# PUBLIC TRANSPORT OPTIMIZATION

<u>TEAM MEMBERS</u>

- *SANJAY KUMAR. S*
- *NANDHA KUMAR. M*
- *NISHANTH .U*
- *MANI VEL. S*
- *SRI HARIHARAN. C*

<u>MENTOR</u>

- *PRABHU*

## PROJECT DEFENITION :

- *Public transport optimization refers to the process of improving and maximizing the efficiency, effectiveness, and overall performance of public transportation systems.*
- *It involves various strategies and measures aimed at enhancing accessibility, reducing travel time, increasing ridership, improving cost-effectiveness, and providing a better experience for passengers.*
- *The goal is to create a more sustainable, convenient, and reliable public transport system that meets the needs of the community.*

## PROJECT OBJECTIVES :

- *Reduce travel time for passengers.*
- *Increase the frequency of public transport services.*
- *Improve the reliability of public transport schedules.*
- *Enhance accessibility for all passengers, including those with disabilities.*
- *Minimize overcrowding on public transport vehicles.*
- *Maximize the coverage of public transport networks to serve more areas.*
- *Improve the overall user experience and customer satisfaction with public transport services.*

## DESIGN THINKING

- *IOT sensor design*
- *Real –Transit information platform*
- *Integration approach*
- *Passenger informative system*
- *Integrating with other mobility services*

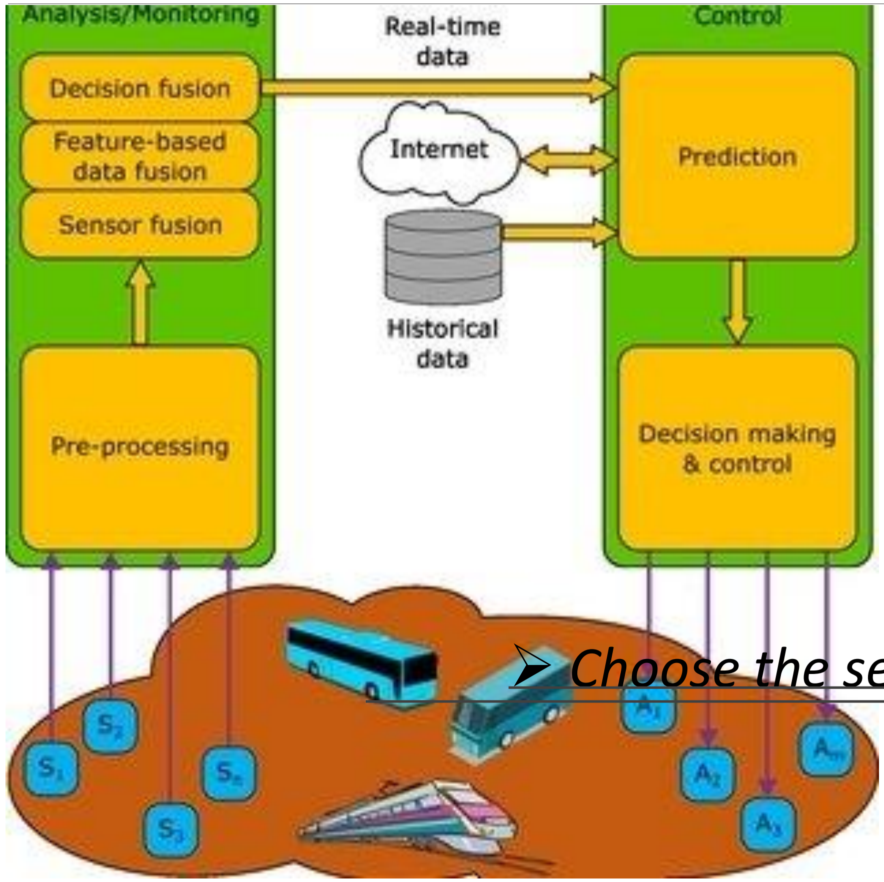
## IOT SENSOR DESIGN :

- ***Traffic monitoring:*** *The system can use traffic sensors to monitor traffic congestion and provide alternative routes to drivers.*
- ***Maintenance monitoring:*** *The system can use sensors to monitor the condition of vehicles and provide alerts when maintenance is required. This will help in reducing breakdowns and ensuring that vehicles are always in good condition.*
- ***Analytics and reporting:*** *The system can provide analytics and reporting on vehicle performance, fuel consumption, and driver behavior.*

# REAL TRANSIT INFORMATION PLATFORM :

- ***Convenient access:*** *Passengers can access real-time transit information through mobile apps or digital signage, making it easy to stay informed about service changes and updates.*
- ***Efficient travel:*** *By providing accurate information on delays or diversions, the system helps passengers find alternative routes and avoid congestion, leading to more efficient travel.*
- ***Enhanced satisfaction:*** *With improved information and reduced wait times, passengers have a better overall experience, leading to increased satisfaction and a higher likelihood of using public transportation.*

