International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 8, Issue 10, pp 94-104, October-2024

https://doi.org/10.47001/IRJIET/2024.810015

Design and Evaluation of a Throttle Controller for Common Rail Diesel Engines

[1]Dedy Novindra, [2]*Nazaruddin Sinaga, [3]Eflita Yohana

[1,2,3]Department of Mechanical Engineering, Faculty of Engineering, Diponegoro University, Jalan Prof. Soedarto, Tembalang, Semarang 50275, Central Java, Indonesia

[1]E-mail: dedynovindra10@gmail.com

*Corresponding Author's E-mail: nsinaga19.undip@gmail.com

***Abstract* - Currently, engine remapping is widely utilized to enhance the performance of diesel and petrol engines. Given that a substantial amount of data must be collected during the optimization process, a data acquisition system is necessary to organize and retrieve data automatically. This research focuses on the design, development, and testing of a data acquisition system for controlling and measuring the throttle pedal in diesel vehicles equipped with a common rail system. The system is designed to manage the Throttle Position Sensor (TPS) and monitor vehicle operational parameters in real-time. The proposed device employs Arduino as the primary hardware, with TechStream software for vehicle data acquisition, and a web-based application developed using the Python programming language, and displays data in real-time in the form of graphs and tables. From the test results, it can be concluded that the developed acquisition system produces precise and sensitive throttle position settings. Its operation is also straightforward, safe, and requires a relatively short amount of time.**

Keywords: Data Acquisition, Arduino, TechStream, Python, Diesel Engines, Throttle Position.

I. INTRODUCTION

The current advancements in science and technology are reflected in the growing use of more efficient automatic control systems, particularly in the automotive industry. These advancements are most evident in Engine Control Units (ECUs), computerized systems that automatically manage various aspects of engine performance, including fuel mixture, ignition timing, and engine speed [1]. ECUs can be reprogrammed to achieve optimal performance based on the owner's preferences, a process known as remapping [2, 3].

With prolonged use and vehicle operation, the performance of the ECU may degrade, necessitating recalibration to restore its functionality. To accomplish this, data from the ECU must be accessed using an engine scanner, allowing the monitoring of various vehicle parameters [4]. Therefore, there is a need to develop a data acquisition system that can facilitate the remapping process by automatically adjusting engine settings

In an electronic throttle system, the Throttle Position Sensor (TPS) plays a crucial role in precisely monitoring and adjusting the throttle position. This accuracy is essential for optimizing engine performance, fuel efficiency, and emission control [5, 6]. In diesel vehicles equipped with a Common Rail system, the precise management of air and fuel flow is central to engine operation. However, despite the advantages offered by this technology, challenges such as backfire, misfire, and knocking remain significant issues that affect engine reliability and performance [7].

Electronic Throttle Control Systems (ETCS), integrated with the Throttle Position Sensor (TPS), offer significant advantages in controlling longitudinal speed and enhancing fuel efficiency. This system allows for more responsive and precise throttle adjustments, with microcontrollers such as the Arduino Mega 2560 playing a key role in refining this control [5, 6]. As a microcontroller, Arduino provides high adaptability and compatibility across various applications, enabling the implementation of flexible and efficient control systems. In this context, data acquisition technology is critical to ensuring the system's overall effectiveness. Software such as TechStream facilitates real-time collection of vehicle parameters, providing valuable insights into operational conditions and system performance. Leveraging this data is essential for evaluating the precision of throttle control, ensuring optimal performance across different operating conditions.

This research aims to develop an automated system for the Throttle Position Sensor (TPS) to simplify the engine remapping process. The focus is on achieving precise control, optimizing performance, and improving the safety and reliability of the system [8]. This system is specifically designed to address common engine performance issues, such as aggressive driving styles that can cause inconsistent engine loads and negatively impact fuel consumption [4]. To

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 8, Issue 10, pp 94-104, October-2024

https://doi.org/10.47001/IRJIET/2024.810015

accurately monitor transitions in vehicle status, advanced modelling and computing technologies are utilized [9, 10].

The objectives of the research are to design a data acquisition system for controlling and adjusting the throttle pedal in vehicles and to develop a system capable of collecting real-time vehicle parameters. Additionally, the precision of the throttle position control system and the accuracy of the data acquisition system will be tested through a series of evaluations.

II. LITERATURE REVIEW

2.1 Data Acquisition

Data acquisition is the process of collecting information from various sources and making it available for analysis or storage [11, 12]. In the automotive industry, data acquisition analysis plays a crucial role in optimizing driving conditions, reducing the need for test vehicles, and enhancing the overall functionality of vehicles. This research focuses on measuring data acquisition for the control of various vehicle parameters, including the adjustment of throttle position.2.2 Arduino-Based Data Acquisition System.

2.2 Arduino-Based Data Acquisition System

Arduino is an open-source platform that encompasses both hardware and software, making it a popular choice for developing interactive electronic projects [13]. Arduino microcontroller boards can be programmed using the C or C++ programming languages with the aid of the Arduino Integrated Development Environment (IDE). The development of an Arduino-based data acquisition system aims to control the throttle pedal in vehicles and collect engine performance parameters in real time. This system comprises key components that facilitate data collection, integration between Arduino and Python, and analysis of engine performance measurements [14].

A Digital-to-Analog Converter (DAC) is an electronic component that converts digital signals into analogue signals [15]. Its primary function is to regulate the voltage required for calibration and system control. The DAC transforms digital data into an analogue signal that can be controlled or monitored, allowing for the adjustment of output voltage according to specified parameters [16].

2.3 Python: Data Processing Software

2.3.1 Backend: Flask and SQLAlchemy

Flask is a lightweight Python framework for building web applications. It is a versatile tool that allows for the creation of interactive web applications by utilizing HTML, CSS, and JavaScript technologies, with Python as the core programming language [17]. In this system, Flask is used to present the user interface and to provide an Application Programming Interface (API), including endpoints such as API/save-data and API/live-data. SQLAlchemy is a SQL toolkit for Python that provides Object Relational Mapping (ORM) and is used for data modelling and database management.

