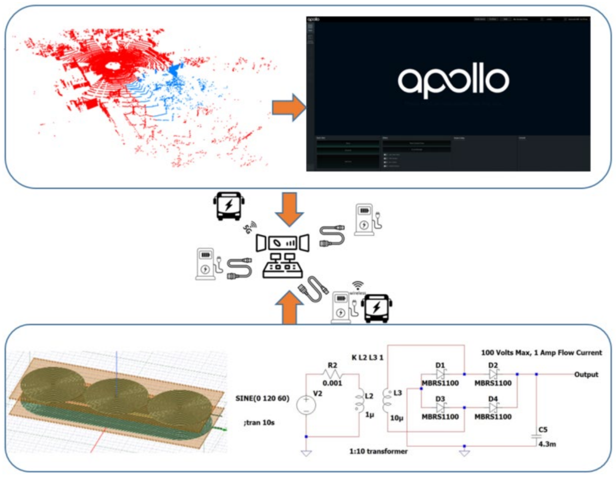
STAR In-Road Electric Vehicle Charging for Parked Vehicles



Prepared by:

Bilal Abdulhamed
Hootan Alavizadeh
Tyler Ricketts
Brandon Schneider
Dr. Hamed Attariani
Dr. Weisong Wang
Dr. Mike Saville

Prepared for:
The Ohio Department of Transportation, Office of Statewide Planning & Research

Project ID Number: *118511*

September 2023

Final Report

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.	
FHWA/OH-2023-28			
4. Title and Subtitle		5. Report Date	
STAR In-Road Electric Vehicle Charging for Parked Vehicles		**September 2023**	
		6. Performing Organization Code	
7. Author(s)		8. Performing Organization Report No.	
Bilal Abdulhamed, Hootan Alavizadeh, Tyler Ricketts, Brandon Schneider, Dr. Hamed Attariani, Dr. Weisong Wang, Dr. Mike Saville			
9. Performing Organization Name and Address		10. Work Unit No. (TRAIS)	
Wright State University, Department of Mechanical and Materials Engineering, Department of Electrical Engineering, Dayton, OH 45435			
		11. Contract or Grant No.	
		38584	
12. Sponsoring Agency Name and Address		13. Type of Report and Period Covered	
Ohio Department of Transportation 1980 West Broad Street Columbus, Ohio 43223		**Final Report**	
		14. Sponsoring Agency Code	
15. Supplementary Notes			
None			
16. Abstract			
This report documents the work conducted as a multi-disciplinary project on the wireless charging of parked Electrical Vehicles (EVs) by Wright State University and supported by the Ohio Department of Transportation (ODOT). This document summarizes our progress on a wide variety of topics, including 1) in-house software to read and label the open-source high-definition maps (HD maps), 2) design of electrical circuitry required for wireless charging, 3) finite element analysis (FEA) of electromagnetic system, 4) conceptual designs of the charging pads and stations, and 5) communication protocol.			
17. Keywords		18. Distribution Statement	
High Definition Map, Vector Map, Wireless Charging, Electromagnetic, Resonant Induction Charging Circuit, Communication Protocol		**No restrictions. This document is available to the public through the National Technical Information Service, Springfield, Virginia 22161**	
19. Security Classification (of this report)	20. Security Classification (of this page)	21. No. of Pages	22. Price
Unclassified	**Unclassified**	**40**	

Form DOT F 1700.7 (8-72) **Reproduction of completed pages authorized**

Credits and Acknowledgments Page

Prepared in cooperation with the Ohio Department of Transportation
and the U.S. Department of Transportation, Federal Highway Administration

We would like to express our sincere appreciation to Nick Hegemier, Jennifer Spriggs, and Aaron Moore from the Ohio Department of Transportation for their invaluable contributions to this project. Their expertise in high-definition maps and unwavering support have been instrumental in guiding us at every step of this endeavor.

Their dedication to moderating meetings, providing technical insights, and offering valuable sources of information has significantly enhanced the quality and scope of our work. Their commitment to our project has been a constant source of inspiration, and we are truly grateful for the generosity of their time and knowledge.

Contents

1. Problem Statement

The global electric vehicle (EV) market is forecasted to grow by 24.3% till 2028 constantly. However, the development in charging infrastructure is still lagging behind that, hindering the EV's widespread application, i.e., 30 million chargers are still needed to support the existing EV demand. Also, based on the survey by Witricity, 86% of EV owners are highly interested in wireless charging for parked vehicles, making it the 3rd most wanted feature in EVs. Here, we aim to design an over-the-pavement "inductive wireless charging ecosystem" for a fleet of autonomous electric vehicles (AEVs). The project's objectives are 1) design a wireless charging station that allows charging AEVs in the parking lot to increase efficiency, eliminate the need for individual charging wires for each EV, turn the passive parking time into a productive time, and increase the autonomy of charging process, 2) suggest the communication system between charging stations and AEVs (charger reservation system), the interaction between charger and AEVs, and safety. This proposed technology is a necessary step to implement a fleet of fully automated shuttles for the future generation of smart cities to connect people to the workplace, health centers, and recreational sites efficiently and with less carbon footprint. It also provides a safe charging technique for extreme climatic conditions (heavy snow and rain), which pose significant importance due to the climate of the state of Ohio.

2. Research Background

The main objective is to design an inductive charging station for a fleet of automated shuttles. In this scenario, an automated shuttle should be able to find an available wireless charger, reserve the charger, optimally park over the charging station, start the charging (docking), and undock after charging completion. This document summarizes our progress on a wide variety of topics, including 1) in-house software to read and label the open-source high-definition maps (HD maps), 2) design of electrical circuitry required for wireless charging, 3) finite element analysis (FEA) of electromagnetic system, 4) conceptual designs of the charging pads and stations, and 5) communication protocol.

3. Research Approach

3.1. High-Definition Maps (HD):

The main objective of this section is to label a charging station on an HD map that is used for the navigation of the Autonomous Vehicle (AV). After extensive research on various open-source HD datasets, such as KITTI [1], Sensat Urban [2], nuScenes, Toronto 3D, and Waymo, we chose the KITTI dataset to implement our labeling algorithm on the existing open-access HD Maps. The KITTI dataset is a popular computer vision dataset commonly used for object detection, tracking, and scene understanding in autonomous driving applications. It provides a comprehensive collection of sensor data recorded from real-world driving scenarios. It also includes Camera Data, LIDAR data, Radar Data, Velodyne Data, GPS/IMU Data, Object Annotation, Calibration Files, and Data Synchronization. These features make the KITTI dataset valuable for developing and evaluating computer vision algorithms for autonomous driving applications. Researchers can leverage this data to train and test their models on real-world scenarios, contributing to advancements in the field of autonomous driving. In addition, we used Open3D [3], an open-source library for point cloud data processing in Python, which helps the rapid development of software dealing with 3D data. Finally, the Apollo environment was used with our code to check our code's functionality and propose a platform.

3.2. Review of Standards and Charging Station Conceptual Design:

A comprehensive review of standards, such as ISO, IEC, UL, and SAE, was performed to understand overall recommended design parameters, e.g., dimension, frequency, and structure.

3.3. Resonant Induction Charging Circuit:

The LTSpice software models the Resonant Induction Charging Circuitry for the receiver and transmitter.

3.4. Electromagnetic Simulation:

Among various simulation software, such as MATLAB/SIMULINK, EM work, SPICE, and ANSYS MAXWELL/SIMPLORER. We chose ANSYS as the most suitable simulator for this project due to its extensive application in academia and industry for evaluating the design and performance of coils. ANSYS Maxwell can solve physics, including frequency-domain and time-varying magnetic and electric fields.

3.5. Communication:

ISO 15118 was selected as the best communication protocol.

4. Research Findings and Conclusions

4.1. High-Definition Maps (HD):

Here, we have developed an efficient algorithm that enables the labeling and classifying of points based on arbitrary latitude and longitude coordinates. By taking the desired location's latitude and longitude as input, our algorithm accurately determines and highlights (labels) the precise location. This objective is achieved through a conversion process that transforms the latitude and longitude values into cartesian coordinates (x, y, and z), allowing for calculating relative distances in meters. The Google Colab was used for coding, an online platform provided by Google that allows users to run and execute Python code entirely on the cloud. Google's infrastructure provides all the computational resources, including CPU, GPU, and RAM. It also provides a Jupyter Notebook interface, allowing users to create and edit notebooks containing executable code, rich text, images, and visualizations. Notebooks are organized into cells, making running code interactively and document work easy. In the next step, we modified our code to be compatible with the Apollo file format by reverse engineering Apollo maps and exploring methods of extracting/labeling point clouds in Apollo. Finally, different approaches were suggested to create an Apollo-compatible dataset based on the marked location of a wireless charging station to simulate navigation/driving in the Apollo environment.

4.1.1. Code Description

To work with 3D data, we installed the Open3D Python library using the pip install command. This library provides the tools and functionality to handle and manipulate 3D data effectively.

```
!pip install open3d
```

The command below downloads and extracts the KITTI dataset's data, such as point clouds, metadata, images, and other information related to cameras and calibrations.

```
!wget https://s3.eu-central-1.amazonaws.com/avg-kitti/raw_data_downloader.zip
!unzip raw_data_downloader.zip
!chmod +x raw_data_downloader.sh
!./raw_data_downloader.sh
```

The data structure of downloaded files is shown in the Fig. 1.

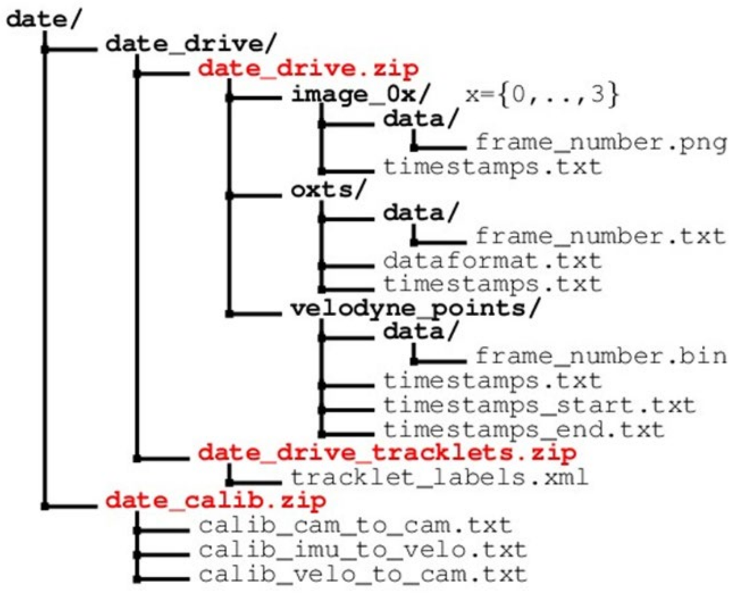


Figure 1. KITTI dataset structure

The point cloud data are in the "velodyne_points" folder, saved in different timeframes. Furthermore, essential metadata information, including GPS/IMU data, camera data, and other supplementary information, is stored within the "oxts" folder (Figure 1).
Our algorithm operates by iterating through the timeframes within the KITTI dataset folders. It searches for the point cloud file that has the closest distance to the ***user-provided input*** in latitude and longitude format. This process allows us to identify and retrieve the most relevant point cloud data corresponding to the specified location.

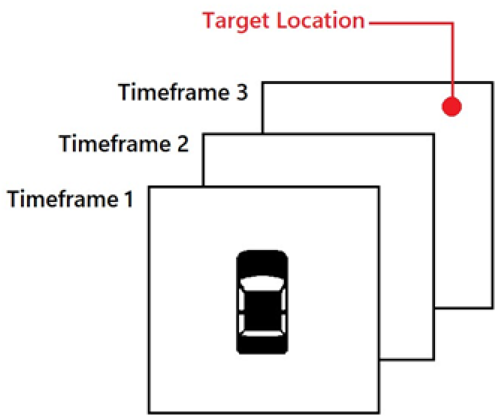


Figure 2. Schematic of different timeframes in the KITTI dataset.

