TABLE OF CONTENT

SMART PARKING

ABSTRACT

Welcome to the revolution of parking efficiency! In this presentation, we will explore how the Internet ofThings (IoT) is transforming parking solutions. Discover the power of IoT in optimizing parking spaces, reducing congestion, and enhancing user experience. Get ready to delve into the world of smart parking

Parking congestion, inefficient utilization of spaces, and frustrated drivers are common problems in urban areas. Traditional parking systems lack real-time information and smart management. IoToffers a solution by enabling real-time monitoring, intelligent parking guidance, and seamless payment mechanisms. Let's explore how IoT can revolutionize parking efficiency.

By leveraging IoTtechnology, parking becomes smarter and more efficient. Connected sensors and devices provide real-time data on available parking spaces, allowing drivers to find vacant spots effortlessly. Smart parking solutions also enable automated payment systems, reducing the hassle of manual transactions. Let's dive deeper into the components of IoT-enabled smart parking.

Sensors play a crucial role in smart parking solutions. They detect the presence or absence of vehicles in parking spaces, transmitting data to a central system. Realtimedata on parking occupancy enables efficient management, accurate guidance, and predictive analytics. With IoT, parking operators can make data-driven decisions to optimize parking resources and enhance overall efficiency.

Finding an available parking spot can be frustrating. Intelligentparkingguidance systems use IoT data to direct drivers to vacant spaces quickly. Mobile apps, digital signage, and in-car navigation systems provide real-time guidance, reducing search time and traffic congestion. With IoT-enabled guidance, drivers can have a stress-free parking experience

Traditional parking payment methods often involve manual transactions and long queues. IoT-enabledsmart parking solutions offer seamless payment mechanisms. Integrated mobile payment apps and automatic billing systems make the payment process convenient and efficient. By eliminating the need for physical tickets or cash, IoT simplifies the parking payment experience for both drivers and parking operators.

The adoption of IoT for smart parking brings numerous benefits. It optimizes parking space utilization, reduces traffic congestion, and enhances user experience. Drivers save time and fuel

by quickly finding parking spots, while parking operators gain insights for efficient resource allocation. With IoT, revolutionize parking efficiency and create a seamless parking ecosystem

CHAPTER 1

INTRODUCTION

Welcome to the revolution of parking efficiency! In this presentation, we will explore how the Internet ofThings (IoT) is transforming parking solutions. Discover the power of IoT in optimizing parking spaces, reducing congestion, and enhancing user experience. Get ready to delve into the world of smart parking!

CURRENT PARKING CHALLENGES:

Parking congestion, inefficient utilization of spaces, and frustrated drivers are common problems in urban areas. Traditional parking systems lack real-time information and smart management. IoToffers a solution by enabling real-time monitoring, intelligent parking guidance, and seamless payment mechanisms. Let's explore how IoT can revolutionize parking efficiency.

IOT AND SMART PARKING:

By leveraging IoTtechnology, parking becomes smarter and more efficient. Connected sensors and devices provide real-time data on available parking spaces, allowing drivers to find vacant spots effortlessly. Smart parking solutions also enable automated payment systems, reducing the hassle of manual transactions. Let's dive deeper into the components of IoT-enabled smart parking.

SENSORS AND DATA COLLECTION:

Sensors play a crucial role in smart parking solutions. They detect the presence or absence of vehicles in parking spaces, transmitting data to a central system. Realtimedata on parking occupancy enables efficient management, accurate guidance, and predictive analytics. With IoT, parking operators can make data-driven decisions to optimize parking resources and enhance overall efficiency.

INTELLIGENT PARKING GUIDANCE:

Finding an available parking spot can be frustrating. Intelligentparkingguidance systems use IoT data to direct drivers to vacant spaces quickly. Mobile apps, digital signage, and in-car navigation systems provide real-time guidance, reducing search time and traffic congestion. With IoT-enabled guidance, drivers can have a stress-free parking experience

SEAMLESS PAYMENT MECHANISMS:

Traditional parking payment methods often involve manual transactions and long queues. IoT-enabledsmart parking solutions offer seamless payment mechanisms. Integrated mobile payment

apps and automatic billing systems make the payment process convenient and efficient. By eliminating the need for physical tickets or cash, IoT simplifies the parking payment experience for both drivers and parking operators.

BENEITS OF IOT IN PARKING:

The adoption of IoT for smart parking brings numerous benefits. It optimizes parking space utilization, reduces traffic congestion, and enhances user experience. Drivers save time and fuel by quickly finding parking spots, while parking operators gain insights for efficient resource allocation. With IoT, revolutionize parking efficiency and create a seamless parking ecosystem

LIST OF FIGURE:

CHAPTER 2

SYSTEM ANALYSIS:

Planning and Assessment:

Sensors:

Smart parking systems use sensors (e.g., ultrasonic, infrared, or camera-based) to detect the presence or absence of vehicles in parking spaces. This data can be used to provide real-time information on parking availability.

Mobile Apps:

Users can access information about available parking spaces through mobile apps. They can reserve parking spots, pay for parking, and get navigation assistance to the parking area.

Payment Solutions:

These systems often offer various payment options, including mobile payments, credit cards, or contactless payments, reducing the need for cash transactions.

Real-time Data:

Users can check the real-time availability of parking spaces, reducing the time spent searching for a parking spot.

Navigation Assistance:

Smart parking systems can provide turn-by-turn navigation to the nearest available parking spot, saving time and reducing frustration.

Reduced Traffic Congestion:

By helping drivers find parking quickly, smart parking systems can reduce traffic congestion and emissions.

Data Analytics:

The data collected by these systems can be used to analyze parking patterns and optimize parking space usage.

Integration with IoT:

Smart parking systems often use the Internet of Things (IoT) to connect and communicate between various devices and systems, enhancing their functionality.

Environmental Benefits:

Efficient parking can lead to a reduction in fuel consumption and emissions since drivers spend less time circling for parking.

2.1.1 DISADVANTAGE:

Reserved Parking Spaces:

Smart parking systems often include reserved parking spaces for people with disabilities, ensuring they have easier access to facilities.

Real-time Availability:

Users can check the availability of accessible parking spaces in real-time through mobile apps or websites, saving time and effort.

Parking Guidance:

Some systems offer guidance to the nearest available accessible parking space, reducing the need to search for one.