## INTEGRATION APPROACH :

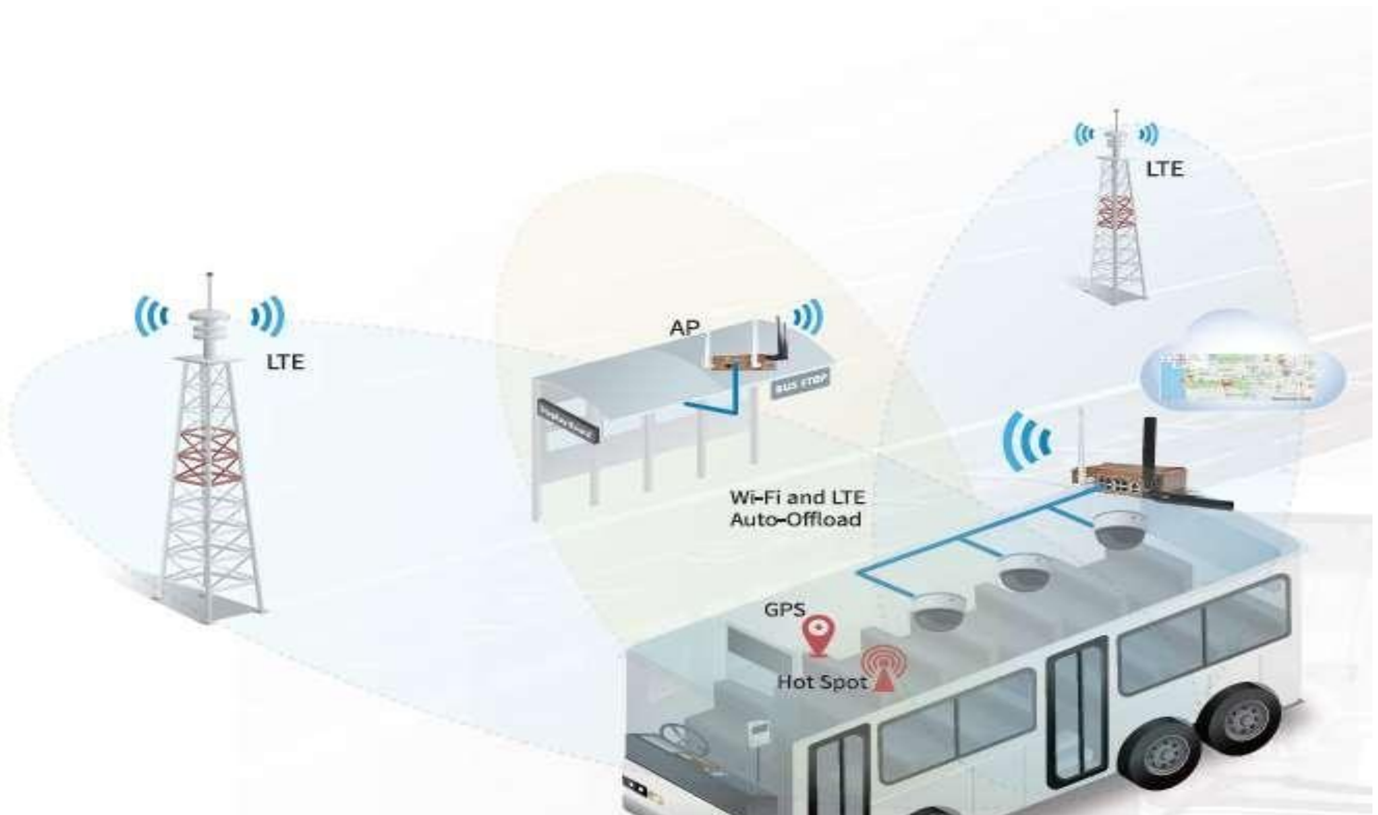


- *Integration approach is to leverage data from various sources and systems to improve the overall efficiency and effectiveness of the transportation network.*

- *Integrating data from different modes of transport, such as buses, trains, and trams, to provide a seamless and interconnected experience for passengers.*

- *Choose the sensors that are relevant to your application requirement*

## *INTEGRATION WITH TRANSPORT*



## *INTEGRATION WITH TRANSPORT:*

- DEFINITION:

- *1. Real-Time Transit Information: Integrate real-time updates on bus and train schedules into popular mapping apps like Google Maps or Apple Maps. This allows users to easily access accurate information on arrival times and plan theirjourneys accordingly.*
- *2. Intermodal Connectivity: Improve integration between different modes of transport, such as buses, trains, and bike-sharing services. This includes optimizing transfer points, providing clear signage, and ensuring seamless connections to enhance the overall travel experience.*