The code snippet below retrieves the user target coordinates in latitude and longitude format and stores them in the "target" variable.

```
target_lat = 49.012605023656
target_lng = 8.41632208753425
```

Then, the necessary libraries are imported using the following commands. We have included the pyplot library to facilitate rendering the results in Colab.

```
import os
import open3d as o3d
import open3d.visualization.rendering as rendering
from matplotlib import pyplot
```

Then, we defined a function called "read_metadata" to extract the metadata information from each timeframe stored in the KITTI dataset. This function allows us to retrieve details related to each timeframe, such as latitude and longitude.

```
def read_metadata(url):
print(f'url: {url}')
with open(url, 'r') as f:
```

```
cam_info = f.readline().split(' ')
lat, lng, alt = [float(item) for item in  cam_info[:6]]
return lat, lng, alt
```

Next, we have defined the "calculate_distance" function to compute the distance between two geographical coordinates. This function operates by converting the latitude and longitude points from degrees to radians. Subsequently, it utilizes the Earth's radius to calculate the distance, employing the Haversine distance and Angular distance formulas. The output of this function is returned in kilometers.

```
from math import sin, cos, atan2, sqrt
def calculate_distance(input_lat, input_lng, camera_lat, camera_lng):
# calculation of distances and coordinates on the Earth's surface
pi = 3.14159265359

# Convert the latitude and longitude coordinates of points from
degrees to radians
lat_src = camera_lat * pi / 180
lng_src = camera_lng * pi / 180

lat_dst = input_lat * pi /180
lng_dst = input_lng * pi /180

delta_lat = lat_dst - lat_src
delta_lng = lng_dst - lng_src
# Calculate the Haversine distance
a = (sin(delta_lat/2)**2) + cos(lat_src) * cos(lat_dst) *
(sin(delta_lng/2)**2)
# print(f'a={a}')

# Calculate the angular distance in radians
c = 2 * atan2(sqrt(a), sqrt(1-a))
# print(f'c={c}')

# Calculate the distance using the Earth's radius
# R: Radius of the Earth
R = 6371.07103
distance = R * c

return distance
```

The code snippet below runs through each folder in the KITTI dataset, utilizing the functions we defined earlier to identify the point cloud file with the minimum distance to the input geographical coordinates.

```
import os
import pandas as pd
import warnings
warnings.filterwarnings("ignore")

path = '/content/kitti'
```

```
show_output = False
list_of_files = {}
df = pd.DataFrame()

for (root, dirnames, filenames) in os.walk(path):
cur_depth = root.count(os.path.sep)

if cur_depth == 3:
for dirname in dirnames:
print(dirname)
timeframes = sorted(os.listdir(os.path.join(root, dirname,
'oxts/data')))
for timeframe in timeframes:
if show_output:
print(f'timeframe: {timeframe} \t '+ os.path.join(root,
dirname, 'oxts/data', timeframe))

lat, lng, alt =
read_metadata(os.path.join(root, dirname, 'oxts/data',
timeframe))
if show_output:
print(f'lat: {lat} lng: {lng} alt: {alt}', lat, lng, alt)

d = calculate_distance(target_lat, target_lng, lat, lng)
print(f'distance: {d}')
data = {
"distance": d,
"file": os.path.join(root, dirname, 'oxts/data', timeframe)
}
df = df.append([{'distance': d, 'file': os.path.join(root,
dirname, 'oxts/data', timeframe), 'path': os.path.join(root,
dirname), 'frame': timeframe[:-4]}])
```

Then, the results are stored in the following variables for subsequent usage. The variable "distance_to_point" represents the distance to the target point in meters. Similarly, "target_file" corresponds to the point cloud file we identified with the closest distance to our desired location. However, we still need to calculate the relative distance to determine the precise location of the point we intend to classify.

```
target_file = df.iloc[0]['file']
target_path = df.iloc[0]['path']
target_frame = df.iloc[0]['frame']
distance_to_point = df.iloc[0]['distance'] * 1000
target_file
```

We needed to define a function that converts spherical coordinates to Cartesian coordinates to calculate the relative distance. The function below inputs latitude and longitude and converts them to the x, y point format.

```
def convert_spherical_to_cartesian(latitude, longitude):
pi = 3.14159265359
#Convert from Degrees to Radians
latRad = latitude * (pi)/180
lonRad = longitude * (pi)/180
R = 6371.07103
earthRadius = R
posX = earthRadius * cos(latRad) * cos(lonRad)
posY = earthRadius * cos(latRad) * sin(lonRad)
return posX, posY
```

Then, by calling this function, we can obtain the x-y coordinates of the camera and target points. These coordinates allow us to calculate the relative distance to the target point.

```
camera_x, camera_y = convert_spherical_to_cartesian(current_camera_lat, current_camera_lng)
target_x, target_y = convert_spherical_to_cartesian(target_lat, target_lng)

relative_target_x = camera_x - target_x
relative_target_y = camera_y - target_y

# convert to meter
relative_target_x = relative_target_x * 1000
relative_target_y = relative_target_y * 1000
```

Finally, once we have determined the relative_target_x and relative_target_y coordinates, we can utilize the methods provided by the Open3D library to crop and segment the points. This function will allow us to extract the points within a 15-meter radius of the target point.

```
pcd = o3d.io.read_point_cloud(target_pointcloud_file)

CUBOID_EXTENT_METERS = 30
METERS_BELOW_START = 1
METERS_ABOVE_START = 20

def get_cuboid_points(start_position):
return np.array([
[start_position['x'] + (CUBOID_EXTENT_METERS / 2),
start_position['y'] + (CUBOID_EXTENT_METERS / 2),
start_position['z'] + METERS_ABOVE_START],
[start_position['x'] - (CUBOID_EXTENT_METERS / 2),
start_position['y'] + (CUBOID_EXTENT_METERS / 2),
start_position['z'] + METERS_ABOVE_START],
[start_position['x'] - (CUBOID_EXTENT_METERS / 2),
start_position['y'] - (CUBOID_EXTENT_METERS / 2),
start_position['z'] + METERS_ABOVE_START],
[start_position['x'] + (CUBOID_EXTENT_METERS / 2),
start_position['y'] - (CUBOID_EXTENT_METERS / 2),
start_position['z'] + METERS_ABOVE_START],
# Vertices Polygon 2
[start_position['x'] + (CUBOID_EXTENT_METERS / 2),
start_position['y'] + (CUBOID_EXTENT_METERS / 2),
start_position['z'] - METERS_BELOW_START],
[start_position['x'] - (CUBOID_EXTENT_METERS / 2),
start_position['y'] + (CUBOID_EXTENT_METERS / 2),
start_position['z'] - METERS_BELOW_START],
[start_position['x'] - (CUBOID_EXTENT_METERS / 2),
start_position['y'] - (CUBOID_EXTENT_METERS / 2),
start_position['z'] - METERS_BELOW_START],
[start_position['x'] + (CUBOID_EXTENT_METERS / 2),
start_position['y'] - (CUBOID_EXTENT_METERS / 2),
start_position['z'] - METERS_BELOW_START],
]).astype("float64")
points = np.array(mesh_sphere.vertices).reshape([-1, 3])

point_cloud = o3d.geometry.PointCloud()
point_cloud.points = o3d.utility.Vector3dVector(points)

start_position = {'x': relative_target_x, 'y': relative_target_y, 'z': -1.9}
cuboid_points = getCuboidPoints(start_position)

points = o3d.utility.Vector3dVector(cuboid_points)
oriented_bounding_box = o3d.geometry.OrientedBoundingBox.create_from_points(points)
point_cloud_crop = point_cloud.crop(oriented_bounding_box)



inliers_indices = oriented_bounding_box.get_point_indices_within_bounding_box(pcd.points)

inliers_pcd = pcd.select_by_index(inliers_indices, invert=False)
outliers_pcd = pcd.select_by_index(inliers_indices, invert=True)
```

The code snippet below visualizes the inlier and outlier data points. In this implementation, the inliers represent the points we have cropped and are about to be classified.

```
bounding_box = inliers_pcd.get_axis_aligned_bounding_box()
bounding_box.color = (1, 0, 0)
o3d.visualization.draw_geometries([pcd, inliers_pcd, bounding_box])
```

The labeled results are shown in Figure 3.

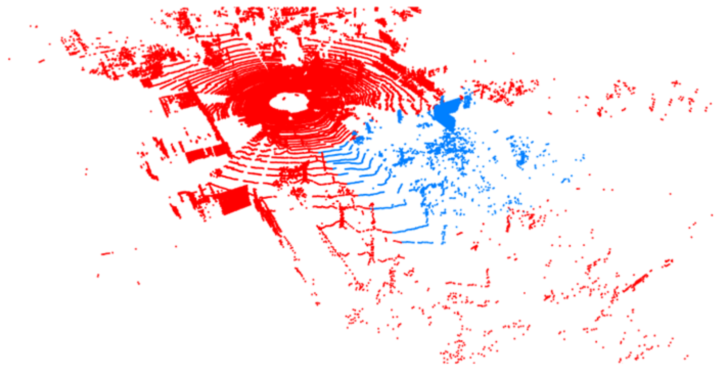


Figure 3*. Inliers and outliers*

In this step, we assign labels to the points divided into inliers and outliers. Subsequently, we concatenate these labeled points.

```
# inliers (marked location)
labels_column                                                                    =
np.ones(np.asarray(inliers_pcd.points).shape[0]).reshape(np.asarray(inliers_pcd.points).shape[0],
1)

inliers_numpy = np.asarray(inliers_pcd.points)
all_classified_data = np.append(inliers_numpy, labels_column, 1)

# outliers
labels_column                                                                    =
np.zeros(np.asarray(outliers_pcd.points).shape[0]).reshape(np.asarray(outliers_pcd.points).shape[
0], 1)
outliers_numpy = np.asarray(outliers_pcd.points)
all_unclassified_data = np.append(outliers_numpy, labels_column, 1)

all_data = np.concatenate((all_classified_data, all_unclassified_data), axis=0)
```

As a result, we can save the concatenated points, including the labeled points, to a single PCD file named "pointcloud.pcd" compatible with various commercial and open-source software like Cloud Compare.

```
xyzi = all_data
xyz = xyzi[:,0:3]
i = [[i] for i in xyzi[:,3]]

pcd_out = o3d.t.geometry.PointCloud()
pcd_out.point["positions"] = o3d.core.Tensor(xyz)
pcd_out.point["classification"] = o3d.core.Tensor(i)

o3d.t.io.write_point_cloud("pointcloud.pcd", pcd_out)
```

4.1.2. Integrating Developed Code for Labeling with Apollo Software

Apollo Auto, developed by Baidu, Inc., is a cutting-edge and comprehensive platform for autonomous driving and intelligent transportation solutions. This innovative technology suite is at the forefront of the global autonomous vehicle industry, offering a wide range of software and hardware solutions designed to enable safe, efficient, and autonomous mobility.
Apollo Auto represents Baidu's ambitious vision to transform how we move and interact with transportation. As one of the world's leading technology companies, Baidu has leveraged its artificial intelligence (AI), machine learning, and data analytics expertise to create a robust ecosystem that powers self-driving vehicles and intelligent transportation systems.
This platform has gained significant attention and acclaim in the automotive and tech industries for its open-source approach, fostering collaboration among stakeholders, including automakers, hardware manufacturers, software developers, and researchers. Apollo Auto's open platform provides the tools and resources necessary to accelerate the development and deployment of autonomous vehicles, making it a pivotal player in shaping the future of mobility.
In this section, we changed our labeling code to be compatible with Apollo's file format by reverse engineering an existing Apollo map and exploring methods of extracting point clouds.

4.1.3. Apollo Installation

Before installing Apollo [4], some prerequisites, such as Ubuntu Linux, NVIDIA GPU Driver, Docker Engine, and NVIDIA Container Toolkit, must be installed.
After installing prerequisites, one needs to run the command below to clone the repository

```
# Using SSH
git clone git@github.com:ApolloAuto/apollo.git

# Using HTTPS
git clone https://github.com/ApolloAuto/apollo.git
```

The command below will define APOLLO_ROOT_DIR environment variable to the root directory.

```
echo "export APOLLO_ROOT_DIR=$(pwd)" >> ~/.bashrc  && source ~/.bashrc
```

The next step is starting the Apollo development docker container by entering the command below in the Linux terminal (root directory).

```
bash docker/scripts/dev_start.sh
```

After starting the Apollo development container, the command below should be run to login into the newly started container.

```
bash docker/scripts/dev_into.sh
```

In this step, Apollo needs to be built by running this command inside the Apollo docker container.

```
./apollo.sh build
```

After a successful build by running this command, the Apollo Dreamview backend will be started.

```
./scripts/bootstrap.sh [start | stop | restart]
```

Then, by opening http://localhost:8888 in the browser, the Apollo starting menu should pop up, as shown in Figure 4.

