PHASE -5 INNOVATION

SMART PARKING SYSTEM

INTRODUCTION:

Smart parking system using IoT. This is a great system and project for techies, hobbyists, and project makers.

What do you need to make this system and how you can make it we will share all the details. We have uploaded a lot of projects before from our smart irrigation system using Arduino.

PROBLEM DEFINITION:

Smart Parking is based on lot (Internet of Things) and Al (Artificial intelligence) technologies. It gives access to real-time parking occupancy and parking availability, predicts peak and now peak times, makes deamic pricing acconting to congestion and provides parking cperators with big real-time data and reporting.

Smart parking system is an integrated system to organize vehicles in public

It eliminates unnecessary travelling of vehicles across the parking slots

DESIGN THINKING:

1.PROJECT OFECTIVE

Smartparking tecnologies ensure to reduce the number of cars caching around the eets for finding a parking spot. This ultimately smoothens the traffic flow and minin the search traffic on soets as much as possible

Increase in safety Drivers are less distracted looking around for a spot because they know where they can park their car. They will have the full attention on the road. By having their eyes on the road accidents will decrease and safety will increase for themselves, other divers and pedestrians

IOT SENSOR DESIGN:

1.IR SENSOR

Anfared sensor (IR) is a diation-sensitive optoelectronic component with a spectral sensitivity in the infrared wavelength range IR sensor now widely used in motion detectors, which are used in building services to switch on lamps or in a systems to detect unwelcome guests door Riccant that works the principle of Infared light emision and reflection to detect an object distance andinction

REAL TIME TRANSIT INFORMATION PLATFORM:

The Cayenne dashboard is the main screen where you can setup customzit, monitor, manage and control your connected devices. It is user friendly platform

The Cayenne Library is a collection of code known as sketch files that makes it easy for you to connect and send data to and from sensors, actuators and devices connected to Antuno boards.

INTEGRATION APPROACH:

Data is captured at the source through sensors (which are either inbuilt like in a robot or through external sensors installed on the connected device being monitored)

Data is then transmitted to a system for storage and organization purposes

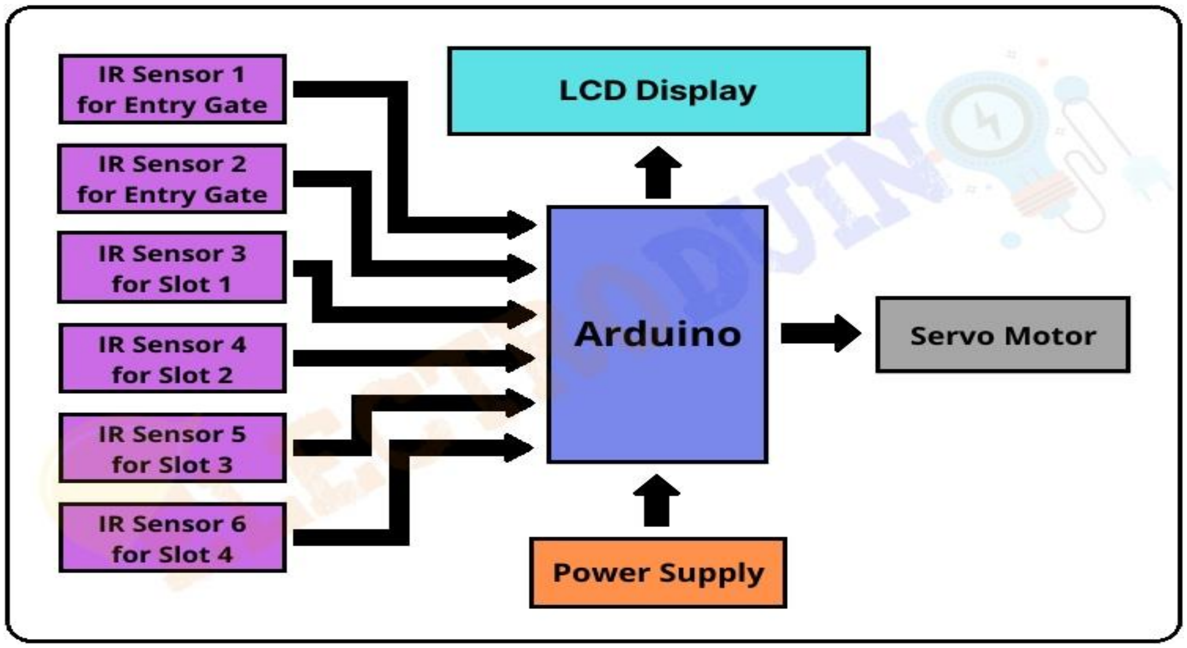
Data is consumed and analyzed by the system to generate actionable insights

