IoT-based Autonomou ElectricVehicle Communication System

[1]Voruganti Bharath Kumar, [2]Mohammad Moin, [3]MD.Samreen Sulthana,[4]K David Raju

[1]Assistant Professor, Dept of CSE, St. Martin's Engineering College, Secunderabad.

[2]Assistant professor, Dept of CSE, Guru Nanak Institute of technology, Hyderabad.

[3]Assistant professor, Dept of EEE, Guru Nanak Institute of technology, Hyderabad.

[3]Assistant Professor, Dept of EEE, Malla reddy Engineering College for women, Misammaguda.

Abstract—**This paper discusses the development of an Internet of Things (IoT)-based communication system for Electric Vehicles (EVs) that utilizes a communication protocol that is bandwidth-efficient, lightweight, and uses the least amount of battery power. This system was designed using the Message Queuing Telemetry Transport (MQTT) protocol, which is a lightweight protocol that consumes very little system power. A Quality of Service (QoS) levels comparison was carried out to determine the suitable QoS level for this system. This paper proposes a Lane Detection System which is an Advanced Driver-Assistance System (ADAS) feature, and its parameters, such as radius of curvature, the off-center distance of the vehicle, and curve direction, are calculated and communicated via the communication system. In the Lane Detection System, color selection and edge detection techniques were used. This system can detect lanes in real-time and store and send data to a remote location for vehicle monitoring.**

Keywords—Internet of Things (IoT), Electric Vehicle (EV), Message Queuing Telemetry Transport (MQTT), Lane Detection System, Quality of Service (QoS)

I. INTRODUCTION

The Internet of Things (IoT) is a network of internet-connected devices that can gather and transfer data across a wireless network without requiring human intervention. With the introduction of IC (Internal Combustion engine) engines, the problem of air pollution has become considerably worse because of the gases generated by burning fuel. Furthermore, fossil fuels are a finite resource that cannot be used indefinitely. As a result, the automobile industry is gradually transitioning to Electric Vehicles to address the issues mentioned above [1]. To monitor the EV from remote locations and make it autonomous, factors such as battery percentage, state of charge, vehicle location, and Advanced Driver-Assistance System (ADAS) feature parameters must be communicated by using an efficient communication system.

A larger number of electric vehicle charging stations in parking systems (grid) will be required as the number of electric vehicles will be increasing. As a result, this charging occurs in a G2V (Grid to Vehicle) manner. Energy can also be transferred in various ways, such as through Vehicle to Grid (V2G), vehicle-to-pedestrian (V2P), or vehicle-to-home (V2H). Energy exchange between electric vehicles that are moving or parked can be accomplished with the help of V2V communication. By using bi-directional chargers in homes or charging stations, V2G communication allows energy to be returned to the grid [2]. The Vehicle to Grid system is a two-way charging system that allows drivers to charge their automobiles from the grid or sell the excess energy generated by their vehicles to the charging station or the grid. This is known as V2G energy transfer and is also referred to as the intelligent energy management system. The advantages of V2G include helping to balance electricity consumption and avoiding excessive costs [3]. To carry out the above processes, a communication medium must be developed, allowing for the smooth transfer of critical data from the vehicle system to the charging system. As a result, we attempted to build a communication system in this project.

Wireless communication is preferred over wired communication because it does not require a physical medium, such as cables, to transfer data from one point to another. The world is evolving in the field of self-driving automobiles, making it even more critical to design intelligent systems that provide safe transportation and travel. Although the ADAS will not be able to prevent accidents, it will be able to help us to limit the number. ADAS features are in-vehicle devices that assist drivers in their daily driving activities. The term "ADAS" refers to a rising variety of safety features aimed at improving the safety of drivers, passengers, and pedestrians by lowering the overall frequency of vehicle accidents. When a car is moving, whether it is self-driving or being controlled by a human, there is always the possibility that it will deviate from its present lane or into a side lane. This could result in a deadly accident. To avoid this, the lane departure warning system is one of the most significant ADAS elements. We have attempted to implement the same in this project.

The paper is organized as follows: In the Literature Survey, EV details, Vehicle Communication techniques, and Lane Detection System - an ADAS feature are discussed. The mechanism for the implementation of the overall communication system is discussed in Section 3. In section 4, the Lane Detection System, which is utilized to detect lanes, is briefly illustrated. The results cover the lanes detected in different cases as well as the QoS comparison. Section 6 concludes the work that has been completed.

II. LITERATURE SURVEY

The goal of the project is to develop an IoT architecture that will allow a Charging Station and an Electric Vehicle (EV) to communicate in a better way. When it comes to communication between V2V or V2G, there are several obstacles to overcome, including maintaining optimal latency, transmission range, and transmission throughput. Wireless communication is efficient, secure, and reliable, thanks to enhanced encryption technologies. Constrained Application Protocol (CoAP), MQTT, Advanced Message Queuing Protocol (AMQP), and other wireless IoT communication technologies and protocols can be used to connect smart devices. In IoT communication networks, Quality of Service (QoS) is a set of traffic requirements for data flow. QoS is a data traffic management technique that reduces latency and packet loss. It assists in achieving maximum bandwidth, and it allows us to regulate and manage network resources by establishing network priorities. A comparison of QoS parameters should be performed to determine the most efficient protocol [4].