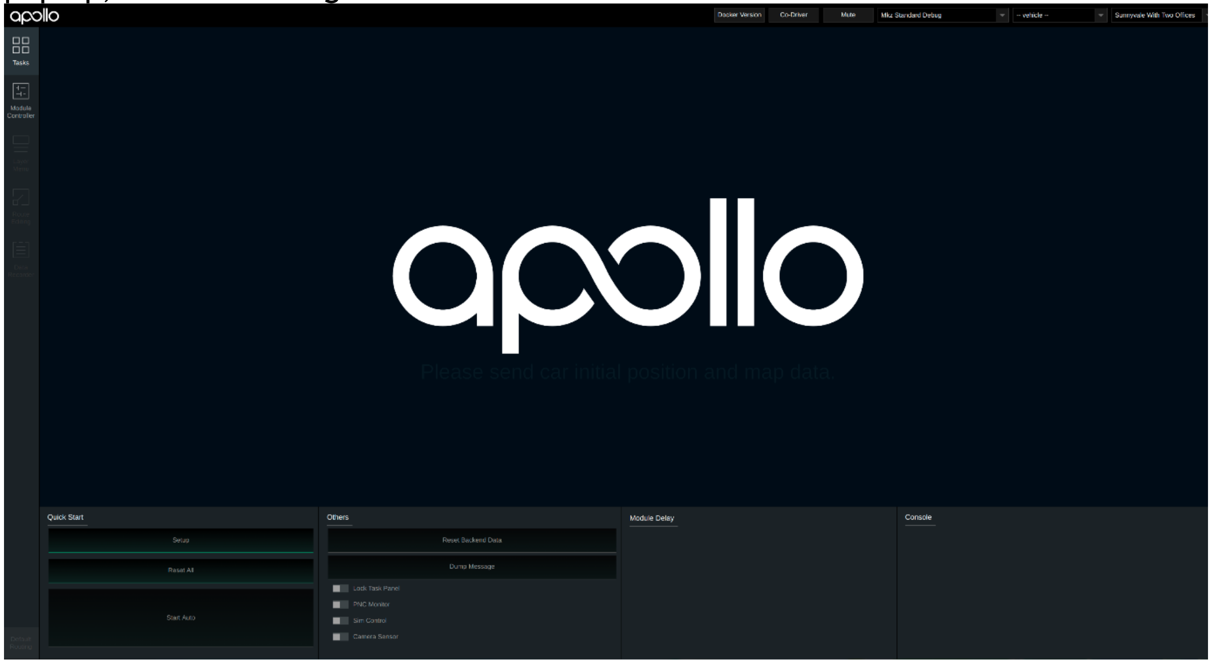


Figure 4. *The Apollo starting menu*

4.1.4. Point Cloud Extraction in Apollo

Point cloud extraction in Apollo involves several steps, which will be discussed below.

1- First, the sensor data needed to be collected through the command below.

```
cyber_recorder record -c imu_topic localization_pose_topic lidar_topic
```

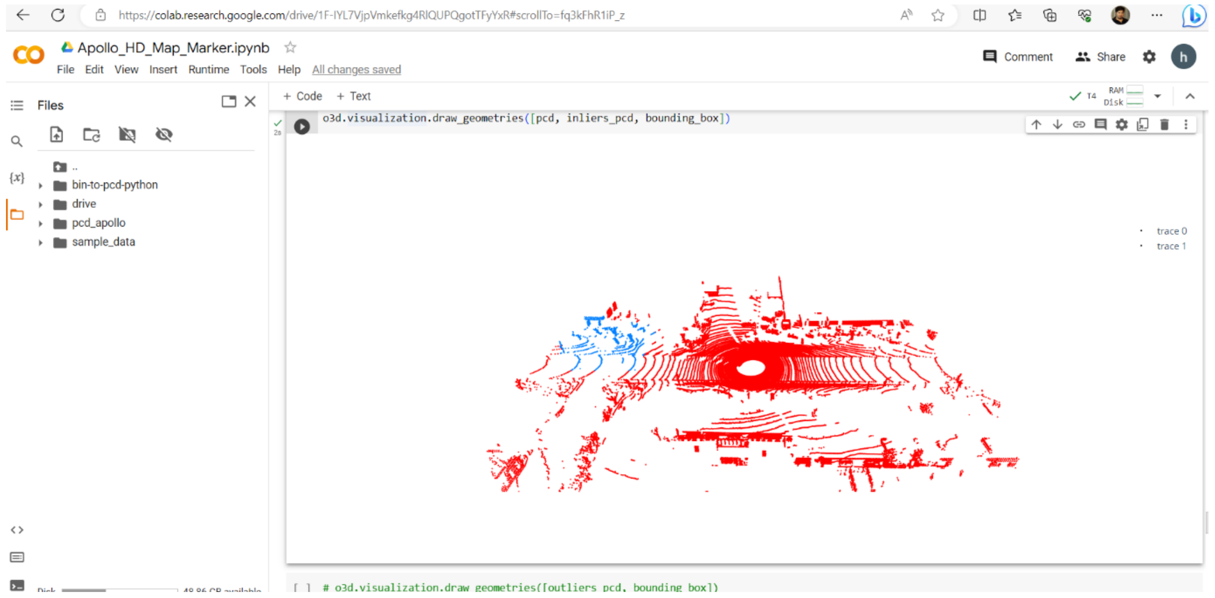
2- Next, decompress the record file in Apollo to get the pcd results:

```
./bazel-bin/modules/localization/msf/local_tool/data_extraction/cyber_record_parser --bag_file=data/bag/demo_sensor_data_for_vision.record --out_folder=data --cloud_topic=/apollo/sensor/velodyne64/compensator/PointCloud2
```

We used "cyber record parser" to parse and save information about fusion, GNSS, lidar, odometry location, point clouds, etc., to generate output pcd files. ***This tool will generate pcd outputs in timeframes***. Therefore, to get the final point cloud map, first, we need to run pose interpolator command as below.

```
./bazel-bin/modules/localization/msf/local_tool/map_creation/poses_interpolator --input_poses_path=data/pcd/odometry_loc.txt --ref_timestamps_path=data/pcd/pcd_timestamp.txt --extrinsic_path=modules/localization/msf/params/velodyne_params/velodyne64_novatel_extrinsics_example.yaml --output_poses_path=data/pcd/poses.txt
```

3- Then we interpolated the position according to the external parameters and time-stamp of the lidar. The corrected positions will be saved in –output_poses_path.


https://colab.research.google.com/drive/1F-IYL7VjpVmkefkg4RIQUPQgotTFyYxR#scrollTo=fq3kFhR1iP_z
Apollo_HD_Map_Marker.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
bin-to-pcd-python
drive
pcd_apollo
sample_data
+ Code
+ Text
T4
RAM
Disk
trace 0
trace 1

```
o3d.visualization.draw_geometries([pcd, inliers_pcd, bounding_box])
```

```
# o3d.visualization.draw_geometries([outliers_pcd, bounding_box])
```

Figure 5*. Labeled cloud point data*

4- Finally, we can see the output result of the classified Apollo map after running our map marker algorithm (Figure 5). Additionally, the ndt_mapping technique, a localization technique based on the normal distributions transform algorithm, can make preloaded maps. NDT aligns a real-time LiDAR point cloud with a high-definition map represented as a 3D point cloud. This method suits environments where LiDAR data is reliable and readily available.

4.1.5. Road Network

An HD map can generally be separated into point-cloud and vector maps. A vector map, also known as a road network map, represents the geo-referenced position of the objects of interest in a driving environment (lanes, traffic signs, lights, etc.) with points, lines, and polygons and is used for strategic planning (navigation). Implementing HD map navigation and path-finding involves several major steps: Acquiring HD Map Data, Map Data Processing and labeling, localization, Route Planning, Map Matching, Turn-by-Turn Navigation, and Real-Time Updates. A brief description of each step is covered in the following section.

4.1.6. Vector Map

In order to generate the vector map, usually machine learning methods are used to extract road network semantic information (e.g., road types, lane information, speed limits, traffic signals and stop signs, turn restrictions, pedestrian crosswalks, bicycle lanes and paths, public transportation stops, land marks and points of interest, traffic flow and congestion, accident and incident report, road surface condition, elevation, and terrain information). Fig. 6 shows the point cloud map and associated vector map with road network semantic information [6] to illustrate the concept.

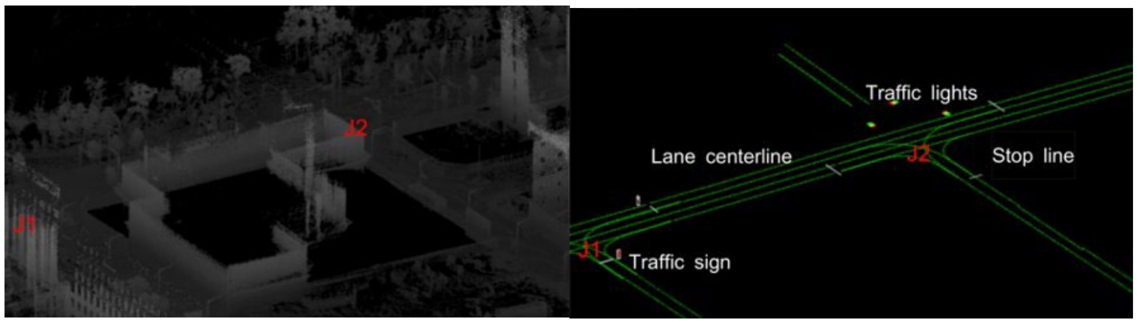


Figure 6. *Point cloud database and corresponding vector map between two points, J1 and J2, adapted from [6].*

However, creating a vector map is a manual digitization process and takes time, so researchers are trying to find methods to automate this task using Graph Neural Networks for Vector Map Automation.
VectorMapNet [7] proposes a neural network architecture that directly generates vectorized HD maps from raw sensor data and eliminates the need for manual map creation and post-processing steps. The "end-to-end" nature suggests that the entire process, from input sensor data to the output vectorized HD map, is integrated into a single learning framework. ***More importantly, the "class labels" can be automatically assigned with this program, e.g., charging stations.*** Fig.7 shows the architecture of this end-to-end algorithm.

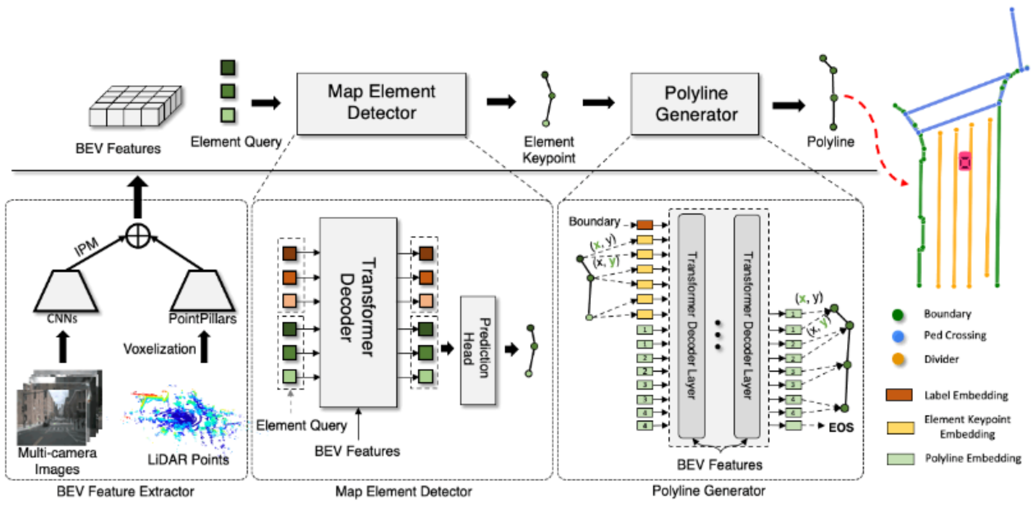


Figure 7*. The pipeline of VectorMapNet includes a feature extractor, map element detector, and polyline generators [7].*

4.1.7. Vector Map Tools

We review other vector map generation tools here as viable options. The Vector Map Tools usually input Point Cloud Data (pcd) and generate vector maps with road and lane information as output. Some vector map creation tools that can be used to generate vector maps are listed below:

- **MapToolbox (Autoware AI):** This unity plugin gets pcd files as input and help users generate output vector map.
- **Vector Mapper [8]:** This is a free online tool; it requires a Point Cloud Data file, and lanes must be drawn manually point by point, which could be time-consuming.
- **Simple Vector Mapper Tool (SVMT)[9]:** It is the recommended software to generate vector maps for Autoware. Autoware is a driving platform developed on ROS (Robot Operating System). SVMT is a program that helps to create vector maps, especially for path planning in Autoware software.

4.1.8. Route Planning

The next step after generating the vector map is route planning, which relies on vector maps to determine the optimal path from a starting point to a destination. Route planning involves several steps.

- **Define the graph:** The vector map serves as the basis for creating a graph representation of the road network. Nodes represent key points (e.g., intersections), and edges represent road segments between them.

- **Define cost metrics:** Cost metrics are used to evaluate different routes based on factors like distance, traffic, and road conditions. These metrics help in selecting the most suitable path.

- **Apply path-finding algorithms:** Path-finding algorithms, such as Dijkstra's algorithm or A* search, are applied to the graph to determine the optimal route from the current position to the destination.

4.1.9. Map Matching

Map Matching is the process of aligning sensor data from the Vehicle with the vector map to determine the Vehicle's precise position on the road. This process involves multiple steps:

- **Sensor data collection:** Data from sensors, such as GPS, IMU, and odometry, is collected to track the Vehicle's movement.
- **Preprocess sensor data:** Raw sensor data is processed and filtered to remove noise and inaccuracies.
- **Map matching algorithm:** A map matching algorithm compares the sensor data to the vector map and estimates the Vehicle's position on the road.
- **Position estimation:** The algorithm accurately estimates the Vehicle's position, which is crucial for navigation and control.