DESIGN PRINCIPLE

As mentioned above, the proposed smart parking lot circuit will be equipped with several sensors, inexpensive microcontrollers and Wi-Fi module using which a car / any vehicle owner can check if there is vacant space in a parking lot using his / her phone or tablet or even on computer.

The number of vacant spaces in the smart parking lot can be viewed from anywhere in the world using a URL link or the user can scan a QR code. The scanned / shared URL can be browsed on any web browser to know how many empty parking spot exist in real time.

BLOCK DIAGRAM:



COMPONENTS :

1 NodeMCU (ESP-8266)

2 ARDUINO UNO

3 Ultrasonic Sensor SRF- 04

4 Servo Motor SG90

5 Infrared Sensor (IR)

SOFTWARE DEVELOPMENT:

1 Arduino IDE

2 Blynk Platform

WORKING PRINCIPLE :

After assembling all components according to the circuit diagram and uploading the code to the Arduino board. Now place the sensors and servo motor at accurate positions.

There are four parking slots in this project, IR sensor-3, 4, 5, and 6 are placed at slot-1, 2, 3, and 4 respectively. IR sensor-1 and 2 are placed at the entry and exit gate respectively and a servo motor is used to operate the common single entry and exit gate. The LCD display is placed near the entry gate.The system used IR

Sensor-3, 4, 5, and 6 to detect whether the parking slot is empty or not and IR sensor-1, and 2 for detecting vehicles arriving or not at the gate.

In the beginning, when all parking slots are empty, then the LCD display shows all slots are empty.

When a vehicle arrives at the gate of the parking area then the IR sensor-1 detects the vehicle and the system allowed to enter that vehicle by opening the servo barrier. After entering into the parking area when that vehicle occupies a slot then the LED display shows that the slot is full. In this way, this system allows 4 vehicles.

In case the parking is full, the system blocked the entrance gate by closing the servoBarrier. And the LED display shows that slot-1, 2, 3, and 4 all are full.

When a vehicle leaves a slot and arrives at the gate of the parking area then the IR sensor-2 detects that vehicle and the system open the servo barrier. Then the LED display shows that the slot is empty. Again the system will allow entering a new vehicle.

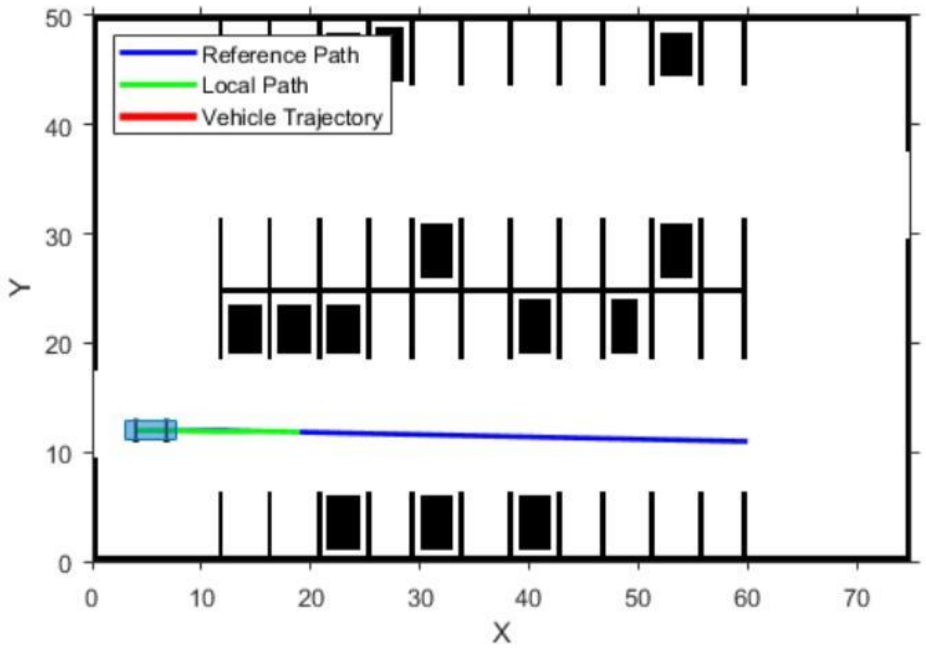
PROGRAMMING:

