PRELIMINARY INVESTIGATION

1.1 Introduction

Seeking a vacant parking space during peak hours in areas like Hospitals, Hotels & Shopping Centers, Airports, Universities, and Exhibitions & Convention Center has always been frustrating for many drivers. Surveys says that traffic generated by cars searching for vacancies in Parking Spaces is up to 40% of the total traffic. Now that is a serious issue to look after, and Smart Parking System is one of the best available solutions to at least reduce the traffic congestion caused due to the above problem. This application gives information about the occupancy status of the spaces in the parking lot equipped with sensors that detect the presence of vehicles.Smart Parking is an Internet of Things (IoT) based application, used to detect the available parking slots. This app uses ultrasonic sensor to detect the presence of a vehicle (whether the parking slot is occupied or not). Based on the parking slot occupancy, the status (occupied/unoccupied) is displayed on the web application dashboard. In real time, the environment have sensors and devices embedded into parking spaces, transmitting data on the occupancy status; and the vehicle drivers can search for parking availability using their mobile phones or any infotainment system that is attached to the vehicle. Hence the driver would know where there is an available spot to park his vehicle in less time, reducing the energy consumption and air pollution.

- **Advantages:**

- The proposed system is user friendly, easy to use and efficient.
- Reduces traffic congestion.
- Reduces pollution.
- Saves Time.
- Check parking status over the internet.

- **Disadvantages:**
    - This requires an active internet connection

1.2 Existing System & Its Limitation

- Existing system in parkingLot is a manually handled system.
- Parking status is not visible to everyone
- Sometimes parking spaces are hoarded .
- Difficutly in finding a parking spot.
- Loss of fuel and time because of searching for parking.

1.3 Proposed System

- Proposed system is an IOT based system which will allow users/drivers to find parking at any given time .
- Available parking can be seen by the users
- Total free parking can be see by the user.
- Parking available counter will be reduced by one as soon as car enters parking lot.
- Parking available counter will be increased by one as soon as car leaves parking lot.
- Parking slot light will turn green if that slot is available to show that it is vacant.
- Parking slot light will turn red if that slot is not available to show that it is not vacant.
- Entry/Exit gates will be blocked by barricade which will open automatically when a car comes near them.

1.4 Hardware and Software to be used

- **Hardware Requirement: -**
  - Laptop or PC: -
    1. I5 Processor Based Computer or higher
    2. 2GB RAM
    3. 10 GB Hard Disk
    4. Esp8266
    5. Led lights.
    6. Infrared sensors.
    7. Ultrasonic sensors.
    8. Servo Motors.

- **Software Requirement: -**
  - Laptop or PC: -
  1. Windows 7 or higher
  2. Arduino ide
  3. Firebase libraries

1.5 Fact Finding Technique

1. Do you find parking easily ?

2. How often do you have to wait for finding a parking slot?

3. In your opinion, the website interface is?

4. In your opinion, moving from normal to a smart parking system, is it good or bad?

5. Should the reserve parking spot option be available?

6. How easy would it be finding a parking slot in smart parking compared to traditional parking ?

7. In your opinion how can we make parking system easy and convenient for the user?

8. In your opinion, how can this smart parking system be improved?

1.6 Feasibility study

What Is a Feasibility Study?

As the name implies, a feasibility study is used to determine the viability of an idea, such as ensuring a project is legally and technically feasible as well as economically justifiable. It tells us whether a project is worth the investment—in some cases, a project may not be doable. There can be many reasons for this, including requiring too many resources, which not only prevents those resources from performing other tasks but also may cost more than an organization would earn back by taking on a project that isn't profitable.
A well-designed study should offer a historical background of the business or project, such as a description of the product or service, accounting statements, details of operations and management, marketing research and policies, financial data, legal requirements, and tax obligations. Generally, such studies precede technical development and project implementation.

Benefits of feasibility study:

The importance of a feasibility study is based on organizational desire to "get it right" before committing resources, time, or budget. A feasibility study might uncover new ideas that could completely change a project's scope. It's best to make these determinations in advance, rather than to jump in and to learn that the project won't work. Conducting a feasibility study is always beneficial to the project as it gives you and other stakeholders a clear picture of the proposed project.

Five Areas of Project Feasibility

A feasibility study evaluates the project's potential for success; therefore, perceived objectivity is an essential factor in the credibility of the study for potential investors and lending institutions. There are five types of feasibility study—separate areas that a feasibility study examines, described below.

1. Technical Feasibility:

This assessment focuses on the technical resources available to the organization. It helps organizations determine whether the technical resources meet capacity and whether the technical team is capable of converting the ideas into working systems. Technical feasibility also involves evaluation of the hardware, software, and other technical requirements of the proposed system. As an exaggerated example, an organization wouldn't want to try to put Star Trek's transporters in their building—currently, this project is not technically feasible.

2. Economic Feasibility:

This assessment typically involves a cost/ benefits analysis of the project, helping organizations determine the viability, cost, and benefits associated with a project before financial resources are allocated. It also serves as an independent project assessment and enhances project credibility—helping decision-makers determine the positive economic benefits to the organization that the proposed project will provide.

3. Legal Feasibility:

This assessment investigates whether any aspect of the proposed project conflicts with legal requirements like zoning laws, data protection acts or social media laws. Let's say an organization wants to construct a new office building in a specific location. A feasibility study might reveal the organization's ideal location isn't zoned for that type of business. That organization has just saved considerable time and effort by learning that their project was not feasible right from the beginning.

4. Operational Feasibility:

This assessment involves undertaking a study to analyze and determine whether—and how well—the organization's needs can be met by completing the project. Operational feasibility studies also examine how a project plan satisfies the requirements identified in the requirements analysis phase of system development.

5. Scheduling Feasibility:

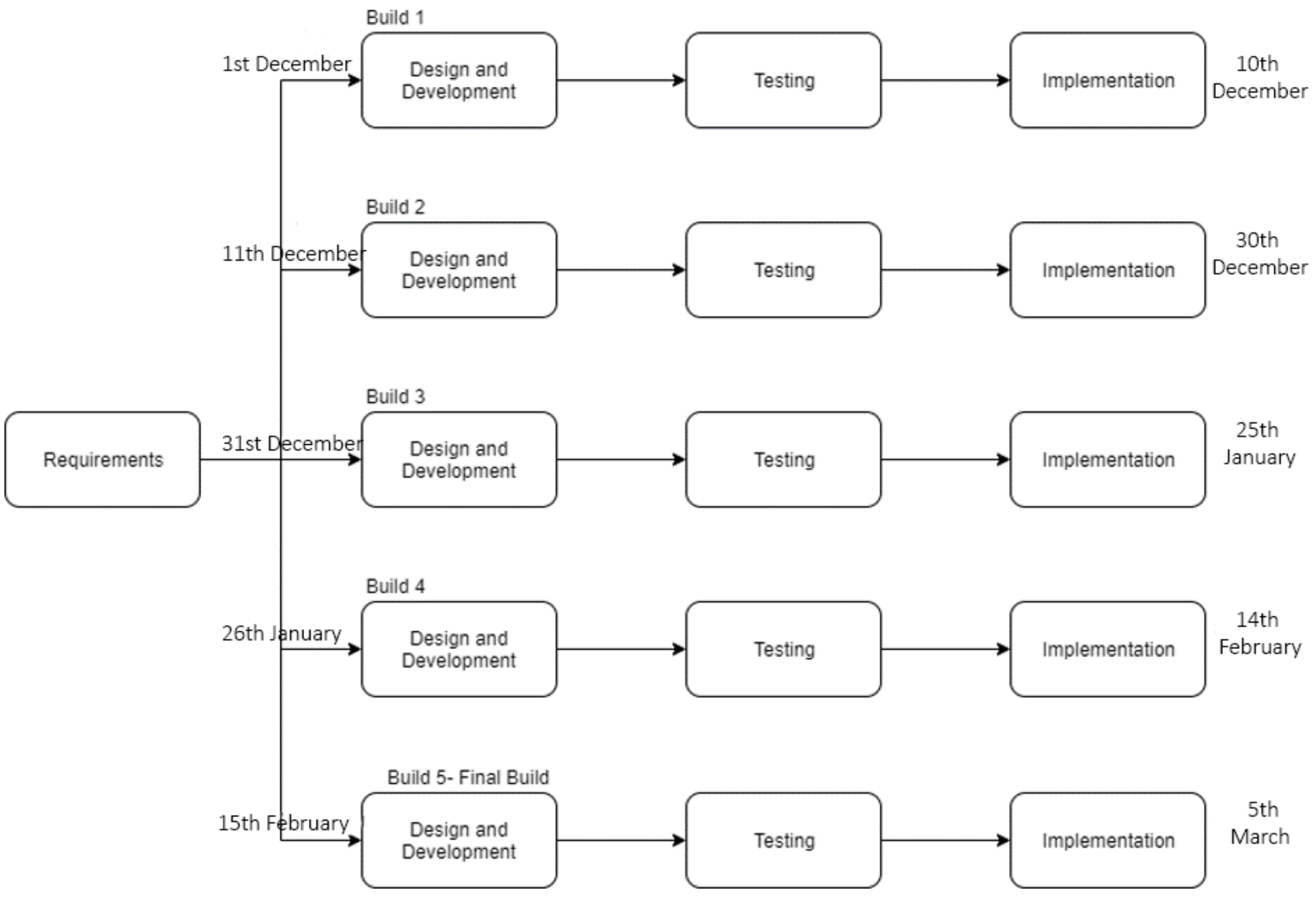
This assessment is the most important for project success; after all, a project will fail if not completed on time. In scheduling feasibility, an organization estimates how much time the project will take to complete.

When these areas have all been examined, the feasibility study helps identify any constraints the proposed project may face, including:

- Internal Project Constraints: Technical, Technology, Budget, Resource, etc.
- Internal Corporate Constraints: Financial, Marketing, Export, etc.
- External Constraints: Logistics, Environment, Laws and Regulations, etc.

1.7 Project Schedule

Sr. No.	Task	From	To	Time
1	Build 1	1st December, 2019	10th December, 2019	11 days
2	Build 2	11th December, 2019	30th December, 2019	20 days
3	Build 3	31st December, 2019	25th January, 2020	25 days
4	Build 4	26th January, 2020	14th February, 2019	18 days
5	Build 5	15th February, 2020	5th March , 2020	19 days



1.8 SDLC Model:

Iterative Model

In the Iterative model, iterative process starts with a simple implementation of a small set of the software requirements and iteratively enhances the evolving versions until the complete system is implemented and ready to be deployed.