4.1.10. Turn-by-Turn Navigation

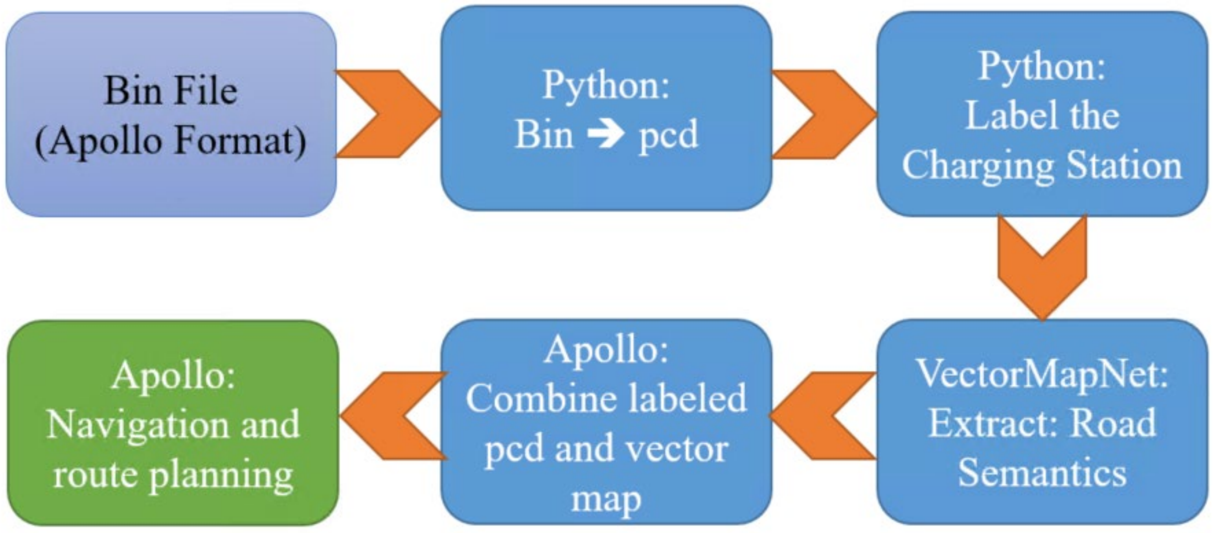
Turn-by-turn navigation takes the information from the route planning and map-matching processes to guide the driver or autonomous Vehicle through the selected path. It involves:

- **Maneuver detection:** Identifying upcoming maneuvers, such as turns, merges, or exits, based on the selected route and the Vehicle's current position
- **Maneuver instructions:** Providing clear and timely instructions to the driver or autonomous system on executing each maneuver.

4.1.11. A Proposed Flowchart for Labeling and Navigation of/to Charging Stations:

We propose the following flowchart (Fig. 8) for labeling an existing location as a charging station on Apollo HD maps and modeling autonomous vehicle navigation. The point clouds will be extracted from bin files using an existing Python extension code (bin.to.pcd). The generated PCD file can be read with our labeling Python code to mark the location of the charging station. In the next step, the marked point cloud file is imported into the "VectorMapNet" software to extract road semantic information, including the road map network.
In the final step, these files will be combined to generate a bin file that includes all labels and road information. This file can be opened within the Apollo environment

(Apollo Dream View) to simulate the navigation based on our created map. Therefore, we are generating HD maps compatible with Apollo, which has modules for navigation and path-finding algorithms. To change the default map and send commands to Apollo Dream View, we can also utilize the LGSVL Simulator API in Python.



Bin File (Apollo Format)
Python: Bin ➔ pcd
Python: Label the Charging Station
VectorMapNet: Extract: Road Semantics
Apollo: Combine labeled pcd and vector map
Apollo: Navigation and route planning

Figure 8*. The flowchart of the proposed approach for labeling charging stations on existing Apollo datasets and generating the roadmap network for navigation.*

4.2. Review of Standards and Charging Station Conceptual Design:

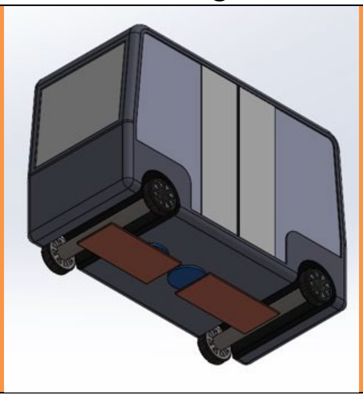
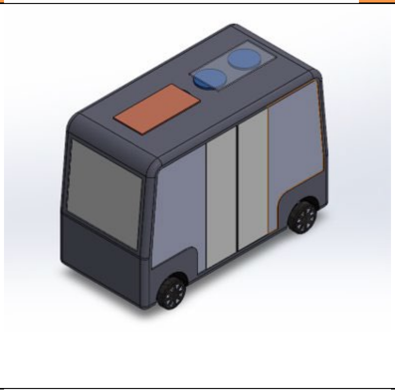
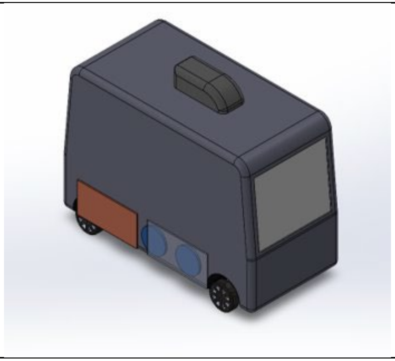
4.2.1. Design Criterion from Standards:

After reviewing the standards SAE J2954, SAE J2845/6202009, IEC TS61980-2, IEC 61980-3, ISO 19363, UL 9741, SAE J2836/6_201305, and SAE J1773, we have gathered information about the design criteria for the charging station, e.g., frequency, structure, and dimensions of transmitter/receiver pads. The SAE standard J2954-2 provides the overall dimensions of the pads for heavy-duty (HD) and Fully electrified HD Vehicles based on the available floor pan. The focus was on electrified HD Vehicles because of a particular interest in electrified autonomous minibus. Due to copyright issues, the recommended dimensions and structure of transmitter/receiver pads from SAE J2954 can not be added to the public report.

4.2.2. Conceptual Designs:

Four conceptual designs are proposed for various transmitter/receiver configurations, and their advantages and disadvantages are summarized in Table 1. The first design (highlighted in green) was selected as the final design because of its straightforward method of maintaining a constant gap between the transmitter and receiver, leading to higher magnetic coupling among coils. However, this design can also customize the number of transmitting and reviving pads. Other methods, such as adding pads to the front or side of the vehicle, raise concerns about maintaining a constant distance between the transmitter and the receiver.

Table 1. Decision Matrix for placement of Transmitter and Receivers.

Pros	Cons	Design
• A fixed receiver-to-transmitter distance • High efficiency • A lower loss of the electromagnetic field • Allows for multiple receivers for a single transmitter. • Doesn't need to be parked accurately.	• Affected by the weather • Potential damage due to snow plowing, etc.	
• Unaffected by the weather • A fixed distance between the transmitter and the receivers. • Allows for multiple receivers for a single transmitter. • Doesn't need to be parked accurately.	• Expensive to create and produce/May not have the room at the location • Requires extra wiring or electronics to run the power from the top of the bus to the bottom. • Needs additional shelter	
• Harder for the weather to affect the design • Can have multiple receivers to a single transmitter • Could charge more than one bus at a time/Use up less space	• Distance is not fixed, so some receivers may lack power transfer • Harder to align the bus with chargers unless going in a straight line.	

• Harder for the weather to affect the design • Can have a variable distance between the receiver and transmitter • Compact and simple design.	• The size constraint of the receiver/May only allow for a single receiver • It would require backing out/ pulling out depending on how the bus was parked. • Variable distance, If not parked close enough, can lose efficiency.	

Finally, Figure 11 illustrates the CAD models of the initial design. A long elliptical charging pad along a wall-mounted or pole-mounted control box will be used. Three receiver coils will be installed on the Vehicle. This design is based on the electromagnetic simulation presented in Section 4.

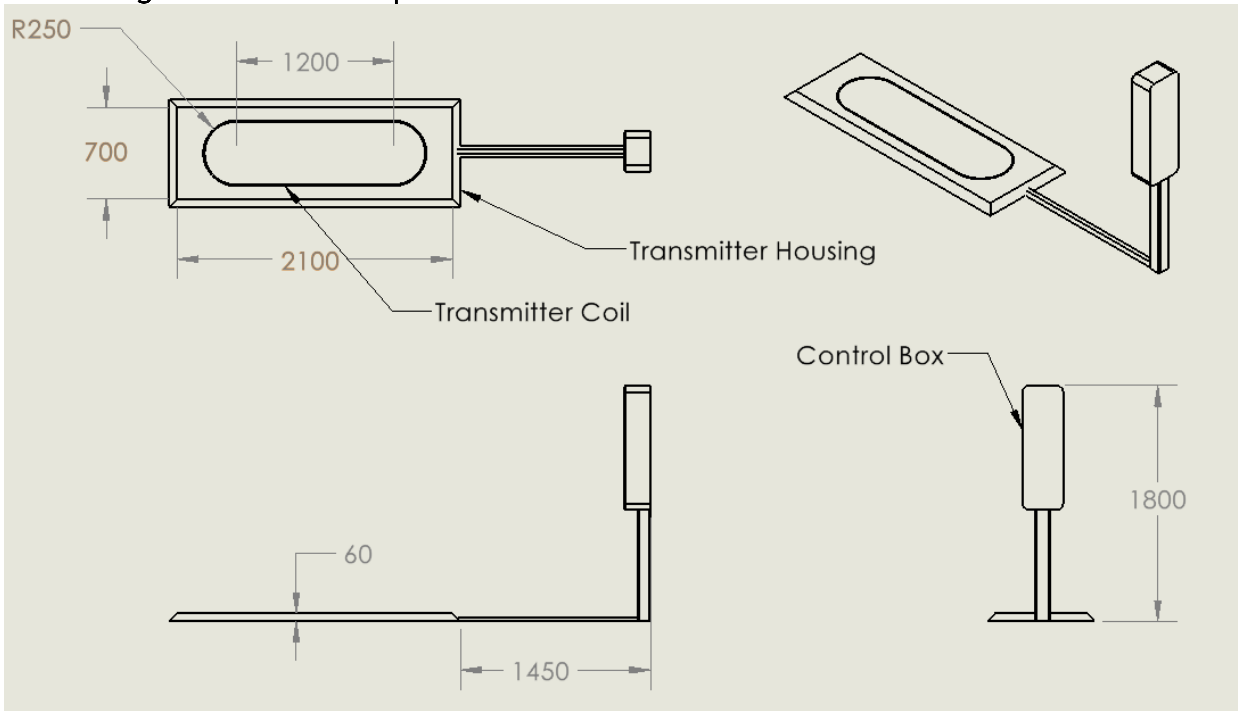


Figure 9. *The proposed CAD models for the initial design of the charging station.*

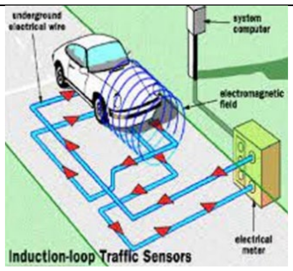
4.2.3. Receiving Pad (Vehicle Attached (VA)) and Transmitter Pad (Ground Attached (GA)) Designs

In this section, we focus on the dimensions and structure of the receiver and transmitter pads based on SAE standards. The receiving pad, also known as the Vehicle Attached pad (VA), is mounted underneath the Vehicle. It uses the coils to receive the induction waves and convert them into power for the batteries. The SAE J2954 provides the dimensions and structure of the VA, although we cannot report it here due to copyright issues.

4.2.4. Parking Spot Reservation System:

We explored various existing technologies for parking spot reservations. Table 2 shows the pros/cons for potential technologies to be integrated into our system for parking reservations. After concluding the pros and cons of each option, the G5 wireless was the best solution for reserving parking spaces (highlighted in green). The G5 combines the advantages of the induction loop and the Lidar sensors. Using two detection types, we can differentiate the vehicles from other objects. The data they collect can be used to reserve a spot in planning software. These devices use LTE communications, meaning they are extremely easy to integrate with other systems.