Lane detection is a key component of the ADAS that aids in the development of intelligent automobiles. By delivering data to the driver such as detected lanes, curve radius, and off-center distance of the vehicle, this technique ensures the safety of drivers and passengers. This approach also aids the driver in safely turning the car while keeping an eye on the surroundings and avoiding colliding with other vehicles or objects [5]. By enabling Computer Vision technology for vehicles, the vehicle's surroundings may be sensed. Computer Vision detects lanes by performing image processing on real-time videos or images. Computer Vision is different than other techniques as it can detect lanes in different scenarios. Deep Learning recognizes different objects by comparing millions of different data sets. But deep learning has limitations and can also fail in unpredictable ways. Suppose there is a shadow of a tree on the road and if a deep learning technique has not been trained in such a situation, then it will fail to detect lanes which can cause trouble to the driver. With computer vision, it is possible to perform different color selection algorithms and detect lanes in such situations by using real-time data [6]. Further for the process of lane detection, various techniques such as Hough Transform, Bilateral filter, and Edge detection were studied [7].

In Advances in Vision-based Lane Detection Algorithm Based on Reliable Lane Markings paper [8], methods to detect lanes based on Reliable Lane markings are discussed. The research uses an image processing technique to remove shade effects from RGB images and convert them to grayscale images. The technique uses a Gaussian filter to eliminate sound from images, a Canny edge detection algorithm for edge detection, and a Hough Transform to retrieve image pixels in a straight line. As a result, if the vehicle deviates from the lane, lane correction actions are evaluated.

Sahil Samantaray and co-authors developed a lane detecting method that can be employed by automated bots engaging in the Intelligent Ground Vehicle Challenge (IGVC) [9]. The algorithm primarily makes use of OpenCV functions and uses an image pipeline to appropriately segment and return the direction vector in terms of magnitude, which can then be transmitted to the bot as steering signals.

The paper Low Complexity Lane Detection Methods for Light Photometry System aims to analyze effective solutions for accurate lane detection on the roads [10]. It focuses on correctly driving a light-measurement trailer by detecting airport runways and taxiways. Line detection using edge detection, Scharr mask, and Hough transform, and hyperbola fitting line detection method to find the optimal path are some of the approaches used for lane detection.

In [11], a secure V2V and V2I communication approach are provided, in which trusted cloudlets are used to approve, verify, and ensure the authenticity, integrity, and anonymity of messages sent, rather than direct peer-to-peer connection. Moving vehicles are dynamically connected to adjacent cloudlets, which have security controls in place to prevent rogue vehicles from sending false signals.

III. METHODOLOGY

Electric Vehicle System, Charging Station Monitoring System, Charging Infrastructure, and Cloud Service Platform are the primary components for communication between an electric vehicle and a charging station. This paper focuses on the communication between the Charging Station and Vehicle Monitoring System using a Cloud Service Platform. Fig. 1 shows how data flows in the communication system. EV charging stations use the controller and communicate with the cloud using different IoT protocols such as MQTT, CoAP, or AMQP.

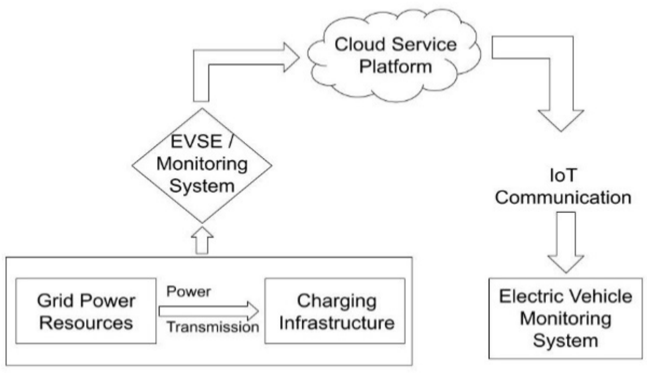


Fig. 1. Intelligent service system structure

As shown in Fig. 1, the Cloud obtains data from the Electric Vehicle Supply Equipment (EVSE) by subscribing to the topics it delivers. Data is exchanged between the cloud and the IoT devices in the Electric Vehicle Monitoring System using lightweight messaging protocol standards. The data is sent using the publish/subscribe or request/response IoT architectures. The data is published by the client in the form of a topic, such as the amount of charging, distance traveled, and position, and it is then sent to the cloud and then to the client who has subscribed to the topic. The subscribed message will be transferred onto the broker/cloud, which will pass it to a designated database as per the IoT communication design.

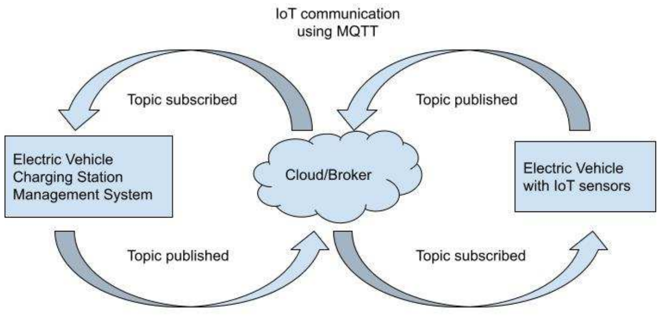

IoT communication using MQTT
Topic subscribed
Topic published
Electric Vehicle Charging Station Management System
Cloud/Broker
Electric Vehicle with IoT sensors
Topic published
Topic subscribed

Fig. 2. Cloud Design for MQTT

The data can be extracted for different data processing using cloud services. The primary goal is to process the data, analyze it, and store it. Understanding background activities, varying charging habits, controlling the entire infrastructure, and making troubleshooting jobs easier will all benefit from this knowledge. We will transfer vehicle data generated by the ADAS function swiftly and securely through IoT communication.

