

PHASE 3: IMPLEMENTATION OF PROJECT TITLE :

Smart Fleet Management and Telematics System Using IOT

Objective

The goal of Phase 3 is to implement the core components of the Smart Fleet Management and Telematics System based on the design and planning carried out in Phase 2. This includes the development of the telematics AI engine, dashboard interface, optional IoT hardware integration, and the implementation of data security protocols.

1. Telematics AI Engine Development

Overview

The core of the Smart Fleet Management System is the AI-based telematics engine, which collects and analyzes vehicle and driver data to provide insights and recommendations.

Implementation

- Sensor Data Processing: The AI engine interprets GPS, speed, fuel usage, and maintenance sensor input to generate actionable reports.
- Predictive Analytics: Uses machine learning to predict maintenance needs and optimize route planning.- Data Source: Real-time data from vehicles and cloud-connected databases are used for training and inference.

Outcome

By the end of this phase, the AI engine will identify inefficiencies, recommend maintenance actions, and monitor compliance with driving standards.

2. Dashboard & User Interface Development

Overview

The system will include a centralized web-based dashboard for fleet managers to monitor all fleet-related metrics in real time.

Implementation

- Fleet Overview: Live data visualization of all active vehicles, showing location, status, and driver information.
- Alerts & Reports: Custom notifications for speeding, engine issues, and maintenance schedules.
- Multilingual UI: Currently supports English; other languages to be added in future phases.

Outcome

The dashboard will allow fleet operators to manage operations effectively through a user-friendly and data-rich interface.

3. IoT Device Integration (Optional)

Overview

While optional in this phase, we aim to connect the system to in-vehicle IoT devices for enhanced real-time data capture.

Implementation

- Hardware Integration: Devices such as OBD-II dongles, GPS trackers, and fuel sensors will be integrated.
- API Use: Standardized APIs from IoT platforms will be used for data ingestion.

Outcome

If hardware is available, the system will capture live vehicle diagnostics and driving behavior. Otherwise, mock data will be used for simulation.

4. Data Security Implementation

Overview

Data security is a high priority given the sensitive nature of location and operational data in fleet management.

Implementation

- Encryption: All fleet data (vehicle location, driver behavior, trip history) will be encrypted both in transit and at rest.
- Secure Access: Role-based access controls and secure cloud storage will be implemented to comply with industry regulations.

Outcome

All data processed and stored by the system will be protected by enterprise-grade security protocols.

5. Testing and Feedback Collection

Overview

This phase includes the initial rollout of the system for testing with a small fleet segment.

Implementation

- Pilot Testing: Select vehicles and drivers will be enrolled for testing various system modules.
- Feedback Loop: Input from fleet managers and drivers will be collected on usability, accuracy, and performance.

Outcome

Feedback will be analyzed to refine AI models and enhance UI/UX in Phase 4.

Challenges and Solutions

Challenge: AI Prediction Accuracy

Solution: Continuous model training with real data and expert feedback.

Challenge: User Adoption

Solution: Training sessions and UI enhancements based on early feedback.