Payment Options:

Integrated payment solutions can make it easier for individuals with disabilities to pay for parking without the need for physical tickets or coins.

Notifications:

Users can receive notifications when their parking time is about to expire, making it easier to manage their parking needs.

Accessibility Features:

Smart parking systems can have accessible interfaces, such as voice commands or larger text options, to assist those with visual or hearing impairments.

Security and Surveillance:

Some systems incorporate security features like CCTV cameras to enhance safety and security in parking areas.

2.2 PROPOSED SYSTEM:

Real-time Parking Availability:

Sensors and cameras could monitor parking spaces and provide real-time data on available spots.

Mobile App Integration:

A dedicated app for users to check parking availability, reserve spots, and make payments.

Automated Payment:

Users can pay for parking through the app, reducing the need for physical tickets or cash.

Navigation Assistance:

The app could provide directions to available parking spots.

IoT Sensors:

Sensors can detect when a car occupies or vacates a parking space, helping to manage occupancy.

Sustainability:

Implementing eco-friendly features like electric vehicle charging stations.

Security:

Surveillance cameras for safety and security.

Data Analytics:

Collect and analyze data to optimize parking operations and enhance the user experience.

2.2.2 ADVANTAGES:

Cost:

Developing and installing a smart parking system can be expensive. This includes the cost of sensors, infrastructure, software, and ongoing maintenance.

Complexity:

Smart parking systems can be complex to set up and maintain, requiring specialized knowledge and expertise. They may also be prone to technical issues.

Privacy Concerns:

Collecting data on vehicles and drivers raises privacy concerns. Users may be uncomfortable with the amount of data collected and how it's used.

Integration Challenges:

Integrating the smart parking system with existing infrastructure and technologies can be challenging, especially in older parking facilities.

Reliability:

These systems rely on sensors and technology, which can fail or malfunction. This can lead to incorrect information about parking availability.

Accessibility:

Smart parking systems may not be user-friendly for all demographics, especially those who aren't tech-savvy or don't have access to smartphones.

Dependency on Technology:

If the system experiences a failure or outage, it can disrupt parking operations and inconvenience users.

Environmental Impact:

The production and maintenance of technology used in smart parking systems can have environmental implications.

Job Displacement:

Automation of parking processes could potentially displace jobs for parking attendants and related roles.

CHAPTER 3

SYSTEM SPECIFICATION:

3.1 SOFTWARE SPECIFICATION:

- LINUX
- Programming Language python

3.2 HARDWARE SPECIFICATION:

- **Sensors:**

  Smart parking systems typically use various sensors to detect the presence of vehicles in parking spaces. These can include ultrasonic, infrared, magnetic, or video-based sensors. Sensor specifications may include detection range, accuracy, and power consumption.

- **Cameras:**

For video-based parking monitoring and security, cameras are essential. Specifications for cameras may include resolution, field of view, frame rate, and low-light performance.

- **Communication Devices:**

These devices enable sensors and cameras to transmit data to a central control system or cloud. Common communication protocols include Wi-Fi, Bluetooth, LoRa, cellular, and Ethernet. The choice of communication method depends on the range and coverage requirements.

- **Processing Unit:**

A central processing unit (CPU) or microcontroller is needed to collect and process data from sensors and cameras. The specifications may include processor type, speed, and memory capacity.

- **Power Supply:**

Smart parking systems require a reliable power source, which may include both AC and DC power options. Backup power sources like batteries or uninterruptible power supplies (UPS) may also be necessary.

- **Display and User Interface:**

If the system includes user-facing displays or interfaces for drivers or parking attendants, these may require screens, input devices, and interface hardware.

- **Control Software:**

The software running on the hardware should be tailored to the specific needs of the smart parking system. It manages data from sensors, processes it, and provides a user interface.

- **Data Storage:**

If historical data needs to be stored, a storage system with specific capacity and speed requirements may be necessary.

- **Security Components:**

Hardware for data encryption, access control, and physical security (e.g., tamper-resistant enclosures) is essential to protect the system.

- **Environmental Considerations:**

Depending on the location and climate, the hardware may need to be designed to withstand extreme temperatures, humidity, or other environmental factors.

- **Scalability:**

Hardware should be chosen with scalability in mind, allowing for the addition of more sensors or cameras as the parking system expands.

- **Maintenance and Serviceability:**

Consider hardware that is easy to maintain, upgrade, and replace if necessary.

- **Cost Considerations:**

The choice of hardware should align with the budget and cost constraints of the project.

3.4 SOFTWARE DESCRIPTION:

PYTHON:

Python is a general-purpose language which means it is versatile and can be used to program many different types of functions. Because it is an interpreted language, it precludes the need for compiling code before execution and because it is a high-level programming language, Python is able to abstract details from code. In fact, Python focuses so much attention on abstraction that its code can be understood by most novice programmers.

Python code tends to be short and when compared to compiled languages like C and C++, it executes programs slower. Its user-friendliness makes it a popular language for citizen developers working with machine learning algorithms in low-code no-code (LCNC) software applications.

Python has a simply syntax and is known for having a large community that actively contributes to a growing selection of software modules and libraries. Python's initial development was spearheaded by Guido van Rossum in the late 1980s. Today, Python is managed by the Python Software Foundation.

Techopedia Explains Python

Python offers several frameworks for web development. A Python Web framework is a group of modules and libraries that enable programmers to re-use another developer's code. This collaborative approach can developers avoid dealing with low-level issues such as protocols, sockets and process/thread management.

Python Frameworks

Here are 10 frameworks that web developers, machine learning teams and data analytics teams should consider when using Python:

Open-source Django is a popular Python web framework that facilitates quick web design and development. Django is a free-to-use framework that enables developers to reuse code to build high-quality web apps and APIs. Django is known for:

- Helping programmers avoid security blunders.

- Supporting a data-driven architecture.
- Moving software from concept to launch quickly.

Pyramid is a compact open-source web framework that works in all supported versions of Python. It offers the essential elements required for online applications including delivering static content and converting URLs to code. Some of Pyramid's attributes include:

- Security APIs that support authentication and authorization.
- A cookiecutter that generates sample Pyramid projects from project templates.
- Supporting the SQLAlchemy project and using its object-relational mapper (ORM) to interface with databases.

