MINI PROJECT REPORT

HARDWARE INTEGRATION OF SENSING AND CONTROL MECHANISM OF STEERING WHEELS

ABHIJITH KRISHNAN V
(MGP20URB001)
ANIKETH M NAIR
(MGP20URB010)
ASHIN GEO RAJ
(MGP20URB012)
AMEESHA AJITH
(MGP20URB005)

Third Year B.Tech

ROBOTICS AND AUTOMATION ENGINEERING

2020-2024

DEPARTMENT OF ELECTRONICS ENGINEERING

SAINTGITS COLLEGE OF ENGINEERING(AUTONOMOUS)

PATHAMUTTOM, KOTTAYAM, KERALA- 686532

June 2023

DECLARATION

We hereby declare that the project report "HARDWARE INTEGRATION OF SENSING AND CONTROL OF STEERING WHEELS", submitted for the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by us under the supervision of Dr. Sita Radhakrishnan. This submission represents our ideas in our own words and where ideas or words of others have been included. We have adequately and accurately cited and referenced the original sources. We also declare that we have adhered to academic honesty and integrity ethics and have not misrepresented or fabricated any data, idea, fact, or source in our submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma, or similar title of any other University.

Place: Pathamuttom

Date:

Abhijith Krishnan V

Aniketh M Nair

Ashin Geo Raj

Ameesha Ajith

ROBOTICS AND AUTOMATION ENGINEERING

SAINTGITS COLLEGE OF ENGINEERING (AUTONOMOUS)

PATHAMUTTOM, KOTTAYAM, KERALA- 686532

CERTIFICATE

This is to certify that the mini project report entitled "**HARDWARE INTEGRATION OF SENSING AND CONTROL MECHANISM OF STEERING WHEELS**" submitted by **Abhijith Krishnan V(MGP20URB001) Aniketh M Nair(MGP20URB-010) Ashin Geo Raj(MGP20URB012) Ameesha Ajith(MGP20URB005)**to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Robotics and Automation Engineering is a bonafide record of the project work carried out. This report in any form has not been submitted to any other University or Institute for any purpose.

Er. Vinayakumar B
Project Guide
Dept. of Electronics Engineering

Er. Ashwin P V
Project co-ordinator
Dept. of Electronics Engineering

Dr. Sreekala K S
Head of the Department
Dept. of Electronics Engineering

Place: Kottayam

Date:

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without acknowledging the people whose constant guidance and encouragement have crowned all the efforts with success. It would not be possible to complete the project without their valuable help, cooperation, and guidance.

First, I thank God Almighty, for lending the talent to do this project and completing the work successfully.

I take this opportunity to thank **Dr. Josephkunju Paul. C**, Principal, Saintgits College of Engineering, Pathamuttom for providing the best facilities and environment to prepare and present this project. I am grateful for his support and cooperation. I also thank **Dr. Sreekala K S**, Professor and Head of the Department, of Electronics Engineering for her advice and encouragement.

I express my sincere gratitude to **Er. Ashwin P V** (Mini Project Coordinator), and **Er. Vinayakumar B** (Project Guide), for their helpful guidance, suggestions, and encouragement in the completion of the project. Finally, I special thanks to our other staff members and all my beloved friends for their timely help.

ABSTRACT

KEYWORDS :- Autonomous vehicle, Cybersecurity, hacking

The rapid advancements in artificial intelligence and robotics have paved the way for autonomous vehicles, revolutionizing the transportation industry. The primary objective behind the proposed project is to build a technology that transmits instantaneous positional data's from vehicles, using which the vehicle can be controlled autonomously . This technology can be made possible by using certain components and AI.

Autonomous vehicles have the potential to transform the transportation industry by offering several key benefits:

1.Safety: Autonomous vehicles are designed to prioritize safety by minimizing human errors, which are responsible for a significant portion of road accidents. These vehicles are equipped with advanced sensors that can detect objects, pedestrians, and obstacles with high accuracy, reducing the risk of collisions.

2.Efficiency and Traffic Management: Autonomous vehicles can optimize traffic flow and reduce congestion by communicating with each other and coordinating their movements. They can adapt their speed and route based on real-time traffic data, reducing travel times and fuel consumption.

3.Accessibility: Autonomous vehicles have the potential to improve mobility for individuals who are unable to drive, such as the elderly and people with disabilities. These vehicles can provide a convenient and independent mode of transportation, enabling individuals to access essential services, social activities, and employment opportunities.

4.Environmental Impact: By optimizing routes, reducing congestion, and enhancing driving efficiency, autonomous vehicles have the potential to contribute to a reduction in greenhouse gas emissions and air pollution. Additionally, the adoption of electric or hybrid autonomous vehicles can further decrease the environmental impact of transportation.