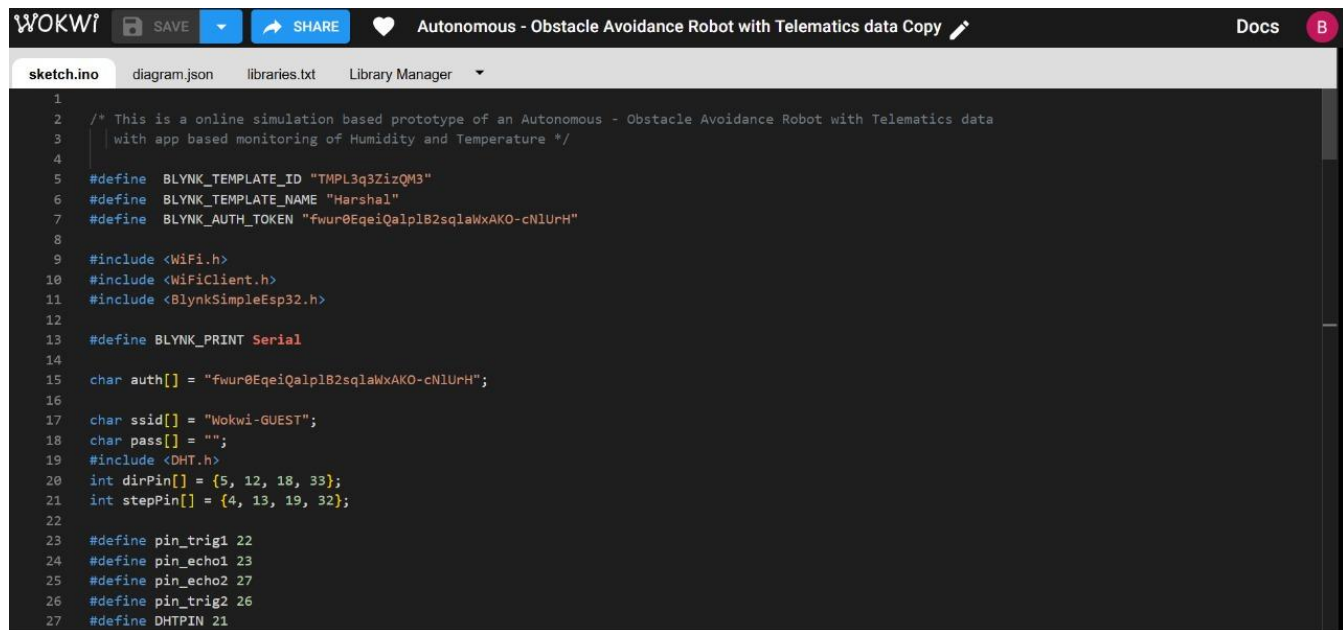
Challenge: Hardware Compatibility

Solution: Use of open-standard APIs and modular system design.

Outcomes of Phase 3

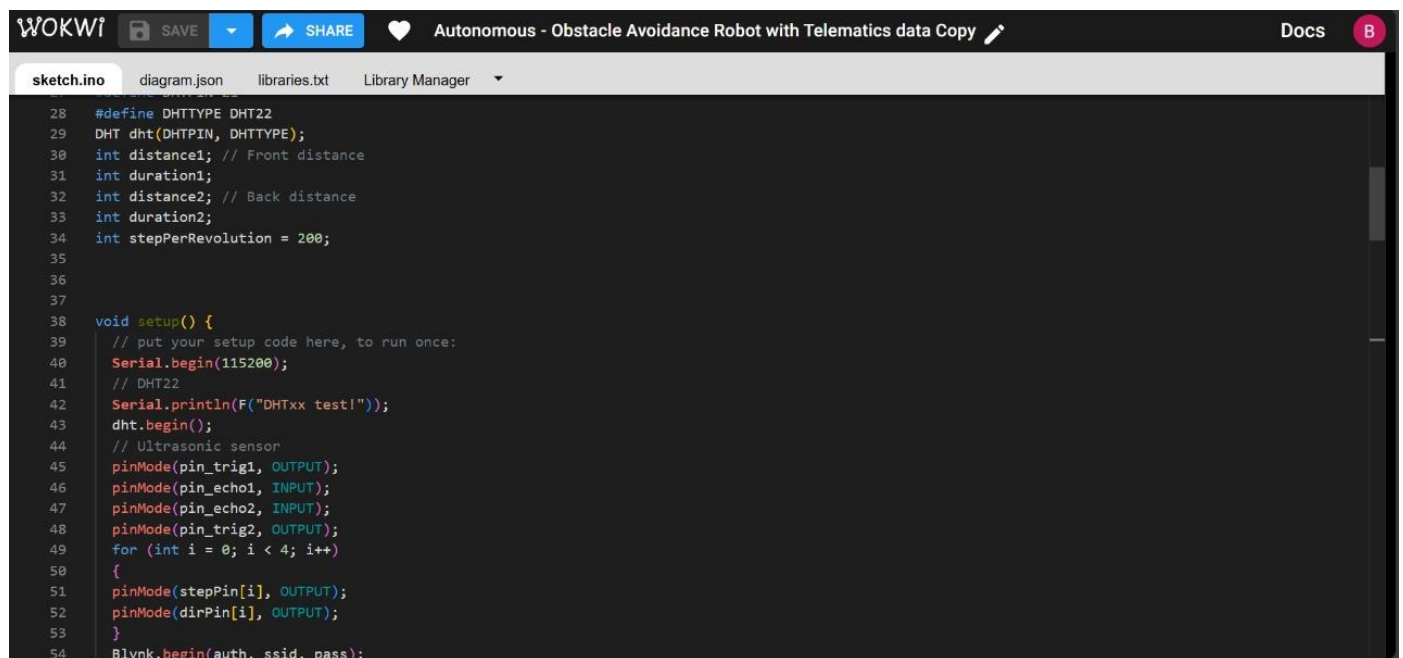
1. Functional AI Engine: Able to analyze fleet data and offer insights.
2. Operational Dashboard: Real-time fleet management capabilities.
3. IoT Integration (Optional): Basic connectivity with vehicle hardware.
4. Secure Data Framework: Encrypted and access-controlled data environment.
5. Initial User Testing: Feedback collected to guide future improvements.