Lane detection is a feature of an ADAS that uses data from a front-facing camera to perform functions such as vehicle routing and lane departure warning. As a result, it is the approach for detecting lanes in automobile applications including travel control, self-directed driving, lane departure detection, and warning.

The Lane detection system can provide the data such as:

- Curve direction
- Radius of curvature
- Off-center distance of the vehicle from the center of the lane

As shown in Fig. 2, the lane detecting system communicates all of the above data to a subscriber at any place who subscribes to the topic using the MQTT protocol.

IV. System Description

The work is organized as follows:

A. *MQTT Protocol:*

Protocols are a collection of rules for sharing data and other services across different devices that provide a way of communication. They're a collection of guidelines that govern how data is transported across a network. Billions of electrical devices must communicate and share information by using different protocols. We ran simulations to compare the protocols and determine which one is best for our project in terms of all aspects, such as latency and throughput.

Wireshark was used to create graphical representations of parameters like throughput and packets vs time graphs for proper packet transmission analysis. Wireshark captures the packets and allows us to inspect them. We performed a basic communication using the MQTT protocol and discovered that the message is published using the MQTT protocol and the acknowledgment is sent using the Transmission Control Protocol (TCP). Wireshark is a tool for breaking down data packets that are sent across many networks. Data packets can be searched and filtered by users. Users can also view the flow of packets across the network.

MQTT offers three levels of QoS.

1. QoS 0 is the lowest level of service, in which the broker or client only sends the message once and receives no confirmation.
2. QoS 1 is the level at which the broker or client delivers the message at least once and requires confirmation or acknowledgment.
3. QoS 2 is the highest level of service, at which the broker or client uses a four-step handshake to send the message exactly once and sends the next message after receiving the acknowledgment.

QoS 0 and QoS 1 deliver messages much faster than QoS 2. QoS 2 requires more system power to operate than QoS 0 and QoS 1 [12].

B. *Lane Detection System:*

A lane detection system is a technology that alerts drivers if they leave their lanes without signaling. This technology aids in the reduction of accidents caused by driver error or distraction.

This system monitors lane markings on highways and arterial roads using real-time video images to detect when a vehicle leaves its lane. When a vehicle leaves its lane, the Lane Detection system alerts the driver so that the vehicle can be driven back into it.

The Open-Source Computer Vision (OpenCV) library and the Python programming language were used to develop the Lane Detection System. Using the different color selection and curve fitting algorithms, OpenCV includes several built-in functions that make it easier to detect lanes.

Procedure for Lane Detection:

a) *Getting Input Data:*

Initially, a video will be captured via a front-facing camera mounted on the car. Because a movie is made up of a succession of images, image processing is performed on these successive images to detect lanes.

b) *Birdseye View:*

Birdseye View is required for fitting curves and identifying lane lines, and detecting lanes will be easier if we can look down on the road rather than out on it from the point of view of the driver. The images have a perspective warp applied to them. The image is then mapped into another matrix to produce a Birdseye view of the lanes by placing four points on it to surround only the lanes that are there.

c) *Color Selection:*

To detect lanes, we need to isolate pixels of lanes from the surrounding image. We are going to perform color selection and edge detection to isolate these pixels. Each pixel has RGB (Red, Green, Blue) values ranging from 0 to 255 because they are 8-bit images. To isolate pixels, we can convert RGB pixels into other color spaces like HSL (for hue, saturation, lightness), HSV (for hue, saturation, value), or Lab (It uses three values L, a, and b). OpenCV makes it easy to convert between these color spaces.

The HLS filtered image was converted to grayscale, then the grayscale image was thresholded to remove any superfluous detections other than lanes, resulting in a blurred image. The algorithm for thresholding is described below. The value of each pixel is compared to the threshold value. If a pixel's value is less than the threshold, its value is set to 0, resulting in a black pixel. If the pixel value is greater than the threshold value, the value is set to 255, resulting in a white pixel.

If $f(x, y) < T$, then $f(x, y) = 0$,

Otherwise, $f(x, y) = 255$,

where $f(x, y)$ is the Coordinate Pixel Value and T is the Threshold Value.

Then converted threshold image into a blurred image using a Gaussian filter. The Gaussian filter convolves the source image with the specified Gaussian kernel. Blur image is used to detect lanes using a canny edge detection algorithm.

d) *Edge Detection:*

Now getting into edge detection, OpenCV has a couple of functions that will detect edges.

Canny Edge Detection Technique:

John F. Canny developed the Canny Edge Detection Technique, which is a popular edge detection algorithm. There are several steps to the algorithm which are used to detect lanes.