2.3.2 Frontend: Data Visualization

Hypertext Markup Language (HTML) is used to structure web pages, facilitating the creation and organization of content on those pages [18]. Cascading Style Sheets (CSS) are employed to style and format web pages, enhancing their visual appeal and user-friendliness. By utilizing CSS, developers can control various visual aspects, including colour, size, font type, spacing, and other properties, thereby improving the overall aesthetics of the page [19]. JavaScript is utilized for interactivity, enabling real-time data updates and graphical visualizations through libraries such as Chart.js.

2.4 Accelerator Pedal Position (APP)

APP are sensors used to detect the position of the accelerator pedal in a vehicle. This is shown in Figure1.

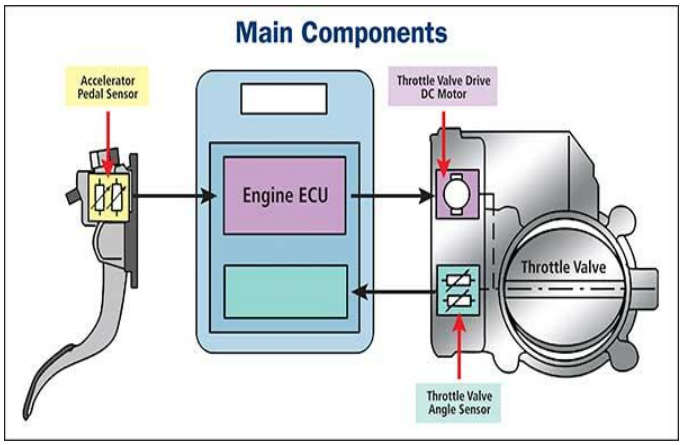

Figure 1: Accelerator Pedal Position [20]

The APP system operates by applying voltage to the terminals, which varies proportionally with the throttle pedal's operating angle. The characteristics of the pedal sensor exhibit a linear voltage range of 0–5V corresponding to an angle range of 0 to 125 degrees [21]. The signal from the main terminal (MAIN) represents the actual throttle pedal angle, which is utilized to control the engine, while the signal from the backup terminal (SUB) conveys the status of the MAIN circuit and is employed to verify the APP sensor [22]. The positions of the APP sensor components are illustrated in Figure 2.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 8, Issue 10, pp 94-104, October-2024

https://doi.org/10.47001/IRJIET/2024.810015

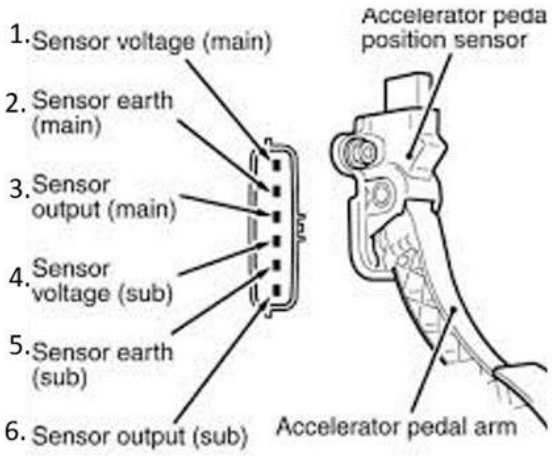

Figure 2: APP Sensor

The sensor structure consists of six terminals:

1. Sensor Voltage (Main)
2. Sensor Ground (Main)
3. Sensor Output (Main)
4. Sensor Voltage (Sub)
5. Sensor Ground (Sub)
6. Sensor Output (Sub)

2.5 TechStream Toyota

To access sensor parameter data and reset and read error codes on Toyota vehicles. This is shown in Figure 3.

Figure 3: TechStream Toyota Software

2.6 Percentage Error

Percentage error quantifies the accuracy of an estimate by comparing it to the known true value [23, 24].

$$PE = \frac{Av - Ev}{Ev} \; x \; 100\%$$

Notation:

PE: Percentage Error
Ev: Measured Value
Av: Actual Value.

III. RESEARCH METHODOLOGY

3.1 Research Workflow

The research workflow includes the stages of design, development, testing, results, and evaluation. It is illustrated in Figure 4.

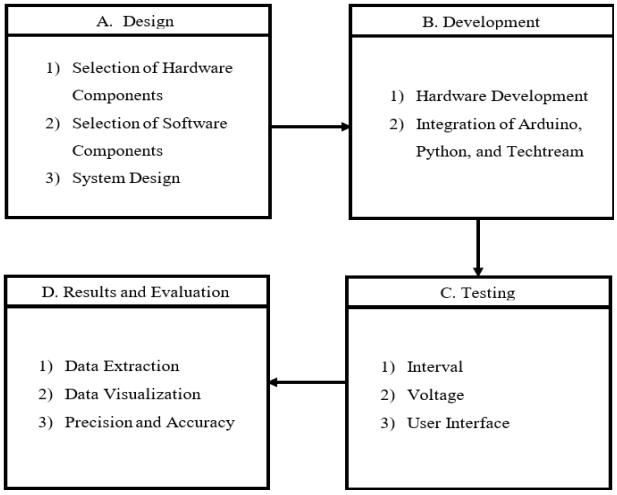

Figure 4: Research Workflow

3.2 Design of a Data Acquisition System

The developed system consists of two main components:

a) Hardware component: utilizes Arduino to control the TPS.
b) Software component: employs Flask as a web server to manage the user interface, and SQLAlchemy for data storage in the database.

3.3 Hardware Development

The hardware utilized in this system is illustrated in Figure 5. These components include various essential elements that support the operation and functionality of the data acquisition system.