An iterative life cycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which is then reviewed to identify further requirements. This process is then repeated, producing a new version of the software at the end of each iteration of the model.

Advantages:

The advantage of this model is that there is a working model of the system at a very early stage of development, which makes it easier to find functional or design flaws. Finding issues at an early stage of development enables to take corrective measures in a limited budget.

The advantages of the Iterative and Incremental SDLC Model are as follows −

- Some working functionality can be developed quickly and early in the life cycle.
- Results are obtained early and periodically.
- Parallel development can be planned.
- Progress can be measured.
- Less costly to change the scope/requirements.
- Testing and debugging during smaller iteration is easy.

- Risks are identified and resolved during iteration; and each iteration is an easily managed milestone.
- Easier to manage risk - High risk part is done first.
- With every increment, operational product is delivered.
- Issues, challenges and risks identified from each increment can be utilized/applied to the next increment.
- Risk analysis is better.
- It supports changing requirements.
- Initial Operating time is less.
- Better suited for large and mission-critical projects.
- During the life cycle, software is produced early which facilitates customer evaluation and feedback.

Disadvantages:

The disadvantage with this SDLC model is that it is applicable only to large and bulky software development projects. This is because it is hard to break a small software system into further small serviceable increments/modules.

The disadvantages of the Iterative and Incremental SDLC Model are as follows −

- More resources may be required.
- Although cost of change is lesser, but it is not very suitable for changing requirements.
- System architecture or design issues may arise because not all requirements are gathered in the beginning of the entire life cycle.
- Defining increments may require definition of the complete system.

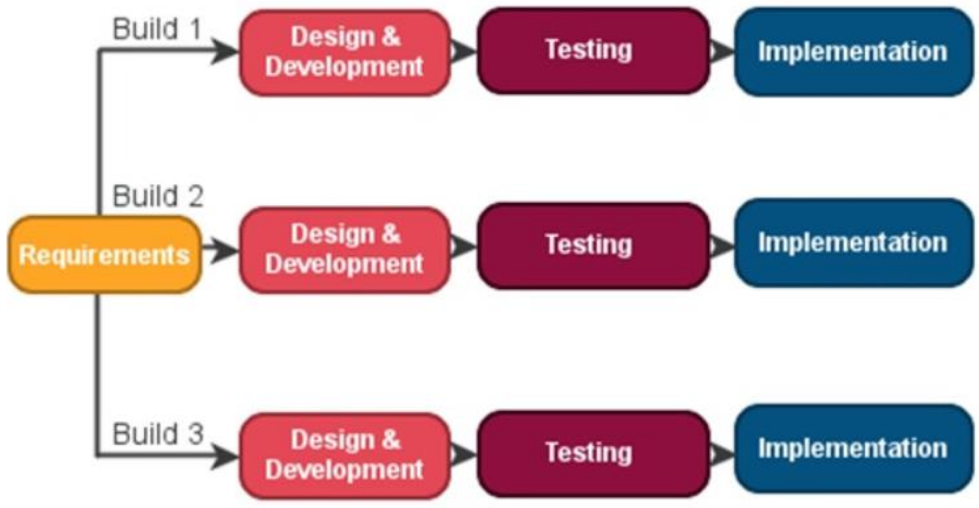
- Not suitable for smaller projects
- Management complexity is more.
- End of project may not be known which is a risk.
- Highly skilled resources are required for risk analysis.
- Projects progress is highly dependent upon the risk analysis phase.

Application:

Like other SDLC models, Iterative and incremental development has some specific applications in the software industry. This model is most often used in the following scenarios −

- Requirements of the complete system are clearly defined and understood.
- Major requirements must be defined; however, some functionalities or requested enhancements may evolve with time.
- There is a time to the market constraint.
- A new technology is being used and is being learnt by the development team while working on the project.
- Resources with needed skill sets are not available and are planned to be used on contract basis for specific iterations.

- There are some high-risk features and goals which may change in the future.

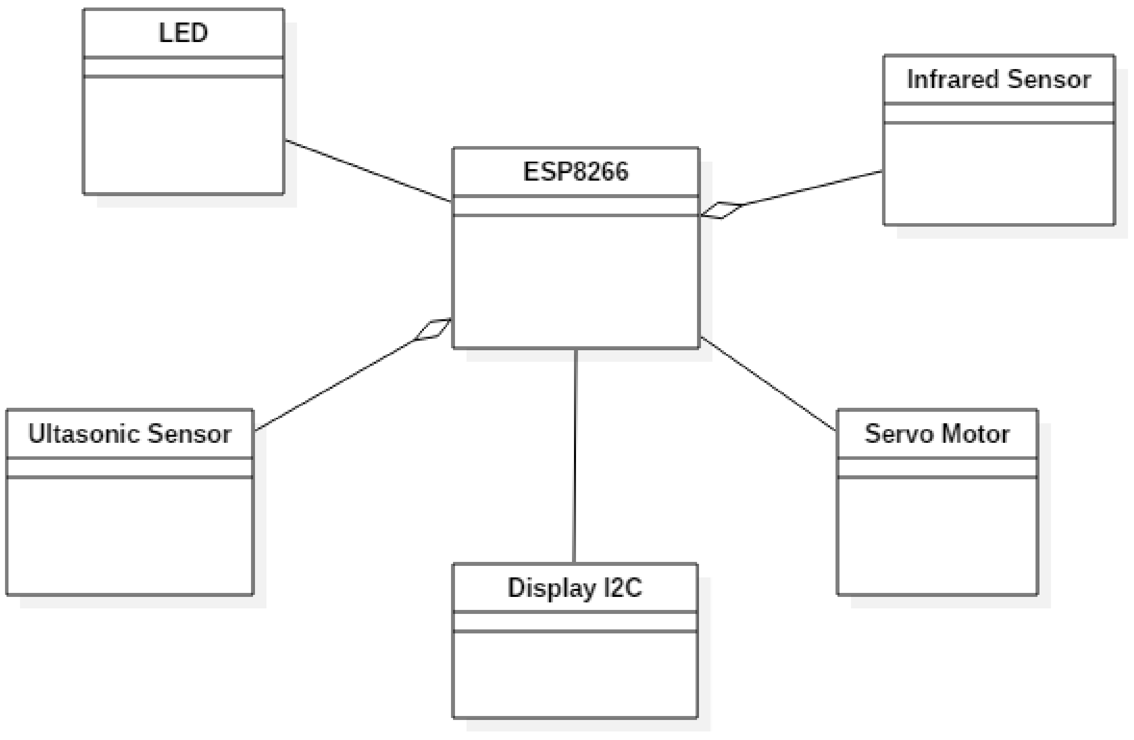


SYSTEM ANALYSIS

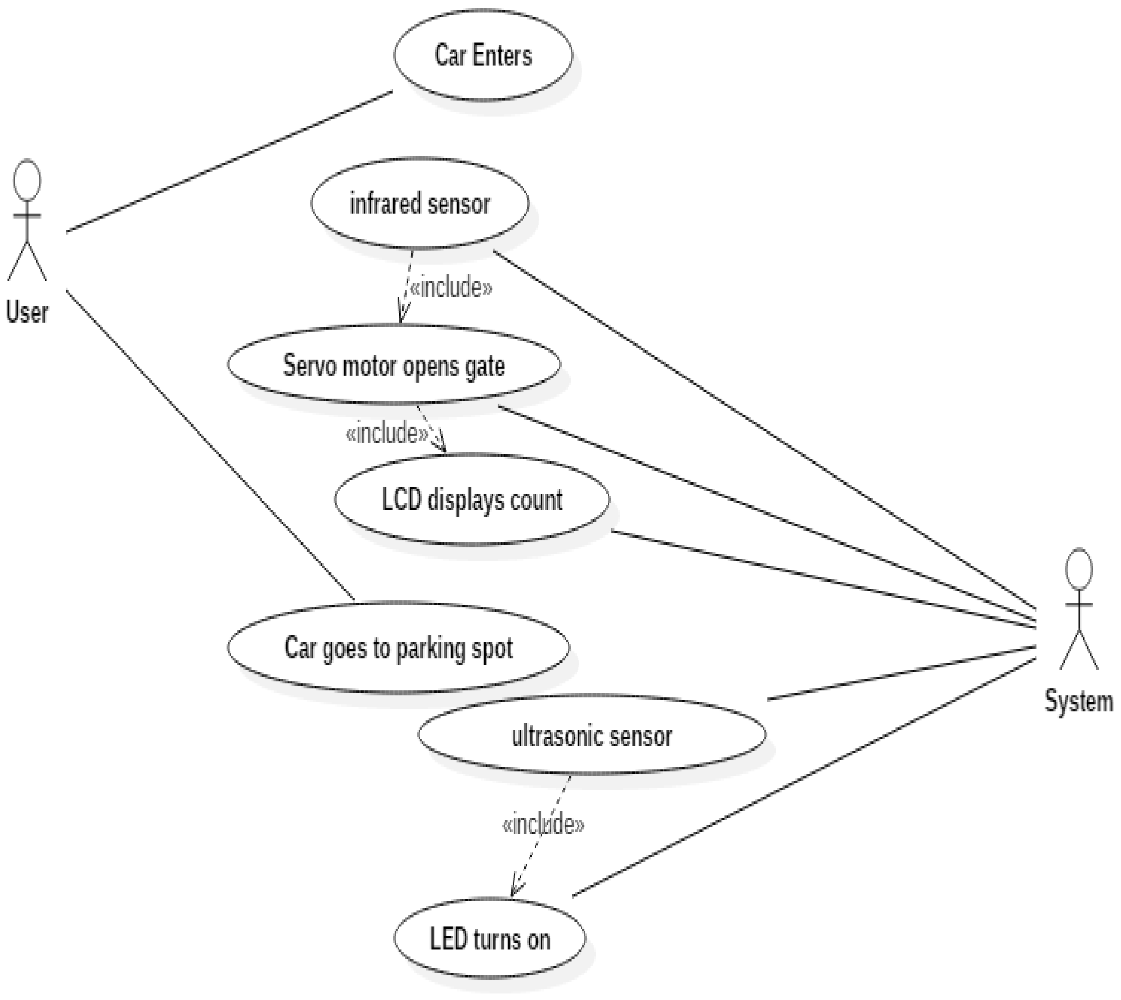
2.1 Event Table

EVENT	TRIGGER	SOURCE	ACTIVITY	RESPONSE	DESTINATION
Car comes in	Infrared sensor	Car	increment count	Opens gate , display count	user
Car parks in spot	Ultrasonic sensor	Car	Use the spot	LED color changes	User
Car goes out	Infrared sensor	Car	Decrement count	Opens gate , display cont	User