1. *Noise Reduction:*

Because edge detection is prone to noise, the first step is to eliminate it from the image. A 5 x 5 Gaussian filter is used to reduce the noise.

2. *Finding Intensity Gradient of the Image:*

The Sobel system is used in the Canny edge detection approach. The image is smoothed and then filtered with a Sobel Kernel to obtain the image's derivative in both the horizontal (Gx) and vertical (Gy) directions. These derivates can be used to find the edge gradient and direction of each pixel:

$$\text{Edge Gradient } (G) = \sqrt{G_x^2 + G_y^2} \tag{1}$$

$$\text{Angle } (\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right) \tag{2}$$

3. *Non-maximum Suppression:*

After determining the gradient magnitude and direction, a thorough scan of the image is performed to exclude pixels that are not part of the edge.

4. *Hysteresis Thresholding:*

After deleting the pixels that aren't part of the lanes, the image is thresholded to try to connect edges that were previously unconnected. We'll need two threshold values for this: minimum and maximum. Any edges with a bigger intensity gradient than the maximum are considered part of the lane and are thus saved. Edges with an intensity gradient smaller than the minimum, however, are eliminated.

e) Curve Fitting:

Using the threshold image, we can run an algorithm to find left and right lines and then we can apply curve fit to those lines.

Curve Fitting Procedure:

1. Take the histogram of the bottom half of the image and the peaks are placed where we believe the line should begin.

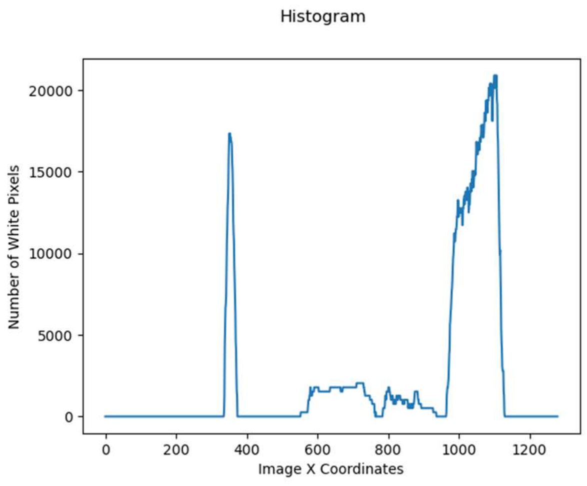


Fig. 3. Histogram of the bottom half of the image

We can notice two separate peaks in the histogram Fig 3 that identify the white pixels, i.e., the spots where the lanes begin. To begin, we must locate the histogram midway. Then determine the x-coordinates of the left and right lanes to determine lane width.

2. We are going to draw a box around those starting points and any of those white pixels from the image that fall in that box, we will put them in a list of x and y values. For the next one, we will just add a box on top of that and then do the same thing.

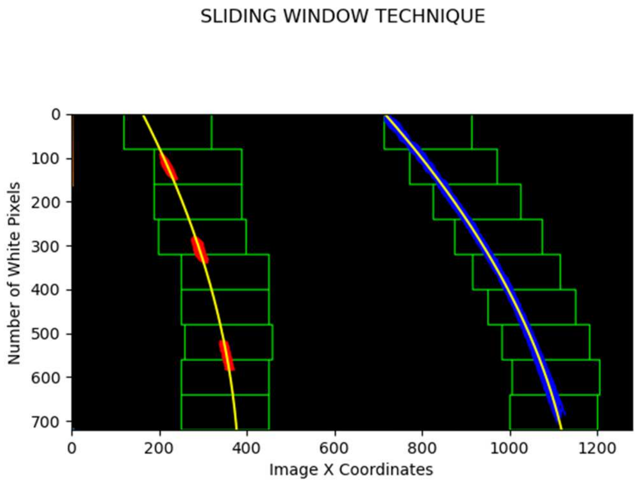


Fig. 4. Sliding Window Technique

As we move up the line, we will move the boxes based on the average of the previous line as shown in Fig. 4. So, if the average starts moving to the left, we will know that the next box needs to be moved to the left.

3. Now we run a NumPy second-order Polyfit to this image of pixels which gives us a curve fit in pixel space.

The values we obtained are in pixel space, which we must convert to meter space. To convert x- and y-coordinates into meter space, multiply them by 3.5 / 720, because the usual lane width is 3.5 m, and lane width of 3.5 m equals 720 pixels in pixel space. To retrieve the output in meter space, run the Polyfit function on these data again.

f) Calculating Lane Detection Parameters:

1. Calculating Radius of Curvature*:*

We can compute the radius of curvature using the previously obtained fits. The radius of curvature is the average of the left line's radius of curvature and the right line's radius of curvature.

2. Curve Direction:

The curve direction is calculated by using the values in lists of x and y coordinates we calculated earlier for creating a stack of windows. The values in these lists are in pixels space and we need to convert them into meter space first which can be done by multiplying each value by 3.5 / 720. To identify curve direction, we need leftx[0] and leftx[-1] coordinates, where leftx[0] is the initial coordinate of the left lane and leftx[-1] is the last coordinate of the left lane, and we may determine the curve direction by computing the difference between these two coordinates.