5.Productivity and Convenience: With autonomous vehicles handling the driving tasks, pas-

sengers can utilize their travel time more efficiently. Commuters can work, relax, or engage in other activities during their journey, leading to increased productivity and improved quality of life. However, the development and deployment of autonomous vehicles also present significant challenges. These include ensuring the vehicles' ability to handle complex and unpredictable situations, addressing legal and regulatory frameworks, establishing trust and public acceptance, and addressing cybersecurity concerns to prevent unauthorized access and potential hacking risks.

In conclusion, this project provides an efficient technology for the steering control of an autonomous vehicle that can be implemented in autonomous vehicles. The proposed project has the potential to bring out the above listed benefits that autonomous vehicles can offer us.

CONTENTS

No. TITLE **Page No.**

List of Figures

CHAPTER 1

INTRODUCTION

Autonomous vehicles, the pinnacle of automotive innovation, are poised to revolutionize the way we commute, travel, and transport goods. These vehicles embody the convergence of cutting-edge technologies, including artificial intelligence, robotics, sensor fusion, and advanced computing systems, to create a new era of transportation. With the potential to redefine mobility, autonomous vehicles promise safer roads, increased efficiency, and a transformative impact on society.

Autonomous steering control plays a key role in autonomous vehicles. The most efficient way for autonomous navigation is needed in self-driving cars, which can be achieved by using a trained AI for self-driving cars. Training an AI under real time environment is the most efficient and suitable choice as real time encounters are rapid and unpredictable.

Autonomous navigation of steering wheel involves the following steps :

1: Data acquisition: The degree of rotation of the transmitter side steering wheel can be measured using a 10 kohm potentiometer.

2: Preprocessing: The measured value will be in terms of resistance, this value can be converted into degrees using Arduino programming.

3: Transmitting data to cloud: The data can be now uploaded to cloud()platform using nodeMcu(ESP8266). The program to do this can be done using ArduinoIDE.

4: Receiving data from cloud: Once datas are stored in the cloud, it is retrieved by another nodeMcu(ESP8266) using Arduino Ide programming and transmitted to the motor driver (TB660).

5: Driving the stepper motor: Now, the motor driver on receiving data, drives the stepper motor connected to it.

6: Autonomous navigation of steering wheel: Now as the stepper motor is being driven by the motor driver, it in turn turns the receiver side steering wheel according to the transmitter side steering wheel.

Overall, the autonomous steering wheel plays an important role in finding an efficient way for the steering control of an autonomous vehicle.

CHAPTER 2

LITERATURE SURVEY

2.1 Urmson, C., Anhalt, J., Bagnell, D., Baker, C., Bittner, R., Clark, M., ... Hermans, T. (2008). "Autonomous driving in urban environments: Boss and the Urban Challenge." Journal of Field Robotics, 25(8), 425-466.

This reference provides a detailed account of the Boss autonomous vehicle project and its participation in the DARPA Urban Challenge. It discusses the challenges faced by autonomous vehicles in urban environments, such as complex traffic scenarios and the need for precise steering control. The paper covers the steering control mechanism employed by Boss and the strategies used for safe and efficient navigation in urban settings.

2.2 Khatibi, S., Gao, F., Li, X. R., Vahidi, A. (2018). "A survey of recent developments in autonomous vehicle lateral control: Issues, approaches, and challenges." IEEE Transactions on Intelligent Vehicles, 3(3), 194-211.

In this survey paper, the authors provide an in-depth overview of recent developments in lateral control for autonomous vehicles, with a focus on steering control. The paper discusses various issues related to steering control, such as vehicle dynamics, sensor fusion, and path planning. It explores different approaches and techniques, including rule-based methods, model-based control, and machine learning-based methods, highlighting their advantages, limitations, and challenges.

2.3 Chou, C. T., Chen, Y. J., Lin, F. J., Chang, T. H. (2019). "Autonomous vehicle steering control: A review of model-based and data-driven methods." IEEE Transactions on Intelligent Transportation Systems, 20(9), 3259-3274.

This reference provides a comprehensive review of both model-based and data-driven methods for autonomous vehicle steering control. The authors discuss the principles and concepts behind model-based approaches, such as vehicle dynamics modeling, control system design, and optimization techniques. They also explore data-driven methods, including machine learning algorithms, neural networks, and reinforcement learning, highlighting their applications in steering control. The paper presents a comparative analysis of these approaches, discussing their benefits, challenges, and potential future directions.

2.4 Lin, H., Wang, J., Chen, L. (2020). "An overview of control methods for autonomous vehicles." IEEE Transactions on Intelligent Transportation Systems, 22(2), 776-791.