2.2 Class Diagram

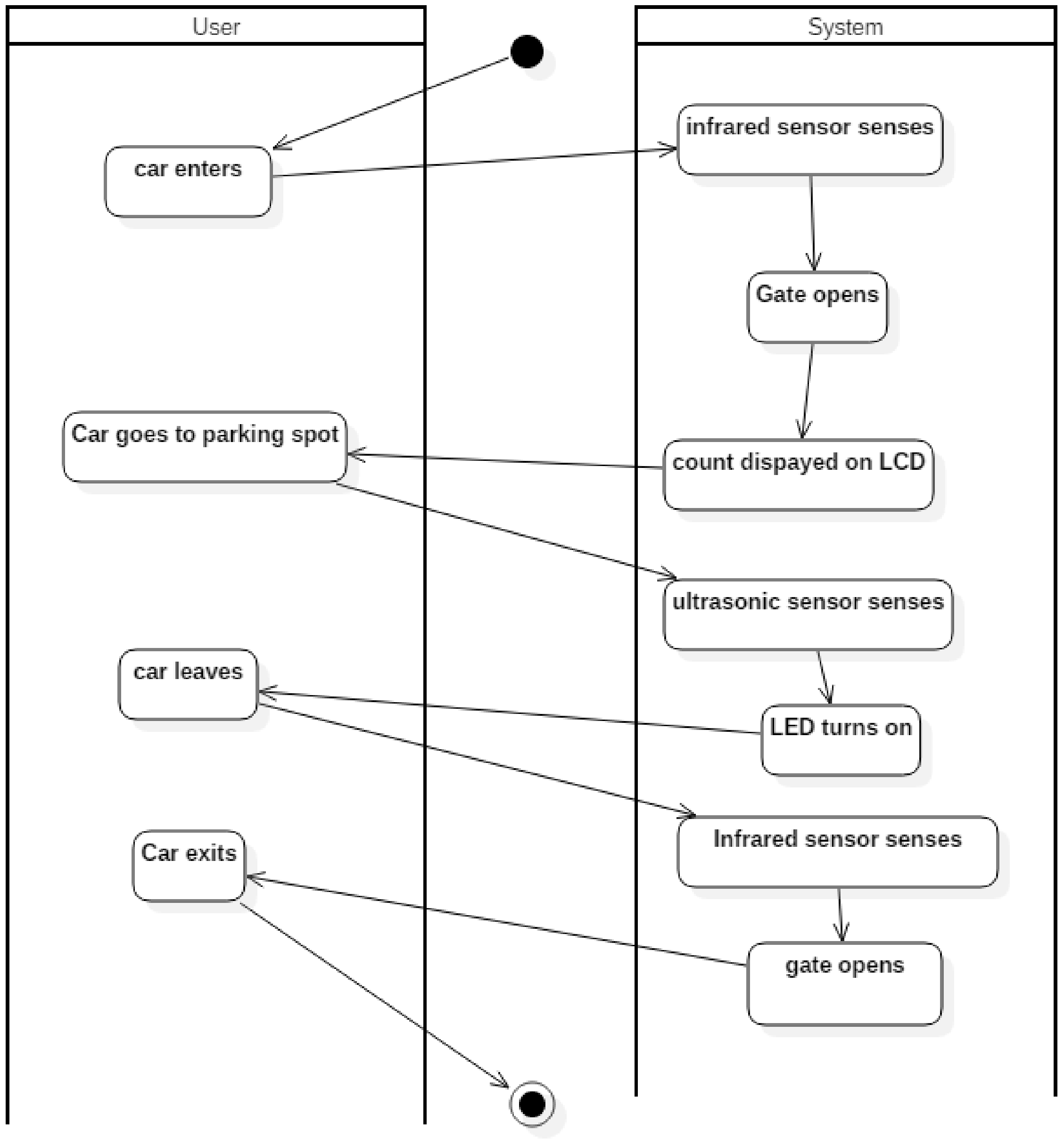


2.3 Use Case

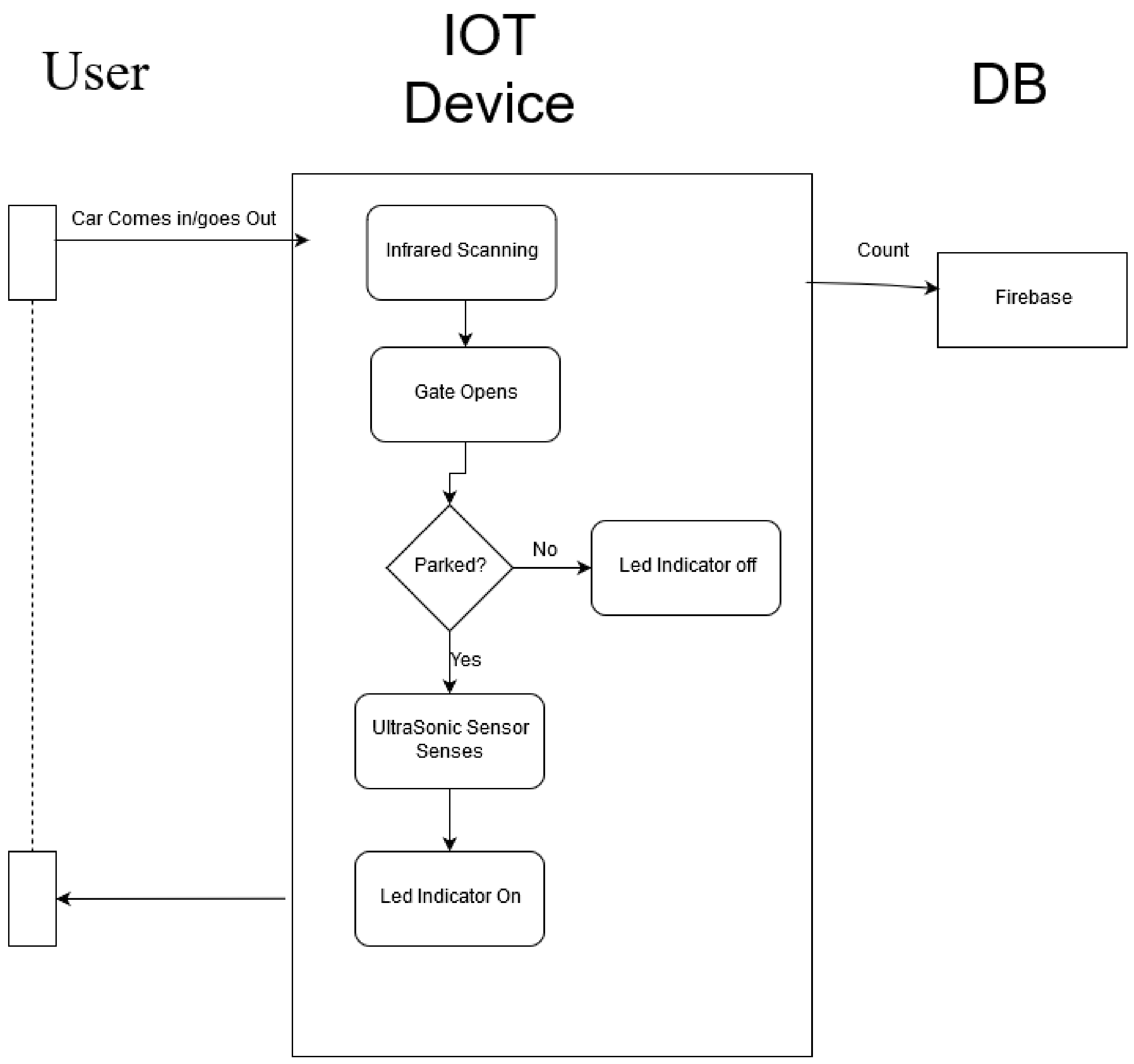


Car Enters
infrared sensor
«include»
Servo motor opens gate
«include»
LCD displays count
Car goes to parking spot
ultrasonic sensor
«include»
LED turns on
User
System