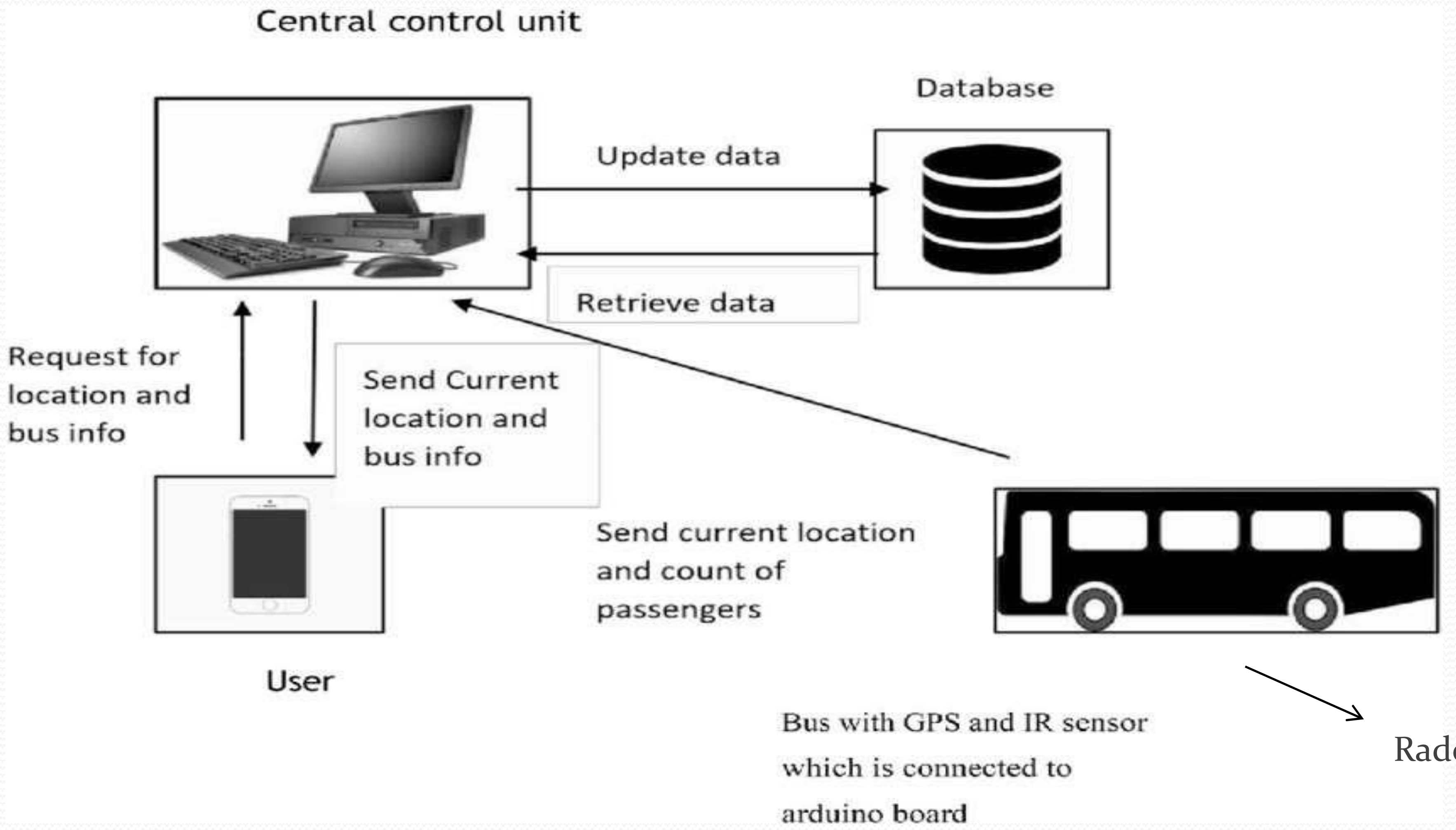
## ➢ *EXISTING SYSTEM:*

- ➢ *The existing system of bus transport typically involves a network of routes and schedules to provide transportation services to passengers. Buses follow designated routes, pick up and drop off passengers at designated stops, and operate on a fixed schedule.*
- ➢ *Passengers pay fares, either with cash or through electronic payment methods, to board the bus.*
- ➢ *Bus transport plays a crucial role in providing affordable and accessible transportation options for people in urban and rural areas.* 🚌

## ➢ *LIABILITIES:*

- ➢ *Traffic congestion becomes worse due to bad roads and constructions. Due to traffic congestions sudden changes of the bus routes were unknown.*
- ➢ *As public transport vehicles don't stop at specific destinations, you must take care of your travel from the stand or station to reach your desired stop. Privacy is a big issue in public transport. There are a lot of crowds, and sometimes you need more space to sit.*
- ➢ *The timings of the public transport timetable may differ from your timings. It leads to increased waiting time and disturbance of your routine. The limited space may be uncomfortable for women with children or disabled people as personal vehicles.*

## ➢ *SENSOR DESIGN:*

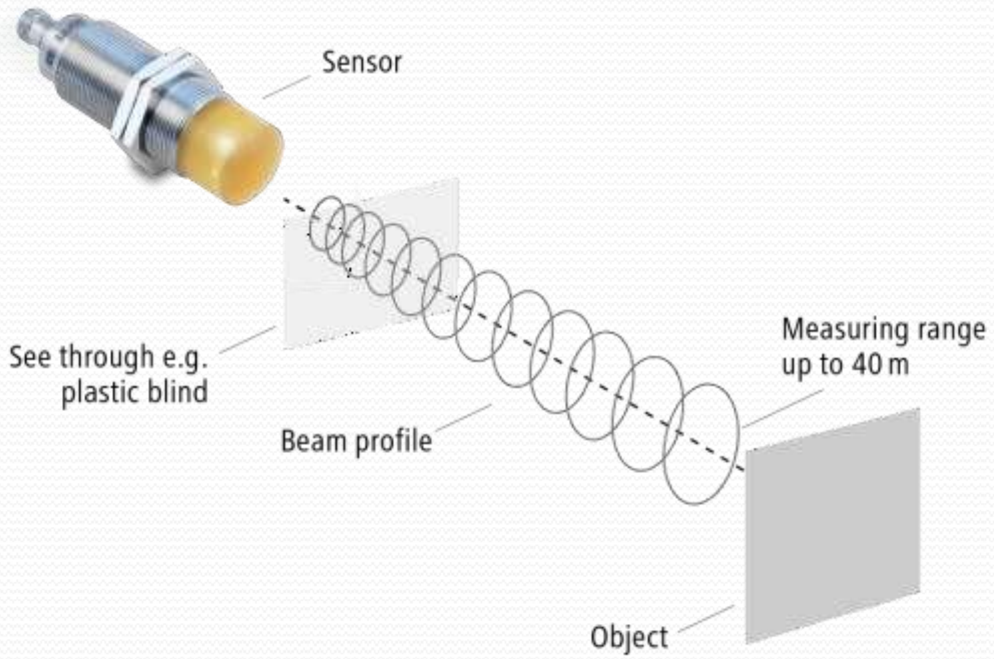

Central control unit
Database
Update data
Retrieve data
Request for location and bus info
Send Current location and bus info
Send current location and count of passengers
User
Bus with GPS and IR sensor which is connected to arduino board

Rador sensor

## ➢ *WORKINGS:*

➢ GPS SENSOR

1. GPS Tracking: GPS sensors installed in buses receive signals from multiple satellites to determine the precise location of the bus. This information is then transmitted to a central server or cloud platform through the internet.
2. Real-Time Monitoring: The data from GPS sensors allows transportation authorities or operators to monitor the position of buses in real-time. This enables them to track the progress of each bus, ensure adherence to schedules, and provide accurate arrival time information to passengers.
3. Route Optimization: By analyzing the GPS data, transportation authorities can identify traffic patterns, congestion points, and areas with high demand. This information can be used to optimize bus routes, improve efficiency, and reduce travel times.