- If leftx[0] — leftx[-1] > 60, then curve direction is left.
- If leftx[-1] — leftx[0] > 60, then curve direction is right.
- If left[-1] — left[0] = 0 then curve is absent which means the lane is straight.

3. Off-Center Distance of Vehicle:

Off-center can be calculated easily if we know x coordinates of the left lane and right lane. To calculate the off-center distance, we need to know the mid-point of the lane and it can be calculated by computing the mean value of right lane (rightPos) and left lane (leftPos) coordinates.

$$\text{laneCenter} = (\text{rightPos} - \text{leftPos}) / 2 + \text{leftPos} \quad (3)$$

The off-center distance is the difference between the center of the lane and the center of the image.

$$\text{off\_center\_distance} = \text{laneCenter} - \text{imageCenter} \quad (4)$$

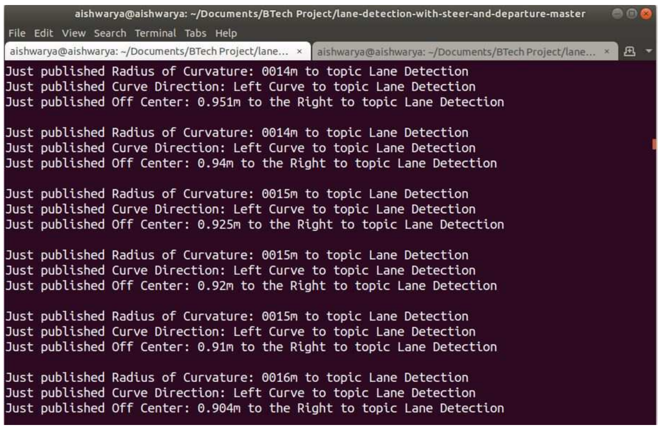
If the off-center distance is positive, the vehicle is on the right side of the lane's center; otherwise, the vehicle is on the left side.

C. *IoT Communication:*

Following the detection of lanes, numerous parameters such as the radius of curvature, curve direction, and the vehicle's off-center distance from the lane's center are determined. All of this data is transmitted via the MQTT protocol and saved in a CSV file.

1. *Publishing the Data:*

We first construct a client called "Electric Vehicle" before publishing the data. We used an EMQX broker with the hostname "broker.emqx.io" for this project, and we'll connect to it using port 1883. We publish the data by submitting the text to be published and the QoS parameter to a topic called "Lane Detection Parameters." We must publish the data three times because we are sending three parameters. The estimated parameters are published after running the publisher program for the Lane Detection feature, as illustrated in Fig 5. These parameters are provided to the EMQX broker over the MQTT protocol using appropriate QoS.


aishwarya@aishwarya: ~/Documents/BTech Project/lane-detection-with-steer-and-departure-master
File Edit View Search Terminal Tabs Help
aishwarya@aishwarya: ~/Documents/BTech Project/lane...
aishwarya@aishwarya: ~/Documents/BTech Project/lane...

```
Just published Radius of Curvature: 0014m to topic Lane Detection
Just published Curve Direction: Left Curve to topic Lane Detection
Just published Off Center: 0.951m to the Right to topic Lane Detection

Just published Radius of Curvature: 0014m to topic Lane Detection
Just published Curve Direction: Left Curve to topic Lane Detection
Just published Off Center: 0.94m to the Right to topic Lane Detection

Just published Radius of Curvature: 0015m to topic Lane Detection
Just published Curve Direction: Left Curve to topic Lane Detection
Just published Off Center: 0.925m to the Right to topic Lane Detection

Just published Radius of Curvature: 0015m to topic Lane Detection
Just published Curve Direction: Left Curve to topic Lane Detection
Just published Off Center: 0.92m to the Right to topic Lane Detection

Just published Radius of Curvature: 0015m to topic Lane Detection
Just published Curve Direction: Left Curve to topic Lane Detection
Just published Off Center: 0.91m to the Right to topic Lane Detection

Just published Radius of Curvature: 0016m to topic Lane Detection
Just published Curve Direction: Left Curve to topic Lane Detection
Just published Off Center: 0.904m to the Right to topic Lane Detection
```

Fig. 5. Publishing the Data

- *Supplying Security to the Communication:*

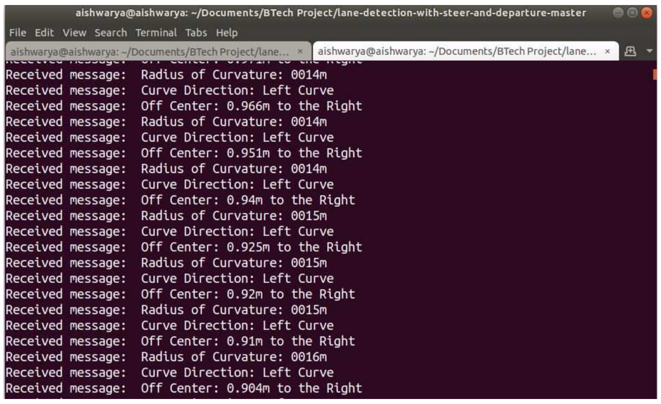
Unsecure MQTT communication takes place on port 1883. We can increase security by using port 8883, which is dedicated to MQTT over TLS (Transport Layer Security). TLS allows a client and a server to communicate securely. Python includes routines for configuring TLS connections and network encryption and authentication. The default certification authority for Python 3.5 is the system's default certification authority.