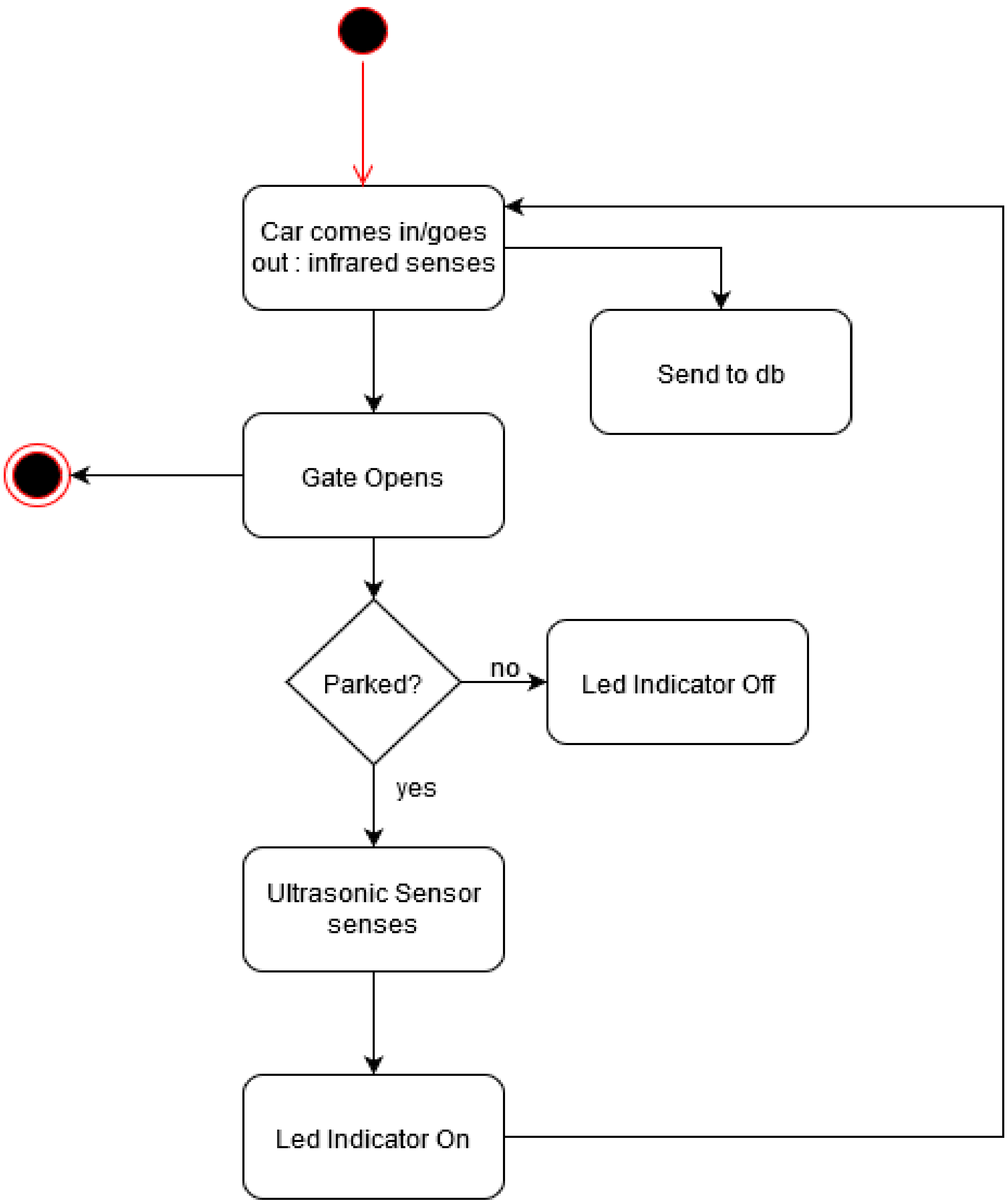
2.4 Activity Diagram



2.5 Sequence Diagram

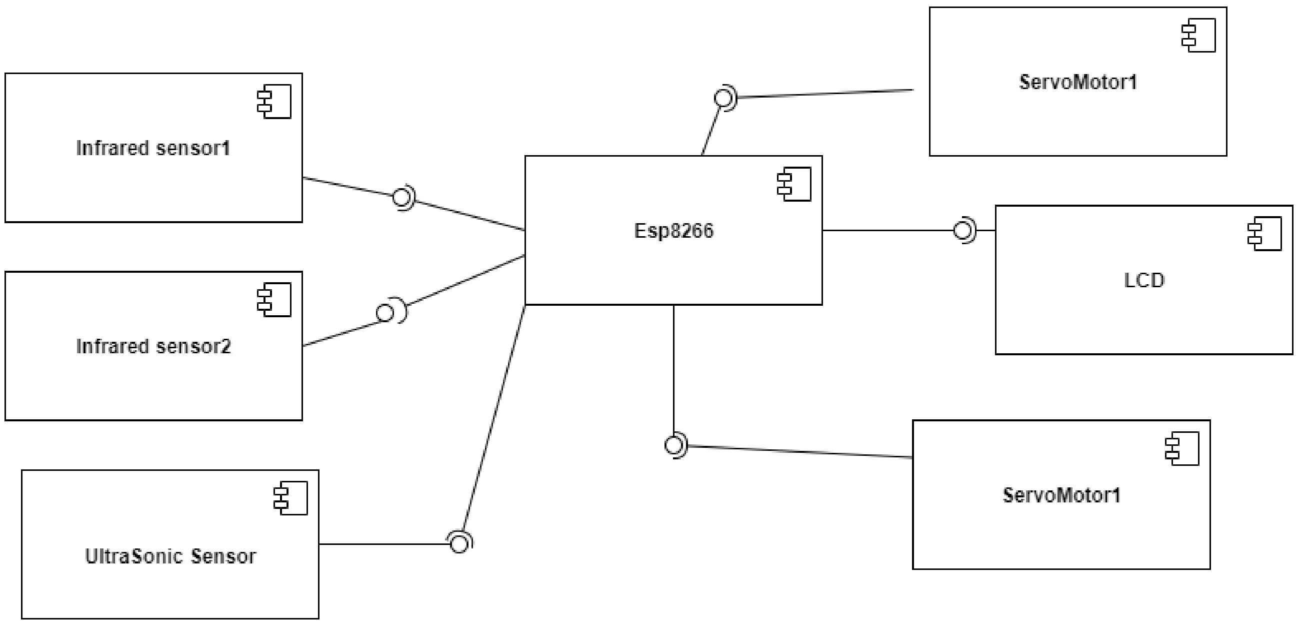


2.6 State Flow

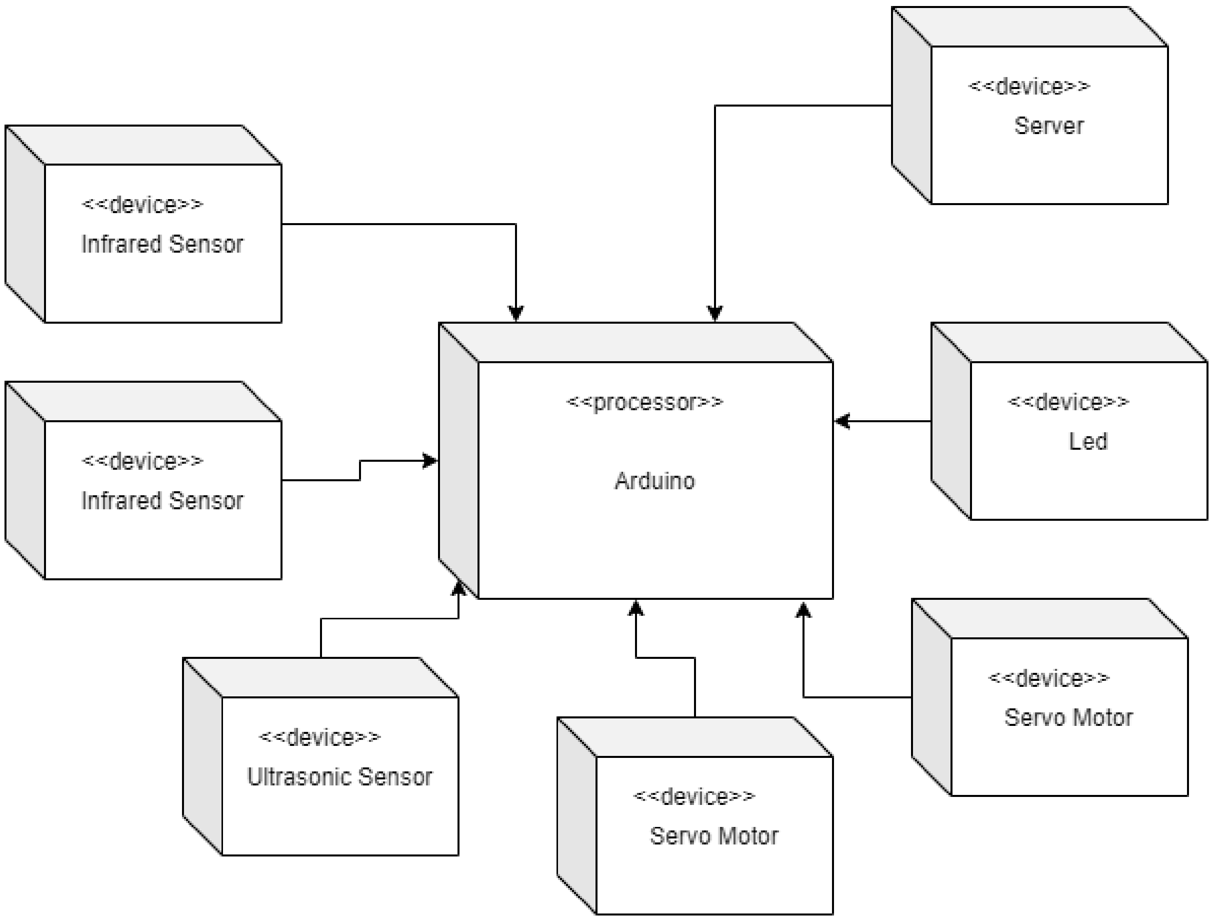


SYSTEM DESIGN

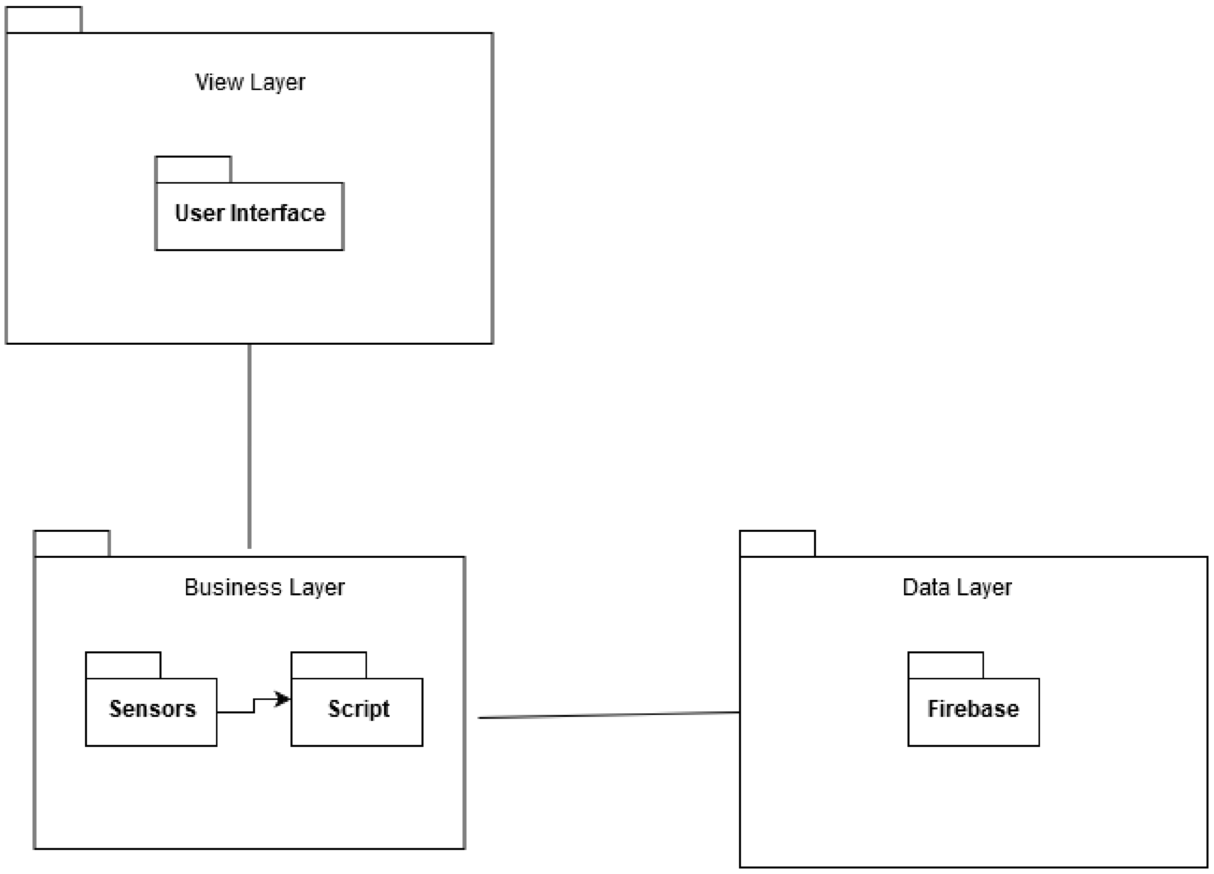
3.1 Component Diagram



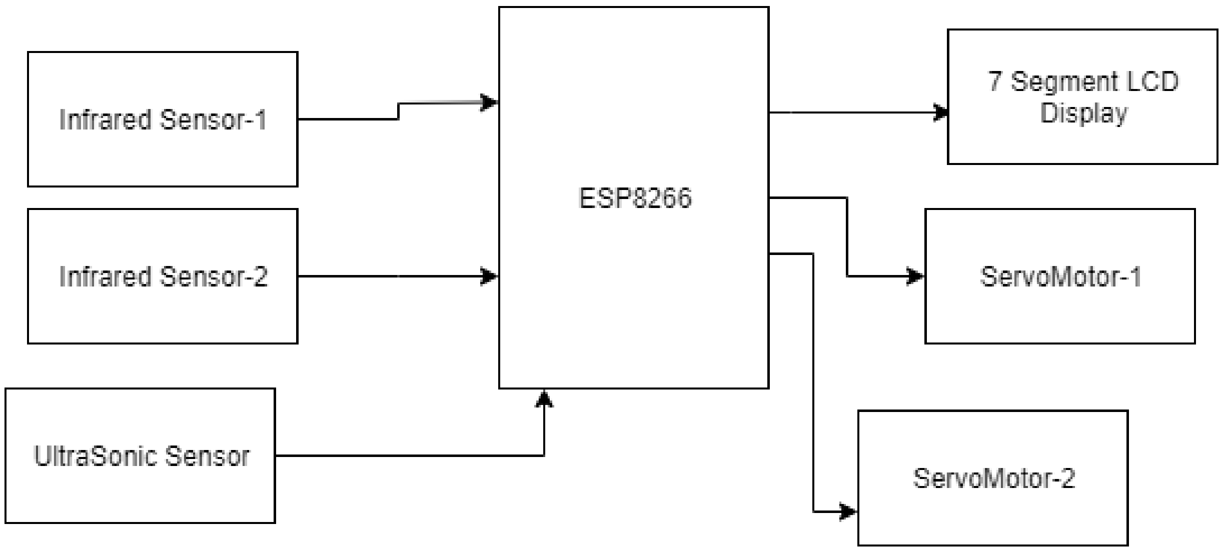
3.2 Deployment Diagram



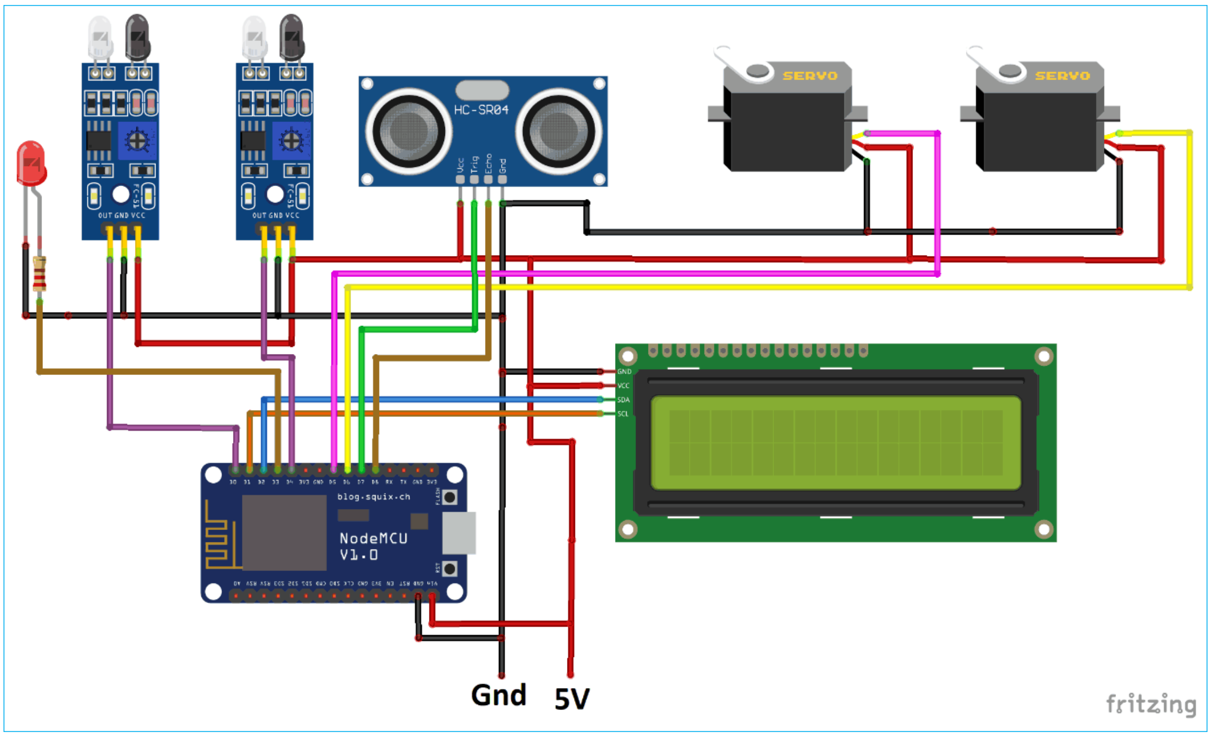
3.3 Package Diagram



3.4 Block Diagram



3.5 Circuit Diagram



SYSTEM CODING

4.1 Code

Park.ino (Arduino Code):

```
 #include <FirebaseArduino.h>

#include <ESP8266WiFi.h>

#include <Servo.h>

#include <Wire.h>

#include <LiquidCrystal_I2C.h>


#define FIREBASE_HOST "park-5f48d.firebaseio.com"              // the project name address
from firebase id

#define FIREBASE_AUTH "p5jbitijOZPYoc21P75IMv02AUmERZMwNvyGEzYf"          // the
secret key generated from firebase


#define WIFI_SSID "iPhone"                                  // input your home or public wifi name

#define WIFI_PASSWORD "COD12345"                            //password for Wifi


String Available = "";                                      //availability string

String fireAvailable = "";


LiquidCrystal_I2C lcd(0x27, 16, 2);        //i2c display address 27 and 16x2 lcd display

Servo myservo;                      //servo as gate

Servo myservos;                        //servo as gate

int Empty;                      //available space integer

int allSpace = 90;
int countYes = 0;

int carEnter = D0;                    // entry sensor

int carExited = D4;                  //exit sensor

int TRIG = D7;                 //ultrasonic trig  pin

int ECHO = D8;               // ultrasonic echo pin

int led = D3;                // spot occupancy signal

int pos;

int pos1;


long duration, distance;

void setup() {

  delay(1000);

  Serial.begin (9600);     // serial debugging

  Wire.begin(D2, D1);       // i2c start

  myservo.attach(D6);      // servo pin to D6

  myservos.attach(D5);       // servo pin to D5

  pinMode(TRIG, OUTPUT);      // trig pin as output

  pinMode(ECHO, INPUT);         // echo pin as input

  pinMode(led, OUTPUT);         // spot indication

  pinMode(carExited, INPUT);    // ir as input

  pinMode(carEnter, INPUT);     // ir as input


  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);                                  //try to connect with wifi
```

```
 Serial.print("Connecting to ");
 Serial.print(WIFI_SSID);                    // display ssid
 while (WiFi.status() != WL_CONNECTED) {