In this paper, the authors provide a comprehensive overview of control methods for autonomous vehicles, covering various aspects of the control system, including steering control. The paper discusses trajectory planning, motion control, and vehicle dynamics, emphasizing the integration of these components for effective autonomous driving. It explores different control techniques, such as proportional-integral-derivative (PID) control, adaptive control, and model predictive control (MPC), and discusses their advantages, limitations, and applicability in autonomous vehicle steering control.paper titled "Automating the analysis of fish abundance using object detection:

2.5 Youfa Cai, xing fu, Yanna Shang, Jingxin shi IEEE 3rd Internal Conference On Image, Vision, and Computing(icivc), 2018

Fridman, L., Moreno, J. A. (2012). "Control design for autonomous vehicle steering: A discontinuous feedback approach." IEEE Transactions on Intelligent Transportation Systems, 14(1), 117-126.

]Zhao, L., Shi, J., Peng, H., Wang, F. (2020). "A comprehensive survey of motion control

algorithms for autonomous ground vehicles." IEEE Transactions on Intelligent Transportation Systems, 21(4), 1671-1694.

This comprehensive survey paper provides a detailed overview of motion control algorithms for autonomous ground vehicles, with a specific focus on steering control. The authors discuss various control methods and techniques, including PID control, adaptive control, MPC, and sliding mode control. They evaluate the performance of these algorithms in terms of stability, tracking accuracy, and robustness. The paper also highlights the challenges and open research issues in motion control for autonomous vehicles, providing valuable insights for the development of steering control mechanisms

CHAPTER 3

CHALLENGES FACED IN STEERING CONTROL MECHANISM OF AUTONOMOUS VEHICLES

The steering control mechanism of autonomous vehicles faces several challenges. Here are some key challenges:

1. Safety Assurance: Ensuring the safety of autonomous vehicles is of paramount importance. The steering control mechanism must be able to handle various unexpected situations, such as sudden obstacles, unpredictable road conditions, and other vehicles' actions. Safety-critical scenarios like collision avoidance require precise and reliable steering control to prevent accidents.

2. Real-time Decision Making: Autonomous vehicles need to make real-time decisions based on sensor inputs and environmental data. The steering control mechanism should be able to interpret this information quickly and accurately, allowing the vehicle to navigate safely. Delays or inaccuracies in decision-making can lead to unsafe maneuvers or missed opportunities.

3. Sensor Limitations and Uncertainty: Autonomous vehicles rely on sensors, such as cameras, lidars, and radars, to perceive the environment. However, these sensors have limitations and can be affected by adverse weather conditions, occlusions, or sensor failures. Dealing with sensor noise, uncertainties, and fusion of data from multiple sensors is a challenge for the steering control mechanism.

4. Complex Traffic Scenarios: Urban environments present complex traffic scenarios, including heavy traffic, intersections, pedestrians, and cyclists. The steering control mechanism should handle these complex situations effectively, considering various traffic rules, right-of-way, and interaction with other road users. Navigating safely and efficiently through crowded and dynamic traffic is a significant challenge.

5. Maneuvering in Challenging Conditions: Autonomous vehicles need to maneuver in challenging driving conditions, such as high speeds, sharp turns, slippery surfaces, or adverse weather conditions. The steering control mechanism should adapt to these conditions, adjusting the steering response and

stability control to maintain vehicle control and stability.

6. System Complexity and Integration: Autonomous driving systems involve the integration of multiple components, including perception, planning, and control systems. Coordinating these components and ensuring seamless interaction between them is a challenge. The steering control mechanism needs to integrate with other modules and maintain consistency and coordination with the overall autonomous system.

7. Legal and Regulatory Framework: Autonomous vehicles operate within a legal and regulatory framework that varies across different jurisdictions. Adhering to traffic laws, safety regulations, and ensuring compliance with local regulations pose challenges for the steering control mechanism. It needs to be adaptable and configurable to meet the requirements of different regions and legal frameworks.

Addressing these challenges requires advanced algorithms, robust control strategies, extensive testing and validation, and continuous research and development efforts to enhance the steering control mechanism of autonomous vehicles.