- RADOR SENSOR

## ➢ *WORKINGS:*

1. Collision Avoidance: Radar sensors can detect objects in the bus's vicinity and provide warnings to the driver if there is a risk of collision. This helps improve safety by alerting the driver to potential hazards, such as pedestrians or vehicles in blind spots.
2. Adaptive Cruise Control: These sensors measure the distance to the vehicle ahead and adjust the bus's speed accordingly, maintaining a safe following distance.
3. Blind Spot Detection: Radar sensors can assist in detecting vehicles or objects in the bus's blind spots. This helps the driver make safer lane change decisions and reduces the risk of accidents.

## ➢ *Sensor implementation:*

- ➢ Step 1: Choose the GPS and RADAR sensor compatible with bus control system. Install the GPS sensor typically on the roof top for better signal reception. Place the RADAR sensor in rear bumper to provide comprehensive view of surrounding.

- ➢ Step 2: The GPS and RADAR data can be integrated into a central control unit where the information from both sensors is processed by a multicore processor.

- ➢ Step 3: Use GNSS\SDR software to process the GPS and MATPLOTLIB to process RADAR data to detect obstacles and provide collision and tracking signal to users.

- ➢ Step 4:To store historical GPS and RADAR data for analysing routes, fuel efficiency and maintaining schedules we use memory elements.

- ➢ Step 5: Create a website or web application to display the processed radar data by using HTML, CSS, JavaScript for framework to streamline development. To plot the GPS integrating the data with website by implementing the Google maps API, Mapbox and create custom visualization of the data from control unit.

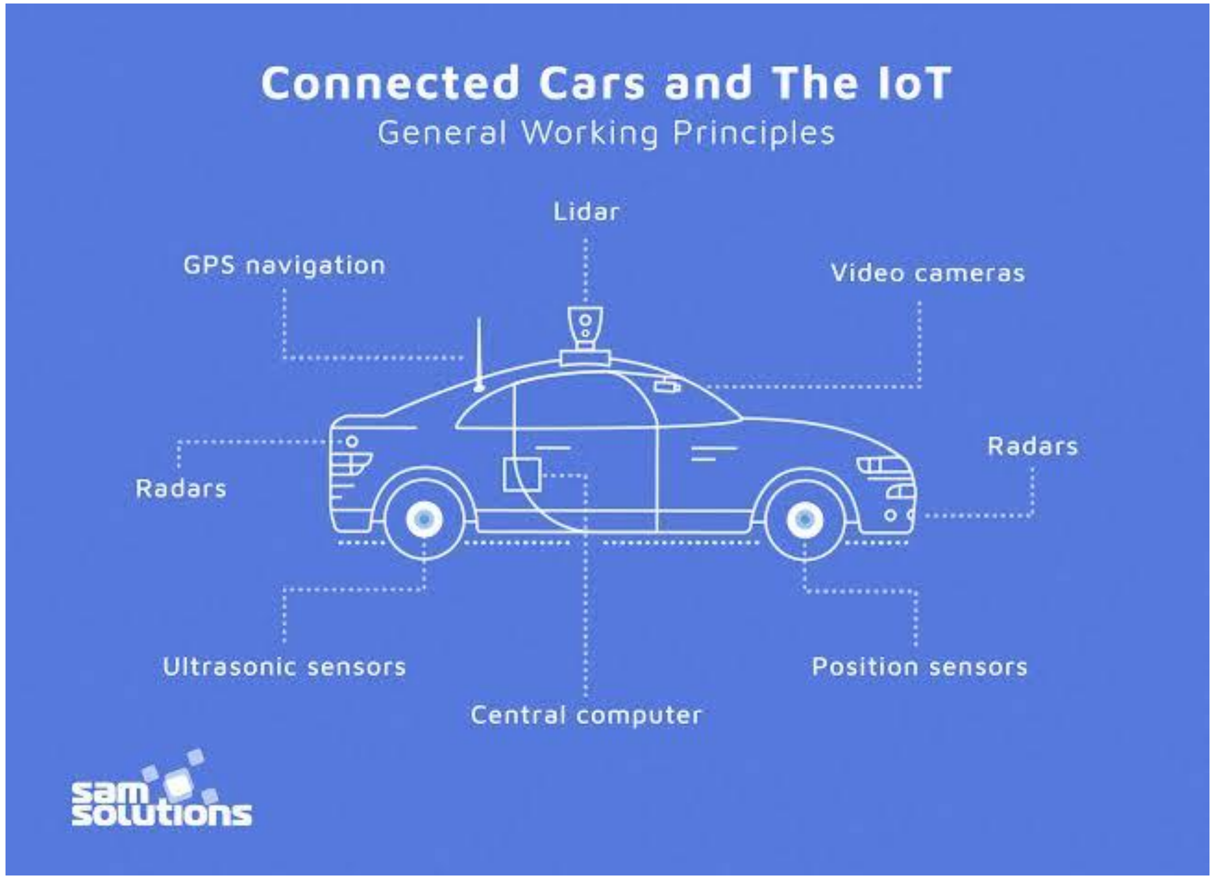
## ➢ *APPLICATIONS:*

1. Navigation: GPS helps drivers and passengers find the best routes, saving time and reducing congestion.
2. Fleet Management: GPS enables real-time tracking of vehicles, allowing companies to monitor their fleet, optimize routes, and improve efficiency.
3. Collision Avoidance: Radar sensors can detect objects in the bus's vicinity and provide warnings to the driver if there is a risk of collision.
4. Adaptive Cruise Control: These sensors measure the distance to the vehicle ahead and adjust the bus's speed accordingly, maintaining a safe following distance.