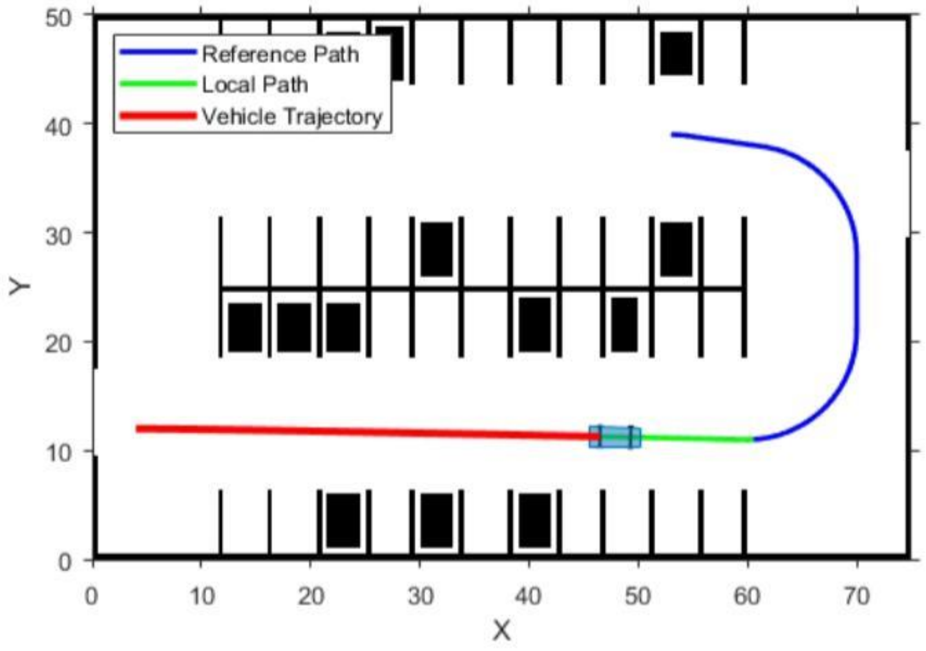
```
Import colorama

From termcolor import colored

Options_message = “””

Choose:

1. To park a vehicle

2. To remove a vehicle from parking

3. Show parking layout

4. Exit

“””

Class Vehicle:

     Def _init_(self, v_type, v_number):

        Self.v_type = v_type

        Self.v_number = v_number

        Self.vehicle_types = {1: ‘c’, 2: ‘b’, 3: ‘t’}

Def _str_(self):

        Return self.vehicle_types[self.v_type]

Class Slot:

  Def _init_(self):

        Self.vehicle = None


    @property

    Def is_empty(self):

        Return self.vehicle is None

Class Parking:
    Def _init_(self, rows, columns):
        Self.rows = rows
        Self.columns = columns
        Self.slots = self._get_slots(rows, columns)


    Def start(self):
        While True:
            Try:
                Print(options_message)


                Option = input(“Enter your choice: “)


                If option == ‘1’:
                    Self._park_vehicle()


                If option == ‘2’:
                    Self._remove_vehicle()


                If option == ‘3’:
                    Self.show_layout()


                If option == ‘4’:
                    Break


            Except ValueError as e:
Print(colored(f”An error occurred: {e}. Try again.”, “red”))
```