Bottle is a Web Server Gateway Interface (WSGI) micro-web framework for Python that is known for being lightweight and easy to use. Bottle is distributed as a single file module and the default Python library is the only dependency of the framework. It is is a popular framework for building mobile applications and supports:

- Python versions 2.7 and above.
- Mako, Jinja2, and Cheetah templates.
- WSGI-capable HTTP servers, including Bjoern, Google App Engine, fapws3 and CherryPy.
- URL mapping using condensed syntax.

CherryPy is an object-oriented HTTP framework that supports Apache and Microsoft IIS. Some of CherryPy's attributes include:

- A robust configuration system suitable for both developers and deployers.
- Built-in support for testing, coverage and profiling.
- Tools for authentication and caching.
- Flexible plugins.
- Robust configuration management.

Flask offers more control than its closest competitor, Django, and features support for unit testing. Along with RESTful request-dispatching and WSGI compatibility, Flask is known for:

- Providing an integrated development server with a debugger.
- Jinja2 templating (tags, filters, macros, and more).
- 100% compliance with WSGI 1.0.

Web2py allows developers to create, distribute, debug, test, manage a database and maintain applications. It has no setup files and can operate from a USB disk. Web2py can:

- Serve as a manual for web developers using the Model View Controller (MVC) paradigm.
- Automatically fix problems that may result in security risks.
- Support a database abstraction layer (DAL) that dynamically writes SQL is part of the framework.

Tornado is an open-source asynchronous framework for I/O operations. Tornado is known for supporting applications that require long-lived connections, real-time location services and allowing the integration of authentication and authorization methods from third parties.

BlueBream is a web application framework, server and library for Python programmers that was initially known as Zope 3. BlueBream is known for being durable, reliable and adaptive. It supports reusable software components as well as:

- WSGI (Web Server Gateway Interface) compatibility for Python.
- A template-development language that complies with XHTML.
- A program for creating forms automatically.

Grok

Grok is a robust framework for creating dependable and adaptable web applications. It supports DRY (Don't Repeat Yourself) software development and has a quick learning curve. Like other full-stack Python web frameworks, Grok features an intuitive UI (user interface).

Quixote

Quixote allows Python programmers to quickly create Web-based apps. This framework's objective is to offer web developers exceptional performance and flexibility for producing HTML with Python code

JSON:

JSON, or JavaScript Object Notation, is a format used to represent data. It was introduced in the early 2000s as part of JavaScript and gradually expanded to become the most common medium for describing and exchanging text-based data. Today, JSON is the universal standard of data exchange. It is found in every area of programming, including front-end and server-side development, systems, middleware, and databases.

This article introduces you to JSON. You'll get an overview of the technology, find out how it compares to similar standards like XML, YAML, and CSV, and see examples of JSON in a variety of programs and use cases.

TABLE OF CONTENTS

SHOW MORE

A little bit of history

JSON was initially developed as a format for communicating between JavaScript clients and back-end servers. It quickly gained popularity as a human-readable format that front-end programmers could use to communicate with the back end using a terse, standardized format. Developers also discovered that JSON was very flexible: you could add, remove, and update fields ad hoc. (That flexibility came at the cost of safety, which was later addressed with the JSON schema.)

[Why Wasm is the future of cloud computing | The rise of WebAssembly]

In a curious turn, JSON was popularized by the AJAX revolution. Strange, given the emphasis on XML, but it was JSON that made AJAX really shine. Using REST as the convention for APIs and JSON as the medium for exchange proved a potent combination for balancing simplicity, flexibility, and consistence.

Next, JSON spread from front-end JavaScript to client-server communication, and from there to system config files, back-end languages, and all the way to databases. JSON even helped spur the NoSQL movement that revolutionized data storage. It turned out that database administrators also enjoyed JSON's flexibility and ease of programming.

Today, document-oriented data stores like MongoDB provide an API that works with JSON-like data structures. In an interview in early 2022, MongoDB CTO Mark Porter noted that, from his perspective, JSON is still pushing the frontier of data. Not bad for a data format that started with a humble curly brace and a colon.

Why developers use JSON

No matter what type of program or use case they're working on, software developers need a way to describe and exchange data. This need is found in databases, business logic, user interfaces, and in all systems communication. There are many approaches to structuring data for exchange. The two broad camps are binary and text-based data. JSON is a text-based format, so it is readable by both people and machines.

JSON is a wildly successful way of formatting data for several reasons. First, it's native to JavaScript, and it's used inside of JavaScript programs as JSON literals. You can also use JSON with other programming languages, so it's useful for data exchange between heterogeneous systems. Finally, it is human readable. For a language data structure, JSON is an incredibly versatile tool. It is also fairly painless to use, especially when compared to other formats.

How JSON works

When you enter your username and password into a form on a web page, you are interacting with an object with two fields: username and passward.

CHAPTER 4

4.SYSTEM DESIGN:

4.1 ALGORITHM:

Iterative random vehicle algorithm (RV):

Randomization method is utilized to develop this algorithms As follows. First, classify the vehicles into three types. Te frst type, vehicle is chosen to be scheduled based on Vehicle index. Te next type is the scheduling of vehicles based on the non-decreasing order of the count of People in the vehicle. Te third type is the vehicle scheduling based on the decreasing order of the people count For each vehicle. For a certain type, choose a vehicle randomly from the set of given vehicles. Afer that, allocate The selected vehicle to the parking space that has the minimum total number of people, then repeat until fnishing all vehicles. Tis process is being repeated for many times. Terefore, for each type, execute the selection of Vehicles for lim times.In this context, the function rand(a, b) is responsible of deriving integers in the range a and b, while Asg(j) Is the function that assigns the vehicle j to the parking that has the minimum number of people. Let In() be the Function that sorts the given vehicles in an increasing order based on the number of people inside it. While, De()

M-vehicles with NI and random choice algorithm (NR):

Tis algorithm works as follows, schedule Part of the vehicles using the NI algorithm, then the remaining vehicles are scheduled by applying the random Choice of any of the remaining vehicles. Te frst chosen part is prepared based on a multiplication by the number of parking spaces, which is called the multiplier and is denoted by M. To illustrate, apply the NI algorithm For the frst 2 × np vehicles to be scheduled, the rest of the vehicles will be chosen randomly and will be allocated To the parking space that has the minimum number of people. For this case, the multiplier M is equal to 2. Iterate this algorithm for lim times. Afer that increment the multiplier M to 3 and so on until $M \times np < 50$ and $M \times np < nv$. Tisalgorithm is given the name NR and Table 3 describes the related execution steps