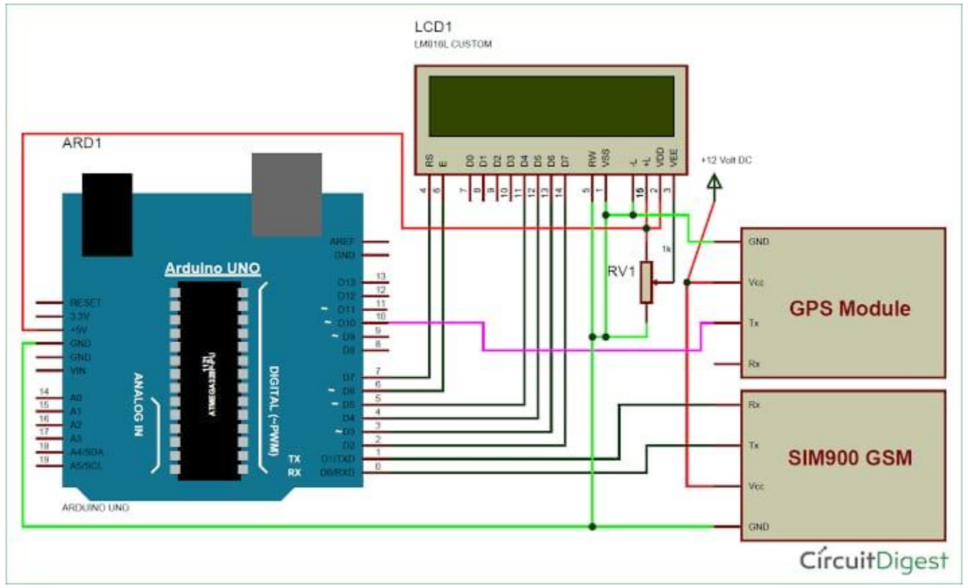
## ➢ *Benefits of proposed system:*

1. GPS (Global Positioning System) provides accurate location information, helping drivers navigate and find the best routes to their destinations. It also enables featureslike real-time traffic updates and turn-by-turn directions, enhancing convenience andefficiency.
2. Radar sensors also play a role in adaptive cruise control systems, maintaining a safedistance from the vehicle ahead by adjusting the vehicle's speed. This enhances safetyand reduces driver fatigue during long drives.
3. In parking situations, radar sensors help drivers detect obstacles and provide assistance in maneuvering the vehicle safely into parking spaces. This makes parkingeasier and reduces the risk of collisions.

