IOT enabled Motor Monitoring system with ADAS for protection

*Aditya . S. Katkar * ,Mohit .D. Hande* , Devang .P. Jalan* , Dr. A. A. Apte*

Department of Elctrical Engineering , AISSMS COE , Pune

Abstract— This paper presents an IoT-based motor monitoring system with an added ADAS feature. Built using ESP32 and MATLAB. Simple idea. But quite powerful.The system continuously measures voltage, current, temperature, and speed. At the same time, an ultrasonic sensor detects obstacles. So it is not just monitoring. It is also aware of surroundings. That adds safety.Data is collected by ESP32. Then sent to cloud. Real-time visualization happens using IoT platform. MATLAB is used for analysis. Graphs are generated. Patterns start to appear. System behavior becomes visible.Results show expected motor characteristics. Current rises and then stabilizes. Speed increases with voltage. Almost linear. That part works well .But not everything is perfect. Distance readings fluctuate at times. Motor speed changes abruptly. Control feels reactive. Voltage signal looks noisy due to PWM switching. These small issues shows limitations in sensing and control.Still, the system works. It connects hardware, software, and cloud together. Monitoring becomes continuous. Remote access becomes possible.With better filtering and advanced control methods, the system can improve further. Become more stable. More reliable. And closer to real ADAS applications.

Keywords— IoT, ESP32, MATLAB, ADAS, Motor Monitoring, Ultrasonic Sensor, PWM Control, Real-Time Monitoring

1.INTRODUCTION

Electric motors are everywhere. Factories. Electric vehicles. Robots. Even the intelligent systems we barely notice. They are preferred for a reason—high efficiency, strong reliability, and simple control. But they are not invincible. They heat up. They slow down. Sometimes, they just fail… without warning. And when that happens, everything stops. Production delays. Costs go up. People start noticing. That's why monitoring parameters like voltage, current, temperature, and speed isn't optional. It's necessary. It keeps the system running. Keeps it alive. Traditionally, motor monitoring was manual. Someone checks readings. Writes them down. Assumes everything is fine till the next inspection. It works… but not really. Faults don't always show themselves early. They develop quietly. No alerts. No signs. And by the time the problem is visible, the damage is already done. This makes traditional systems reactive, not proactive. Continuous monitoring was missing. Predictive maintenance? Almost impossible.

Then IoT changed the story. Suddenly, motors could speak. Sensors collect data continuously. Microcontrollers process it. Wireless communication sends it anywhere. Real-time monitoring becomes possible. Engineers can see everything. Live. From anywhere. IoT-based systems enable continuous acquisition, remote access, and faster analysis. It improves reliability. It improves efficiency. And honestly, it makes life easier.

At the center of this system sits the ESP32 microcontroller. Small device. Big capability. Built-in Wi-Fi. Fast processing. Low power consumption. It connects easily with sensors. It collects data. It transmits data. It just works. Along with hardware, software also plays critical role. MATLAB becomes important here. Very important. It provides hardware support packages for ESP32. Direct interface. Real-time acquisition. Visualization. Analysis. Everything in one place. This integration reduces development complexity. It makes system development faster and more flexible.

Safety adds another dimension. Especially in intelligent vehicles. This is where ADAS comes in. Advanced Driver Assistance Systems are designed to prevent accidents. Ultrasonic sensors help detect obstacles. They measure distance. They provide awareness. When integrated with motor monitoring, system becomes smarter. It not only monitors itself but also watches environment. This dual capability enhances safety. It makes system more intelligent. More reliable.

Many IoT-based motor monitoring systems already exist. But most of them are limited. They focus only on basic monitoring. They use conventional programming tools. MATLAB-based ESP32 programming is still less explored. Integration with ADAS using ultrasonic sensors is also not widely implemented. There is a gap. A clear one.

This paper presents a solution to fill that gap. A motor parameter monitoring system using ESP32, IoT, and MATLAB. Real-time monitoring of voltage, current, temperature, and speed is achieved. Obstacle detection is also implemented using ultrasonic sensor. MATLAB hardware support packages enable programming, interfacing, and analysis. The system improves monitoring capability. Enhances safety. Provides intelligent functionality. It is efficient. It is practical. And it shows how modern tools can transform traditional motor monitoring systems.