M-vehicles with randomized-NI and NI algorithm (RN):

Tis algorithm works as follows, schedule Part of the vehicles using the randomized-NI algorithm, then the remaining vehicles are scheduled by applying the NI algorithm. Te frst chosen part is performed based on a multiplication by the number of parking Spaces, which is called the multiplier and is denoted by M. Te same iteration which is based on the multiplier M Adopted for NR will be utilized in this algorithm. Tis algorithm will be denoted by RN.In the randomized-NI procedure, the randomization is achieved by selecting a probability α to choose vehicle with the largest count of people and with 1 – α for the next vehicle with the largest number of people . Te Algorithm given in Table 4 describes the instructions of the randomized-NI procedure RNI(.). In this algorithm M × np (the input of the procedure) is the set of vehicles that will be set by the multiplier M described in NR.Next, the instructions that elaborate RN as detailed in the algorithm illustrated in Table 5 is given.M-vehicles with randomized-NI and random vehicle algorithm (RR). Tis algorithm works as follows, schedule part of the vehicles using the randomized-NI algorithm as described in the “M-vehicles with Randomized-NI and NI algorithm (RN)” section, then schedule the remaining vehicles by applying the random Choice of any of the remaining vehicles. Te same iteration which is based on the multiplier M adopted for NR Will be used in this algorithm. Tis algorithm is denoted by RR.Part of vehicles with NI and random vehicle algorithm (NRβ). Tis algorithm works as follows, Schedule part of the vehicles using the NI algorithm, then schedule the remaining vehicles by applying the random choice of any of the remaining vehicles. Tis algorithm will introduce the percentage that will be used to Divide the set of given vehicles. First, defne β to be the probability that will be used to apply the division. Ten, Afer applying this division, two subsets S1 and S2, will be generated.

4.2 SYSTEM ARCHITECTURE:

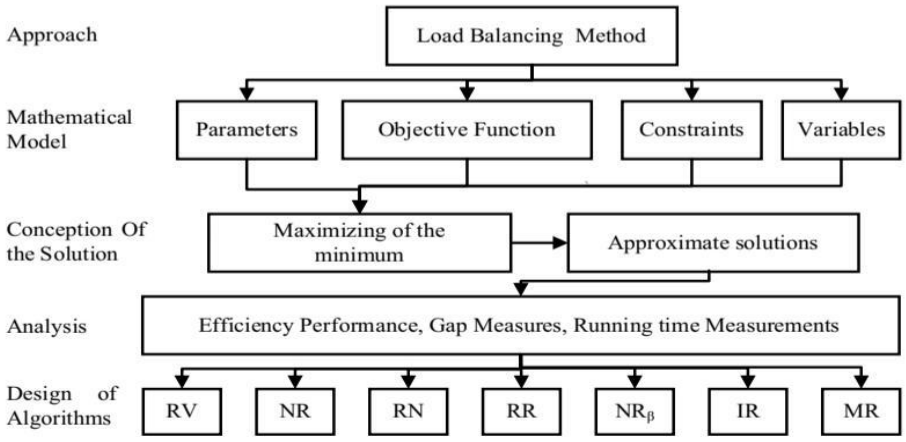

Figure 4. Logical structure diagram.