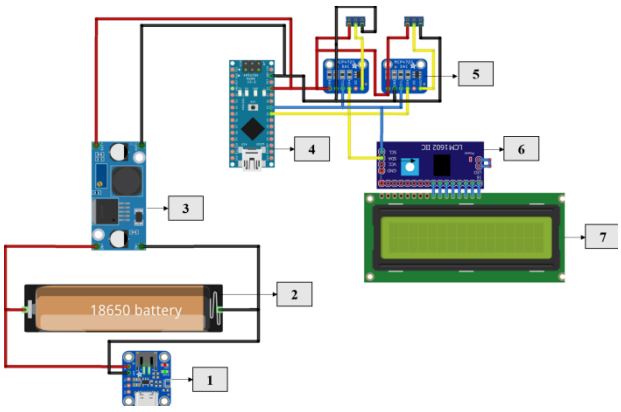

Figure 5: Components of the controller

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 8, Issue 10, pp 94-104, October-2024

https://doi.org/10.47001/IRJIET/2024.810015

Description of Figure 5:

1. USB Type C
2. Battery
3. Step-Up Regulator XL6009E1
4. Arduino Nano
5. DAC MCP 4725
6. LCM 1602
7. LCD Display

3.4 Integration of Arduino, Python Flask, and TechStream

The integration system of Arduino, Python Flask, and TechStreamis illustrated in Figure 6.

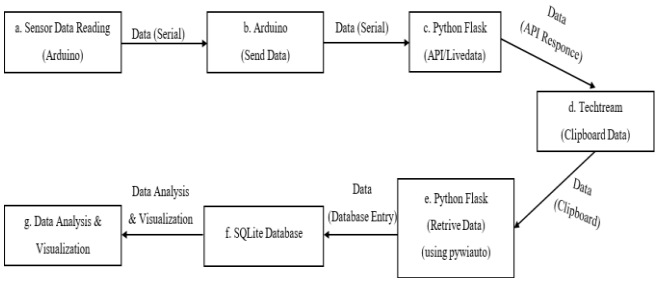

Figure 6: Integration of Arduino, Python Flask, and TechStream

a) Arduino (Sensor Data Reading): Arduino reads data from the vehicle's sensors.
b) Arduino (Send Data): The data read from the sensors is transmitted via the serial port to Python Flask.
c) Python Flask (Receive Data): Python Flask receives data from Arduino through the API endpoint /api/livedata.
d) TechStream (Clipboard Data): TechStream retrieves vehicle diagnostic data and stores it in the clipboard.
e) Python Flask (Retrieve Data using pywinauto): Python Flask uses pywinauto to extract data from the TechStream clipboard.
f) Python Flask (Save Data to SQLite Database): Data received from Arduino and TechStreamis stored in an SQLite database.
g) SQLite Database (Store Data): The SQLite database stores the data collected from Arduino and TechStream.
h) Data Analysis & Visualization: The data stored in the SQLite database is analyzed and visualized according to requirements.

3.5 Development of Python Flask Code

3.5.1 Data Retrieval from TechStream

a) Library Inclusions

To build a web application with Flask and support various additional features, several Python libraries areutilized, as follows:

- Flask: Framework for creating web applications.
- serial: For serial communication with Arduino.
- list_ports: For obtaining a list of available serial ports.
- SQLAlchemy: ORM for managing the SQLite database.
- datetime: For time management.
- pywinauto: For interacting with Windows applications (TechStream).
- random: For generating random data (for testing purposes).

```
from flask import Flask, render_template, jsonify, request
import serial
from serial.tools import list_ports
from sys import platform
from os.path import basename
from time import sleep
from flask_sqlalchemy import SQLAlchemy
import datetime
from pywinauto import Desktop, clipboard
import random
```

Figure 7: Library Inclusions

By using these libraries, the web application can access various features such as database management, serial communication, and interaction with Windows applications.

b) Clipboard Data Processor

The clipboard data processor retrieves data copied to the clipboard and processes it according to predefined headers. Code Snippet for Clipboard Data Processing is depicted in Figure 8.

```
dataDariAplikasi = dict((h, None) for h in headersDataAplikasi)
try:
    desktop = Desktop(backend="Win32")['Techstream (Ver 12.00.127)']
    desktop.menu_select("Function->Copy Data List to Clipboard")
    for l in clipboard.GetData().splitlines()[10:]:
        header = ""
        for h in headersDataAplikasi:
            if h in l: header = h

        if header != "":
            try:
                item = float(l[len(header):].strip().split()[0].replace(",", "."))
            except: continue
            dataDariAplikasi[header] = item
except Exception as e:
    print(e)
sekarang = datetime.datetime.now()
```

Figure 8: Clipboard Data

3.5.2 Voltage and Interval Settings

Voltage settings are configured by sending commands through the serial port using the pyserial library. The steps to configure voltage and interval are as follows:

a) Serial Connection and Voltage Configuration

The serial port is configured for communication with the device. Data is transmitted to adjust the voltage. The code snippet is displayed in Figure 9.