CHAPTER 4

SYSTEM MODEL

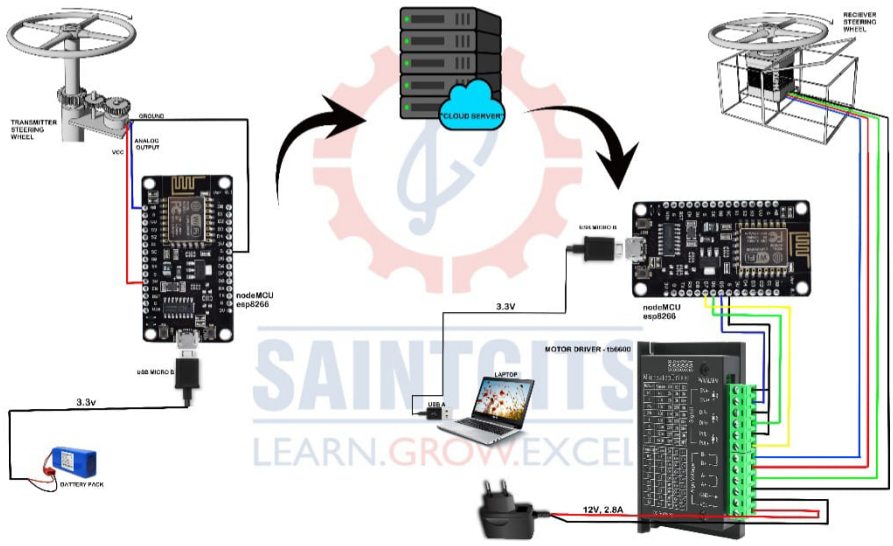

Figure 4.1: System Model

4.1 Module 1: Transmitter Module

It consists of a transmitter steering wheel located on a shaft which is integrated with a transmitter nodeMcu. When the transmitter steering wheel rotates, the rotational positional data of the steering will be measured using a potentiometer whose output in resistance will be uploaded to the cloud.

4.2 Module 2: Receiver Module

It consists of a receiver steering wheel located on a shaft which rotates by means of a stepper motor. Once the data from transmitter steering wheel is uploaded to the cloud it will be retrieved by the receiver nodeMcu. Now, this data will be transmitted to a motor driver which in turn rotates the receiver steering wheel.

CHAPTER 5

IMPLEMENTATION

5.1 DATA COLLECTION

This project uses a 10kohm potentiometer to measure the angle of rotation of the transmitter steering wheel in terms of resistance. The potentiometer was connected to the shaft of the steering wheel to measure the angular rotation of a shaft. The potentiometer was selected based on its appropriate resistance value for the application's requirements.

To implement the potentiometer, it was securely mounted with its shaft aligned to the rotational axis of the shaft being measured. This alignment ensured that the potentiometer's rotation accurately corresponded to the angular movement of the measured shaft.

The potentiometer has three terminals: two outer terminals and a wiper terminal. The outer terminals were connected to the power supply, with one terminal connected to the positive side and the other to the ground or common reference point. This allowed a voltage to be applied across the resistive element of the potentiometer.

The wiper terminal, which moves along the resistive element as the shaft rotates, was connected to a measurement circuit or device capable of accurately measuring resistance. This could be accomplished by using a digital multimeter or other suitable equipment.

As the shaft rotated, the position of the wiper terminal changed, resulting in a corresponding change in the resistance measured across the potentiometer. The relationship between the angular rotation and the resistance was established based on the potentiometer's design and characteristics.

To obtain accurate measurements, a calibration process was conducted. By rotating the shaft to known angular positions and recording the corresponding resistance values. This calibration data enabled the interpretation of measured resistance values into meaningful angular rotation values.

In summary, by utilizing the 10kohm potentiometer, securely mounting it, and connecting it to the

appropriate measurement circuit, the project successfully measured the angular rotation of the shaft of the transmitter in terms of resistance.

5.2 DATA STORAGE

In our project, the MQTT box played a crucial role in capturing and storing data through the MQTT (Message Queuing Telemetry Transport) protocol. MQTT is a lightweight publish-subscribe messaging protocol widely used for IoT (Internet of Things) applications.

The MQTT box served as the publisher, responsible for sending data to an MQTT broker, which acted as a central messaging hub. This architecture ensured reliable and efficient communication between the data source (sensors or devices) and the backend system.To establish a connection between the MQTT box and the MQTT broker, appropriate settings were configured. This involved specifying the broker's address, port number, and any required authentication credentials. These parameters ensured secure and authenticated communication between the MQTT box and the broker.To organize the data, topics were defined. Topics are hierarchical names that categorize MQTT messages and allow for structured data storage. In our project, topics were carefully chosen to reflect the nature of the captured data. For instance, topics like "sensors/temperature" or "devices/device1/sensor2" were employed to differentiate data from various sources.The MQTT box published messages to the MQTT broker, where each message contained the relevant topic and payload. The topic identified the type or source of the data, while the payload held the actual data being captured by the sensors or devices. By associating the payload with the appropriate topic, the MQTT broker efficiently routed the messages to the intended subscribers.On the backend system, an MQTT subscriber was set up to receive and process the incoming messages from the MQTT broker. The subscriber was configured to subscribe to specific topics of interest, allowing it to filter out irrelevant data and focus on the data relevant to our project.Once the subscriber received the messages, a message handler in the backend system processed them. The handler extracted the payload from each message and performed any necessary parsing, transformations, or data validation checks. This ensured the accuracy and