Table 2. Decision Matrix for Parking Spot Reservation System

Description	Pros	Cons	Design
Weight Scale	• Can be scaled to vehicle size • Easy set up	• Must be above ground • Will not be able to pick up smaller objects	
Induction Loop	• Wires can be customized to size of parking spot • In ground and weather resistant • Technology is used at some stoplights	• Will have interference from charger • Parking lot will need cut into to place wires	 underground electrical wire system computer electromagnetic field electrical meter Induction-loop Traffic Sensors
Lidar Sensor	• Can pick up any object • Can use a central computer to control many units • Not effected by weather conditions • Small footprint	• Must be above ground	
G5 Wireless	• Uses micro radar and magnetic field sensing. • Small footprint • Long battery life	• Must be above ground	

	• Takes advantage of LTE, a communication standard we can use		

4.3. Resonant Induction Charging Circuit

Circuit designs were started by researching scientific publications and standard documents over wireless vehicle charging and simulation testing using the LTSpice software. The resonant induction charging system is selected as the charging circuit. Figure 10 shows the components needed to implement inductive charging systems.

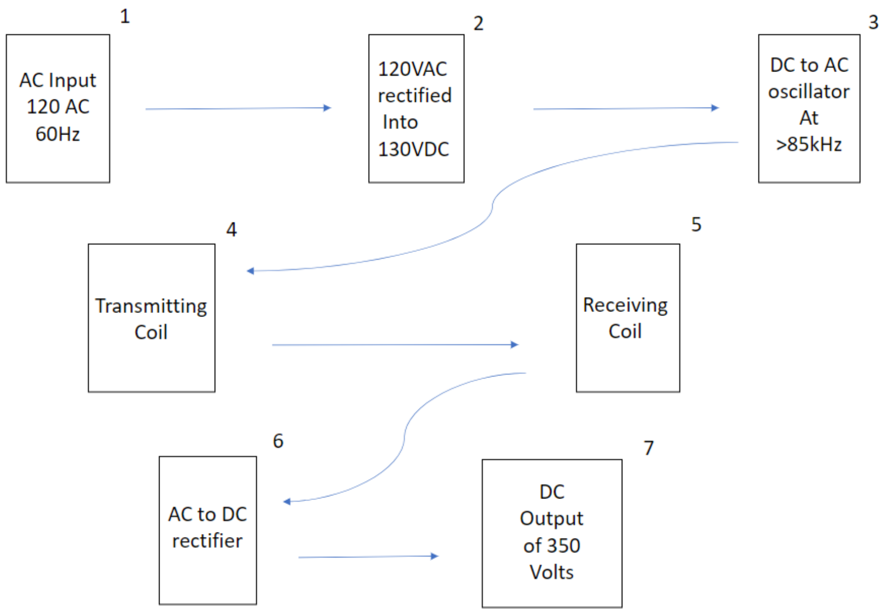


***Figure 10**. Grid power to battery process.*

The resonant induction charging enables a more efficient transfer of electricity through the air without human interaction. Here, we design a circuitry for transmitter and receiver pads. The input is a 120V AC with a frequency of 60 Hz, rectified into a DC voltage with a simulated rectifier design. The input current goes into the oscillator, producing an average of 65 volts AC with 85 kHz. It is known that the higher the frequency, the more efficient the transfer of electricity. However, we chose the frequency of 85 kHz based on the SAE-J2954 standard. This concept has been confirmed through research papers and our simulations with the LTSpice software. According to the SAE J2954, we can obtain the highest frequency allowed, 85 kHz, with a deviation of 50 Hz. After electricity is transferred, AC voltage will be converted back into DC

voltage, which will then get boosted to the desired voltage of 350 V DC, which can then be stored in the EV's battery, Fig. 10.
The current design of the rectifier is obtained by using a transformer with a 1:10 ratio to step up the voltage before it goes through diodes, which convert the AC voltage to DC voltage capable of producing over 100 Volts of DC, Fig. 11. Other rectifiers designs include various types of diodes to achieve the same effect but lack a clean, constant flat DC waveform. The use of the transformer allows for the DC voltage to not vary in frequency and allows for a more predictable output.

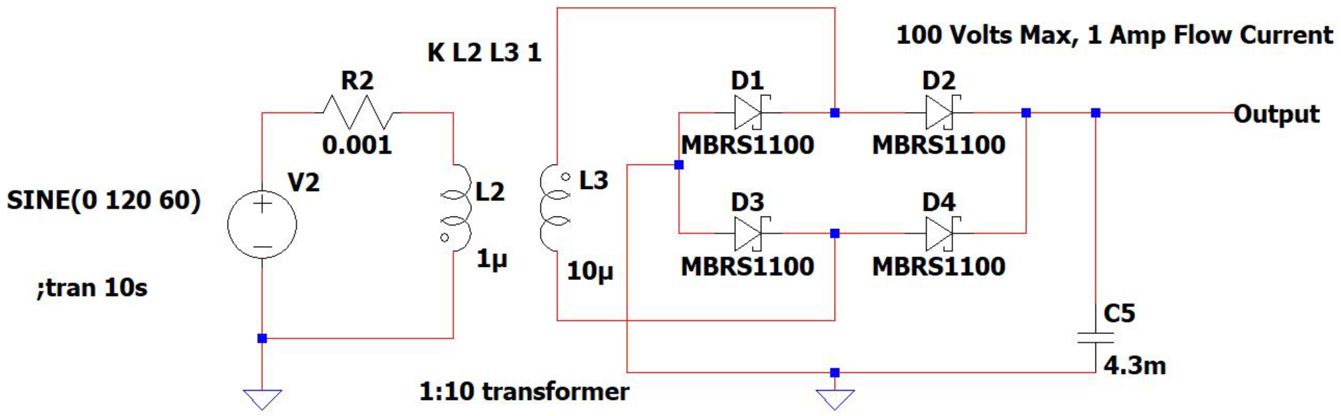


Figure 11. *Rectifier taking 120 V AC at 60 Hz and turning it to 100 V DC*

The oscillator is based on a crystal oscillator. It is constructed with two parallel MOSFETs, forming a loop with an 18 MΩ resistor and a feedback loop consisting of multiple inductors and capacitors, which allows the system to create a high-frequency waveform. The feedback loop creates the oscillations in the voltage while the MOSFET acts as a timer, turning on and off the system at a very high speed. This system allows an input of 100 volts DC and can create an average of 65 volts AC at 85 kHz, Fig. 12.

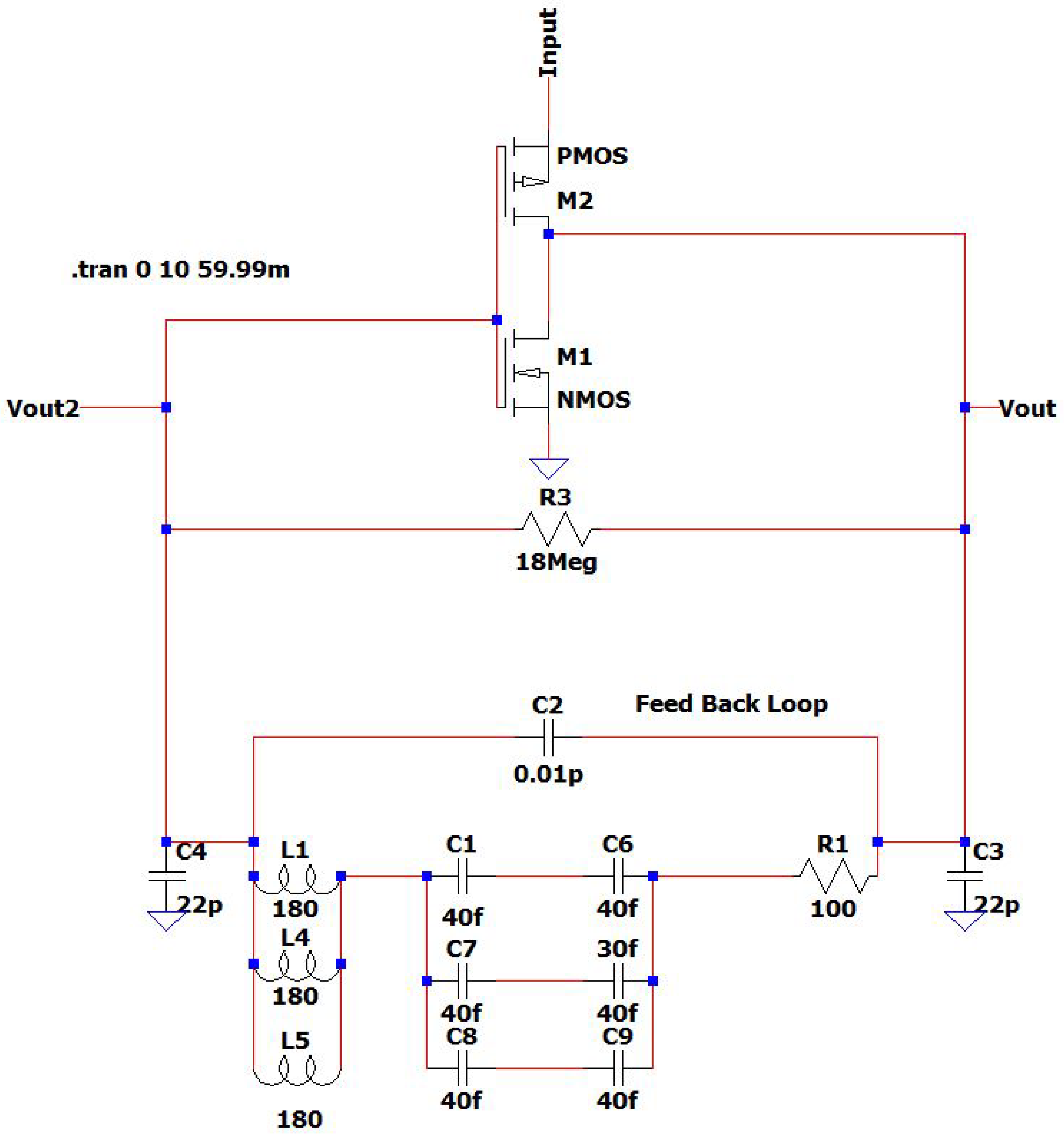


Figure 12. *The Oscillator Circuit takes an input of 100 V DC and can turn it into 65 V AC at 85 kHz.*

The AC voltage on the receiver side should be converted to DC, which is done with a similar rectifier circuit that can produce a clean and constant DC voltage at almost any value. The rectifier can take 65 volts AC at 85 kHz and produce, on average, 120 volts DC through another transformer. The voltage is then stepped up to our desired 350 volts DC, which is needed to charge the electric vehicle, Fig. 13.

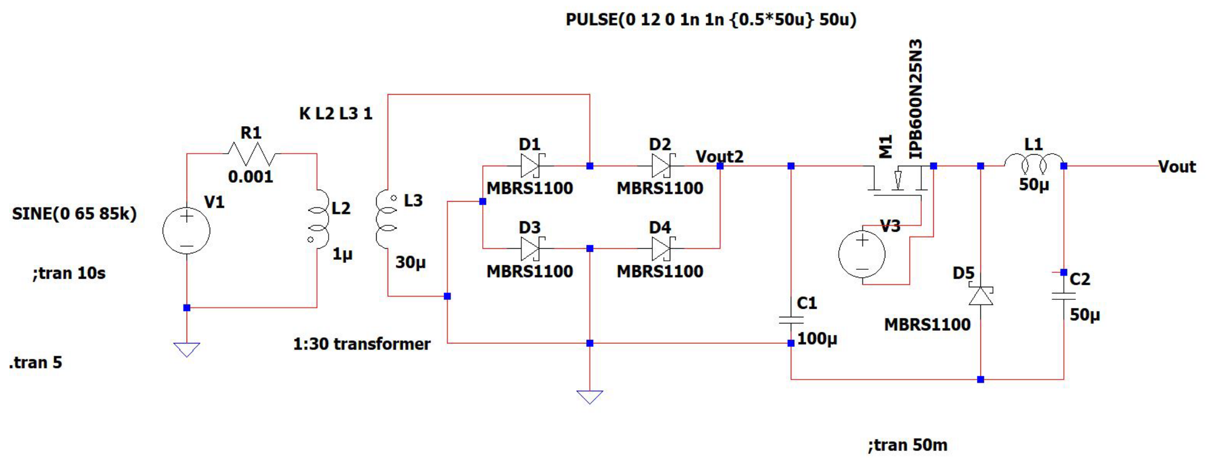
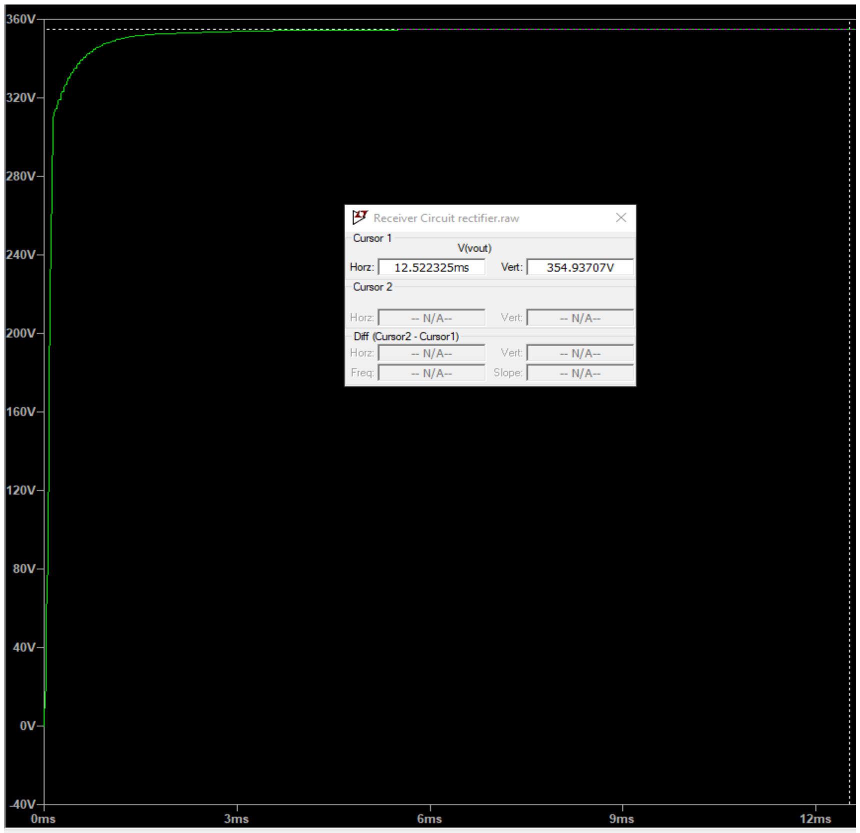