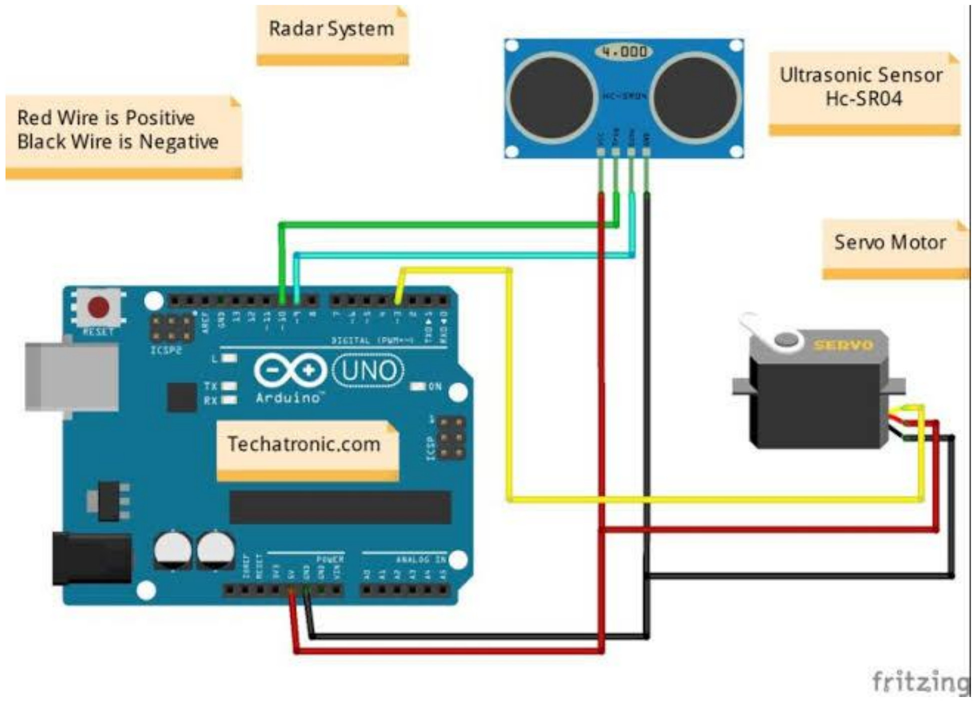
**PROJECT FUNCTION:**



## CIRCUIT DIAGRAM:



*GPS SENSOR



***RADOR SENSOR**

## CODING :

```
% Initialize display for driving scenario example
helperAutoDrivingRadarSigProc('Initialize Display',egoCar,radarParams,...
    rxArray,fc,vMax,rangeMax);

tgtProfiles = actorProfiles(scenario);
tgtProfiles = tgtProfiles(2:end);
tgtHeight = [tgtProfiles.Height];

% Run the simulation loop
sweepTime = waveform.SweepTime;
while advance(scenario)

    % Get the current scenario time
    time = scenario.SimulationTime;

    % Get current target poses in ego vehicle's reference frame
    tgtPoses = targetPoses(egoCar);
    tgtPos = reshape([tgtPoses.Position],3,[]);
    % Position point targets at half of each target's height
    tgtPos(3,:) = tgtPos(3,:)+0.5*tgtHeight;
    tgtVel = reshape([tgtPoses.Velocity],3,[]);

    % Assemble data cube at current scenario time
    Xcube = zeros(Nft,Ne,Nsweep);
    for m = 1:Nsweep

        ntgt = size(tgtPos,2);
    tgtStruct = struct('Position',mat2cell(tgtPos(:).',1,repmat(3,1,ntgt)),...
        'Velocity',mat2cell(tgtVel(:).',1,repmat(3,1,ntgt)),...
        'Signature',{rcsSignature,rcsSignature,rcsSignature});
    rxsig = radar(tgtStruct,time+(m-1)*sweepTime);
    % Dechirp the received signal
    rxsig = dechirp(rxsig,sig);

    % Save sweep to data cube
    Xcube(:,:,m) = rxsig;

    % Move targets forward in time for next sweep
    tgtPos = tgtPos+tgtVel*sweepTime;
end

% Calculate the range-Doppler response
[Xrngdop,rnggrid,dopgrid] = rngdopresp(Xcube);

% Beamform received data
Xbf = permute(Xrngdop,[1 3 2]);
Xbf = reshape(Xbf,Nr*Nd,Ne);
Xbf = beamformer(Xbf);
Xbf = reshape(Xbf,Nr,Nd);

% Detect targets
Xpow = abs(Xbf).^2;
[detidx,noisepwr] = cfar(Xpow,idxCFAR);

% Cluster detections
[~,clusterIDs] = clusterer(detidx.');
```

```
% Estimate azimuth, range, and radial speed measurements
[azest,azvar,snrdB] = ...
    helperAutoDrivingRadarSigProc('Estimate Angle',doaest,...
    conj(Xrngdop),Xbf,detidx,noisepwr,clusterIDs);
azvar = azvar+radarParams.RMSBias(1)^2;

[rngest,rngvar] = rngestimator(Xbf,rnggrid,detidx,noisepwr,clusterIDs);
rngvar = rngvar+radarParams.RMSBias(2)^2;

[rsest,rsvar] = dopestimator(Xbf,dopgrid,detidx,noisepwr,clusterIDs);

% Convert radial speed to range rate for use by the tracker
rrest = -rsest;
rrvar = rsvar;
rrvar = rrvar+radarParams.RMSBias(3)^2;

% Assemble object detections for use by tracker
numDets = numel(rngest);
dets = cell(numDets,1);
for iDet = 1:numDets
    dets{iDet} = objectDetection(time,...
        [azest(iDet) rngest(iDet) rrest(iDet)]',...
        'MeasurementNoise',diag([azvar(iDet) rngvar(iDet) rrvar(iDet)]),...
        'MeasurementParameters',{radarParams},...
        'ObjectAttributes',{struct('SNR',snrdB(iDet))});
end

% Track detections
```

```
    tracks = tracker(dets,time);

    % Update displays
    helperAutoDrivingRadarSigProc('Update Display',egoCar,dets,tracks,...
        dopgrid,rnggrid,Xbf,beamscan,Xrngdop);

    % Collect free space channel metrics
    metricsFS = helperAutoDrivingRadarSigProc('Collect Metrics',...
        radarParams,tgtPos,tgtVel,dets);
end
```

## CODING 2:

```
import serial

import csv


# Open a connection to the GPS device

gps_serial = serial.Serial('COM3', 9600, timeout=1)
```

```
# Create a CSV file for data storage
csv_file = open('gps_data.csv', 'w', newline='')
csv_writer = csv.writer(csv_file)

# Write header row
csv_writer.writerow(['Timestamp', 'Latitude', 'Longitude', 'Altitude'])

try:
    while True:
        gps_data = gps_serial.readline().decode('utf-8')
        if gps_data.startswith('$GPGGA'):
            data_fields = gps_data.split(',')
            if len(data_fields) >= 10:
                timestamp = data_fields[1]
                latitude = data_fields[2]
                longitude = data_fields[4]
                altitude = data_fields[9]
                csv_writer.writerow([timestamp, latitude, longitude, altitude])
```

```
except KeyboardInterrupt:
    # Close the CSV file and serial connection on keyboard interrupt
    csv_file.close()
    gps_serial.close()
```

# Workflow Template:

## Phase 1: Setting up the Work Environment

- ☐ Install necessary Python packages
    - ☐ Install pandas
    - ☐ Install numpy
    - ☐ Install matplotlib
    - ☐ Install geopandas

- ☐ Set up a Python environment for coding #EnvironmentSetup

## Phase 2: Data Collection

- ☐ Connecting to GPS device #DataCollection
  - ☐ Check GPS device connection
  - ☐ Configure GPS device settings
- ☐ Collect GPS signals
  - ☐ Start GPS data collection
  - ☐ Monitor GPS data collection process
  - ☐ Stop GPS data collection
- ☐ Save collected data into a file

## Phase 3: Data Processing

- ☐ Load the collected data into Python #DataProcessing
- ☐ Convert GPS signals into readable format
  - ☐ Define the GPS signal structure

- ☐ Parse the GPS signals
- ☐ Extract useful data from GPS signals
    - ☐ Extract longitude and latitude
    - ☐ Extract altitude
    - ☐ Extract timestamp

## Phase 4: Data Analysis and Visualization

- ☐ Analyze the extracted data #DataAnalysis
    - ☐ Calculate distance travelled
    - ☐ Calculate speed
- ☐ Visualize the data
    - ☐ Plot the GPS data on a map
    - ☐ Plot speed over time
- ☐ Save and export the results

**Note:**

- Bulleted lists represent tasks that can be done in any order.