2 . LITRETURE REVIEW

Electric motors are the workhorses of modern industry, powering factories, farms, and automation systems while often going unnoticed during normal operation. Their maintenance and continuous monitoring are essential, as even small unnoticed faults can lead to complete system failure, production delays, and increased costs.

Traditionally, motor monitoring was done manually. Engineers check voltage, current, temperature during routine inspection. Then they leave. Motor keeps running for long time without checking again. This creates problem. Because faults don't follow schedule. They can develop anytime. Slowly. Silently. And by the time fault becomes visible, damage already done. Manual monitoring helps, but not enough. It is reactive, not preventive.

Then IoT started changing things. Motors could now be monitored continuously. Sensors collect data. Microcontroller process it. Wi-Fi sends data to cloud. Engineers can see motor condition in real time. They don't have to wait. They don't have to guess. Everything visible. This improves reliability. And also reduces unexpected failures. Maintenance becomes more planned.

ESP32 microcontroller plays important role here. It is small. Low cost. Powerful enough. And most important, built-in Wi-Fi. It can connect with sensors easily. It collects data and transmit it wirelessly. Because of this, many researchers started using ESP32 for motor monitoring systems.

Shukla et al. developed IoT-based monitoring system using ESP32 and sensors. It measured temperature, current, vibration, and humidity. Data was sent to ThingSpeak cloud platform. Operators could monitor motor remotely. If abnormal condition appears, it can be detected early. This helped in preventing unexpected failures. And reduced maintenance cost also [1].

Similarly, Tiwari proposed motor monitoring system using ESP32. It measured speed, temperature, and current using PT100 and ACS712 sensors. Data transmitted wirelessly to cloud platform. Engineers could observe motor behavior and detect abnormal conditions before failure happens. This improved motor reliability and reduced downtime. Simple system. But useful [2].

Kadam et al. developed DC motor monitoring system using ESP32 and IoT. The system continuously monitored voltage, current, and temperature. It compared real-time values with predefined limits. If values exceeded safe limits, alerts were generated. Protective actions like relay control activated. This prevented motor damage and improved safety [3].

Kshatriya et al. designed monitoring system using ESP32 Devkit V1. It measured voltage, current, temperature, vibration, and speed using different sensors. Data transmitted wirelessly for remote monitoring. Alerts generated when parameters exceeded safe range. This allowed early action. System became more reliable. More safe [4].

Some researchers also worked on monitoring with control. Dhangare et al. developed motor speed control and monitoring system using ESP32 and IoT. Users could control motor speed remotely using Blynk platform. At same time, voltage and current monitored continuously. This improved efficiency and reduced manual work [5].

Awaar et al. developed similar system using ESP32 and Firebase. It monitored motor parameters and also controlled speed using PWM signals. Data available remotely. Users could monitor motor anytime. This improved system performance and flexibility [6].

Marzuki et al. integrated ESP32 with Variable Frequency Drive for industrial motor monitoring. It allowed real-time monitoring of voltage, current, and speed. System also improved energy efficiency. And provided better control flexibility [7].

Fanani et al. developed multi-sensor monitoring system using ESP32. It measured voltage, current, temperature, vibration, and speed. Data transmitted to cloud platform. Operators could see data in graphical form. This helped in easier monitoring and fault detection [8].

Espinosa-Gavira et al. studied performance of ESP32 in IoT sensor networks. Results showed reliable data transmission and stable communication. ESP32 suitable for real-time monitoring applications. Because of its built-in Wi-Fi and good processing capability [9].

From these studies, it is clear ESP32 provides effective solution for motor monitoring. It enables real-time monitoring. Remote access possible. Fault detection becomes easier. But still some systems monitor limited parameters only. Some lack proper monitoring capability. Some not optimized for complete motor monitoring.

So there is need for better monitoring system. A system that monitor voltage, current, temperature, and speed continuously. With reliable communication. With accurate data acquisition. The goal is clear. Detect faults early. Improve reliability. Reduce failures. And make motor monitoring more practical for real industrial use.