Screenshots of Code and System in Use



The screenshot shows the Wokwi IDE interface. The top bar includes the Wokwi logo, a 'SAVE' button, a 'SHARE' button, a heart icon, the project name 'Autonomous - Obstacle Avoidance Robot with Telematics data Copy', and a 'Docs' button with a 'B' icon. Below the top bar is a tabbed interface with 'sketch.ino' selected. The code editor displays the first 27 lines of the sketch.ino file. The code includes comments, Blynk template definitions, includes for WiFi and Blynk libraries, and pin definitions for DHT and ultrasonic sensors.

```
1
2  /* This is a online simulation based prototype of an Autonomous - Obstacle Avoidance Robot with Telematics data
3     with app based monitoring of Humidity and Temperature */
4
5  #define BLYNK_TEMPLATE_ID "TMPL3q3ZizQM3"
6  #define BLYNK_TEMPLATE_NAME "Harshal"
7  #define BLYNK_AUTH_TOKEN "fwur0EgeiQalplB2sqlaWxAKO-cNlUrH"
8
9  #include <Wifi.h>
10 #include <WifiClient.h>
11 #include <BlynkSimpleEsp32.h>
12
13 #define BLYNK_PRINT Serial
14
15 char auth[] = "fwur0EgeiQalplB2sqlaWxAKO-cNlUrH";
16
17 char ssid[] = "Wokwi-GUEST";
18 char pass[] = "";
19 #include <DHT.h>
20 int dirPin[] = {5, 12, 18, 33};
21 int stepPin[] = {4, 13, 19, 32};
22
23 #define pin_trig1 22
24 #define pin_echo1 23
25 #define pin_echo2 27
26 #define pin_trig2 26
27 #define DHTPIN 21
```



The screenshot shows the Wokwi IDE interface, continuing from the previous one. The top bar is identical. The 'sketch.ino' tab is selected, and the code editor displays lines 28 to 54 of the sketch.ino file. The code defines DHT22 sensor type, initializes DHT and ultrasonic sensor variables, and defines the setup() function which initializes serial communication, DHT22, and pins for the ultrasonic sensor and stepper motor.

```
28 #define DHTTYPE DHT22
29 DHT dht(DHTPIN, DHTTYPE);
30 int distance1; // Front distance
31 int duration1;
32 int distance2; // Back distance
33 int duration2;
34 int stepPerRevolution = 200;
35
36
37
38 void setup() {
39   // put your setup code here, to run once:
40   Serial.begin(115200);
41   // DHT22
42   Serial.println(F("DHTxx test!"));
43   dht.begin();
44   // Ultrasonic sensor
45   pinMode(pin_trig1, OUTPUT);
46   pinMode(pin_echo1, INPUT);
47   pinMode(pin_echo2, INPUT);
48   pinMode(pin_trig2, OUTPUT);
49   for (int i = 0; i < 4; i++)
50   {
51     pinMode(stepPin[i], OUTPUT);
52     pinMode(dirPin[i], OUTPUT);
53   }
54   Blynk.begin(auth, ssid, pass);
```

sketch.ino

diagram.json

libraries.txt

Library Manager

```
55 }
56
57 void loop()
58 {
59   Blynk.run();
60   // Front ultrasonic sensor
61   digitalWrite(pin_trig1, LOW);
62   delayMicroseconds(2);
63   digitalWrite(pin_trig1, HIGH);
64   delayMicroseconds(10);
65   digitalWrite(pin_trig1, LOW);
66   duration1 = pulseIn(pin_echo1, HIGH);
67   distance1 = duration1 * 0.034/2;
68   Serial.print("FRONT-DISTANCE = ");
69   Serial.print(distance1);
70
71   delay(10); // this speeds up the simulation
72
73   // Second ultrasonic sensor
74   digitalWrite(pin_trig2, LOW);
75   delayMicroseconds(2);
76   digitalWrite(pin_trig2, HIGH);
77   delayMicroseconds(10);
78   digitalWrite(pin_trig2, LOW);
79   duration2 = pulseIn(pin_echo2, HIGH);
80   distance2 = duration2 * 0.034/2;
81   Serial.print(" BACK-DISTANCE = ");
```