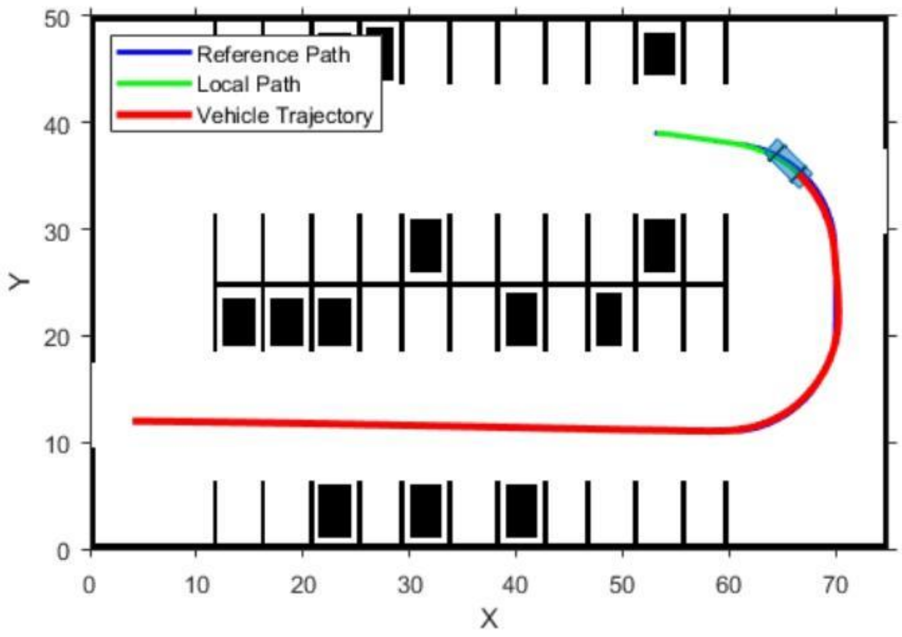
```
        Print(colored("Thanks for using our parking assistance system", "green"))
    Def _park_vehicle(self):
        Vehicle_type = self._get_safe_int("Available vehicle types: 1. Car\t2. Bike\t3. Truck.\nEnter your choice: ")

        If vehicle_type not in [1, 2, 3]:
            Raise ValueError("Invalid vehicle type specified")

        Vehicle_number = input("Enter vehicle name plate: ")
        If not vehicle_number:
            Raise ValueError("Vehicle name plate cannot be empty.")
        Vehicle = Vehicle(vehicle_type, vehicle_number)

        Print('\n')
        Print(colored(f"Slots available: {self._get_slot_count()}\n", "yellow"))
        Self.show_layout()
        Print('\n')

        Col = self._get_safe_int("Enter the column where you want to park the vehicle: ")
        If col <= 0 or col > self.columns:
            Raise ValueError("Invalid row or column number specified")

        Row = self._get_safe_int("Enter the row where you want to park the vehicle: ")
        If row <= 0 or row > self.rows:
            Raise ValueError("Invalid row number specified")
        Slot = self.slots[row-1][col-1]
        If not slot.is_empty:
            Raise ValueError("Slot is not empty. Please choose an empty slot.")
Slot.vehicle = vehicle


    Def _remove_vehicle(self):
        Vehicle_number = input("Enter the vehicle number that needs to be removed from parking
slot: ")
        If not vehicle_number:
            Raise ValueError("Vehicle number is required.")


        For row in self.slots:
            For slot in row:
                If slot.vehicle and slot.vehicle.v_number.lower() == vehicle_number.lower():
                    Vehicle: Vehicle = slot.vehicle
                    Slot.vehicle = None
                    Print(colored(f"Vehicle with number '{vehicle.v_number}' removed from parking",
"green"))
                    Return
        Else:
            Raise ValueError("Vehicle not found.")


    Def show_layout(self):
        Col_info = [f'<{col}>' for col in range(1, self.columns + 1)]
        Print(colored(f"|{''.join(col_info)}|columns", "yellow"))
Self._print_border(text="rows")


        For I, row in enumerate(self.slots, 1):
```