Figure 13. *Rectifier turning 65 Volts AC into 350 Volts DC on the receiver side.*

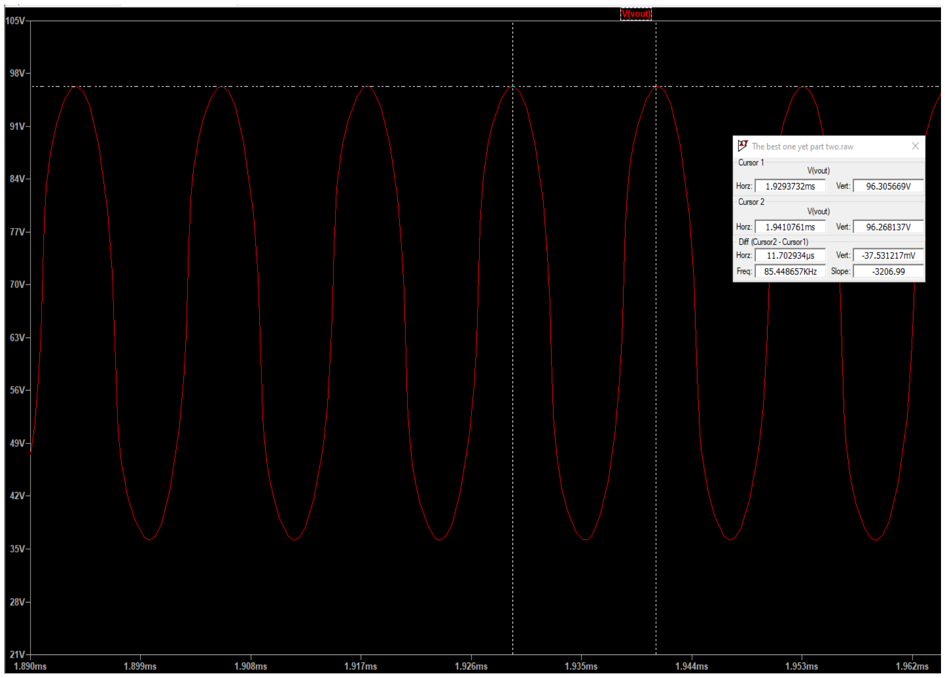
Figs. 14-15 show the capability of the proposed circuit to generate the 350 V DC on the receiver part for charging the battery and the generated AC voltage on the transmitter circuitry.



Receiver Circuit rectifier.raw

Cursor 1
V(vout)
Horz: 12.522325ms Vert: 354.93707V
Cursor 2
Horz: -- N/A-- Vert: -- N/A--
Diff (Cursor2 - Cursor1)
Horz: -- N/A-- Vert: -- N/A--
Freq: -- N/A-- Slope: -- N/A--

Figure 14. *Output Waveform of the voltage after rectification through the Receiver Circuit*



```
The best one yet part two.raw
Cursor 1
                 V(vout)
Horz: 1.9293732ms      Vert: 96.305669V
Cursor 2
                 V(vout)
Horz: 1.9410761ms      Vert: 96.268137V
Diff (Cursor2 - Cursor1)
Horz: 11.702934µs      Vert: -37.531217mV
Freq: 85.448657KHz     Slope: -3206.99
```

***Figure 15**. Waveform of AC Voltage after oscillation, showing Max and Min Voltages and RMS Values*

4.4. Electromagnetic Simulations:

Heinrich Hertz was the first to implement wireless power transfer (WPT) in 1888. He used a spark gap and parabolic reflectors to demonstrate a high-frequency power transfer between the transmitter and the receiver. Figure 16 shows the Hertz spark gap and parabolic reflectors that Heinrich Hertz used in experiments of 1888.

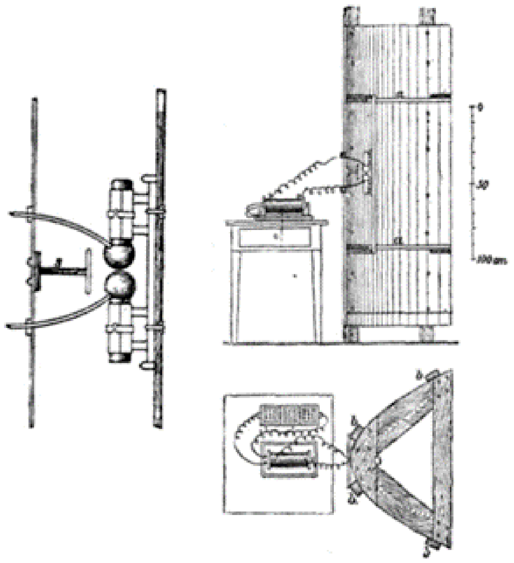


***Figure 16**. Hertz spark gap and parabolic reflectors for wireless power transfer (WPT) in 1888.*

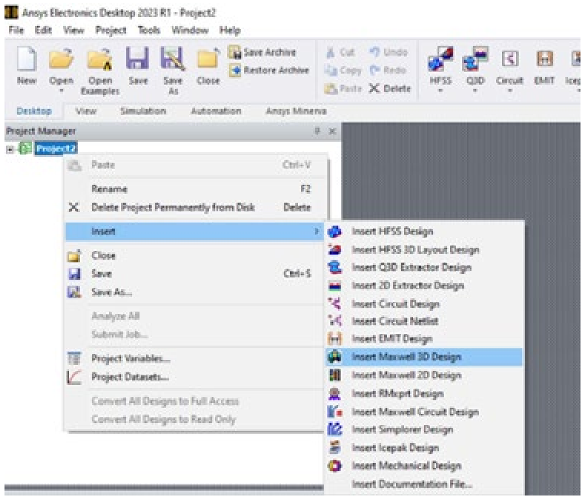
Wireless power transfer (WPT) has been used for decades in applications like satellite communications, telemetry, and radio frequency identification (RFID) tags. Those applications transfer meager power, microwatts to milliwatts. To transfer a high amount of energy, kilowatts, we need to use ***inductive coupling***, which was invented by Nikola Tesla.

Wireless power transfer (WPT) allows energy transmission through an air gap without interconnecting cables. Therefore, the heart of the Wireless power transfer (WPT) system is the Transmitting and receiving coils. Before implementing any coil design physically, a simulation through a suitable simulator must be done. There are a lot of simulators, such as MATLAB/SIMULINK, EM work, SPICE, and ANSYS MAXWELL/SIMPLORER. We have determined through researching online reports and documents that the most suitable simulator for this project is ANSYS. Researchers and industries have widely used this software for evaluating the design and performance of coils.

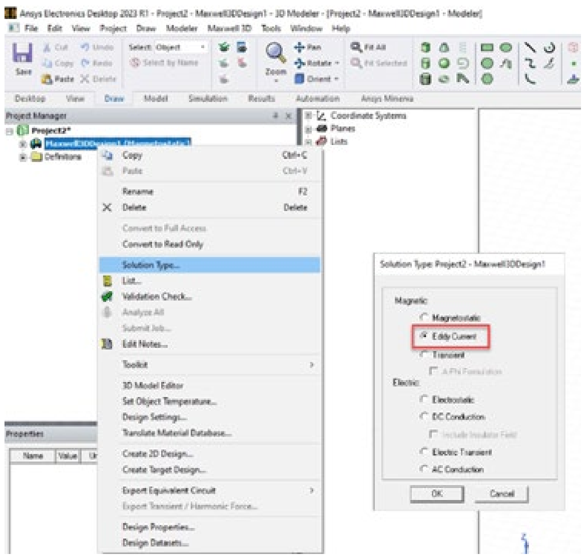
In Wireless power transfer (WPT), the design of the coils is essential because it controls how much power is transmitted and received as well as the efficiency. In our implementation, we use ANSYS Maxwell, an EM field solver for wireless charging, transformers, and other electric-mechanical devices. ANSYS Maxwell can solve physics, including frequency-domain and time-varying magnetic and electric fields.

4.4.1. Simulation Procedure:

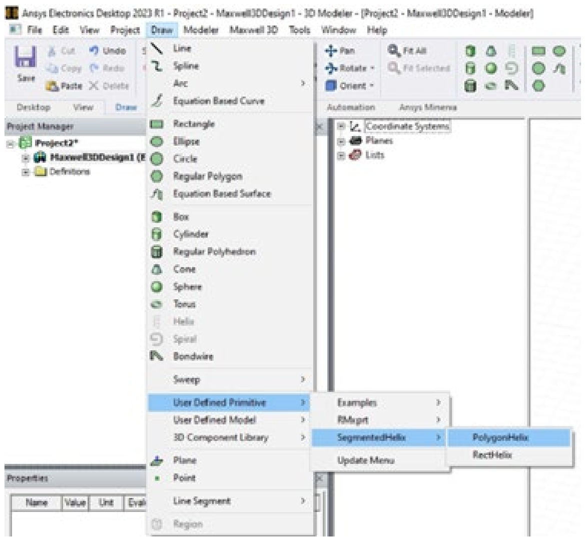
1) **Design Tx and Rx coils:** After running the ANSYS Electronics Desktop software, the simulation type should be selected in the Project manager area (Project2)
   a) Right-click on the project, then Insert, and from Insert, select Insert Maxwell 3D design.


Ansys Electronics Desktop 2023 R1 - Project2
File Edit View Project Tools Window Help
New Open Open Examples Save Save As Close
Save Archive
Restore Archive
Cut Copy Paste Undo Redo Delete
HFSS Q3D Circuit EMIT
Desktop View Simulation Automation Ansys Minerva
Project Manager
Project2
Paste Ctrl+V
Rename F2
Delete Project Permanently from Disk Delete
Insert
Close
Save Ctrl+S
Save As...
Analyze All
Submit Job...
Project Variables...
Project Datasets...
Convert All Designs to Full Access
Convert All Designs to Read Only
Insert HFSS Design
Insert HFSS 3D Layout Design
Insert Q3D Extractor Design
Insert 2D Extractor Design
Insert Circuit Design
Insert Circuit Netlist
Insert EMIT Design
Insert Maxwell 3D Design
Insert Maxwell 2D Design
Insert RMxprt Design
Insert Maxwell Circuit Design
Insert Simplorer Design
Insert Icepak Design
Insert Mechanical Design
Insert Documentation File...

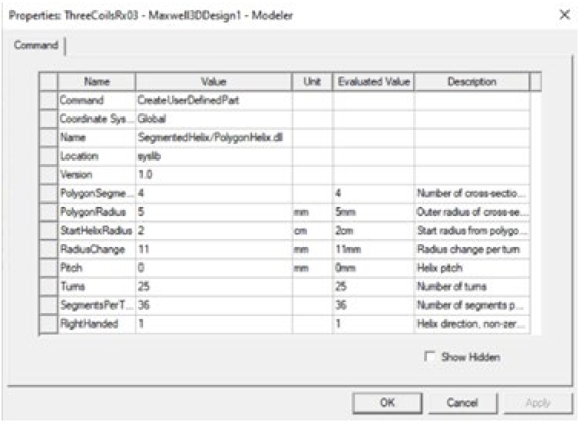
   b) Right-click on Maxwell3Desing1 ==> select Solution Type.. ==> Eddy Current ==> OK.