International Research Journal of Innovations in Engineering and Technology (IRJIET)
ISSN (online): 2581-3048
Volume 8, Issue 10, pp 94-104, October-2024
https://doi.org/10.47001/IRJIET/2024.810015

```python
@app.route("/api/serial/<port>")
def koneksi_serial_api(port):
    global SERIAL
    if port == "disconnected":
        if SERIAL is not None:
            try: SERIAL.close()
            except: pass
            SERIAL = None
        return jsonify({"status": "oke"})

    if platform == "linux": port = f"/dev/{port}"
    try:
        SERIAL = serial.Serial(port, timeout=1, baudrate=9600)
    except:
        SERIAL = None
        return jsonify({"status": "failed"})

    return jsonify({"status": "oke"})
```

Figure 9: Serial Connection and Voltage Configuration

b) Transmission of Voltage Data

The voltage is transmitted to the device via the serial port. The code snippet is shown in Figure 10.

```python
@app.route("/api/tps/<value>")
def tps_api(value):
    data = f"{value}\r\n".encode("utf-8")
    print(data)
    if SERIAL is not None:
        try:
            SERIAL.write(data)
            sleep(0.1)
        except: return jsonify({"status": "failed"})

    return jsonify({"status": "oke"})
```

Figure 10: Transmission of Voltage Data

3.5.3 Data Visualization

Data visualization is presented in the form of graphs and tables within the web application. The web page for visualization is rendered using Flask. The code snippets are shown in Figures 11 and 12.

a) Route for the Graphs Page

```python
@app.route("/grafik5")
def grafik5_page():
    return render_template("grafik5.html")
```

Figure 11: Route for the Graphs Page

b) Route for the Tables Page

```python
@app.route("/table")
def table_asli():
    return render_template("table-asli.html")
```

Figure 12: Route for the Tables Page

3.5.4 Data Storage

The collected data is stored in an SQLite database using SQLAlchemy. The storage process involves receiving data through an API endpoint and saving it into database tables as shown on Figure 13 and 14.

a) API Endpoint for Data Storage

```python
@app.post("/api/simpan-data")
def simpan_data_api():
    req = request.get_json(force=True)
    waktu = datetime.datetime.now()
    for key, value in req.items():
        tps_set, putaran_set = key.split(",")
        data = Data()
        data.tps_set = tps_set
        data.putaran_set = putaran_set

        data.tps = value["tps"]
        data.putaran_mesin = value["putMesin"]
        data.putaran_roda = value["putRoda"]
        data.load_cell = value["loadCell"]
        data.torsi_roda = value["torsiRoda"]
        data.daya_roda = value["dayaRoda"]
        data.torsi_mesin = value["torsiMesin"]
        data.bbm = value["bbm"]
        data.waktu = waktu

        db.session.add(data)
        db.session.commit()
    return jsonify({"status": "ok"})
```

Figure 13: API Endpoint for Data Storage

b) Data Model

```python
class Data(db.Model):
    id = db.Column(db.Integer, primary_key=True)

    tps_set = db.Column(db.Double)
    putaran_set = db.Column(db.Double)

    tps = db.Column(db.Double)
    putaran_mesin = db.Column(db.Double)
    putaran_roda = db.Column(db.Double)
    load_cell = db.Column(db.Double)
    torsi_roda = db.Column(db.Double)
    daya_roda = db.Column(db.Double)
    torsi_mesin = db.Column(db.Double)
    bbm = db.Column(db.Double)
    waktu = db.Column(db.DateTime, nullable=False, default=datetime.datetime.now)
```

Figure 14: Data Model

3.6 Arduino Voltage Settings

Voltage Settings: The voltage is controlled through a Digital-to-Analog Converter (DAC) to adjust the throttle position sensor (TPS).

```
void setVoltageAtas(float voltageAtas) {
  //if (voltageAtas < 1.6 || voltageAtas > 5.0) return;
  //Serial.println(voltageAtas);
  int vAtas = (voltageAtas * 4095.0) / 5.0;
  int vBawah = ((voltageAtas - 0.8) * 4095.0) / 5.0;
  dacAtas.setVoltage(vAtas, false);
  dacBawah.setVoltage(vBawah, false);
}
```

- int vAtas and int vBawah: Calculate the voltage values for the DAC based on the input voltageAtas.
- dacAtas.setVoltage(vAtas, false): Sets the voltage on the upper DAC.
- dacBawah.setVoltage(vBawah, false): Sets the voltage on the lower DAC.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 8, Issue 10, pp 94-104, October-2024

https://doi.org/10.47001/IRJIET/2024.810015

IV. RESULTS AND ANALYSIS

4.1 Results of Data Acquisition Control

4.1.1 Data Retrieval from TechStream

Data from the vehicle system is obtained using the TechStream application. This data is copied to the clipboard and subsequently accessed by the Python Flask application. This process is illustrated in Figure 15.

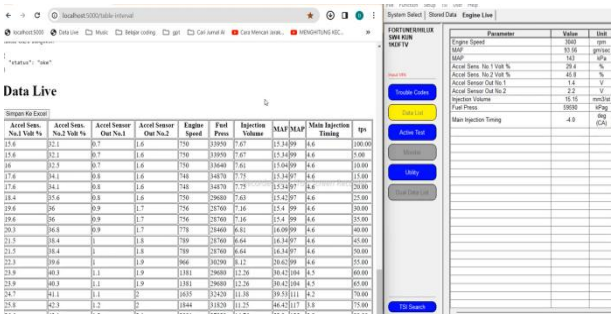

Figure 15: Data Extraction from TechStream

4.1.2 Voltage Settings and Intervals

The required voltage and data acquisition intervals can also be adjusted according to needs. This is illustrated in Figure 16.

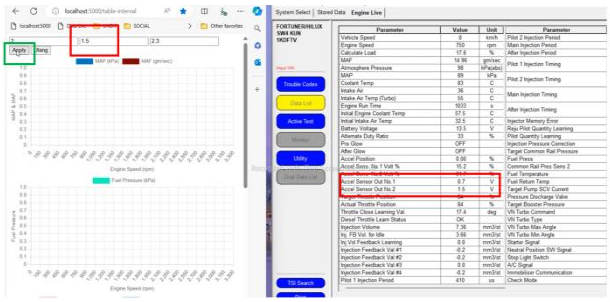

Figure 16: Voltage Settings and Intervals

4.1.3 Data Visualization

The collected data is displayed in graphical form, facilitating easier visual analysis. This is illustrated in Figure 17.