  Serial.print(".");                    // if not connected print this
  delay(500);
 }
 Serial.println();
 Serial.print("Connected to ");
 Serial.println(WIFI_SSID);
 Serial.print("IP Address is : ");
 Serial.println(WiFi.localIP());                              //print local IP address
 Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);       // begin firebase authentication

 lcd.begin();                     //begin lcd
 lcd.home();
 lcd.setCursor(0, 0);                  // 0th row and 0thh column
 lcd.print("Smart Parking");
}

void loop() {

 digitalWrite(TRIG, LOW);        // make trig pin low
 delayMicroseconds(2);
```

```
digitalWrite(TRIG, HIGH);       // make trig pin high

delayMicroseconds(10);

digitalWrite(TRIG, LOW);

duration = pulseIn(ECHO, HIGH);

distance = (duration / 2) / 29.1;     // take distance in cm


  Serial.print("Centimeter: ");

  Serial.println(distance);


Firebase.setString("ParkingStatus/available",fireAvailable);

int carEntry = digitalRead(carEnter);     // read ir input

if (carEntry == LOW) {                  // if high then count and send data


  countYes++;                          //increment count


  if (countYes > 90)

  {

    Serial.print("Parking Full :");

     lcd.setCursor(0, 1);

     lcd.print("Parking Full");


    Serial.println(countYes);

    countYes = 90;
  delay(3000);

  lcd.clear();


  }

  else{

 Serial.print("Car Entered = " );

 Serial.println(countYes );

 lcd.setCursor(0, 1);

 lcd.print("Car Entered");

 for (pos = 140; pos >= 45; pos -= 1) {       // change servo position

   myservos.write(pos);

   delay(5);

 }

 delay(3000);


 for (pos = 45; pos <= 140; pos += 1) {       // change servo position

   // in steps of 1 degree

   myservos.write(pos);

   delay(5);

 }


//  Firebase.pushString("ParkingStatus/available/",fireAvailable);    // send string to firebase

   Firebase.setString("ParkingStatus/available",fireAvailable);
 lcd.clear();

}

}


int carExit = digitalRead(carExited);              //read exit ir sensor

if (carExit == LOW) {                            //if high then count and send


 countYes--;                                    //decrement count


 if (countYes < 0)

 {

  Serial.print("Parking Empty");

  Serial.println(countYes);

   lcd.setCursor(0, 1);

   lcd.print("Parking empty");

  countYes = 0;

  delay(3000);

  lcd.clear();

  }

  else{

 Serial.print("Car Exited = " );

 Serial.println(countYes);

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