2. *Subscribing the Data:*

To subscribe to the data, the main part is to connect to the broker and subscribe to the topic.

```
client.connect("broker.emqx.io",        8883,        60)
client.subscribe("Lane Detection Parameters",qos=1)
```

After subscribing to the topic, we will start receiving the data and will be displayed on the terminal shown in Fig. 6.


aishwarya@aishwarya: ~/Documents/BTech Project/lane-detection-with-steer-and-departure-master
File Edit View Search Terminal Tabs Help
aishwarya@aishwarya: ~/Documents/BTech Project/lane...
aishwarya@aishwarya: ~/Documents/BTech Project/lane...

```
Received message:  Radius of Curvature: 0014m
Received message:  Curve Direction: Left Curve
Received message:  Off Center: 0.966m to the Right
Received message:  Radius of Curvature: 0014m
Received message:  Curve Direction: Left Curve
Received message:  Off Center: 0.951m to the Right
Received message:  Radius of Curvature: 0014m
Received message:  Curve Direction: Left Curve
Received message:  Off Center: 0.94m to the Right
Received message:  Radius of Curvature: 0015m
Received message:  Curve Direction: Left Curve
Received message:  Off Center: 0.925m to the Right
Received message:  Radius of Curvature: 0015m
Received message:  Curve Direction: Left Curve
Received message:  Off Center: 0.92m to the Right
Received message:  Radius of Curvature: 0015m
Received message:  Curve Direction: Left Curve
Received message:  Off Center: 0.91m to the Right
Received message:  Radius of Curvature: 0016m
Received message:  Curve Direction: Left Curve
Received message:  Off Center: 0.904m to the Right
```

Fig. 6. Subscribing the Data

V. RESULTS

Following the execution of the code, the output video is displayed on the screen, displaying detected lanes in green color, as seen in Fig. 7. The calculated parameters are also displayed as text in the upper left corner. As we progress through the video, a sequence of images is recorded, and image processing is applied to each image to detect lanes and calculate parameters, which are presented on the screen.

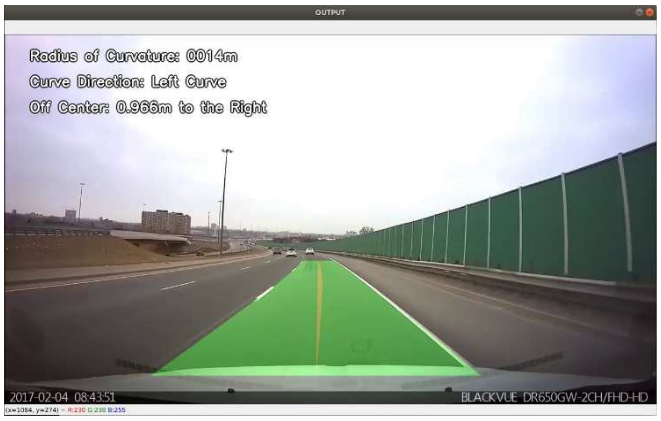

OUTPUT
Radius of Curvature: 0014m
Curve Direction: Left Curve
Off Center: 0.966m to the Right
2017-02-04 08:43:51
BLACKVUE DR650GW-2CH/FHD-HD

Fig. 7. Lane Detection System Output – Left curve detected

Because the curve direction is straight in Fig. 8, no curve is detected, and so the radius of curvature is not applicable (N/A).

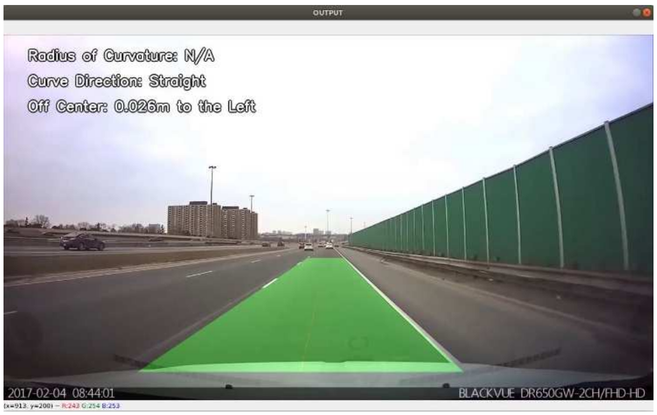

OUTPUT
Radius of Curvature: N/A
Curve Direction: Straight
Off Center: 0.026m to the Left
2017-02-04 08:44:01
BLACKVUE DR650GW-2CH/FHD-HD

Fig. 8. No curve detected

The right curve is recognized in Fig. 9 since the curve direction is right, and the calculated radius of curvature and off-center distance of the vehicle from the lane center is indicated.