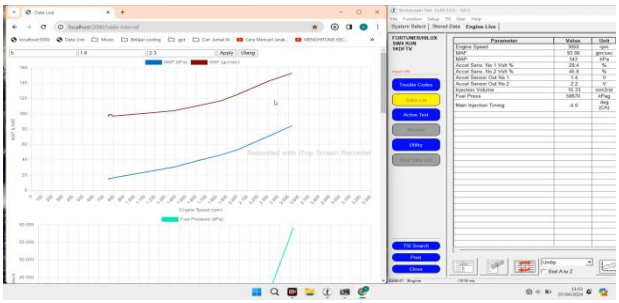

Figure 17: Data Visualization

4.1.4 Data Storage

The data obtained from Arduino and TechStreamis stored in Excel format. This is illustrated in Figure 18.

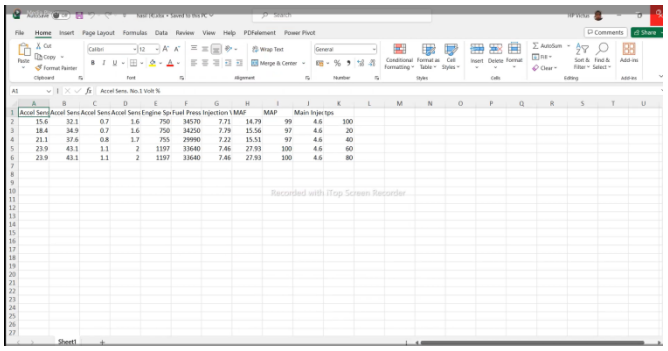

Figure 18: Data Storage

Overall, the system provides an effective solution for data acquisition, analysis, and visualization from vehicles, leveraging the integration of Arduino, Python Flask, and TechStream.

4.1.5 Application of Data Acquisition in Vehicles

The application of data acquisition in vehicles is illustrated in Figure 19, which depicts the implementation of data acquisition tools within the real-world context of vehicles.

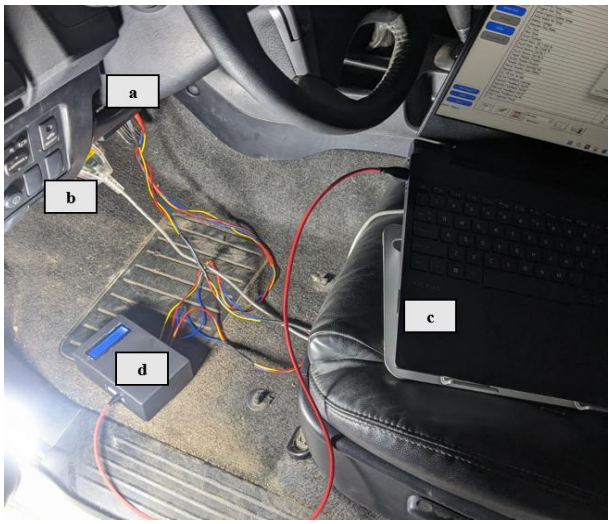

Figure 19: Application of Data Acquisition in Vehicles

a) Data acquisition cable connecting Arduino to the pedal for Throttle Position Sensor (TPS) adjustment.
b) TechStream cable for extracting data from the vehicle.
c) Laptop serving as the user interface for adjusting voltage and intervals.
d) Custom-built tool for TPS control data acquisition.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 8, Issue 10, pp 94-104, October-2024

https://doi.org/10.47001/IRJIET/2024.810015

4.2 Measurement Results (Percentage Error for Data Comparison)

In this analysis, the percentage error is used to compare the acquisition data from the designed system with reference data obtained from TechStream across various common rail diesel vehicle parameters. The parameters analyzed include,

1) Engine Speed
2) MAF
3) Accel Sensor No.1 Volt
4) Accel Sensor No.2 Volt
5) Accel Sensor 1 %
6) Accel Sensor 2%.

The percentage error is calculated to assess the accuracy of the data collection system. Testing was conducted at intervals of 1%, 5%, and 20%, and for voltages of 1.7 V, 1.9 V, 2.1 V, and 2.3 V.

4.2.1 Engine Speed

The percentage error for engine speed tends to be low at 1% and 5% intervals, with error values at 0% for most measurements. However, at the 20% interval, the percentage error shows greater variation, particularly at 2.3V, where the error reaches 0.39%. This information is presented in Table1 and Figure 20.

Table 1: Measurement Results of Percentage Error for Engine Speed

Engine Speed			
Percentage Error (%)			
Voltage	Interval		
(V)	1%	5%	20%
1.7	0.00	0.00	0.00
1.9	-0.04	-0.02	0.00
2.1	0.01	0.08	0.00
2.3	0.02	0.00	0.39

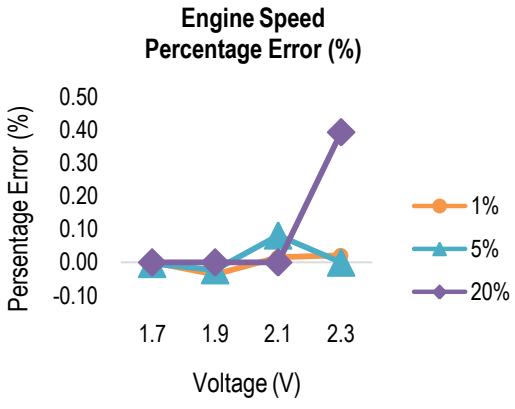

Figure 20: Measurement Results of Percentage Error for Engine Speed

4.2.2 MAF

Data for MAF indicates that the percentage error at the 1% interval is generally very small, with limited positive and negative variation. However, at the 20% interval, the percentage error shows a significant increase, particularly at 2.3V, where the error reaches 0.44%. This may suggest instability in MAF readings at this voltage or issues with sensor sensitivity at higher intervals. This information is presented in Table 2 and Figure 21.