Fig 4.2 System architecture

4.3 CIRCUIT DIAGRAM :

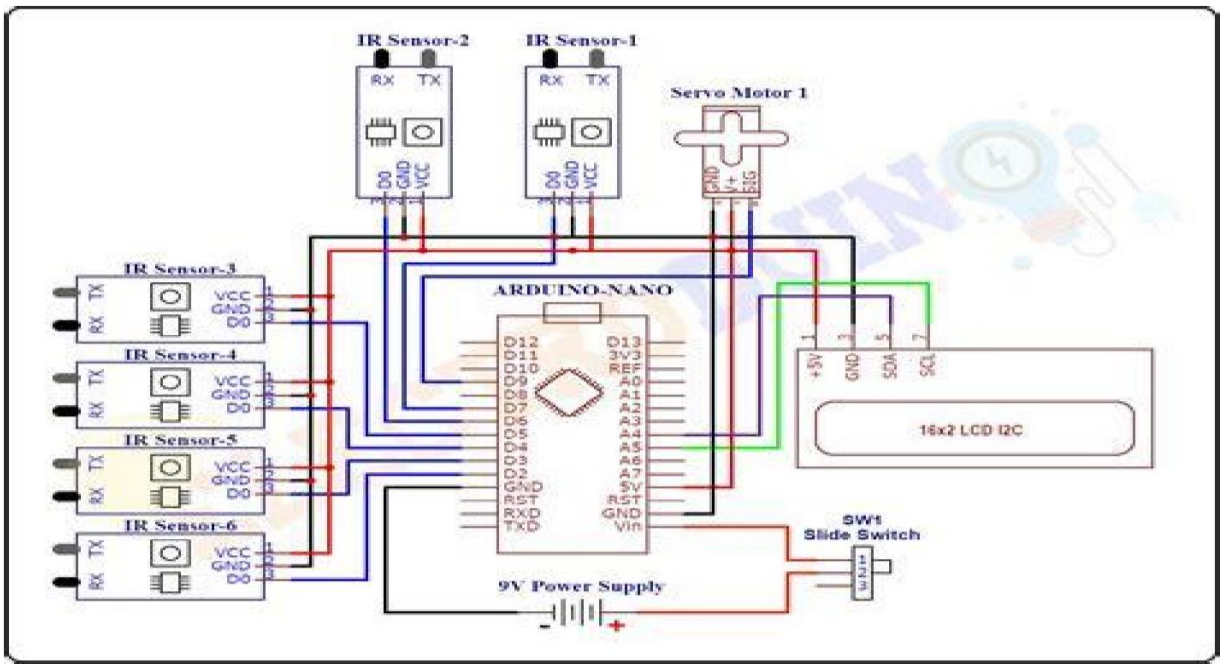

Fig 4.3 Circuit diagram

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 FRONT END SOURCE CODE:

```
#define ECHO_PIN1 15 //Pins for Sensor 1

#define TRIG_PIN1 2 //Pins for Sensor 1

#define ECHO_PIN2 5    //Pins for Sensor 2
#define TRIG_PIN2 18   //Pins for Sensor 2

#define ECHO_PIN3 26  //Pins for Sensor 3
#define TRIG_PIN3 27   //Pins for Sensor 3


int LEDPIN1 = 13; int
LEDPIN2 = 12; int
LEDPIN3 = 14;

void setup() { Serial.begin(115200);
pinMode(LEDPIN1, OUTPUT);
pinMode(TRIG_PIN1, OUTPUT);
pinMode(ECHO_PIN1, INPUT);

pinMode(LEDPIN2, OUTPUT); pinMode(TRIG_PIN2,
OUTPUT);
pinMode(ECHO_PIN2, INPUT);

pinMode(LEDPIN3, OUTPUT); pinMode(TRIG_PIN3,
OUTPUT);
pinMode(ECHO_PIN3, INPUT);
}
```

```
float readDistance1CM() { digitalWrite(TRIG_PIN1,
LOW); delayMicroseconds(2);
digitalWrite(TRIG_PIN1, HIGH);

delayMicroseconds(10); digitalWrite(TRIG_PIN1, LOW);
int duration = pulseIn(ECHO_PIN1, HIGH); return
duration * 0.034 /2 ;
}

float readDistance2CM() { digitalWrite(TRIG_PIN2,
LOW); delayMicroseconds(2); digitalWrite(TRIG_PIN2,
HIGH); delayMicroseconds(10); digitalWrite(TRIG_PIN2,
LOW); int duration = pulseIn(ECHO_PIN2, HIGH); return
duration * 0.034 / 2;
}

float readDistance3CM() { digitalWrite(TRIG_PIN3,
LOW); delayMicroseconds(2); digitalWrite(TRIG_PIN3,
HIGH); delayMicroseconds(10); digitalWrite(TRIG_PIN3,
LOW); int duration = pulseIn(ECHO_PIN3, HIGH); return
duration * 0.034 / 2;
}

void loop() { float distance1 =
readDistance1CM(); float distance2 =
readDistance2CM(); float distance3 =
readDistance3CM();

bool isNearby1 = distance1 > 200;
digitalWrite(LEDPIN1, isNearby1);

bool isNearby2 = distance2 > 200;
digitalWrite(LEDPIN2, isNearby2);



bool isNearby3 = distance3 > 200; digitalWrite(LEDPIN3, isNearby3);

Serial.print("Measured distance: ");
Serial.println(readDistance1CM());
```

```
Serial.println(readDistance2CM());
Serial.println(readDistance3CM()); delay(100);
}
```

5.2 BACK END SOURCE CODE:

```
{
"version": 1,
"author": "Surya K",
"editor": "wokwi",
"parts": [

{ "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": 168.01, "left": -54.47, "attrs": {} },
{ "type": "wokwi-hc-sr04", "id": "ultrasonic1", "top": 10.18, "left": 222.47, "attrs": {} },
{ "type": "wokwi-hc-sr04", "id": "ultrasonic2", "top": 10.18, "left": 7.37, "attrs": {} },
{ "type": "wokwi-hc-sr04", "id": "ultrasonic3", "top": 11.1, "left": -199.42, "attrs": {} }, {
"type": "wokwi-led",
"id": "led1",
"top": 215.93,
"left": -245.43,
"attrs": { "color": "green" }
},
{
"type": "wokwi-led",
"id": "led2",
"top": 217.94,
"left": -202.14,
"attrs": { "color": "green" }
},
{
"type": "wokwi-led",
"id": "led3",

[ "led3:C", "esp:GND.2", "black", [ "v0" ] ],
[ "led1:A", "esp:D13", "green", [ "v0" ] ],
```

["led2:A", "esp:D12", "green", ["v0"]],
["led3:A", "esp:D14", "green", ["v0"]],
["ultrasonic3:TRIG", "esp:D27", "green", ["v0"]],
["ultrasonic3:ECHO", "esp:D26", "green", ["v0"]],
["ultrasonic2:GND", "esp:GND.1", "black", ["v0"]],
["ultrasonic2:ECHO", "esp:D15", "green", ["v0"]],
["ultrasonic2:TRIG", "esp:D2", "green", ["v0"]],
["ultrasonic1:ECHO", "esp:D5", "green", ["v0"]],
["ultrasonic1:TRIG", "esp:D18", "green", ["v0"]]
],
"dependencies": {}
}