3 . Methodology

3.1 System architecture

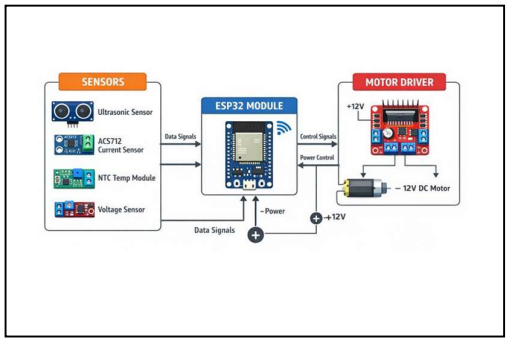

Fig 1. Circuit Diagram

3.1.1.

The system architecture starts with the DC motor and its integration with IoT monitoring and ADAS obstacle detection system. It is not just a motor now. It becomes part of a connected system. At the center, ESP32 microcontroller is used. This acts like the brain of the system. Everything is connected to it. It have built-in Wi-Fi and Bluetooth, so wireless communication becomes possible without using extra modules. Data can be transmitted easily, and sometimes faster than expected.

3.1.2.

The DC motor is connected to the L298N motor driver module. This driver controls the motor using PWM signals. Short pulses. Fast switching. Speed of motor can be controlled depending on PWM value. The motor driver also provides two output channels, which makes possible to control more than one motor if needed. Sensors are connected across the motor and supply lines. They monitor the motor continuously. The

ACS712 sensor measures the current flow. The LM35 sensor measures the temperature, which increases slowly during operation. Voltage is measured using RC-based voltage sensing circuit, giving approximate value of supply voltage.

3.1.3.

All the sensors outputs are connected to analog input pins of ESP32. Analog sensors was selected because continuous data was required, not only fixed ON or OFF values. The ultrasonic sensor is also connected to ESP32. It sends sound waves. Waits for echo. Then distance is calculated. If obstacle comes near, system detects it immediately.

3.1.4.

ESP32 collects data from all sensors. It process the signals and prepares it for transmission. Then data is sent through Wi-Fi to IoT platform for monitoring. MATLAB is used with this architecture for receiving and analyzing the data. Graphs can be plotted. System behavior can be observed. In this way, the overall architecture connects motor, sensors, ESP32, IoT, and MATLAB together, making the system capable of monitoring and obstacle detection, though sometimes small variations in readings may occurs.

4. Hardware Components used

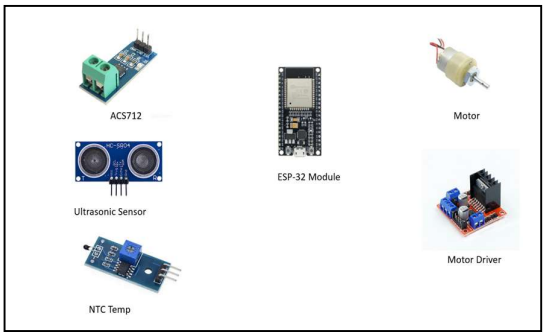

Fig 2. Hardware Components

The hardware setup starts with the DC motor. It is the main device. Everything is connected around it. The motor is controlled using the L298N motor driver, which uses PWM signals. Fast switching. Speed changes depending on the signal.
At the center, ESP32 microcontroller is used. It collects sensor data and also sends it through Wi-Fi for IoT communication. It acts like the brain of the system. The ACS712 current sensor measures motor current, while LM35 sensor measures temperature. Voltage sensor based on RC circuit is used to measure supply voltage. These sensors continuously sends analog signals.
For ADAS feature, ultrasonic sensor HC-SR04 is used. It detects obstacle and measures distance. The power supply provides required voltage to all components. In this way, all hardware are connected together, forming a complete monitoring and control system, though small variations in readings may occurs.

5. Software Used

MATLAB Software – MATLAB is used as the main software in this system. It is not only used for analysis, but also for programming. It connects with the ESP32. Communication happens. Data is received. Sometimes commands are also sent back. This makes control and monitoring easier than expected.
MATLAB reads sensor values like current, voltage, temperature, and distance. It process the data continuously. Then graphs are generated. Speed vs time. Current vs time. Distance vs time. These plots helps to see what is really happening inside the system.
MATLAB is also used for deeper analysis. Patterns starts to appear. Some variations can be seen, which normally goes unnoticed. With its hardware support packages, MATLAB makes programming and analysis possible in same environment. Because of this, it becomes an important part of the overall system, even though small communication delays may occurs sometimes.