4.2.3 Accel Sensor 1 Volt

For Accel Sensor No.1 Volt, the percentage error is generally small, although there are minor fluctuations across various voltages and intervals. At the 20% interval, the error reaches 0.19%, showing an increase at higher voltages, which may indicate a higher error margin at larger intervals. This information is presented in Table 3 and Figure 22.

Table 2: Measurement Results of Percentage Error for MAF

MAF			
Percentage Error (%)			
Voltage	Interval		
(V)	1%	5%	20%
1.7	0.00	-0.01	0.04
1.9	-0.03	0.00	-0.06
2.1	0.04	0.09	0.23
2.3	0.02	0.00	0.44

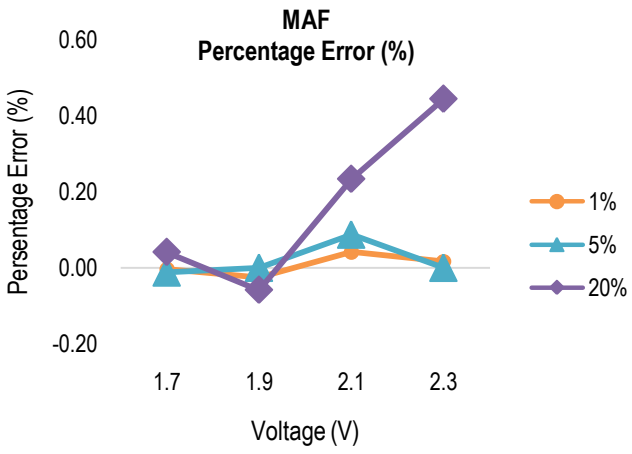

Figure 21: Measurement Results of Percentage Error for MAF

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 8, Issue 10, pp 94-104, October-2024

https://doi.org/10.47001/IRJIET/2024.810015

Table 3: Measurement Results of Percentage Error for Accel Sensor1 Volt

Accel Sensor 1 Volt			
Percentage Error (%)			
Voltage	Interval		
(V)	1%	5%	20%
1.7	0.00	-0.02	-0.02
1.9	-0.02	-0.02	0.00
2.1	0.00	0.00	0.03
2.3	0.01	0.00	0.19

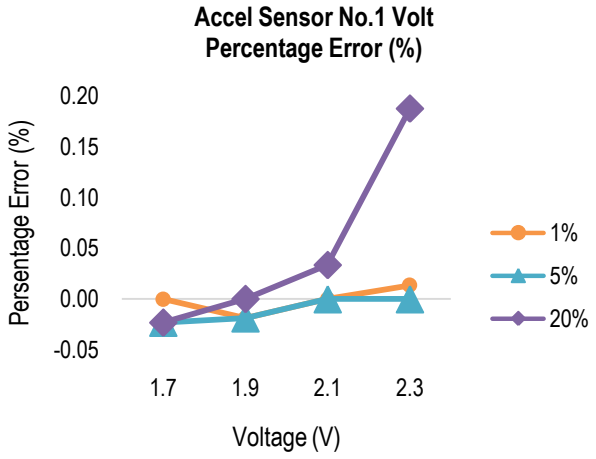

Figure 22: Measurement Results of Percentage Error for Accel Sensor 1 Volt

4.2.4 Accel Sensor 2 Volt

Accel Sensor No.2 Volt exhibits very small percentage errors across all intervals, with a slight increase at the 20% interval at higher voltages. This indicates that this sensor has excellent accuracy, particularly at higher voltages and larger intervals. This information is presented in Table 4 and Figure 23.

Table 4: Measurement Results of Percentage Error for Accel Sensor2 Volt

Accel Sensor 2 Volt			
Percentage Error (%)			
Voltage	Interval		
(V)	1%	5%	20%
1.7	0.00	-0.01	-0.01
1.9	0.00	-0.01	0.00
2.1	0.00	0.00	0.05
2.3	0.00	0.00	0.06

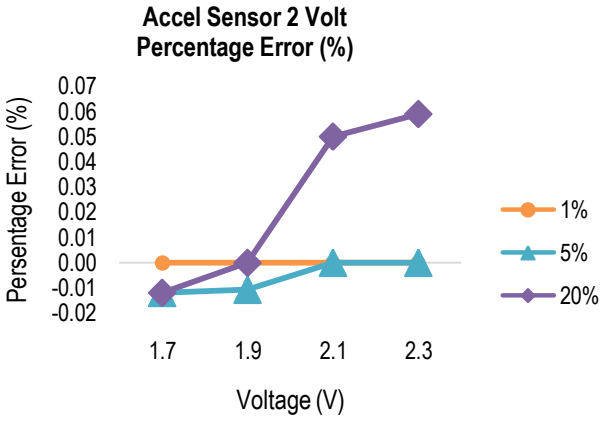

Figure 23: Measurement Results of Percentage Error for Accel Sensor 2 Volt

4.2.5 Accel Sensor 1 %

For the Accel Sensor 1 % output, the percentage error is exceptionally low at the 1% and 5% intervals, remaining at 0% across most voltage levels. However, at the 20% interval and 2.3V, there is a slight increase in the percentage error to 0.15%. This suggests that the sensor exhibits high consistency, with only a minor increase in error at larger intervals. This information is detailed in Table 5 and Figure 24.