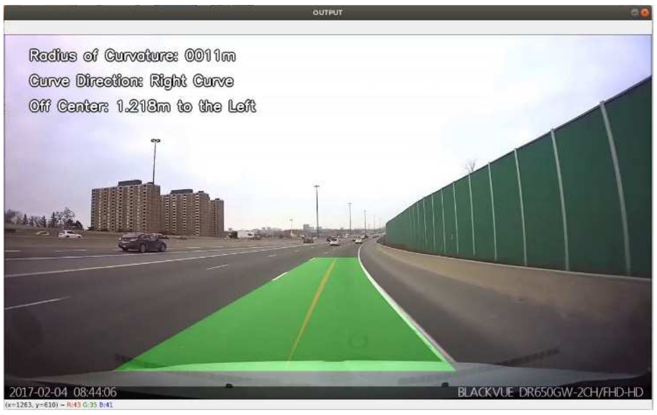
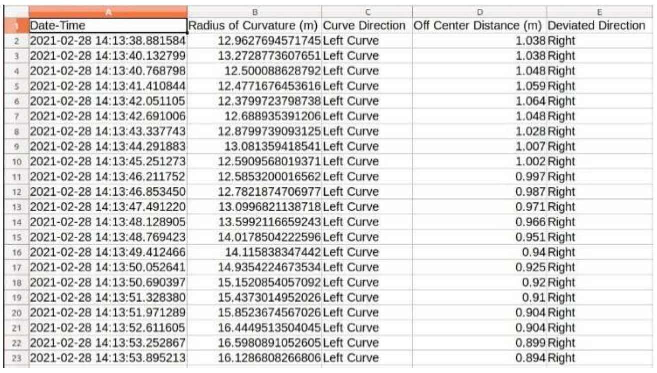

OUTPUT
Radius of Curvature: 0011m
Curve Direction: Right Curve
Off Center: 1.218m to the Left
2017-02-04 08:44:06
BLACKVUE DR650GW-2CH/FHD-HD

Fig. 9. Right curve detected

A CSV file is used to store the calculated parameters. The snippet of the file in which the published data is inserted is shown in Fig. 10. The parameters calculated at the respective date and time are stored to analyze the data properly. Rows 14 to 21 record the data corresponding to the results as shown in Fig. 5 and Fig. 7. Similar results can be obtained when the left curve or no curve is detected.



	A	B	C	D	E
1	Date-Time	Radius of Curvature (m)	Curve Direction	Off Center Distance (m)	Deviated Direction
2	2021-02-28 14:13:38.881584	12.9627694571745	Left Curve	1.038	Right
3	2021-02-28 14:13:40.132799	13.2728773607651	Left Curve	1.038	Right
4	2021-02-28 14:13:40.768798	12.500088628792	Left Curve	1.048	Right
5	2021-02-28 14:13:41.410844	12.4771676453616	Left Curve	1.059	Right
6	2021-02-28 14:13:42.051105	12.3799723798738	Left Curve	1.064	Right
7	2021-02-28 14:13:42.691006	12.688935391206	Left Curve	1.048	Right
8	2021-02-28 14:13:43.337743	12.8799739093125	Left Curve	1.028	Right
9	2021-02-28 14:13:44.291883	13.081359418541	Left Curve	1.007	Right
10	2021-02-28 14:13:45.251273	12.5909568019371	Left Curve	1.002	Right
11	2021-02-28 14:13:46.211752	12.5853200016562	Left Curve	0.997	Right
12	2021-02-28 14:13:46.853450	12.7821874706977	Left Curve	0.987	Right
13	2021-02-28 14:13:47.491220	13.0996821138718	Left Curve	0.971	Right
14	2021-02-28 14:13:48.128905	13.5992116659243	Left Curve	0.966	Right
15	2021-02-28 14:13:48.769423	14.0178504222596	Left Curve	0.951	Right
16	2021-02-28 14:13:49.412466	14.115838347442	Left Curve	0.94	Right
17	2021-02-28 14:13:50.052641	14.9354224673534	Left Curve	0.925	Right
18	2021-02-28 14:13:50.690397	15.1520854057092	Left Curve	0.92	Right
19	2021-02-28 14:13:51.328380	15.4373014952026	Left Curve	0.91	Right
20	2021-02-28 14:13:51.971289	15.8523674567026	Left Curve	0.904	Right
21	2021-02-28 14:13:52.611605	16.4449513504045	Left Curve	0.904	Right
22	2021-02-28 14:13:53.252867	16.5980891052605	Left Curve	0.899	Right
23	2021-02-28 14:13:53.895213	16.1286808266806	Left Curve	0.894	Right

Fig. 10. Lane Detection System Data Storage

A QoS comparison was performed while sending this information. Table I illustrates the data gathered when conducting identical communications for various QoS. Compared to QoS 1 and QoS 0, QoS 2 has the lowest TCP error and captures most of the data packets. QoS 2 has higher throughput and it delivers the message exactly once which leads to high overhead and takes more time to deliver the message. Implementation of QoS 2 for data transfer Is expensive than QoS 0 and QoS 1. QoS 0 results indicate a high TCP error, which means a lot of data is lost during communication.

TABLE I. QUALITY OF SERVICE (QOS) COMPARISON

Factors	**Quality of Service (QoS)**		
	QoS 0	***QoS 1***	***QoS 2***
Throughput (bits/second)	1608	1624	1624
TCP Error	High	Medium	Low
Total Number of Packets Captured	7600	8121	9572

After analyzing all of the considerations, QoS 1 is determined to be the best option for our project because it consumes a moderate amount of system power and captures the majority of packets, which is sufficient for gathering data and obtaining the desired result.