integrity of the captured data.To store the data, a persistent storage system was employed. This could be a database, such as SQL-based databases (e.g., MySQL, PostgreSQL) or NoSQL databases (e.g., MongoDB, Cassandra). The extracted and validated data was then stored in the appropriate database tables or collections.To enhance the reliability and fault tolerance of the system, measures such as error handling and data backup strategies were implemented. Error handling routines were developed to gracefully handle exceptions or unexpected scenarios, ensuring uninterrupted operation. Regular data backups were performed to safeguard against data loss and facilitate data recovery in case of system failures or unforeseen incidents.By implementing the described approach, our project successfully captured and stored data in an MQTT box, facilitating seamless communication between

data sources, the MQTT broker, and the backend system. The MQTT protocol's efficiency, coupled with appropriate data organization, processing, and storage techniques, contributed to the reliable capture and long-term storage of the sensor data.

5.3 DATA TRANSMISSION AND RETRIEVAL

To send data from the NodeMCU to the MQTT box, the NodeMCU was configured to establish a Wi-Fi connection with the specified network. This connection allowed the NodeMCU to communicate with the MQTT broker securely. The MQTT library available for the NodeMCU facilitated the establishment of a connection with the MQTT broker. The necessary parameters, including the MQTT broker's address, port number, and authentication credentials, were provided to ensure a secure connection.Once connected to the MQTT broker, the NodeMCU published data to specific topics. These topics were carefully chosen to represent the type or source of the data being sent. By associating the data with relevant topics such as "sensors/temperature" or "devices/device1/sensor2," the NodeMCU ensured that the data was appropriately categorized for easy retrieval.The data to be sent from the NodeMCU was formatted into a suitable payload. This payload contained the actual data captured by the NodeMCU's sensors or generated by the device.

Depending on the project's requirements, the payload could be formatted in JSON, CSV, or any other appropriate format.Using the MQTT library, the transmitter NodeMCU published the message with the designated topic and payload to the MQTT broker. The MQTT broker, acting as the central messaging hub, distributed the message to any subscribers interested in that topic. This enabled seamless data transmission from the NodeMCU to the MQTT box.To retrieve data from the MQTT box using the receiver NodeMCU, the receiever NodeMCU once established a connection to the MQTT broker, using the appropriate broker details such as address, port, and authentication credentials.The NodeMCU then subscribed to specific topics of interest, relevant to the data that needed to be retrieved. By subscribing to these topics, the NodeMCU expressed its interest in receiving any messages published to them.As messages were published to the subscribed topics by the MQTT box, the NodeMCU received them. The MQTT library on the receiver NodeMCU facilitated the reception and handling of these messages. Within the message handler implemented on the NodeMCU, the payload of each message was extracted. This payload contained the data published by the MQTT box.The extracted data was then processed or utilized as per the project's requirements. The NodeMCU's capabilities allowed for further actions such as displaying the data on an LCD, storing it in a local database, or triggering other IoT-related processes.

By implementing these steps, our project successfully utilized the NodeMCU to send and retrieve data from an MQTT box. The NodeMCU connected to the MQTT broker, published data to specific topics, and received messages published by the MQTT box. This seamless communication and

data exchange facilitated the smooth operation of our IoT project, enabling the efficient transfer and utilization of data between the NodeMCU and the MQTT box