Table 5: Measurement Results of Percentage Error for Accel Sensor 1%

Accel Sensor Out 1 %			
Percentage Error (%)			
Voltage	Interval		
(V)	1%	5%	20%
1.7	0.00	0.00	0.00
1.9	0.00	0.00	0.00
2.1	0.00	0.00	0.00
2.3	0.00	0.00	0.15

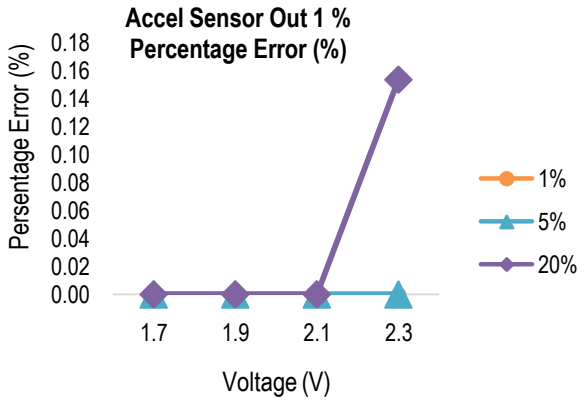

Figure 24: Measurement Results of Percentage Error for Accel Sensor 1%

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 8, Issue 10, pp 94-104, October-2024

https://doi.org/10.47001/IRJIET/2024.810015

4.2.6 Accel Sensor 2 %

The Accel Sensor 2% output demonstrates exceptionally low percentage errors across all voltages and intervals. Although there is a slight increase at the 20% interval at 2.3V, the percentage error remains very small, at 0.05%. This indicates that the sensor is highly stable and accurate within the tested measurement range. This information is presented in Table 6 and Figure 25.

Table 6: Measurement Results of Percentage Error for Accel Sensor 2%

Accel Sensor Out2 %			
Percentage Error (%)			
Voltage	Interval		
(V)	1%	5%	20%
1.7	0.00	0.00	0.00
1.9	0.00	0.00	0.00
2.1	0.00	0.00	0.00
2.3	0.00	0.00	0.05

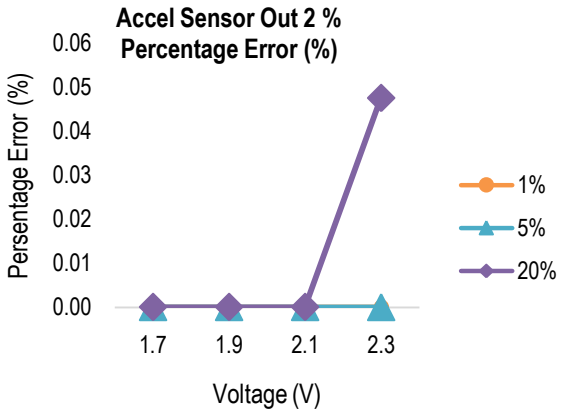

Figure 25: Measurement Results of Percentage Error for Accel Sensor 2%

Based on the data analysis presented, it is observed that the maximum error occurs at a voltage of 2.3 V and an interval of 20%. Overall, the comparison results indicate that the data acquisition system has good accuracy at lower intervals and voltages. However, inaccuracies tend to increase at higher voltages and more extreme intervals. This may suggest limitations in the system's design or calibration that need to be addressed to improve measurement accuracy.

V. CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

The research results indicate that the designed data acquisition system successfully collects vehicle parameters with sufficient accuracy

1) Data Acquisition System Design

The designed data acquisition system successfully meets its primary objective of regulating the throttle on the vehicle's gas pedal. The system is capable of collecting various vehicle parameters in real-time, including engine speed, airflow, and data from acceleration sensors.

2) Development of the Data Acquisition System

The developed data acquisition system demonstrates good capability in collecting real-time vehicle parameters.

3) Testing of Precision and Accuracy

Testing of the accelerator pedal position control system's precision and the data acquisition system's accuracy revealed some measurement errors. Variations in errors were detected in several sensors, particularly at 2.3 volts and higher 50% intervals. This indicates the need for more precise calibration of certain sensors.

5.2 Recommendations

Based on the results and conclusions above, the following recommendations are suggested for further research and development:

1) Calibration and Adjustment of Sensors

Conduct more in-depth calibration of sensors, particularly at extreme voltages and higher intervals. Additional testing and calibration adjustments can enhance accuracy and reduce errors.

2) Broader Testing

Further testing is required across a wider range of voltages and a broader variety of intervals to ensure the system's accuracy under different operational conditions.

3) System Control Improvements

Evaluate and improve the accelerator pedal position control system to enhance responsiveness and accuracy. This is important to ensure that the system can control the throttle precisely according to the vehicle's operational requirements.

REFERENCES

[1] G. Zamboni and M. Capobianco, "Effects of rail pressure control on fuel consumption, emissions and combustion parameters in a turbocharged diesel engine," *Cogent Eng*, vol. 7, no. 1, Jan. 2020, doi: 10.1080/23311916.2020.1724848.

ISSN (online): 2581-3048
Volume 8, Issue 10, pp 94-104, October-2024
https://doi.org/10.47001/IRJIET/2024.810015

[2] G. Macias-Bobadilla, J. D. Becerra-Ruiz, A. A. Estévez-Bén, and J. Rodríguez-Reséndiz, "Fuzzy control-based system feed-back by OBD-II data acquisition for complementary injection of hydrogen into internal combustion engines," *Int J Hydrogen Energy*, vol. 45, no. 51, pp. 26604–26612, Oct. 2020, doi: 10.1016/j.ijhydene.2020.07.084.

[3] J. Xue and X. Jiao, "Speed cascade adaptive control for hybrid electric vehicle using electronic throttle control during car-following process," *ISA Trans*, vol. 110, pp. 328–343, Apr. 2021, doi: 10.1016/j.isatra.2020.10.058.

[4] D. Rimpas, A. Papadakis, and M. Samarakou, "OBD-II sensor diagnostics for monitoring vehicle operation and consumption," in *Energy Reports*, Elsevier Ltd, Feb. 2020, pp. 55–63. doi: 10.1016/j.egyr.2019.10.018.

[5] M. Bathre and P. K. Das, "Design & implementation of smart power management system for self-powered wireless sensor nodes based on fuzzy logic controller using Proteus & Arduino Mega 2560 microcontroller," *J Energy Storage*, vol. 97, Sep. 2024, doi: 10.1016/j.est.2024.112961.