Ansys Electronics Desktop 2023 R1 - Project2 - Maxwell3DDesign1 - 3D Modeler - [Project2 - Maxwell3DDesign1 - Modeler]
Solution Type...
Solution Type: Project2 - Maxwell3DDesign1
Magnetic
Magnetostatic
Eddy Current
Transient
Electric
Electrostatic
DC Conduction
Electric Transient
AC Conduction
OK
Cancel

c) Now, to insert the coil, click on Draw ==> Used defined Primitive ==> SegmentedHelix ==> PolygonHelix



d) The simulation parameters should be imported below.


Properties: ThreeCoilsRx03 - Maxwell3DDesign1 - Modeler

Command

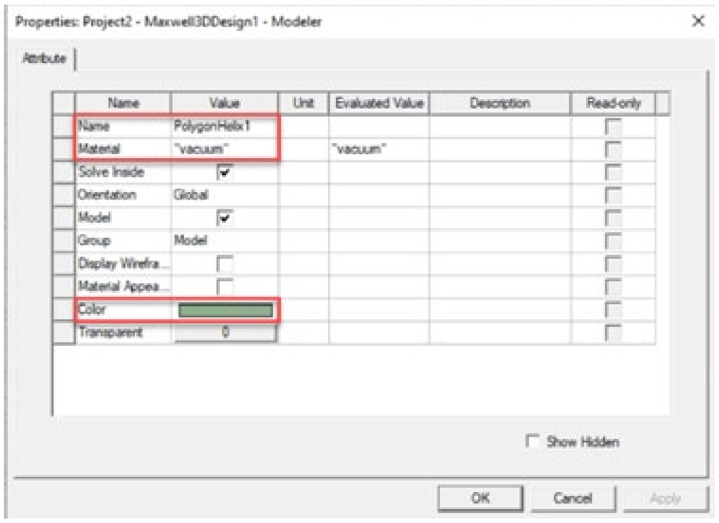
Name	Value	Unit	Evaluated Value	Description
Command	CreateUserDefinedPart			
Coordinate Sys...	Global			
Name	SegmentedHelix/PolygonHelix.dll			
Location	syslib			
Version	1.0			
PolygonSegme...	4		4	Number of cross-sectio...
PolygonRadius	5	mm	5mm	Outer radius of cross-se...
StartHelixRadius	2	cm	2cm	Start radius from polygo...
RadiusChange	11	mm	11mm	Radius change per turn
Pitch	0	mm	0mm	Helix pitch
Turns	25		25	Number of turns
SegmentsPerT...	36		36	Number of segments p...
RightHanded	1		1	Helix direction, non-zer...

☐ Show Hidden

OK Cancel Apply

2) Coil Setting and Properties:

a) Under Vacuum, double-click on PolygonHelix1, then the coil specification window will pop up. The name, Material, and color of the component could be edited from the window below:



Properties: Project2 - Maxwell3DDesign1 - Modeler

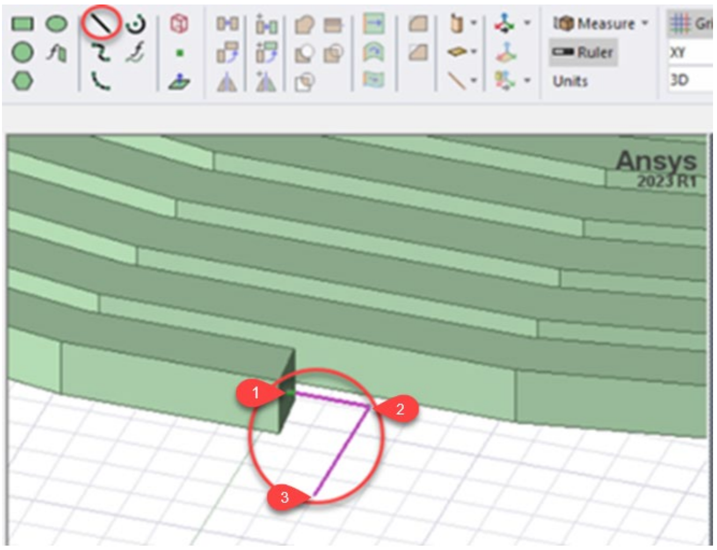
Attribute

Name	Value	Unit	Evaluated Value	Description	Read-only
Name	PolygonHelix1				☐
Material	"vacuum"		"vacuum"		☐
Solve Inside	☑				☐
Orientation	Global				☐
Model	☑				☐
Group	Model				☐
Display Wirefra...	☐				☐
Material Appea...	☐				☐
Color	🟩				☐
Transparent	0				☐

☐ Show Hidden

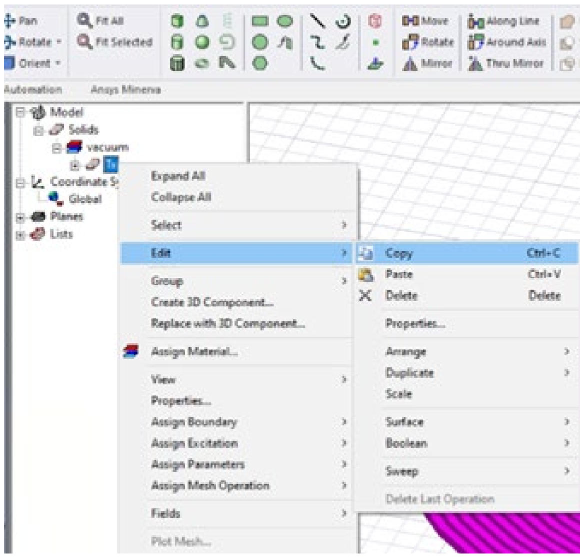
OK Cancel Apply

b) The next step is drawing two lines along the Y-axis and X-axis. The two lines had three points: Point 1, Point 2, and 3, as shown below.

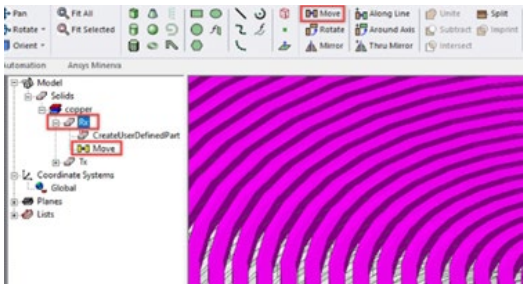


c) The next step is combining the coil and the two lines, Lines 1 and 2. That could be done by following the steps below: Hold Ctrl, then select the coil, PolygonHelix1, then Line 1 and Line 2, CreatePolyLine

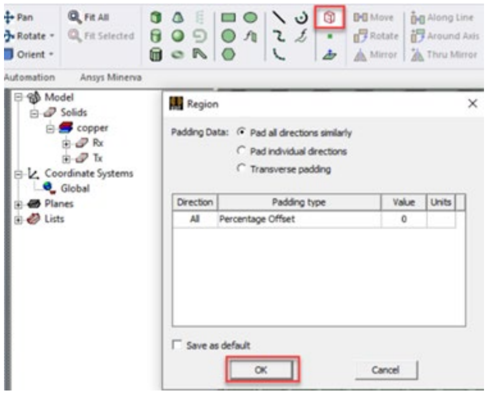
3) **Duplicating Transmission Coil into Three Receiver Coils:** Steps 1 and 2 covers creating a Tx coil; the same steps could be followed to create Rx coils. The other way is to copy and paste the design, as shown below.



4) **Creating Gap between Tx and Rx:** The gap is a critical parameter that can impact the performance of the Tx and Rx coils. In order to set the gap, select the Rx coils and click move. Then, double click moves inside Rx coil and set x-value and y-value = 0 mm and z-value = a variable value as 'Gap.' The idea behind setting Z to Gap is to allow the design to test different gap distances (parametric sweeping over Gap distance).



5) **Creating A Region:** To simulate the coil, we need to create a Box of Air Region as below.



Padding Data: (● Pad all directions similarly) Pad individual directions / Transverse padding

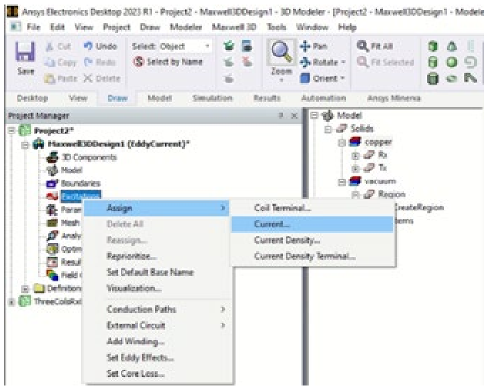
Direction	Padding type	Value	Units
All	Percentage Offset	0	

☐ Save as default

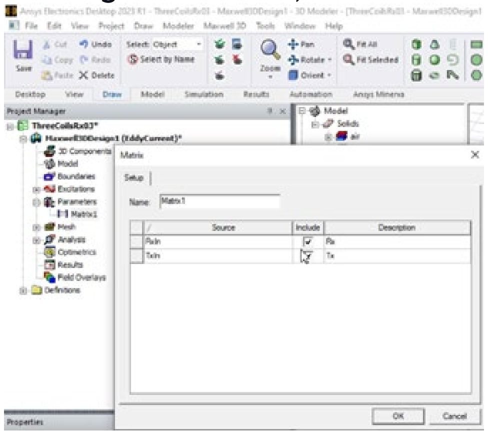
OK Cancel

6) **Steps for simulation settings:**
   a) **3D Components:** We do not change this parameter because everything is designed under the predefined setting of Maxwell3D.

b) **Model:** All the components will be listed here.
c) **Boundaries:** We do not change this parameter because of default settings.
d) **Excitation:** The current excitation in the Tx Coil will flow from Tx to Rx, leading the Rx Coil to be excited when it is in the Tx Coil magnetic field range.



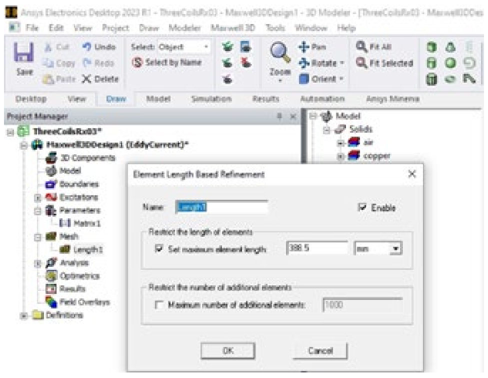
e) **Parameters:** we will set simulation parameters as below:
Right-click on Parameter ==> Assign ==> Matrix, then select Rxin and Txin as shown



Name: Matrix1

Source	Include	Description
RxIn	✓	Rx
TxIn	✓	Tx

f) Mesh: Two meshes will be assigned, one for region and one for coils.
For Coil: right-click on Mesh ==> Initial Mesh Setting, then Mesh Method ==> Auto, Curved Surface Meshing ==> Use Slider and Mesh Size ==> Coarse(Small).
For region: right-click on Mesh ==>Assign Mesh Operation ==>Inside Selection ==>Length Based.


Element Length Based Refinement

Name: Length1 ☑ Enable

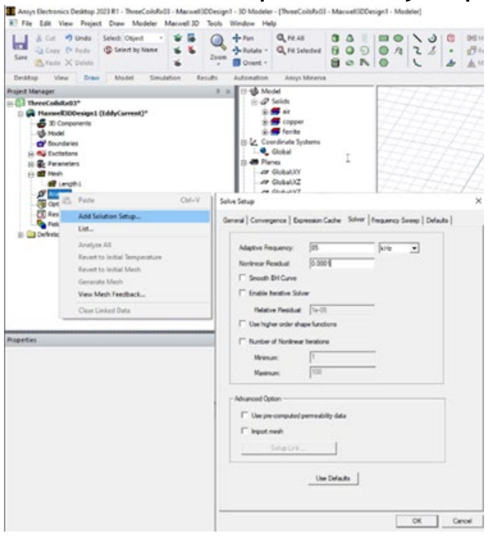
Restrict the length of elements

☑ Set maximum element length: 388.5 mm

Restrict the number of additional elements

☐ Maximum number of additional elements: 1000

g) Analysis: Below is how to set up the analysis parameter.