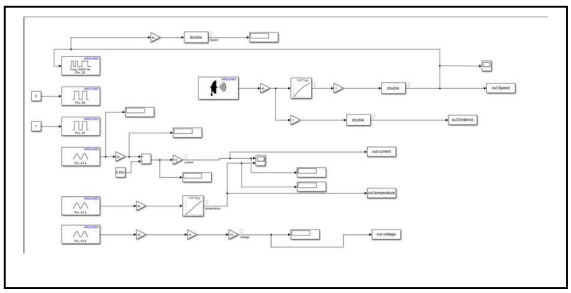

Fig 3. Simulink block diagram of system

6. System Working

The system begins with a DC motor. Simple at first. But here, it becomes part of something bigger. It is integrated with an IoT monitoring system and also an ADAS obstacle detection unit. At the center, ESP32 microcontroller is used. This becomes the main controller. Everything connects here. It have built-in Wi-Fi and Bluetooth, which makes wireless communication possible without extra modules.

The DC motor is connected to the L298N motor driver. This driver controls motor using PWM signals. Fast switching happens. Speed can be adjusted depending on the signal. Sensors are placed across the motor and supply line. Watching continuously. ACS712 measures the motor current. LM35 measures temperature, which slowly increases during operation. Voltage is measured using RC-based sensing circuit. It gives supply voltage information, though sometimes small variations appears.

Ultrasonic sensor is also connected. It detects obstacle. Measures distance. This makes the ADAS functionality

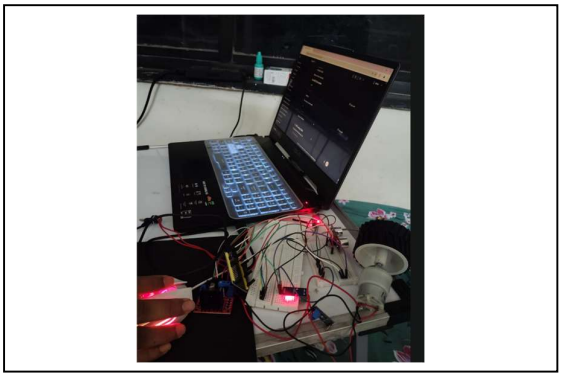

Fig 4. Hardware setup of the system

possible. All sensor outputs goes to ESP32. ESP32 collects the data. Processes it. Then sends it through Wi-Fi to IoT platform and MATLAB. This allows monitoring in real time.

6.2. IoT Monitoring Implementation

Sensors continuously measure motor parameters. Current. Voltage. Temperature. Distance. All this data goes to ESP32. ESP32 processes it. Then transmits through Wi-Fi.

Data reaches IoT platform and MATLAB. Monitoring becomes possible remotely. System condition can be observed in real time. This helps in detecting abnormal behavior early, before major issue happens.

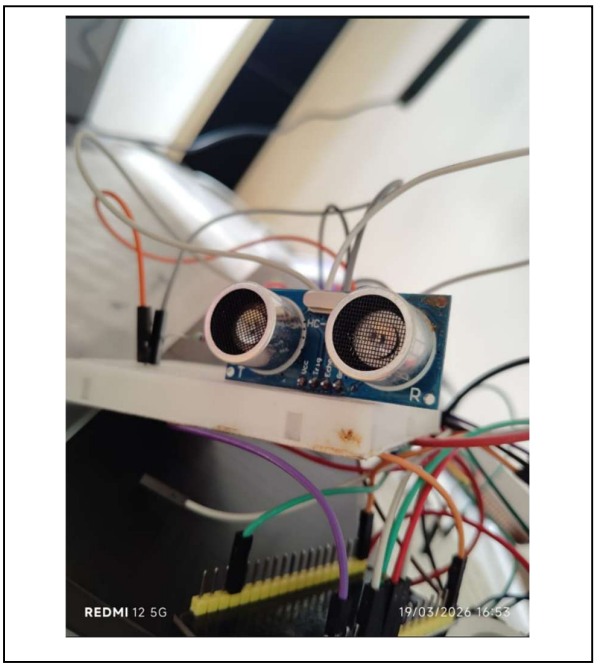

Fig 5. Ultrasonic Sensor

6.3. ADAS Obstacle Detection Implementation

Ultrasonic sensor is used for obstacle detection. It sends sound waves. Waits for echo. Time is measured. Distance is calculated using time-of-flight principle.