- Numbered lists represent tasks that should be done in a specific order.
- Checkboxes represent tasks that can be marked as complete or incomplete.
- Indentation is used to show hierarchy and relationship between tasks and sub-tasks.
- Hashtags are used for prioritizing, labeling, organizing, grouping, and tagging the tasks.

**CODE FOR WEBSITE:**

```
<!DOCTYPE html>

<html>

<head>

   <title>Vehicle Essentials and Navigation</title>

   <link rel="stylesheet" type="text/css" href="styles.css">

</head>

<body>

   <header>
```

```
    <h1>Welcome to Vehicle Essentials</h1>
    <nav>
        <ul>
            <li><a href="#">Home</a></li>
            <li><a href="#">Vehicles</a></li>
            <li><a href="#">Contact</a></li>
        </ul>
    </nav>
</header>

<main>
    <section>
        <h2>Featured Vehicles</h2>
        <div class="vehicle-listing">
        <h3>Vehicle 1</h3>
        <p>Details about Vehicle 1.</p>
    </div>
    <div class="vehicle-listing">
        <h3>Vehicle 2</h3>
        <p>Details about Vehicle 2.</p>
    </div>
</section>

<section>
    <h2>Search for Vehicles</h2>
    <form>
        <label for="search">Search:</label>
        <input type="text" id="search" name="search">
```

```
          <input type="submit" value="Submit">
        </form>
      </section>
    </main>

    <footer>
      &copy; 2023 YourWebsiteName
    </footer>
  </body>
</html>
```

CSS:STYLE

```
body {
   font-family: Arial, sans-serif;
}

header {
   background-color: #333;
   color: #fff;
   text-align: center;
   padding: 10px;
}

nav ul {
   list-style: none;
```

```
}

nav ul li {
   display: inline;
   margin-right: 20px;
}

main {
   padding: 20px;
}

.vehicle-listing {
   border: 1px solid #ddd;
   padding: 10px;
```

```
    margin: 10px 0;
}

footer {
    background-color: #333;
    color: #fff;
    text-align: center;
    padding: 10px;
}
```

JAVASCRIPT FOR COMPLETEING A WEBSITE

```
// Importing classes from java.servlet package
// for connectivity of application class
```

```
import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;


// Since userlogin is used in action="userlogin"

// form tag in jsp page

@WebServlet("/userlogin")


// Class 1

// Helper class extending HttpServlet main class

public class LoginServlet extends HttpServlet

{
```

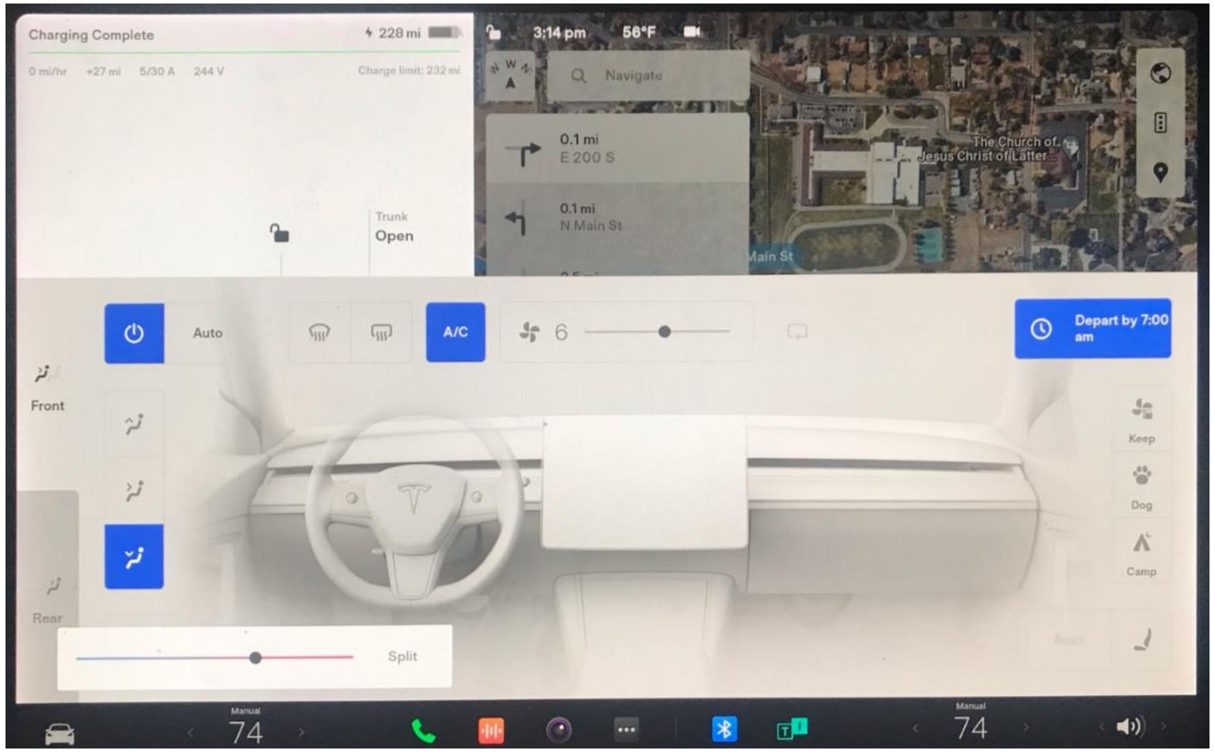
```
// Method
@Override
protected void doPost(HttpServletRequest request,
                    HttpServletResponse response)
throws ServletException, IOException
{

   response.setContentType("text/html");

   PrintWriter out = response.getWriter();

   // Customly setting name and password
   String name = request.getParameter("name");
```

```
        String password = request.getParameter("password");
        ....
        // Further programming
        // Code can be appended here onward so
    }
}
let currentIndex = 0;
const images = document.querySelectorAll('#slideshow img');

function nextSlide() {
  images[currentIndex].style.display = 'none';
  currentIndex = (currentIndex + 1) % images.length;
  images[currentIndex].style.display = 'block';
}
```