```
  lcd.print("Car Exited");

  for (pos1 = 140; pos1 >= 45; pos1 -= 1) {        // change servo position

    myservo.write(pos1);

    delay(5);

  }

  delay(3000);


  for (pos1 = 45; pos1 <= 140; pos1 += 1) {           // change servo position

    // in steps of 1 degree

    myservo.write(pos1);

    delay(5);

  }

  //Firebase.pushString("ParkingStatus/available/",fireAvailable);  // send string to firebase

    Firebase.setString("ParkingStatus/available",fireAvailable);


  lcd.clear();

}

}




if (distance < 6) {                    //if distance is less than 6cm then on led

      Serial.println("Occupied ");

      Firebase.setString("ParkingSpace/p1","1");
    digitalWrite(led, HIGH);
  }

  if (distance > 6) {                      //if distance is greater than 6cm then off led
       Serial.println("Available ");
               Firebase.setString("ParkingSpace/p1","0");

    digitalWrite(led, LOW);
  }

  Empty = allSpace - countYes;         //calculate available data

  Available = String("Available= ") + String(Empty) + String("/") + String(allSpace);       //
convert the int to string
  fireAvailable = String("Available=") + String(Empty) + String("/") + String(allSpace);

  lcd.setCursor(0, 0);
  lcd.print(Available);                //print available data to lcd

}
```

Website Code:

Index.html

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <script type="text/javascript" src="index.js"></script>

  <title>CodePen - Parallel park - pure css</title>

  <link rel="stylesheet" href="./style.css">

</head>

<body>

  <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">

  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">

  <script src="https://code.jquery.com/jquery-3.4.1.js" integrity="sha256-
WpOohJOqMqqyKL9FccASB9O0KwACQJpFTUBLTYOVvVU="
crossorigin="anonymous"></script>

  <!-- Insert these scripts at the bottom of the HTML, but before you use any Firebase
services -->

  <script src="https://www.gstatic.com/firebasejs/7.7.0/firebase-app.js"></script>

  <!-- Firebase App (the core Firebase SDK) is always required and must be listed first -->



  <!-- If you enabled Analytics in your project, add the Firebase SDK for Analytics -->

  <script src="https://www.gstatic.com/firebasejs/7.7.0/firebase-analytics.js"></script>
```

```
<!-- Add Firebase products that you want to use -->
<script src="https://www.gstatic.com/firebasejs/7.7.0/firebase-auth.js"></script>
<script src="https://www.gstatic.com/firebasejs/7.7.0/firebase-firestore.js"></script>
<script src="https://www.gstatic.com/firebasejs/7.7.0/firebase-database.js"></script>
<!-- partial:index.partial.html -->
<header>
  <h1>
    <span class="header-subtitile">Car Parking System</span>
    <p class="header-title">
      Parallel park
    </p>
  </h1>
  <h1>Parking Availability</h1>
  <h1 id="available">
  </h1>
  <p style="color: green">Available</p>
  <p style="color: red">Taken</p>
</header>
<script>
  // Your web app's Firebase configuration
  var firebaseConfig = {
    apiKey: "AIzaSyBKfVvyjus15cJVevxHv9_fW3_BRqsUrb0",
    authDomain: "park-5f48d.firebaseapp.com",
        databaseURL: "https://park-5f48d.firebaseio.com",
        projectId: "park-5f48d",
        storageBucket: "park-5f48d.appspot.com",
        messagingSenderId: "1049006017736",
        appId: "1:1049006017736:web:ba168b9383c070b5e3a9fd",
        measurementId: "G-WSRESFB7WT"
    };
    // Initialize Firebase
    firebase.initializeApp(firebaseConfig);
    firebase.analytics();
</script>
<div id="container">
    <div id="p1">P1</div>
    <div id="p2">P2</div>
    <div id="p3">P3</div>
    <div id="p4">P4</div> <span></span>
</div>
<script src="./script.js"></script>
<script>
    $(function() {
        setInterval(oneSecondFunction, 1000);
    });
    function oneSecondFunction() {
        // stuff you want to do every second
        var ref = firebase.database().ref("/ParkingStatus");
        //var database = firebase.database();
        ref.once('value', gotUserData)
        function gotUserData(snapshot) {
            var parkk = (snapshot.val() && snapshot.val().available) || 'Anonymous';
            console.log(parkk);
            document.getElementById('available').innerHTML = parkk;
        }
    }
</script>
<script>
    $(function() {
        setInterval(oneSecondfunction, 1000);
    });
    function oneSecondfunction() {
        // stuff you want to do every second
        var reff = firebase.database().ref("/ParkingSpace");
        //var database = firebase.database();
        reff.once('value', gotUserDataa)
        function gotUserDataa(snapshot) {
            var p1 = (snapshot.val() && snapshot.val().p1) || 'Anonymous';
            var p2 = (snapshot.val() && snapshot.val().p2) || 'Anonymous';
            console.log(p1);

            console.log(p2);

            if (p1 == 1) {

              document.getElementById("p1").style.backgroundColor = "red";

            } else {

              document.getElementById("p1").style.backgroundColor = "green";

            }

            if (p2 == 1) {

              document.getElementById("p2").style.backgroundColor = "red";

            }

            if (p3 == 1) {

              document.getElementById("p3").style.backgroundColor = "red";

            }

            if (p4 == 1) {

              document.getElementById("p4").style.backgroundColor = "red";

            }

          }

        }

    </script>

</body>

</html>
```

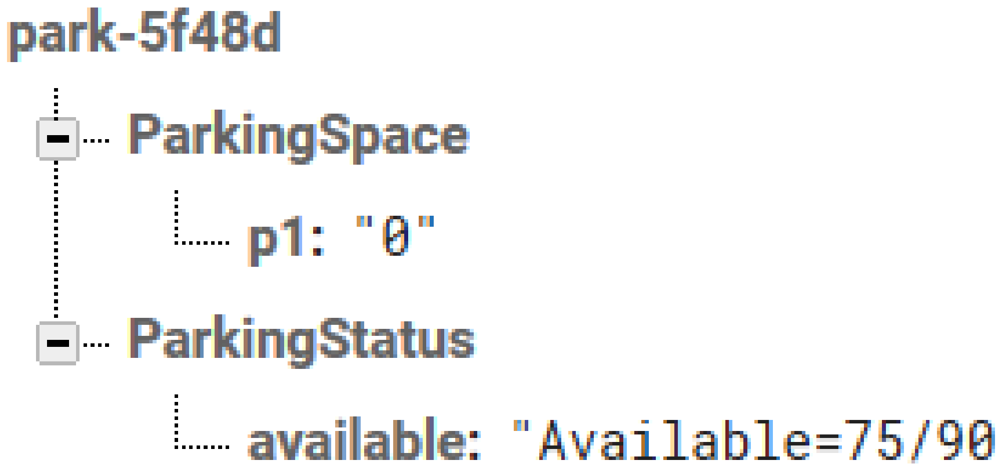
Style.css

```
@import url(https://fonts.googleapis.com/css?family=Montserrat:400,700);

@import url(https://fonts.googleapis.com/css?family=Lato);

input {

   display: none;

}

main {

   position: relative;

}

header h1 {

   font-family: Montserrat, "sans-serif";

   font-size: 45px;

   text-align: center;

   color: #34495e;

}

header h1 .header-subtitile {

   font-size: 0.35em;

   margin-bottom: 5px;

   position: relative;

}


header h1 .header-subtitile:before,

header h1 .header-subtitile:after {
   content: "";

   position: absolute;

   width: 25%;

   height: 2px;

   top: 50%;

   margin-top: -1px;

   background-color: #34495e;

}

header h1 .header-subtitile:before {

   left: -35%;

}

header h1 .header-subtitile:after {

   right: -35%;

}


header h1 .header-title {

   margin-top: 0;

}


html {

   padding: 0 20px;

   background-color: #ecf0f1;

   font-family: "Lato", sans-serif;
```

```
}
main {
    width: 985px;
    display: block;
    background-color: #e8e3e3;
    min-height: 500px;
    margin: 0 auto;
}
.park {
    background-color: #dcd6d6;
    padding: 20px;
    position: relative;
    border-top: 2px solid #fff;
    -moz-box-shadow: 0px -20px 0px #c4baba;
    -webkit-box-shadow: 0px -20px 0px #c4baba;
    box-shadow: 0px -20px 0px #c4baba;
    margin-top: 50px;
}
#container {
    text-align: justify;
    -ms-text-justify: distribute-all-lines;
    text-justify: distribute-all-lines;
    width: 100%;
```

```
}

#container>div {

   width: 100px;

   height: 100px;

   vertical-align: top;

   display: inline-block;

   *display: inline;

   zoom: 1;

   background: green;

}

span {

   width: 100%;

   display: inline-block;

   font-size: 0;

   line-height: 0

}
```

4.2 Data Dictionary



```
park-5f48d
 ⊟ ParkingSpace
     p1: "0"
 ⊟ ParkingStatus
     available: "Available=75/90
```

4.3 Programming Description

Sr. No.	**Name**	**Description**
1	Park.ino	This is the code written for esp8266 to get input from the sensors and give output to servo motor , LCD and send details to firebase.
2	Index.html	This is the website part , here the user will be able to see no. of available parking at any given point of time. It will also show user which parking slots are free. Data is retrived from firebase with the help of JavaScript.
3	Style.css	This contains all the CSS part (Cascading Style sheet) HTML part takes all the styling from here.

4.4 Naming Conventions

Sr No.	Controls	Naming
1	Servo	myservo,myservos
2	Infrared sensor	carEnter,carExit
3	Ultrasonic Sensor	TRIG,ECHO
4	lcd	lcd

PROGRAM LISTING

5.1 Cost Estimation

Cocomo (Constructive Cost Model) is a regression model based on LOC, i.e **number of Lines of Code**. It is a procedural cost estimate model for software projects and often used as a process of reliably predicting the various parameters associated with making a project such as size, effort, cost, time and quality. It was proposed by Barry Boehm in 1970 and is based on the study of 63 projects, which make it one of the best-documented models.