[6] E. Witrant, I. D. Landau, and M. P. Vaillant, "A data-driven control methodology applied to throttle valves," *Control Eng Pract*, vol. 139, Oct. 2023, doi: 10.1016/j.conengprac.2023.105634.

[7] A. H. Khalid *et al.*, "Hydrogen port fuel injection: Review of fuel injection control strategies to mitigate backfire in internal combustion engine fuelled with hydrogen," May 13, 2024, *Elsevier Ltd*. doi: 10.1016/j.ijhydene.2024.04.087.

[8] Y. Srinivasa Rao and T. Getachew Alenka, "Performance and Emission Analysis of Common Rail Diesel Engine with Microalgae Biodiesel," *Journal of Engineering (United Kingdom)*, vol. 2022, 2022, doi: 10.1155/2022/7441659.

[9] M. I. Ozmen, A. Yilmaz, C. Baykara, and O. A. Ozsoysal, "Modelling Fuel Consumption and NO Emission of a Medium Duty Truck Diesel Engine with Comparative Time-Series Methods," *IEEE Access*, vol. 9, pp. 81202–81209, 2021, doi: 10.1109/ACCESS.2021.3082030.

[10] H. Kim, Y. Jeong, W. Choi, D. H. Lee, and H. J. Jo, "Efficient ECU Analysis Technology Through Structure-Aware CAN Fuzzing," *IEEE Access*, vol. 10, pp. 23259–23271, 2022, doi: 10.1109/ACCESS.2022.3151358.

[11] N. Tendikov*et al.*, "Security Information Event Management Data Acquisition and Analysis Methods with Machine Learning Principles," *Results in Engineering*, p. 102254, Jun. 2024, doi: 10.1016/j.rineng.2024.102254.

[12] Y. Chen, S. Li, Z. P. Luo, Y. Huang, Q. P. Yuan, and B. J. Xiao, "Design of real-time data acquisition system for POlarimeter- INTerferometer diagnostic," *Fusion Engineering and Design*, vol. 129, pp. 83–87, Apr. 2018, doi: 10.1016/j.fusengdes.2018.02.080.

[13] H. K. Kondaveeti, N. K. Kumaravelu, S. D. Vanambathina, S. E. Mathe, and S. Vappangi, "A systematic literature review on prototyping with Arduino: Applications, challenges, advantages, and limitations," May 01, 2021, *Elsevier Ireland Ltd*. doi: 10.1016/j.cosrev.2021.100364.

[14] A. S. Kvalsund and D. Winkler, "Development of an Arduino-based, open-control interface for hardware in the loop applications," *HardwareX*, vol. 16, Dec. 2023, doi: 10.1016/j.ohx.2023.e00488.

[15] G. A. Fahmy and M. Zorkany, "Design of a memristor-based digital to analogue converter (Dac)," *Electronics (Switzerland)*, vol. 10, no. 5, pp. 1–16, Mar. 2021, doi: 10.3390/electronics10050622.

[16] R. K. Halder *et al.*, "ML-CKDP: Machine learning-based chronic kidney disease prediction with smart web application," *J Pathol Inform*, vol. 15, Dec. 2024, doi: 10.1016/j.jpi.2024.100371.

[17] M. De Luca, A. R. Fasolino, and P. Tramontana, "Investigating the robustness of locators in template-based Web application testing using a GUI change classification model," *Journal of Systems and Software*, vol. 210, Apr. 2024, doi: 10.1016/j.jss.2023.111932.

[18] R. Tabarés, "HTML5 and the evolution of HTML; tracing the origins of digital platforms," *Technol Soc*, vol. 65, May 2021, doi: 10.1016/j.techsoc.2021.101529.

[19] M. Serdar Biçer and B. Diri, "Defect prediction for Cascading Style Sheets," *Applied Soft Computing Journal*, vol. 49, pp. 1078–1084, Dec. 2016, doi: 10.1016/j.asoc.2016.05.038.

[20] premierautotrade, "Testing Accelerator Pedal Position Sensors (APS)," 2024.

[21] S. P. V., U. P. Borole, R. Kadam, J. Khan, H. C. Barshilia, and P. Chowdhury, "A novel AMR based angle sensor with reduced harmonic errors for automotive applications," *Sens Actuators A Phys*, vol. 324, Jun. 2021, doi: 10.1016/j.sna.2021.112573.

[22] D. W. Gilbert and O. Trinidad, "Toyota Electronic Throttle Control Investigation Preliminary Report Introduction."

[23] I. J. Jin, Y. Y. Park, and I. C. Bang, "Heat transfer performance prediction for heat pipe using deep learning based on wick type," *International Journal of Thermal Sciences*, vol. 197, Mar. 2024, doi: 10.1016/j.ijthermalsci.2023.108806.

ISSN (online): 2581-3048
Volume 8, Issue 10, pp 94-104, October-2024
https://doi.org/10.47001/IRJIET/2024.810015

[24] S. Hosseinpour, M. Aghbashlo, M. Tabatabaei, and E. Khalife, "Exact estimation of biodiesel cetane number (CN) from its fatty acid methyl esters (FAMEs) profile using partial least square (PLS) adapted by artificial neural network (ANN)," *Energy Convers Manag*, vol. 124, pp. 389–398, Sep. 2016, doi: 10.1016/j.enconman.2016.07.027.

Citation of this Article:

Dedy Novindra, Nazaruddin Sinaga, & Eflita Yohana. (2024). Design and Evaluation of a Throttle Controller for Common Rail Diesel Engines. *International Research Journal of Innovations in Engineering and Technology - IRJIET*, 8(10), 94-104. Article DOI https://doi.org/10.47001/IRJIET/2024.810015

****