Solve Setup

General | Convergence | Expression Cache | Solver | Frequency Sweep | Defaults

Adaptive Frequency: 85 kHz

Nonlinear Residual: 0.0001

☐ Smooth BH Curve

☐ Enable Iterative Solver

Relative Residual: 1e-05

☐ Use higher order shape functions

☐ Number of Nonlinear Iterations

Minimum: 1

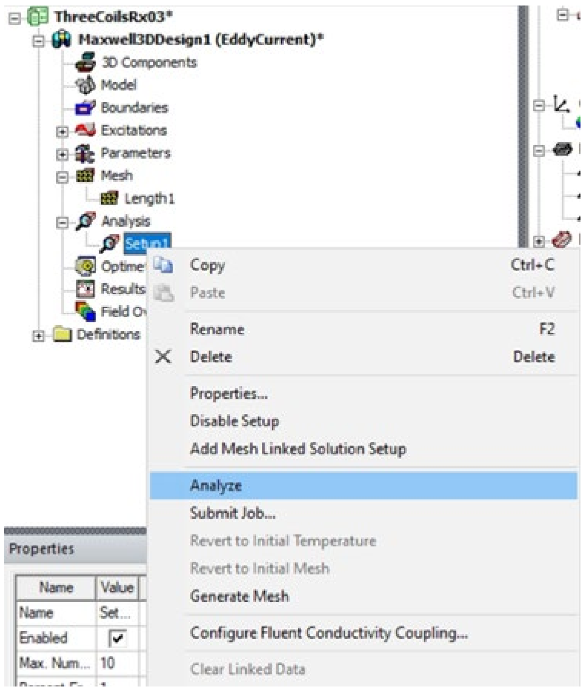
Maximum: 100

Advanced Option

☐ Use pre-computed permeability data

☐ Import mesh

h) In order to run the simulation, Right click on Setup, then click Analyze



Name	Value
Name	Set...
Enabled	✓
Max. Num...	10

4.4.2. Design, Results, and Discussion:

We performed electromagnetic simulations for three transmitter/receiver system designs to predict the magnetic coupling and select the best design with the largest efficiency. The Idea of having three coils for the receiver is to reduce the losses of the electromagnetic flux, leading to improved performance. One of the main problems we face in designing the Wireless power transfer (WPT) is the alignment between the Tx and Rx coils. The more offset we have, the less efficiency we get. In order to solve this problem, our implementation is based on using the three coils as Rx and one long coil as Tx. The Length of the Tx coil could match the Length of the Vehicle. The precise alignment would not be an issue because the Rx coils will receive the maximum possible power from the Tx coil.

The wire used in the coil is 5 mm thick, and the coil size was designed to match the space available for use on the Vehicle. Finally, the gap space between the Tx and Rx was chosen based on the recommendation by SAE standards. Table 3 summarizes our design parameters.

Table 3. Design parameters for electromagnetic simulation.

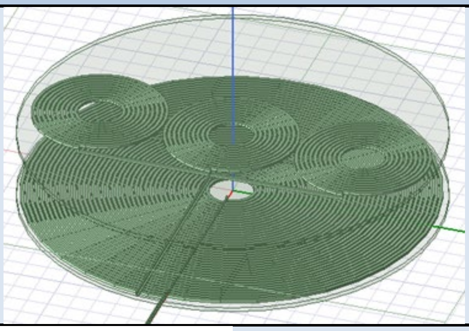
Parameter	Size
Wire diameter	5 mm
Turns Number in Tx and Rx	25
Air Gap between Tx and Rx	5 cm
Tx coils size	25.5 cm
Rx Coil size	76.5 cm

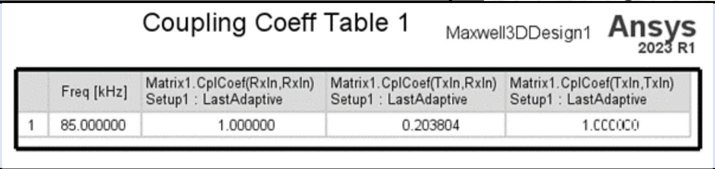
The current that flows in one coil generates a magnetic flux. A portion of the magnetic flux of the transmitter coil connects with the receiver coil, which induces current in the receiver coil. The induced current is proportional to the magnetic coupling coefficient between the two coils, which varies between 0 and 1. When the coupling coefficient is 1, the magnetic flux generated by one coil is linked ideally with the other. When the coupling coefficient is 0, the magnetic flux generated by one coil is not linked with the other coil, and in that case, the coils are known as magnetically isolated. Therefore, the magnetic coupling coefficient was chosen to represent power transmission efficiency, i.e., a higher magnetic coupling coefficient indicates higher efficiency. All simulations are summarized in Table 4.

Table 4. The magnetic coefficient for all

Design 1:

We used a large Tx coil to cover all three Rx coils in this design. The disadvantage of this design is that the large area of the Tx coil is not being used, which leads to low performance. The coupling coefficient of this design is %0.20



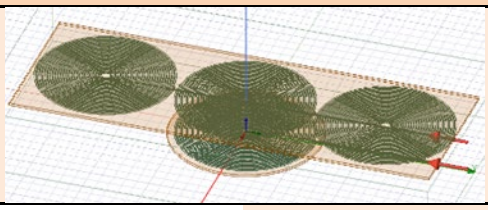


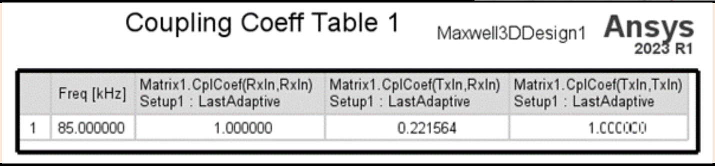
Coupling Coeff Table 1 Maxwell3DDesign1 Ansys 2023 R1

	Freq [kHz]	Matrix1.CplCoef(RxIn,RxIn) Setup1 : LastAdaptive	Matrix1.CplCoef(TxIn,RxIn) Setup1 : LastAdaptive	Matrix1.CplCoef(TxIn,TxIn) Setup1 : LastAdaptive
1	85.000000	1.000000	0.203804	1.CCC0C0

Design 2:

This design used a standard Tx coil to cover the middle Rx coil. The disadvantage of this design is that we have two Rx coils not covered by the Tx coil, leading to low performance. The coupling coefficient of this design is %0.22.



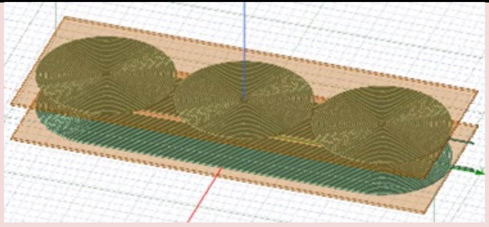


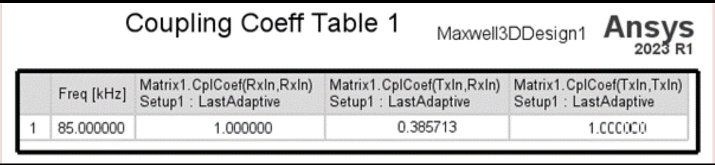
Coupling Coeff Table 1 Maxwell3DDesign1 Ansys 2023 R1

	Freq [kHz]	Matrix1.CplCoef(RxIn,RxIn) Setup1 : LastAdaptive	Matrix1.CplCoef(TxIn,RxIn) Setup1 : LastAdaptive	Matrix1.CplCoef(TxIn,TxIn) Setup1 : LastAdaptive
1	85.000000	1.000000	0.221564	1.CCC0C0

Design 3: **Best Design with High Coupling Coefficient**

We used a long Tx coil to cover all three Rx coils in this design. This design gave the best performance of all the three designs we implemented because the Tx coil covers the Rx coils, and we do not have any unused area of the Tx and Rx coils. The coupling coefficient of this design is %0.38





Coupling Coeff Table 1 Maxwell3DDesign1 Ansys 2023 R1

	Freq [kHz]	Matrix1.CplCoef(RxIn,RxIn) Setup1 : LastAdaptive	Matrix1.CplCoef(TxIn,RxIn) Setup1 : LastAdaptive	Matrix1.CplCoef(TxIn,TxIn) Setup1 : LastAdaptive
1	85.000000	1.000000	0.385713	1.CCC0C0

4.5. Communication Protocol:

The digital communication between the electric vehicle and the charging station should follow communication protocol ISO 15118 to prepare the best charging schedule for the EV. One of the features of the communication protocol ISO 15118 is the high level of security, including encryption. Moreover, the protocol ISO 15118 uses IPv6-based communications, the fastest protocol; many companies and vendors, such as Mercedes Benz, have used ISO 15118 in their EVs. EV charging stations can communicate with a central control station through clouds, providing additional security. Information such as status (idle or busy), scheduling for incoming EV, and predicted availability can be transmitted through wired networks or wireless communications such as 5G to the central control. Each EV also communicates with the central control through 5G wireless networks to report its location and charge level. The central control determines the best route to direct autonomous EVs to the nearest open charging station.
When the EV shuttle arrives at the station, the communication between the EV and the charging station could be through secured wireless to allow the EV to enter and exit the charging station safely and update necessary information such as EV battery level, charging time, and destination.

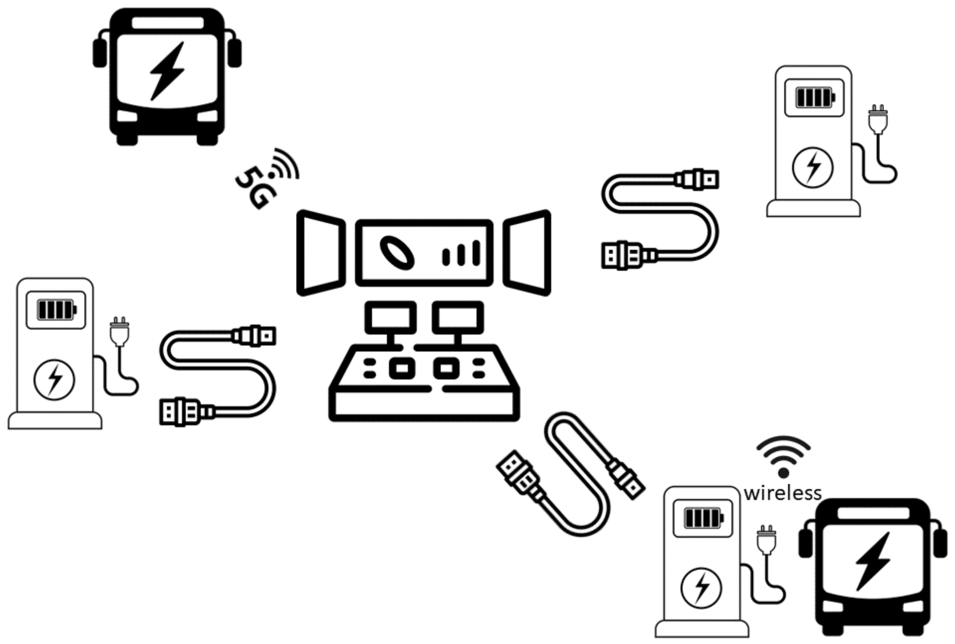


***Figure 17**. Electric shuttle and electric charging station communication demonstration*

5. Conclusions

HD maps: A Python code was developed for marking/labeling a user-input location on an HD map. Also, a flowchart was proposed to make vector maps for navigation in the Apollo environment. In the next step, one can implement the proposed steps with open-source software such as Apollo and Vector Map Net to label the charging station and simulate navigation toward the station.
Electromagnetic Simulation: ANSYS was used to model the magnetic coupling of several designs. The magnetic coupling coefficient was used to represent the power transfer efficiency. The step-by-step tutorial can guide the modeling of any wireless charging design.

Charging Circuitry: The transmitter and receiver circuit was proposed and simulated with LTSpices. The proposed circuitry generates 350 V DC on the receiver sides, which can be used to charge EV batteries. Future plans for the circuit could be updating the overall feedback loop of the circuit to use smaller capacitors; currently, the capacitors used in simulation with the rest of the circuit would be hard to purchase, so with the use of some clever circuity like putting more capacitors in parallel one can lower the individual capacitors capacitance while increasing the number of capacitors.

Communication: ISO 15118 is recommended as the communication protocol between EVs and charging stations.

6. Recommendations for Implementation

None

Bibliography

1. A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets Robotics: The KITTI Dataset," in International Journal of Robotics Research (IJRR), 2013
2. Q. Hu, B. Yang, S. Khalid, W. Xiao, N. Trigoni, and A. Markham, "Towards Semantic Segmentation of Urban-Scale 3D Point Clouds: A Dataset, Benchmarks and Challenges," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021.
3. Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A Modern Library for 3D Data Processing," in arXiv:1801.09847, 2018.
4. Apollo Auto [Source code]. https://github.com/ApolloAuto/apollo.
5. Y. Liu, T. Yuan, Y. Wang, Y. Wang, and H. Zhao, "VectorMapNet: End-to-end Vectorized HD Map Learning," in Proceedings of the 40th International Conference on Machine Learning, A. Krause et al., Eds., vol. 202, pp. 22352-22369, PMLR, Jul. 23-29, 2023.

6. J. Jeong, , J. Y. Yoon, H. Lee, H. Darweesh, and W. Sung, Tutorial on High-Definition Map Generation for Automated Driving in Urban Environments, Sensors 2022, 22, 7056
7. IV Tier, MapTools: VectorMapper, [online] Available: https://maptools.tier4.jp/vector_mapper_description/.
8. W. N. Tun, S. Kim, J.-W. Lee, and H. Darweesh, "Open-Source Tool of Vector Map for Path Planning in Autoware Autonomous Driving Software," 2019 IEEE International Conference on Big Data and Smart Computing (BigComp). IEEE, Feb. 2019. doi: 10.1109/bigcomp.2019.8679340.