The key parameters which define the quality of any software products, which are also an outcome of the Cocomo are primarily Effort & Schedule:

- **Effort:** Amount of labor that will be required to complete a task. It is measured in person-months units.
- **Schedule:** Simply means the amount of time required for the completion of the job, which is, of course, proportional to the effort put. It is measured in the units of time such as weeks, months.

Different models of Cocomo have been proposed to predict the cost estimation at different levels, based on the amount of accuracy and correctness required. All of these models can be applied to a variety of projects, whose characteristics determine the value of constant to be used in subsequent calculations. These characteristics pertaining to different system types are mentioned below.

Boehm’s definition of organic, semidetached, and embedded systems:

1. **Organic** – A software project is said to be an organic type if the team size required is adequately small, the problem is well understood and has been solved in the past and also the team members have a nominal experience regarding the problem.
2. **Semi-detached** – A software project is said to be a Semi-detached type if the vital characteristics such as team-size, experience, knowledge of the various programming environment lie in between that of organic and Embedded. The projects classified as Semi-Detached are comparatively less familiar and difficult to develop compared to the organic ones and require more experience and better guidance and creativity. Eg: Compilers or different Embedded Systems can be considered of Semi-Detached type.

3. **Embedded –** A software project with requiring the highest level of complexity, creativity, and experience requirement fall under this category. Such software requires a larger team size than the other two models and also the developers need to be sufficiently experienced and creative to develop such complex models.

All the above system types utilize different values of the constants used in Effort Calculations.

Types of Models: COCOMO consists of a hierarchy of three increasingly detailed and accurate forms. Any of the three forms can be adopted according to our requirements. These are types of COCOMO model:

1. Basic COCOMO Model
2. Intermediate COCOMO Model
3. Detailed COCOMO Model

The first level, **Basic COCOMO** can be used for quick and slightly rough calculations of Software Costs. Its accuracy is somewhat restricted due to the absence of sufficient factor considerations.

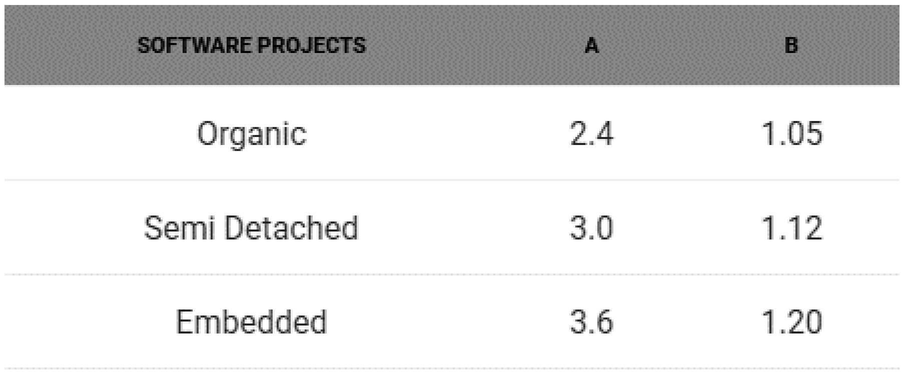
Intermediate COCOMO takes these Cost Drivers into account and **Detailed COCOMO** additionally accounts for the influence of individual project phases, i.e in case of Detailed it accounts for both these cost drivers and also calculations are performed phase wise henceforth producing a more accurate result. These two models are further discussed below.

Estimation of Effort: Calculations –

1) **Basic Model –**

$$\mathrm{E} = \mathrm{a}(\mathrm{KLOC})^{b}$$

The above formula is used for the cost estimation of for the basic COCOMO model, and also is used in the subsequent models. The constant values a and b for the Basic Model for the different categories of system:



SOFTWARE PROJECTS	A	B
Organic	2.4	1.05
Semi Detached	3.0	1.12
Embedded	3.6	1.20

· The effort is measured in Person-Months and as evident from the formula is dependent on Kilo-Lines of code. These formulas are used as such in the Basic Model calculations, as not much consideration of different factors such as reliability, expertise is taken into account, henceforth the estimate is rough.

· · **Intermediate Model –**

The basic Cocomo model assumes that the effort is only a function of the number of lines of code and some constants evaluated according to the different software system. However, in reality, no system's effort and schedule can be solely calculated on the basis of Lines of Code. For that, various other factors such as reliability, experience, Capability. These factors are known as Cost Drivers and the Intermediate Model utilizes 15 such drivers for cost estimation. Classification of Cost Drivers and their attributes:

(i) Product attributes –

- Required software reliability extent
- Size of the application database
- The complexity of the product

(ii) Hardware attributes –

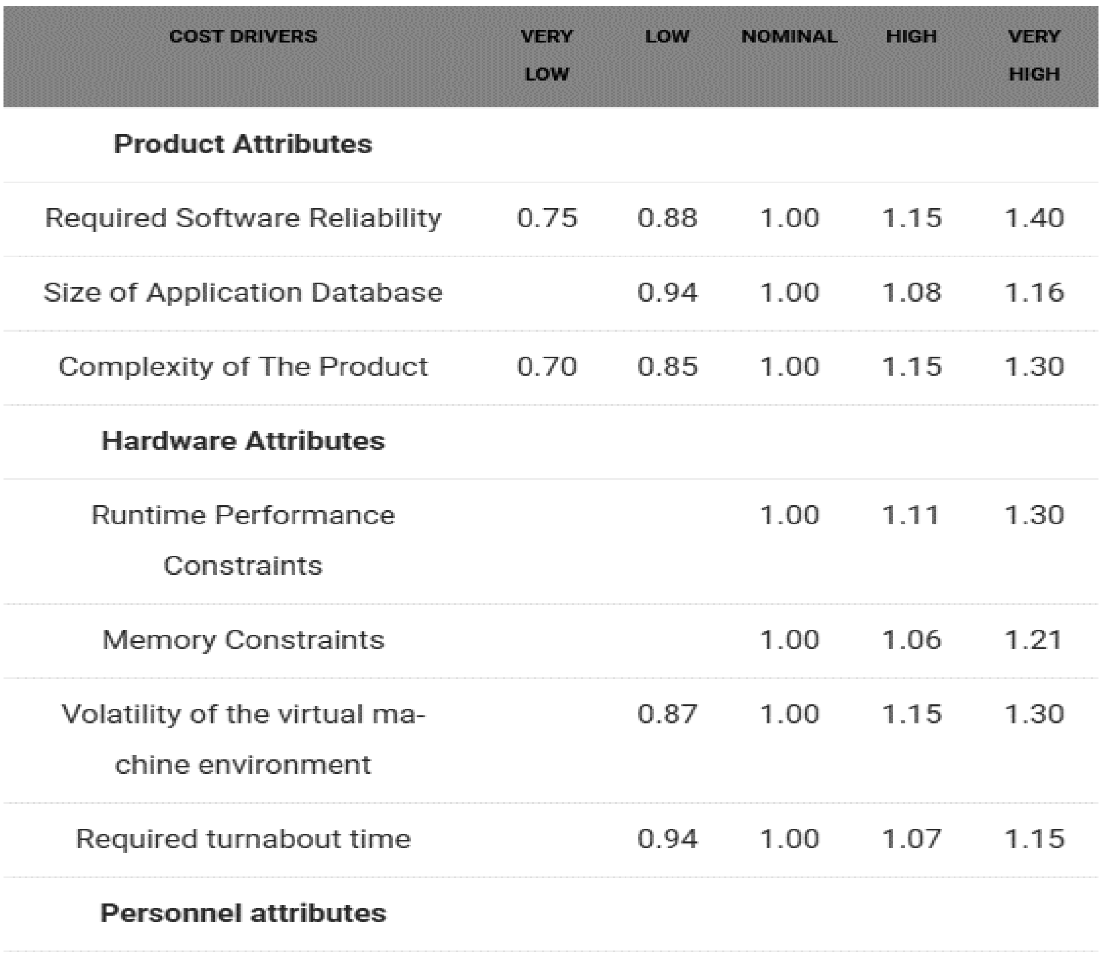
- Run-time performance constraints
- Memory constraints
- The volatility of the virtual machine environment
- Required turnabout time

(iii) Personnel attributes –

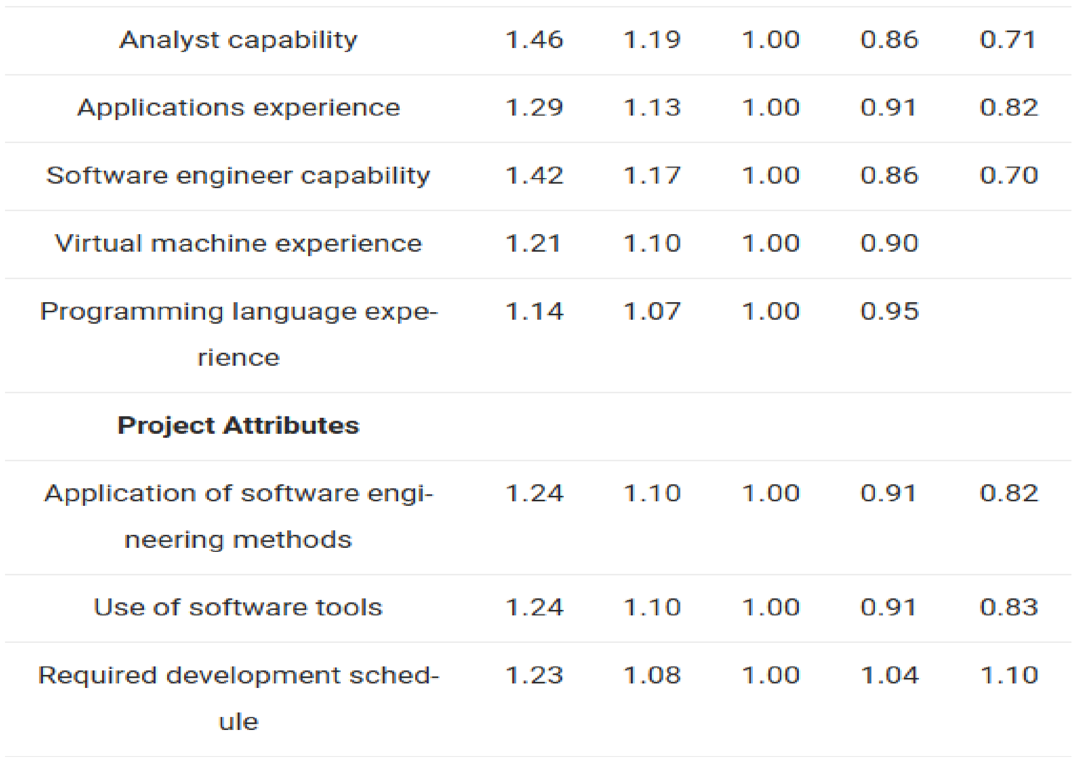
- Analyst capability
- Software engineering capability
- Applications experience
- Virtual machine experience
- Programming language experience