ESP32 reads this distance continuously. If obstacle comes close, system detects it. This improves safety. MATLAB also receives this distance data. Graphs are plotted. System response can be observed, although minor measurement error may occurs sometimes.

6.4. Data Acquisition and Analysis using MATLAB

MATLAB receives sensor data from ESP32. Data is processed immediately. Graphs are generated. Patterns begins to appear.

Motor current, voltage, temperature, and distance are analyzed. Abnormal conditions can be detected. System performance becomes easier to understand using visualization tools.

6.5.1. Distance

Things appear stable at first. The gap increases and remains at about 80 cm for some time. tidy. dependable. Nearly flawless. Then it shifts, though. The readings start to increase. Both up and down. At times, it is near zero, while at other times, it is between 50 and 60 cm. No longer steady.

This narrates a tale. When not much is going on, the sensor performs nicely. However, noise starts to appear soon the system is turned on. Perhaps reflections. Perhaps abrupt motions. Perhaps the readings are being disturbed by the motor.
Thus, the sensor is operational. However, not flawlessly. It must be filtered. Something to ease the situation.

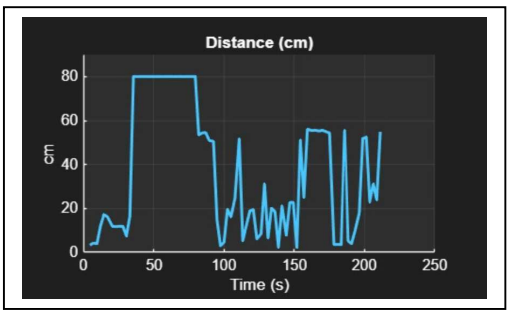

Fig 6. Distance Detected by the sensor

6.5.2. Motor speed

The motor doesn't ease into motion. It jumps. Straight to 100%. No gradual build-up .Then suddenly, it drops. And after that, it keeps fluctuating. Up. Down. Again and again .This kind of behavior usually means one thing — the control logic is too basic. Probably ON/OFF type. No in-between. No smooth transition .It works, technically. But not elegantly. A more refined control, like PID, would make the motor feel… controlled. Right now, it feels reactive.

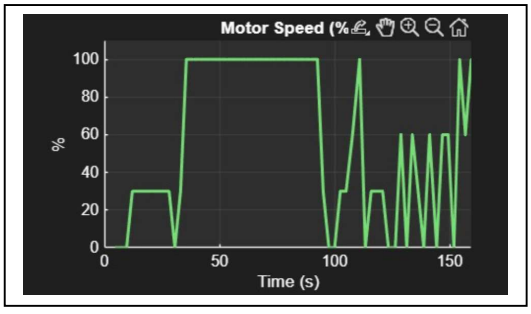

Fig 7. Motor speed variation as per obstacle detection

6.5.3. Current

This graph is calmer. Much calmer.

The current rises quickly at the beginning. Then it settles around 300 mA. And stays there. No major spikes. No sudden drops .That's a good sign. It means the motor reaches its operating condition and holds it. The system is not overloading. Not struggling . In simple words — electrically, things are stable.

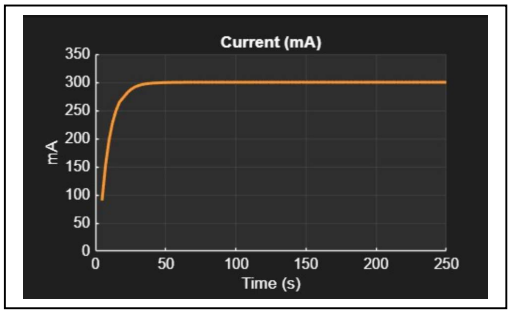

Fig 8. Current is gradually increase and stable (constant load)

6.5.4. Voltage

This one looks messy at first glance. Voltage jumping between 0 and 10 V. Sharp edges. Sudden drops. But there's a reason. This is most likely PWM in action. The voltage isn't actually unstable — it's switching fast. Very fast. What we're seeing is raw behavior. Not averaged. Not filtered. Just the real signal. Still, for analysis, it can be misleading. A smoother representation would help. Maybe average voltage. Or duty cycle.