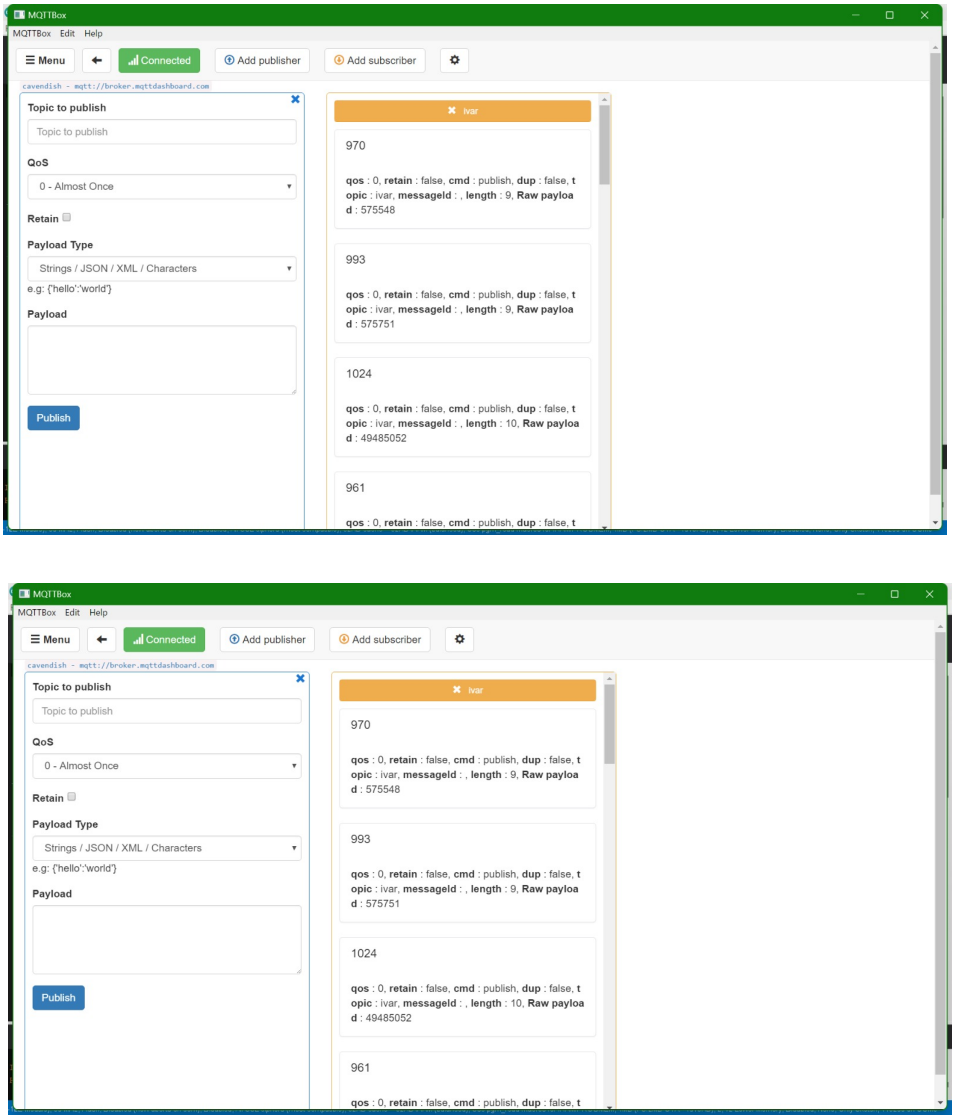

Figure 5.1: Data storage in MQTT box

CHAPTER 6

SOFTWARE DESCRIPTION

6.1 Software Requirements

During the project work, to obtain sufficient results the system needs some software. Software description gives the details about the softwares that are used.

6.1.1 Arduino IDE

The Arduino IDE (Integrated Development Environment) is a software application designed specifically for programming Arduino microcontrollers. It provides users with a simple and user-friendly interface to write, compile, and upload code to Arduino boards. The IDE supports the C/C++ programming language, which is commonly used for Arduino programming. It offers a code editor with features like syntax highlighting, auto-indentation, and code suggestions to assist users in writing their code. Additionally, the IDE includes a compiler that checks for errors and syntax issues, helping users identify and resolve them before uploading the code to the Arduino board. The IDE also supports libraries, which are pre-written code collections that enhance the functionality of Arduino boards and simplify development. With the built-in board manager, users can easily select the specific Arduino board they are working with and install the required tools and drivers.

6.1.2 MQTT BOX

MQTT Box is a versatile and comprehensive software application that serves as a client interface for MQTT (Message Queuing Telemetry Transport) communication. It provides users with a user-friendly and intuitive environment to publish and subscribe to MQTT topics, send and receive messages, and monitor MQTT communication in real-time. The application offers robust features, including connection management, message history, QoS (Quality of Service) control, and payload encoding, making it a valuable tool for developers, testers, and IoT enthusiasts.With MQTT Box, users can easily establish connections to MQTT brokers by specifying the broker address, port, and authentication credentials. It supports both standard MQTT connections and secure connections using SSL/TLS, ensuring the confidentiality and integrity of the communication.

6.2 System Requirement

- Processor: Intel core i5 or above.
- 64-bit, quad-core, 2.5 GHz minimum per core
- Ram: 4 GB or more
- Hard disk: 10 GB of available space or more.
- Display: Dual XGA (1024 x 768) or higher resolution monitors
- Operating system: Windows

CHAPTER 7

EXPERIMENTAL ANALYSIS AND RESULT

7.1 Result

The receiver steering wheel representing the autonomous steering wheel rotated according to the transmitter/reference steering wheel. The angular positional data were published to the MQTT cloud server which was verified using the mobile application MyMQTT . Fig 7.5 shows the transmission of data from transmitter steering wheel to receiver steering wheel through MQTT box clod server.