CHAPTER 6

SAMPLE OUTPUT

6.1 SCREENSHOT:

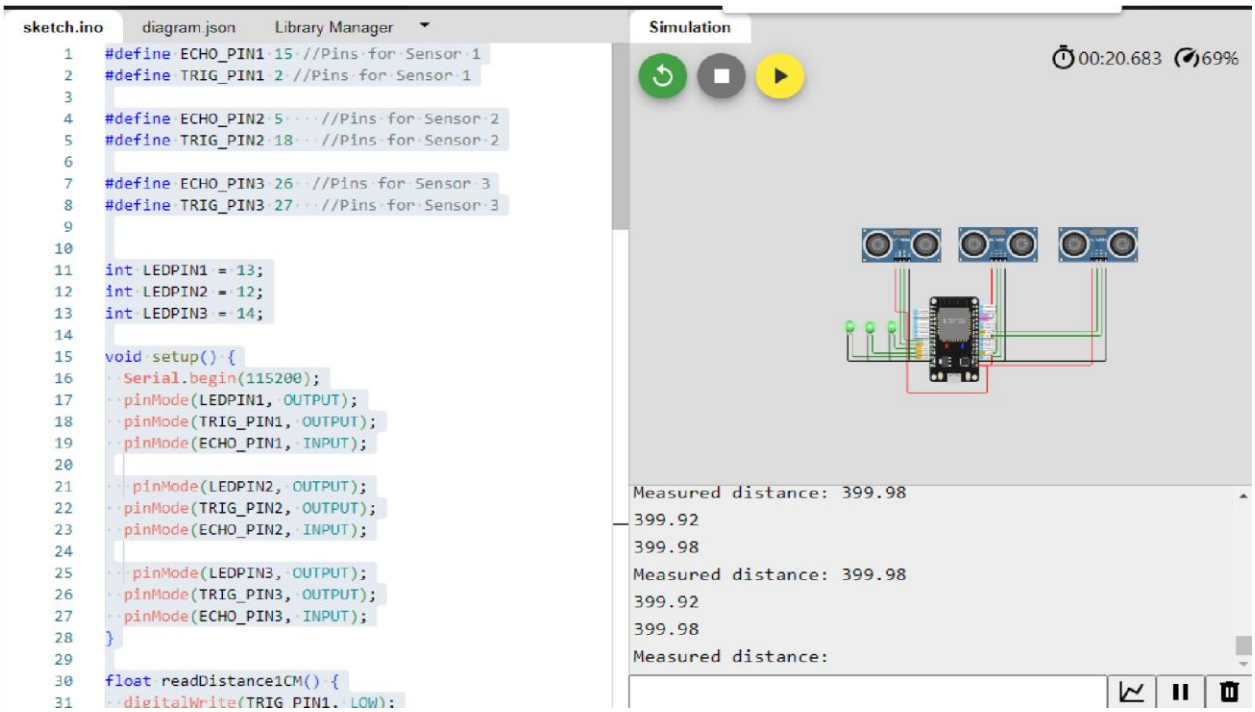

Fig:.4.1.1.initialize the code

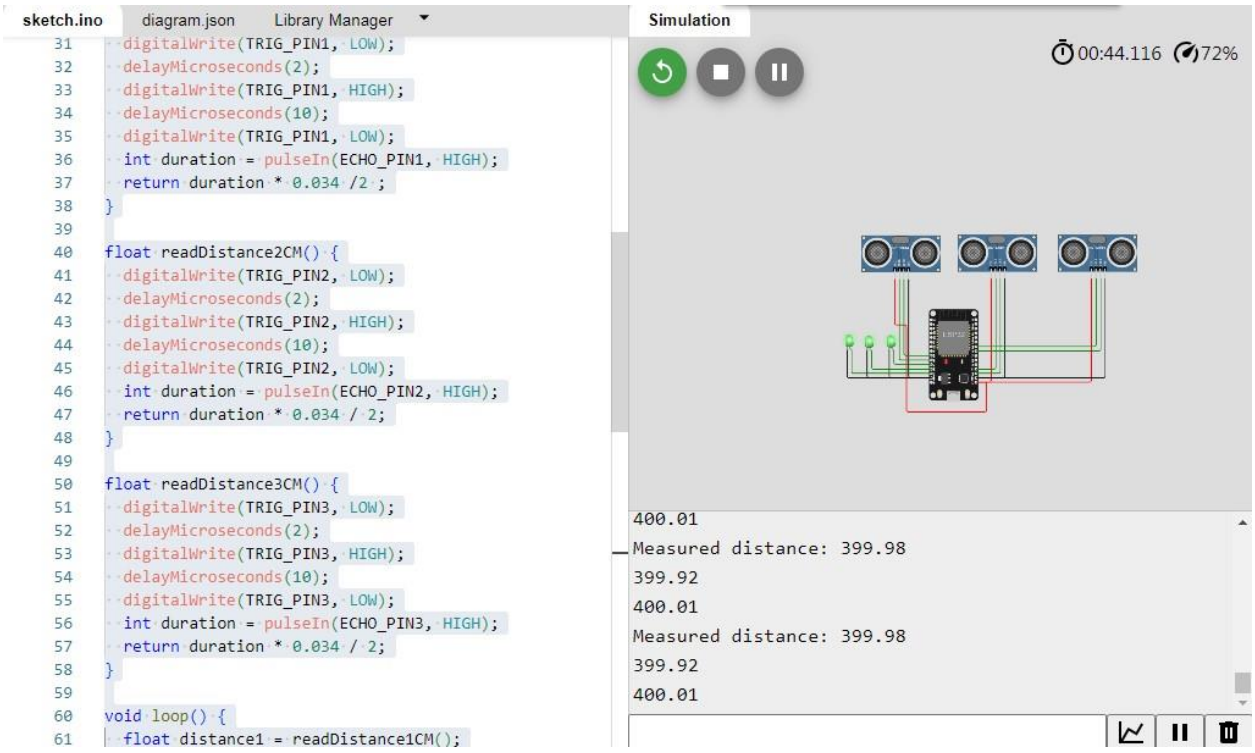

Fig:.4.1.2.processing

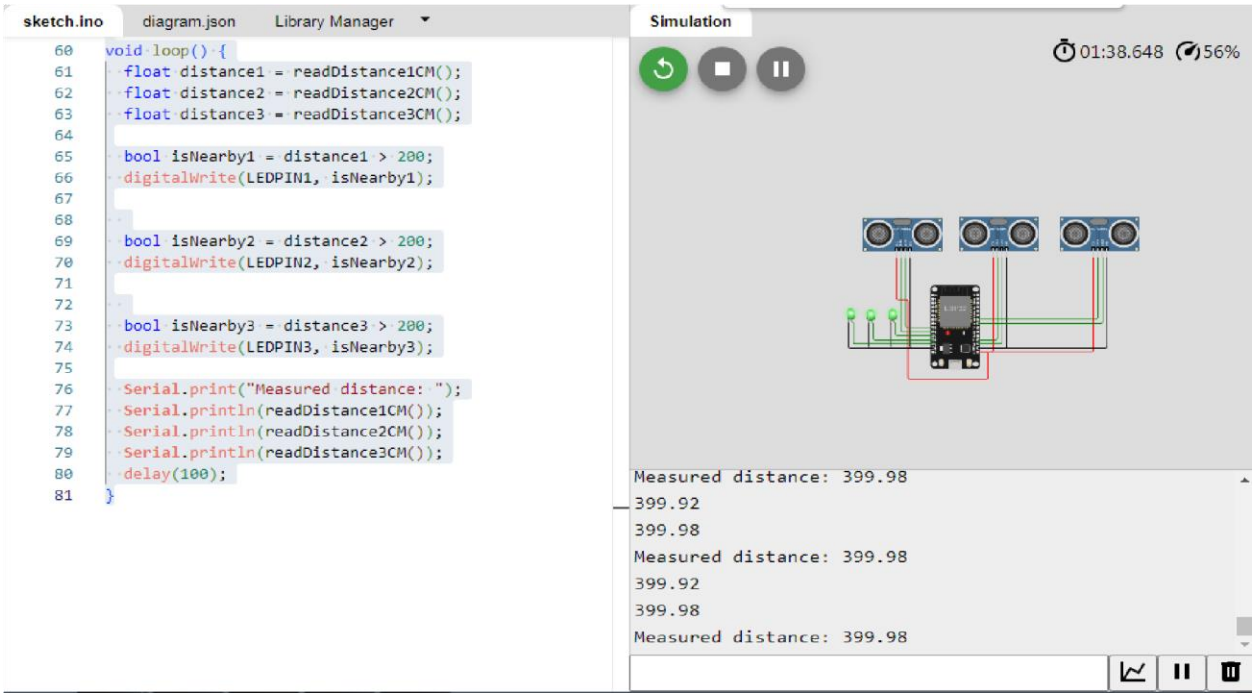

Fig:.4.1.3.ending the process

Fig:.4.2.1.inital state

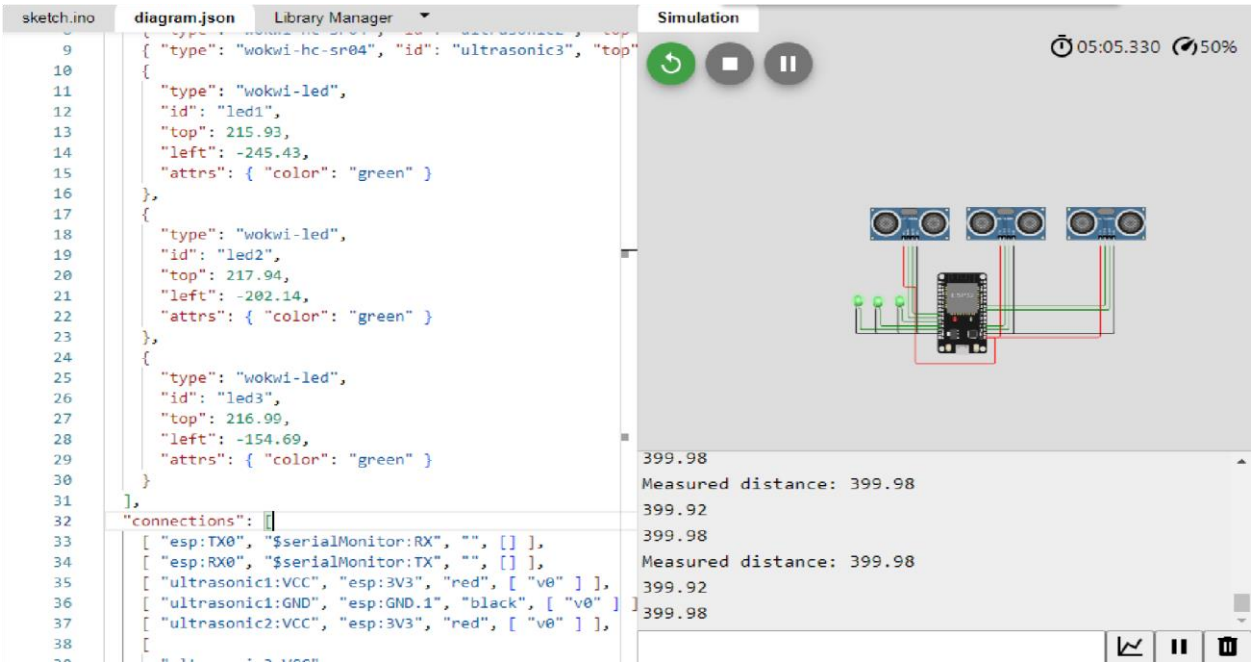

Fig:.4.2.2.processing

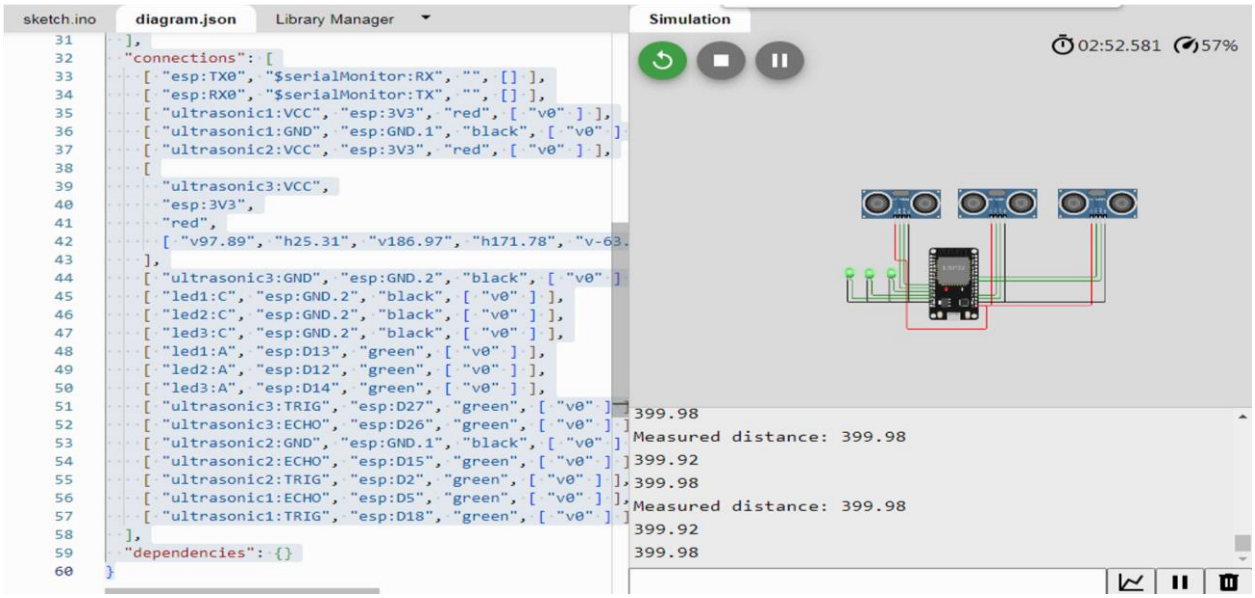

Fig:.4.2.3.Ending of the process

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

The services provided by smart parking have become the essence of building smart cities. This paper focused on implementing an integrated solution for smart parking. The proposed system has several advantages, including detecting parking spaces using the Internet of Things and calculating the time of entry and exit and calculating the expected cost. An attractive and effective application was designed for Android mobile phones. The system benefits from avoiding wasting time andreducing pollution and fuel consumption. Users can book a car park for 24 hours.

