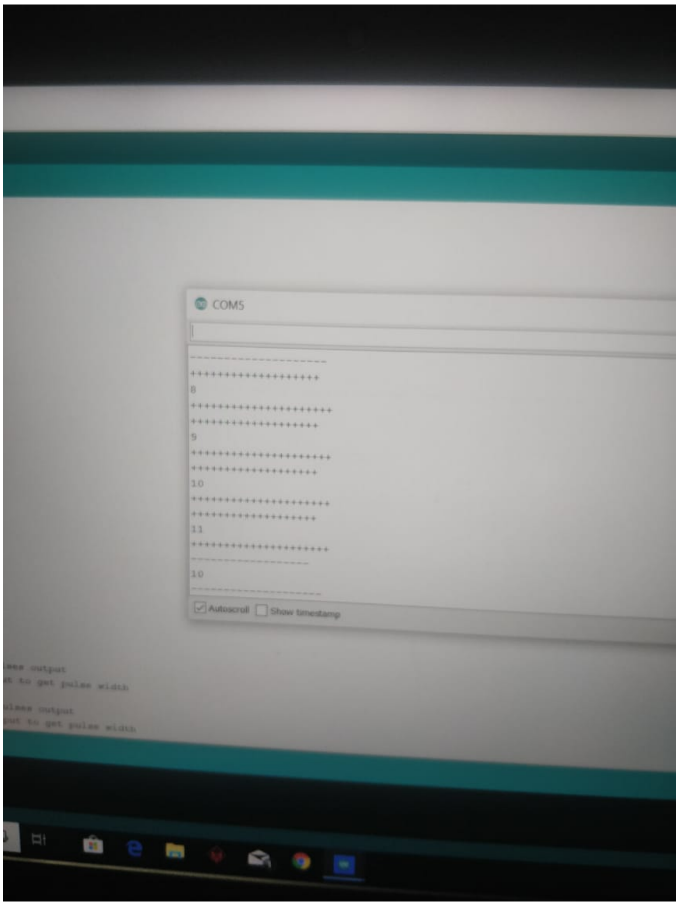


Exit of vehicle:

Detection of exit of vehicle will be confirmed by the glowing of yellow LED at the exit gate



Counter decreases and the number of vehicles is decreased and counted.



```
--------------------
+++++++++++++++++++
8
++++++++++++++++++++
+++++++++++++++++++
9
++++++++++++++++++++
+++++++++++++++++++
10
++++++++++++++++++++
+++++++++++++++++++
11
++++++++++++++++++++
------------------
10
--------------------
```

```
lses output
ut to get pulse width

ulses output
put to get pulse width
```

Result: We have seen the detection of vehicle at entry and exit gate. The limitation of vehicle turns the Red LED on and stops entry of vehicle. Different types of vehicles are detected by ultrasonic sensors and feed to the software system. The vehicle management system works efficiently for all cases.