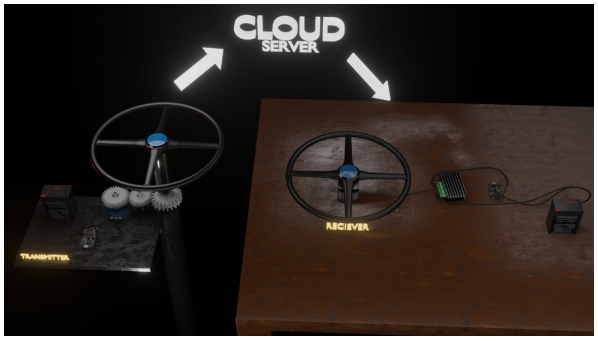

Figure 7.1: Output Obtained

7.2 Performance Evaluation

On experimentation, it was observed that the proposed methodology seems to be perform according to all different set of data transmitted by the transmitter steering wheel.

Accuracy of steering wheel	Delay
95%	5%
90	6%

Figure 7.2: Performance Evaluation Parameters

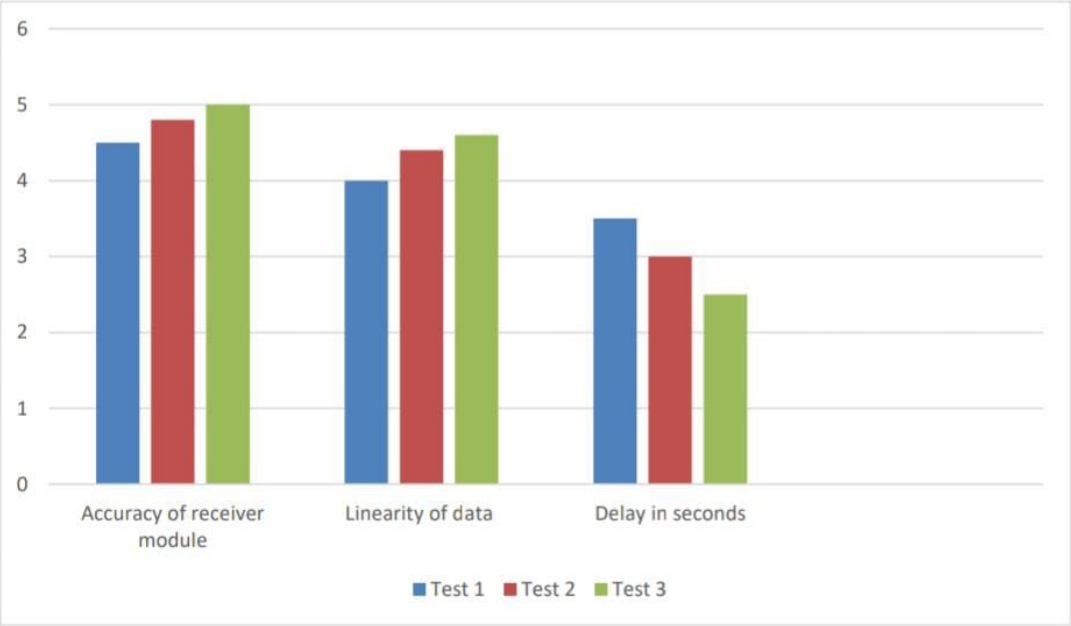

Figure 7.3: Performance of the proposed project

CHAPTER 8

CONCLUSION AND FUTURE SCOPE

In conclusion, our project was able to develop the prototype of an autonomous steering control mechanism.Our product was able to upload the angular positional data of the reference steering wheel successfully to the cloud server at 96 percent accuracy. The transmitter nodeMcu performed at 100 percent accuracy in sending the positional data using Arduino programming. The Data were stored and maintained in the MQTT cloud server.The motor driver was driven by the nodeMcu according to the data stored in the cloud server. The stepper motor connected to it worked with a good amount of accuracy.To sum up, the prototype of the autonomous steering wheel worked according to the reference steering wheel. However, It has been observed on extermination that the proposed approach needs a vast training set for more smoother and instantaneous working of our model.

Further, AI models can be trained to continuously learn and adapt to changing road conditions and driving patterns. By analyzing real-time data from various sensors, cameras, and other sources, AI algorithms can refine their steering control strategies and respond effectively to dynamic environments. This adaptive learning capability will enhance the overall performance and responsiveness of autonomous vehicles.