(iv) Project attributes –

- Use of software tools
- Application of software engineering methods
- Required development schedule



COST DRIVERS	VERY LOW	LOW	NOMINAL	HIGH	VERY HIGH
Product Attributes					
Required Software Reliability	0.75	0.88	1.00	1.15	1.40
Size of Application Database		0.94	1.00	1.08	1.16
Complexity of The Product	0.70	0.85	1.00	1.15	1.30
Hardware Attributes					
Runtime Performance Constraints			1.00	1.11	1.30
Memory Constraints			1.00	1.06	1.21
Volatility of the virtual machine environment		0.87	1.00	1.15	1.30
Required turnabout time		0.94	1.00	1.07	1.15
Personnel attributes					

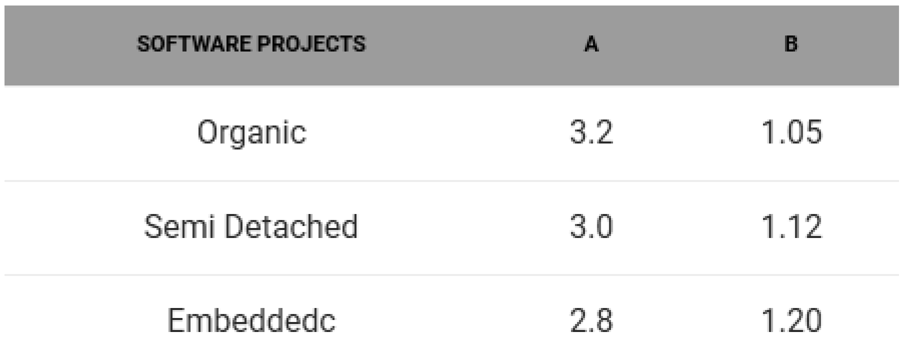


Analyst capability	1.46	1.19	1.00	0.86	0.71
Applications experience	1.29	1.13	1.00	0.91	0.82
Software engineer capability	1.42	1.17	1.00	0.86	0.70
Virtual machine experience	1.21	1.10	1.00	0.90	
Programming language experience	1.14	1.07	1.00	0.95	
Project Attributes					
Application of software engineering methods	1.24	1.10	1.00	0.91	0.82
Use of software tools	1.24	1.10	1.00	0.91	0.83
Required development schedule	1.23	1.08	1.00	1.04	1.10

The project manager is to rate these 15 different parameters for a particular project on a scale of one to three. Then, depending on these ratings, appropriate cost driver values are taken from the above table. These 15 values are then multiplied to calculate the EAF (Effort Adjustment Factor). The Intermediate COCOMO formula now takes the form:

$$\text{E}= (\text{a}(\text{KLOC})^{b}) * EAF$$

The values of a and b in case of the intermediate model are as follows:



SOFTWARE PROJECTS	A	B
Organic	3.2	1.05
Semi Detached	3.0	1.12
Embeddedc	2.8	1.20

- **Detailed Model –**

Detailed COCOMO incorporates all characteristics of the intermediate version with an assessment of the cost driver’s impact on each step of the software engineering process. The detailed model uses different effort multipliers for each cost driver attribute. In detailed cocomo, the whole software is divided into different modules and then we apply COCOMO in different modules to estimate effort and then sum the effort.

The Six phases of detailed COCOMO are:

1. Planning and requirements
2. System design
3. Detailed design
4. Module code and test
5. Integration and test
6. Cost Constructive model

The effort is calculated as a function of program size and a set of cost drivers are given according to each phase of the software lifecycle.

Calculations:

	a	**b**	**c**	**d**
Organic	2.4	1.05	2.5	0.38
Semi-Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.35

Formulae:

Effort = $a(KLOC)^b$ person-month

Development = $c(KLOC)^d$ months

Average Staff Cycle = Effort/Development persons

Productivity = (KLOC/Effort) * 1000 no. of lines of code

No. of lines of code = 383 = 0.383 KLOC

- Organic

  Effort = 2.4 * $(0.383)^{1.05}$ = 0.8761 person-month

  Development = 2.5 * $(0.383)^{0.38}$ = 1.7360 months

  Average Staff Cycle = 0.8761 / 1.7360 = 0.5047 persons

  Productivity = (0.383 /0.8761) * 1000 = 437.1647 no. of lines of code
- Semi- Detached

  Effort = 3.0 * $(0.383)^{1.12}$ = 1.024 person-month

  Development = 2.5 * $(0.383)^{0.35}$ = 1.7867 months

  Average Staff Cycle = 1.024 / 1.7867 =0.5731 persons

  Productivity = (0.383 /1.024) * 1000 = 374.023 no. of lines of code
- Embedded

  Effort = 3.6 * $(0.383)^{1.20}$ = 1.1379 person-month

  Development = 2.5 * $(0.383)^{0.35}$ = 1.7867 months

  Average Staff Cycle = 1.1379 / 1.7867 = 0.6369 persons

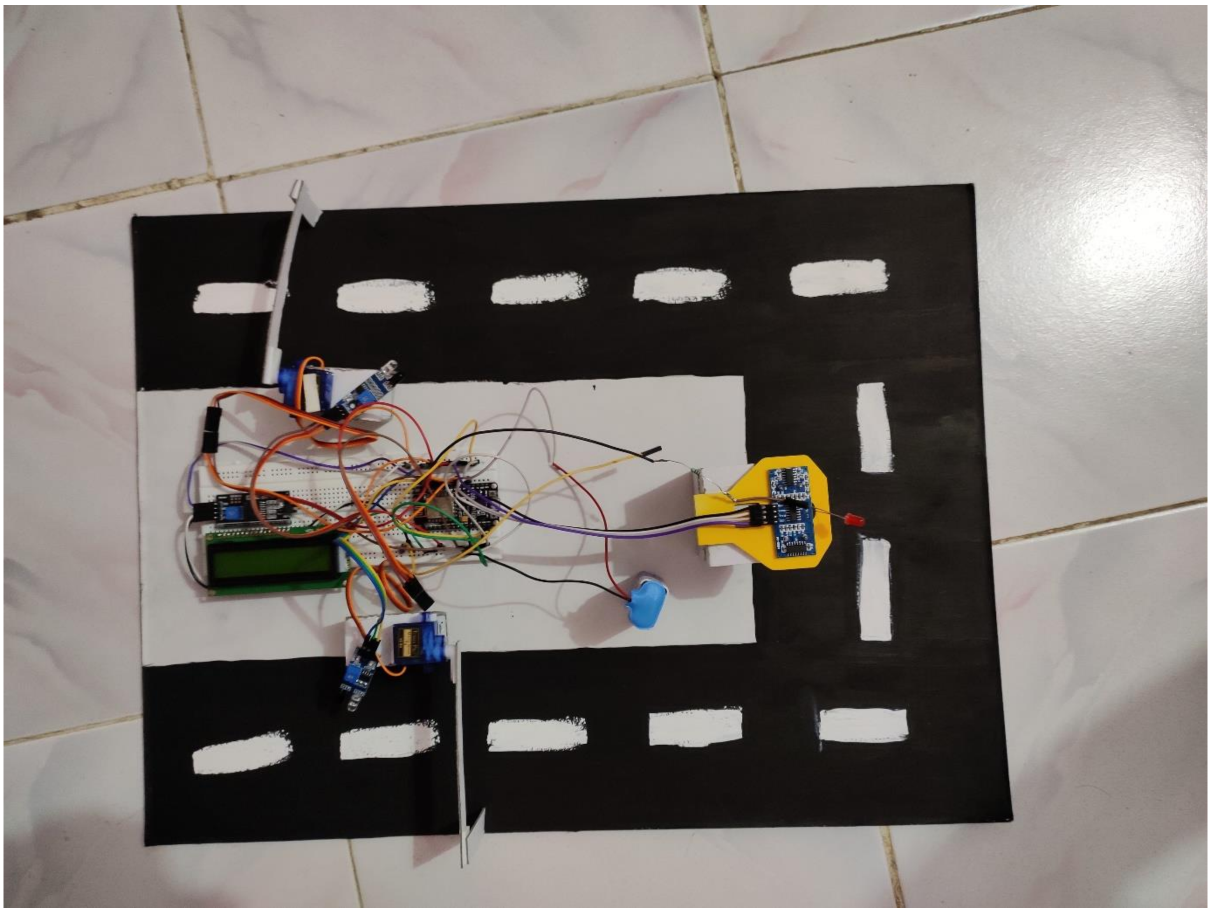
  Productivity = (0.383 /1.1379) * 1000 = 336.5845 no. of lines of code

	Organic	**Semi-Detached**	**Embedded**
Effort	0.8761 person-month	1.024 person-month	1.1379 person-month
Development	1.7360 months	1.7867 months	1.7867 months
Average Staff Cycle	0.5047 persons	0.5371 persons	0.6369 persons
Productivity	437.1647 no. of lines of code	374.023 no. of lines of code	336.5845 no. of lines of code

5.2 Test Cases

Test Conditions	Input Specified	Expected Result	Actual Result
Car parking full	Car enters when car parking is full	Tell user that parking is full	Prints on LCD that parking is full
Car entry	Car tries to enter normally	No. of parking available decrements .properly	Available parking decrements by 1 when 1 car enters.
Car Exit	Car tries to exit normally	No. of parking available increments properly.	Available parking increments by 1 when 1 car enters.
parking slot	Car goes in parking slot	Red led glows once car is parked properly	Red led glows ,once parked also updates on website

5.3 User Manual with Screen Shots



6 .Future Enhancement

- RFID based parking fees duduction.
- Option to book parking slot.
- Increase capacity of parking slots

7. Bibliography

- https://www.stackoverflow.com
- https://firebase.google.com
- https://medium.com/topic/programming