The proposed work involves detecting lanes using OpenCV and communicating the relevant data securely using the MQTT protocol. The lane detection feature was carried out using different color selection algorithm to ensure the detection of lanes in different scenarios. The RGB image was converted into HLS filtered image which is then passed to a Gaussian filter to reduce image noise. The Canny edge detection algorithm was used to detect edges using the Sobel operator. Finally using a curve-fitting algorithm, curve radius, curve direction, and off-center distance of the vehicle from the center of the lane were calculated. This data is then communicated using a secure MQTT connection to monitor the vehicle from a remote location. The work is done using open-source resources which makes the project cost-effective. MQTT protocol used in the project is a machine-to-machine (M2M) IoT protocol that can be used to provide secure and reliable communication to build connected cars.

VI. CONCLUSION

To have effective communication, an electric vehicle communication system was designed utilizing the MQTT protocol. A Lane Detection System was proposed, and its parameters were communicated via MQTT protocol to monitor the vehicle from a remote location. MQTT provides a customizable communication pattern. Based on the foregoing observations and findings, we chose the MQTT protocol for communication between the charging station and the electric vehicle since it is reliable and efficient.

The Radius of Curvature, Curve Direction, and Off-Center Distance of the vehicle from the center of the lane is calculated and recorded in a CSV (comma-separated values) file as Lane Detection parameters. The calculated parameters are then transmitted via the MQTT protocol by a publisher client, and this data can be retrieved by a subscriber client located anywhere in the world by subscribing to the topic. Using port 8883 and TLS, secure MQTT communication was established (Transport Layer Security). TLS is a good choice for establishing a secure MQTT connection since it ensures that data supplied during communication cannot be read or changed by third parties. Because it requires less system power, QoS level 1 is determined to be more suitable for MQTT communication. QoS 1 delivers the data with a successful data transmission rate of 1624 bits/second, medium TCP error, and 84.84% (Total Number of Packets Captured (QoS1) / Total Number of Packets Captured (QoS2)) of the total captured packets are transmitted ensuring that minimum data is lost. The same communication system can be used to transfer data such as data generated by other ADAS features, SoC (State of Charge), SoH (State of Health) of the battery of Electric Vehicle, vehicle location, etc.

VII. References

[1] International Journal of Pure and Applied Mathematics " IoT Enabled smart charging stations for Electric Vehicle", by Arunkumar P, Volume 119 No. 7 2018.

[2] Fast EV charging station integration with grid ensuring optimal and quality power exchange, by Wajahat Khan, Furkan Ahmad, Mohammad Saad Alam, August 2018.

[3] MDPI, energies, "A Comprehensive Review on IoT Protocols' Features in Smart Grid Communication", by Lilia Tightiz and Hyosik Yang, June 2020.

[4] Comparison of Wireless Communication Technologies used in a Smart Home: Analysis of wireless sensor node based on Arduino in home automation scenario, by Oleh Horyachyy, June 2017.

[5] Lane Detection Algorithm for Intelligent Vehicles in Complex Road Conditions and Dynamic Environments, by Jingwei Cao, Chuanxue Song, Shixin Song, Feng Xiao, and Silun Peng1, July 2019.

[6] Lane Detection for Autonomous Vehicles using OpenCV Library, by Aditya Singh Rathore, January 2019.

[7] Introducing Advanced Driver-Assistance System (ADAS) into drivers' training and testing: The young learner drivers' perspective, Master of Science Thesis, Department of Civil Engineering and Geosciences, Delft University of Technology, Tsapi Anastasia, December 2015.

[8] Advances in Vision-based Lane Detection Algorithm Based on Reliable Lane Markings, P. Priyadharshini, Pulugurtha Niketha, K. Saantha Lakshmi, S. Sharmila, R. Divya, 2019

[9] Lane Detection Using Sliding Window for Intelligent Ground Vehicle Challenge, Sahil Samantaray, Rushikesh Deotal, Chiranji Lal Chowdhary, 2021

[10] Low Complexity Lane Detection Methods for Light Photometry System, Jakub Suder, Kacper PodbuckiOrcID,Tomasz MarciniakOrcID and Adam Dąbrowski, 2021

[11] Secure V2V and V2I Communication in Intelligent Transportation using Cloudlets, Maanak Gupta, James Benson, Farhan Patwa, Ravi Sandhu,2021

[12] https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/

[13] "IoT system for any time/anywhere monitoring and control of vehicles' parameters," by Jose A. Afonso; Ruben A. Sousa; Joao C. Ferreira, December 2017, IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI).

[14] Metropolia, "Advanced IoT solution for EV CHAdeMO Fast Charging Applied with Mobile Networks, by Terho Craven, May 2019.

[15] eclipse, "Introduction to MQTT," by Dave Locke, May 2013.

[16] https://github.com/canozcivelek/lane-detection-with-steer-and-departure.

[17] Wireshark.org, "Wireshark," http://www.wireshark.org/, 2013.

[18] A REVIEW OF LANE DETECTION TECHNIQUES GurjyotKaur and Gagandeeop Singh Department of Computer Engineering & Technology Guru Nanak Dev University, Amritsar, Punjab, India, June 2015.