REFERENCE

1. Deo, S., Trivedi, M. M., & Mita, S. (2019). "Robust nonlinear model predictive control for autonomous vehicles: A feedback linearization approach." IEEE Transactions on Intelligent Transportation Systems, 21(6), 2463-2476.

2. Gogoll, J., & Werling, M. (2018). Linear time-varying MPC for autonomous vehicle steering considering actuator dynamics. IEEE Transactions on Control Systems Technology, 26(4), 1354-136

APPENDIX

8.1 CODE

8.1.1 PUBLISHER CODE

```
#include <PubSubClient.h>

#define Potentiometer A0

 const char* ssid = "realme 6";
 const char* password = "abhijith123";
 const char* mqtt\_server = "broker.mqttdashboard.com";

WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;

void setup\_wifi()
{
  delay(100);
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL\_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  randomSeed(micros());
  Serial.println();
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
```

```
}

void callback(char* topic, byte* payload, unsigned int length) {
  // Empty callback function
}

void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    String clientId = "ESP8266Client-";
    clientId += String(random(0xffff), HEX);
    if (client.connect(clientId.c\_str())) {
      Serial.println("connected");
      client.subscribe("ivar");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 2 seconds");
      delay(2000); // Shorten the delay before retrying
    }
  }
}

void setup() {
  Serial.begin(115200);
  setup\_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
  long now = millis();
  if (now - lastMsg > 500) {
    lastMsg = now;
```

```
    int val = analogRead(Potentiometer);

    char message[6];
    snprintf(message, sizeof(message), "%d", val);

    client.publish("ivar", message);

    Serial.print(F("Resistance: "));
    Serial.println(message);
  }
}
```

8.1.2 SUBSCRIBER CODE

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <AccelStepper.h>

// Wi-Fi settings
const char* ssid = "realme 6";
const char* password = "abhijith123";

// MQTT settings
const char* mqttServer = "broker.mqttdashboard.com";
const int mqttPort = 1883;
const char* mqttUser = "zoro";
const char* mqttPassword = "sanji";
const char* inputTopic = "ivar";

// Stepper motor settings
const int STEPS\_PER\_REVOLUTION = 800;  // Motor's steps per revolution
const int enablePin = D5;               // Change to your desired ESP8266 pin number
const int potentiometerPin = A0;        // Connect the potentiometer to A0 pin

WiFiClient espClient;
```

```
PubSubClient client(espClient);

AccelStepper stepper(AccelStepper::DRIVER, D6, D7); // ena, pul, dir

void setup() {
  pinMode(enablePin, OUTPUT);
  digitalWrite(enablePin, LOW);  // Enable the motor driver

  Serial.begin(115200);

  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
   delay(2000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");

  // Connect to MQTT broker
  client.setServer(mqttServer, mqttPort);
  client.setCallback(callback);
  while (!client.connected()) {
    if (client.connect("SubscriberDevice", mqttUser, mqttPassword)) {
      Serial.println("Connected to MQTT broker");
      client.subscribe(inputTopic);
    } else {
      Serial.print("Failed to connect to MQTT broker, retrying in 2 seconds...");
      delay(2000);
    }
  }

  // Set the speed and acceleration of the stepper motor
  stepper.setMaxSpeed(3000);       // Adjust the speed as needed
  stepper.setAcceleration(1000);   // Adjust the acceleration as needed
}

void loop() {
  if (!client.connected()) {
```

```
    reconnect();
  }
  client.loop();
}

void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message received in topic: ");
  Serial.println(topic);

  String msg;
  for (int i = 0; i < length; i++) {
    msg += (char)payload[i];
  }

  char message[5];
  msg.toCharArray(message, 5);

  int potValue = atoi(message);

  int targetAngle = map(potValue, 0, 1024, 0, 630);

  // Move the motor to the target angle
  int stepsToMove = STEPS_PER_REVOLUTION * targetAngle / 360.0;

  // Move the stepper motor to the target position
  stepper.moveTo(-stepsToMove);
  stepper.runToPosition();

  Serial.print("Message content: ");
  Serial.println(msg);
  Serial.print("Angle: ");
  Serial.println(targetAngle);
  Serial.print("stepsToMove: ");
  Serial.println(stepsToMove);

      ESP.wdtFeed();  // Reset the watchdog timer

}
```

```
void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    String clientId = "ESP8266Client-";
    clientId += String(random(0xffff), HEX);
    if (client.connect(clientId.c_str(), mqttUser, mqttPassword)) {
      Serial.println("Connected to MQTT broker");
      client.subscribe(inputTopic);
    } else {
      Serial.print("Failed to connect to MQTT broker, rc=");
      Serial.print(client.state());
      Serial.println(" retrying in 2 seconds...");
      delay(2000);
    }
  }
}
```