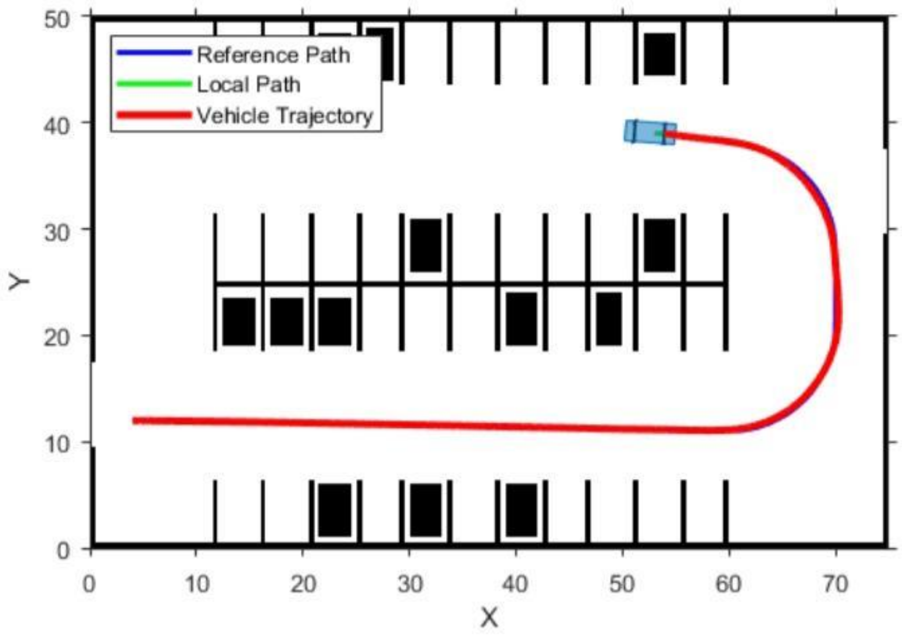
```
        String_to_printed = "|"
        For j, col in enumerate(row, 1):
            String_to_printed += colored(f"[{col.vehicle if col.vehicle else ' '}]",
                                         "red" if col.vehicle else "green")
        String_to_printed += colored(f"|<{i}>", "cyan")
        Print(string_to_printed)

    Self._print_border()

Def _print_border(self, text=""):
    Print(colored(f"|{'-' * self.columns * 3}|{colored(text, 'cyan')}", "blue"))

Def _get_slot_count(self):
    Count = 0
    For row in self.slots:
        For slot in row:
            If slot.is_empty:
                Count += 1
    Return count

@staticmethod
Def _get_slots(rows, columns):
    Slots = []
    For row in range(0, rows):
        Col_slot = []
        For col in range(0, columns):
            Col_slot.append(Slot())
            Slots.append(col_slot)
        Return slots
    @staticmethod
    Def _get_safe_int(message):
        Try:
            Val = int(input(message))
            Return val
        Except ValueError:
            Raise ValueError(“Value should be an integer only”)
Def main():
    Try:
        Print(colored(“Welcome to the parking assistance system.”, “green”))
        Print(colored(“First let’s setup the parking system”, “yellow”))
        Rows = int(input(“Enter the number of rows: “))
        Columns = int(input(“Enter the number of columns: “))
        Print(“Initializing parking”)
        Parking = Parking(rows, columns)
        Parking.start()


    Except ValueError:
        Print(“Rows and columns should be integers only.”)


    Except Exception as e:
        Print(colored(f”An error occurred: {e}”, “red”))
If _name_ == ‘_main_’:
    Colorama.init()  # To enable color visible in command prompt
    Main()
```

OUTPUT:


Reference Path
Local Path
Vehicle Trajectory
X
Y
0
10
20
30
40
50
60
70


Reference Path
Local Path
Vehicle Trajectory
X
Y
0
10
20
30
40
50
60
70


Reference Path
Local Path
Vehicle Trajectory
Y
X
0
10
20
30
40
50
60
70


Reference Path
Local Path
Vehicle Trajectory
Y
X
0
10
20
30
40
50
60
70

CONCLUSION:

This project focuses on implementing car parking place detection by using Internet of Things so time can be saved Smart parking system is an integrated system to organize vehicles in public area & Increase in safety Drivers are less distracted looking around for a spot because they know where they can park their car