The adoption of IoT for smart parking brings numerous benefits. It optimizes parking space utilization, reduces traffic congestion, and enhances user experience. Drivers save time and fuel by quickly finding parking spots, while parking operators gain insights for efficient resource allocation. With IoT, revolutionize parking efficiency and create a seamless parking ecosystem

CHAPTER 8

REFERENCE

1. Abhirup Khanna, Rishi Anand, "IoT based Smart Parking System", Proc., In 2016 International Conference on Internet of Things and Applications (IOTA), 22 Jan - 24 Jan 2016. 2. Anusha, Arshitha M, S, Anushri, Geetanjali Bishtannavar "Review Paper on Smart Parking System," International Journal of Engineering Research & Technology (IJERT), ISSN: 22780181, Volume 7, Issue 08, Special Issue – 2019.

3. S. Senthil, M. Suguna, J. Cynthia, "Mapping the Vegetation Soil and Water Region Analysis of Tuticorin District Using Landsat Images", IJIEST ISSN (2455-8494), Vol.03, No. 01, Jan 2018.International Journal of Computer Science & Information Technology (IJCSIT) Vol 12, No 4, August 2020 66

4. Juhi Seth, Pola Ashritha, R Namith, "Smart Parking System using IoT ElakyaR", International Journal of Engineering and Advanced Technology (IJEAT), ISSN: 2249 – 8958, Volume-9 Issue-1, October 2019.

5. Mimbela, L.Y. and L.A. Klein, "A summary of vehicle detection and surveillance technologies used in intelligent transportation systems", New Mexico State University, Tech. The report, 2007.

6. M. Y. I. Idris, Y. Y. Leon, E. M. Tamil, N. M. Noor, and Z. Razak, "Car parking system: A review of the smart parking system and its technology," Information Technology Journal, pp. 101-113.], 2009.

7. Paidi. V; Fleyeh, H.; Hakansson, J.; Nyberg, R.G.," Smart Parking Sensors, Technologies and Applications for Open Parking Lots: A Review", IET Intel. Transport Syst, 12, 735–741, 2018.

8. Amir O. Kotb, Yao-Chunsheng, and Yi Huang "Smart parking Guidance, Monitoring and Reservation: A Review," IEEE-ITSM, pp.6-16. Apr-2017.

9. Supriya Shinde, AnkitaM Patial, pSusmedha Chavan, Sayali Deshmukh, and Subodh Ingleshwar, "IOT Based Parking System Using Google", Proc., of. I-SMAC,2017, pp.634-636, 2017.

10. Hemant Chaudhary, PrateekBansal., B. Valarmathi," Advanced CAR Parking System using Arduino", Proc., of. ICACCSS, 2017.

11. Wang, M.; Dong, H.; Li, X.; Song, L.; Pang, D. A Novel Parking System Designed for G. Searching page for parking H. View slots of parking Smart Cities. Proc., in 2017 Chinese Automation Congress (CAC), Jinan, China, pp. 3429–3434, 20–22 October 2017.

12. Nastaran Reza NazarZadeh, Jennifer C. Dela," Smart urban parking deducting system", Proc., of. ICSCE, 2016, pp-370-373,2016.

13. PavanKumarJogada and VinayakWarad, "Effective Car Parking Reservation System Based on Internet of things Technologies ", Proc., of. BIJSESC, Vol. 6, pp.140-142, 2016.

14. Yashomati R. Dhumal, Harshala A. Waghmare, Aishwarya S. Tole, Swati R. Shilimkar," Android Based Smart Car Parking System" Proc., of. IJREEIE, Vol. 5, Issue 3, pp-1371-74, mar-2016.

15. Faiz Ibrahim Shaikh, Pratik NirnayJadhav, Saideep Pradeep Bandarakar" Smart Parking System based on embedded system and sensor Network" IJCA, vol.140. pp.45-51. Apr-2016.

16. RicardGarra, Santi Martinez, and Francesc Seb"e" A Privacy-Preserving Pay-by-phone Parking system" IEEE-TVT, pp.1-10, Dec-2016.

17. Khanna, A.; Anand, R.," IoT based Smart Parking System", proc., in 2016 International Conference on Internet of Things and Applications (IOTA), Pune, India, 22–24 January 2016; pp. 266–270.

18. Karthi, M.; Preethi, H. Smart Parking with Reservation in Cloud-based environment. In Proceedings of the 2016 IEEE International Conference on Cloud Computing in Emerging Markets, Bangalore, India, 19–21 October 2016; pp. 164–167.

19. Orrie, O.; Silva, B.; Hancke, G.P. "A Wireless Smart Parking System", prco., in 41st Annual Conference of the IEEE Industrial Electronics Society (IECON), Yokohama, Japan, pp. 4110–4114, 9–12 November 2015.

20. Hsu, C.W.; Shih, M.H.; Huang, H.Y.; Shiue, Y.C.; Huang, S.C., "Verification of Smart Guiding System to Search for Parking Space via DSRC Communication", Proc., in 12th International Conference on ITS Telecommunications, Taipei, Taiwan, pp. 77–81, 5–8 November 2012.

21. Revathi, G., & Dhulipala," Smart parking systems and sensors: A survey", proc., in 2012 International Conference on Computing, Communication, and Applications, 2012.

22. Abhirup Khanna, Rishi Anand," IoT based Smart Parking System", proc., in International Conference on Internet of Things and Applications (IOTA) Maharashtra Institute of Technology, Pune, India 22 Jan - 24 Jan 2016.

23. https://en.wikipedia.org/wiki/MQTT, 18-7-2020.

24. Thusoo, A.; Sarma, J.S.; Jain, N.; Shao, Z.; Chakka, P.; Zhang, N.; Antony, S.; Liu, H.; Murthy, R. HIVE-A,"petabyte-scale data warehouse using Hadoop", proc., In 2010 IEEE 26th International Conference on Data Engineering (ICDE 2010), 2010.

25. https://www.arduino.cc, 18-7-2020.

26. ElakyaR, Juhi Seth, Pola Ashritha, R Namith," Smart Parking System using IoT ", International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-9 Issue-1, October 2019.

27. https://store.fut-electronics.com,18-7-2020.

28. https://www.instructables.com, 18-7-2020.

29. https://components101.com/sensors/tcrt5000-ir-sensor-pinout-datasheet. 18-7-2020.

30. https://engineering.eckovation.com, 18-7-2020.

31. https://www.variohm.com, 18-7-2020.