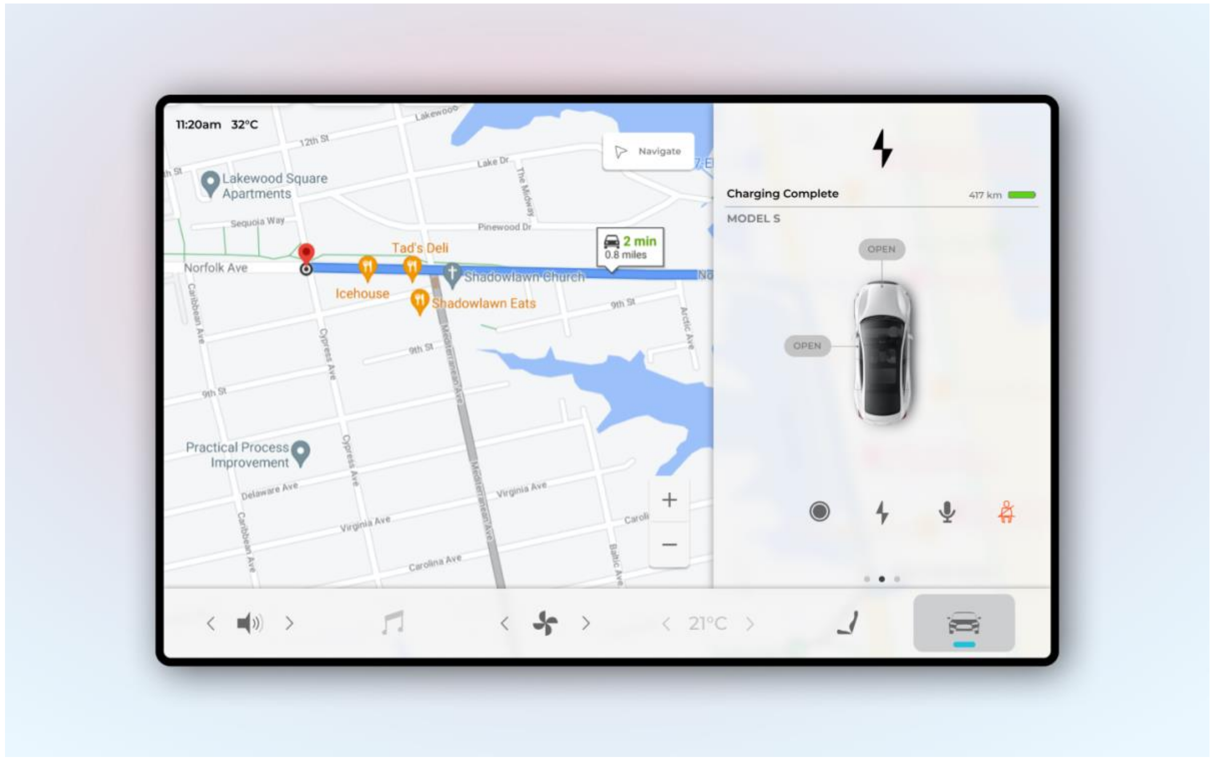
```
setInterval(nextSlide, 3000); // Change slide every 3 seconds

const form = document.getElementById('myForm');
form.addEventListener('submit', function (event) {
  const nameInput = document.getElementById('name');
  const emailInput = document.getElementById('email');

  if (nameInput.value === '' || emailInput.value === '' || !emailInput.checkValidity()) {
    alert('Please fill in the form correctly.');
    event.preventDefault();
  }
});
```

```
function changeText() {
  const demo = document.getElementById('demo');
  demo.innerHTML = 'Text has changed!';
}
```

## SAMPLE DESIGN:



SAMPLE  WEB PAGES:



Departure Time
6 AM-12 PM | 12 PM-6 PM | **6 PM-12 AM** | 12 AM-6 AM

Bus Type
**AC** | Non AC | **Sleeper** | Seater

Price
₹400

Boarding Points
5 Boarding Points Selected

Dropping Points
3 Dropping Points Selected

Bus Operators
SRS Travels, VRL travels, Neeta Volvo

Facilities
WiFi, Charging Point, TV, Water Bottle, Blanket

[✓] Non Stop Buses
[ ] Bus with live tracking

From
PUNE

To
BANGALORE

Journey Date
14 June 17, Wed

**AC** | Non AC | **Sleeper** | Seater

*THANK YOU!*