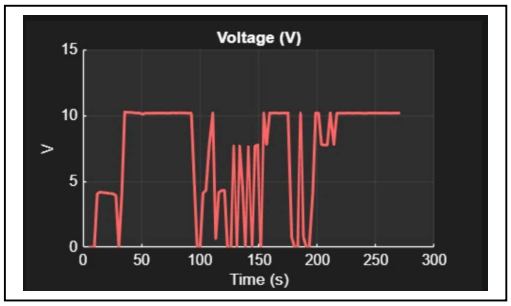

Fig 9. Supply voltage variation

6.5.5. Speed vs Voltage.

As voltage increases, speed increases too. Almost in a straight line. Simple. Predictable .This is exactly what we expect from a DC motor. No surprises here. Just physics doing its job .There are small deviations, yes. But nothing major. Overall, this confirms that the motor behaves correctly.

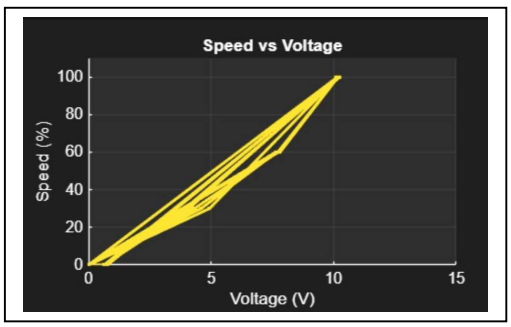

Fig 10. Speed and Voltage

7.2. IOT Results

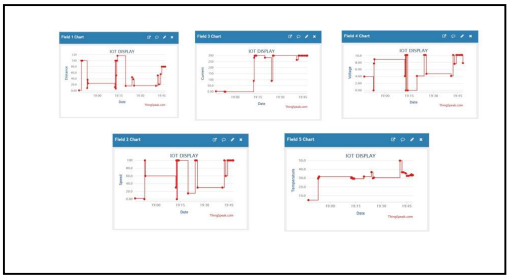

Fig 11. IOT Monitoring

The IoT-based monitoring system was implemented using ThingSpeak for real-time data visualization. Multiple parameters, including distance, motor speed, current, voltage, and temperature, were recorded and analyzed. The system performs well overall. Data is transmitted. Data is visible. And most importantly, it reflects system behavior in real time.

The distance plot shows noticeable variations over time. At certain intervals, the readings remain stable, indicating proper sensing. But then, sudden changes appear. Sharp drops. Unexpected spikes. This suggests the presence of sensor noise or dynamic environmental conditions. In an ADAS context, this becomes critical. Because sensing should be consistent. Always.

The motor speed data follows a similar pattern. It reaches high values quickly, often close to maximum. Then drops. Then rises again. Not smooth. This indicates that the control logic is reactive rather than continuous. Likely threshold-based. It works,but lacks refinement.

The current plot provides a more stable behavior. After an initial rise, the current remains close to a steady value, around 300 mA. This shows that the motor operates under stable load conditions. No major spikes are observed. Which is a good sign.
Voltage readings show frequent fluctuations. Rapid transitions between low and high values are visible. This is expected. The system uses PWM for motor control. So the voltage appears discontinuous. In reality, it is switching fast. Very fast. Still, for better interpretation, an averaged or filtered signal would be more useful.

The temperature data remains relatively stable throughout the operation, mostly around 30–35°C, with a slight peak near 50°C. This indicates that the system does not experience significant thermal stress. The variations are minor. Overall, the IoT system successfully captures and displays real-time data. That's important. It validates the integration of sensing, control, and cloud monitoring. However, the data also reveals system limitations. Noise in sensing. Abrupt control response.

With improved filtering techniques and advanced control strategies, the system performance can be enhanced significantly. The IoT platform, in this case, not only monitors the system but also helps identify these issues clearly.

At this stage, the system works. It communicates. It responds. With refinement, it can become more stable, more accurate, and closer to real-world ADAS implementation.

Conclusion:-

This work presents a MATLAB-based implementation of an ADAS-inspired control system. The system integrates sensing, control, and actuation. And overall, it works.

The results show expected behavior . Motor current rises and then settles. Speed follows voltage. Almost linear. Just like theory predicts. So, the basic model is valid. That part is clear.

But when the system runs in real conditions, things change a bit. The distance readings are not always stable. They fluctuate. Sometimes more than expected. This indicates sensor noise or environmental effects. In an ADAS system, this matters a lot. Accuracy is everything.

The motor response also feels abrupt. It reaches maximum speed too quickly. Then drops. Then fluctuates again. Not smooth. This points to a simple control logic. Likely threshold-based. Not continuous. Not adaptive.

The voltage profile looks irregular as well. Rapid changes. Sudden drops. But this is mostly due to PWM switching. MATLAB captures the raw signal, so it appears noisy. Still, for analysis, a filtered representation would be more meaningful.

So yes, the system is functional. It demonstrates the concept. It connects sensing with action. That's important. But it is not fully refined yet.

With improved filtering and advanced control methods, such as PID, the performance can be significantly enhanced. The system can become smoother. More stable. More reliable.

At this stage, it is a working model. With further improvements, it can move closer to real-world ADAS implementation.

REFERENCES

[1] **[1] A. Shukla, S. P. Shukla, S. T. Chacko, M. K. Mohiddin, and K. A. Fante, "Monitoring of single-phase induction motor through IoT using ESP32 module," Journal of Sensors, vol. 2022, Article ID 8933442, 2022.**

[2] **[2] A. Tiwari, "Prototype health monitoring of induction motor using IoT," International Journal of Engineering Development and Research, vol. 7, no. 2, pp. 502–506, 2019.**

[3] **[3] P. M. Kadam, B. M. Patil, A. D. Rathod, P. P. Wankhede, and S. K. Mittal, "IoT-powered system for continuous monitoring of DC motor load and performance," International Journal of Creative Research Thoughts, vol. 12, no. 11, Nov. 2024.**

[4] **[4] D. S. Kshatriya, R. B. Patil, K. H. Gawali, and S. P. Bharate, "IoT based motor monitoring and control system using ESP32-Devkit-V1," International Journal of Advanced Research in Science, Communication and Technology, vol. 5, no. 5, Mar. 2025.**

[5] **[5] D. Dhangare, G. Naukarkar, A. Gaidhane, and A. Duryodhan, "Adaptive speed control of induction motor with IoT-based real-time monitoring," International Journal for Research Publication and Seminar, vol. 16, no. 1, Jan.–Mar. 2025.**

[6] **[6] V. K. Awaar, S. Rani, K. Kirthi, S. Ch, S. P. Keerthana, and S. Das, "IoT enabled speed control of single-phase induction motor," E3S Web of Conferences, vol. 391, 2023.**

[7] **[7] A. Marzuki, T. Muzakkir, and M. S. Arief, "Three-phase induction motor control and monitoring using VFD and ESP32 based on Modbus RTU protocol," American Journal of Electrical and Computer Engineering, vol. 8, no. 2, pp. 71–80, 2024.**

[8] **[8] W. B. Fanani, A. E. Kristiyono, H. Setiawan, and J. P. Kusanto, "Design and implementation of an IoT-based speed control and monitoring system for 3-phase induction motors," Journal Sainstech Nusantara, vol. 2, no. 3, 2025.**

[9] **[9] M. J. Espinosa-Gavira, A. Aguera-Perez, J. C. Palomares-Salas, J. M. Sierra-Fernandez, P. Remigio-Carmona, and J. J. Gonzalez de-la-Rosa, "Characterization and performance evaluation of ESP32 for real-time synchronized sensor networks," Procedia Computer Science, vol. 237, pp. 261–268, 2024.**

[10] D. S. Kshatriya, R. B. Patil, K. H. Gawali, and S. P. Bharate, "IoT-based real-time motor parameter monitoring system using ESP32," International Journal of Advanced Research in Science, Communication and Technology, 2025.