sketch.ino

diagram.json

libraries.txt

Library Manager

```
73 // Second ultrasonic sensor
81   Serial.print(" BACK-DISTANCE = ");
82   Serial.println(distance2);
83
84   delay(100);
85
86   // DHT22 temperature and humidity reading
87   float humidity = dht.readHumidity();
88   float temperature = dht.readTemperature();
89   Serial.print(F("Humidity: "));
90   Serial.print(humidity);
91   Serial.print(F("% Temperature: "));
92   Serial.print(temperature);
93   Serial.println(F("°C "));
94
95   delay(2000);
96
97 // if in front the distance of ultrasonic sensor and obstacle is less than 20cm
98 if (distance1 <= 5 )
99 {
100   for (int i = 1; i < 4; i += 2) // upside motors
101   {
102     digitalWrite(dirPin[i], LOW);
103   }
104   for (int i = 1; i < 4; i += 2)
105   {
106     digitalWrite(stepPin[i], HIGH);
```

WOKWI

SAVE

SHARE

Autonomous - Obstacle Avoidance Robot with Telematics data Copy

Docs

B

sketch.ino

diagram.json

libraries.txt

Library Manager

```
97 // if in front the distance of ultrasonic sensor and obstacle is less than 20cm
99 {
105 {
106     digitalWrite(stepPin[i], HIGH);
107 }
108 delayMicroseconds(1000);
109 for (int i = 1; i < 4; i += 2)
110 {
111     digitalWrite(stepPin[i], LOW);
112 }
113 delayMicroseconds(1000);
114
115 for (int i = 0; i < 4; i += 2) // downside motors
116 {
117     digitalWrite(dirPin[i], HIGH);
118 }
119 for (int i = 0; i < 4; i += 2 )
120 {
121     digitalWrite(stepPin[i], HIGH);
122 }
123 delayMicroseconds(1000);
124 for (int i = 0; i < 4; i += 2 )
125 {
126     digitalWrite(stepPin[i], LOW);
127 }
128 delayMicroseconds(1000);
129 /*if (distance1 <= 20)
```

WOKWI

SAVE

SHARE

Autonomous - Obstacle Avoidance Robot with Telematics data Copy

Docs

B

sketch.ino

diagram.json

libraries.txt

Library Manager

```
97 // if in front the distance of ultrasonic sensor and obstacle is less than 20cm
99 {
129     /*if (distance1 <= 20)
130     {
131         for (int i = 1; i < 4; i += 2)
132         {
133             digitalWrite(stepPin[i], LOW);
134         }
135         for (int i = 0; i < 4; i += 2) // downside motors
136         {
137             digitalWrite(stepPin[i], LOW);
138         }
139     }*/
140 }
141 // if the front distance between ultrasonic sensor and obstacle is greater than 20 cm
142 else {
143     for (int i = 1; i < 4; i += 2) // upwardside motors
144     {
145         digitalWrite(dirPin[i], HIGH);
146     }
147     for (int i = 1; i < 4; i += 2)
148     {
149         digitalWrite(stepPin[i], HIGH);
150     }
151     delayMicroseconds(1000);
152     for (int i = 1; i < 4; i += 2)
153     {
```


sketch.ino

diagram.json

libraries.txt

Library Manager

```
97 // if in front the distance of ultrasonic sensor and obstacle is less than 20cm
142 else {
149   digitalWrite(stepPin[i], HIGH);
153   {
154     digitalWrite(stepPin[i], LOW);
155   }
156   delayMicroseconds(1000);
157   for (int i = 0; i < 4; i += 2 ) // downside motors
158   {
159     digitalWrite(dirPin[i], LOW);
160   }
161   for (int i = 0; i < 4; i += 2 )
162   {
163     digitalWrite(stepPin[i], HIGH);
164   }
165   delayMicroseconds(1000);
166   for (int i = 0; i < 4; i += 2 )
167   {
168     digitalWrite(stepPin[i], LOW);
169   }
170   delayMicroseconds(1000);
171 }
172
173 int t = dht.readTemperature();
174 Blynk.virtualWrite(V0, t); // Send data to Virtual Pin V0
175 delay(1000);
176 int h = dht.readHumidity(); // Read from GPIO
```

sketch.ino

diagram.json

libraries.txt

Library Manager

```
97 // if in front the distance of ultrasonic sensor and obstacle is less than 20cm
142 else {
171 }
172
173 int t = dht.readTemperature();
174 Blynk.virtualWrite(V0, t); // Send data to Virtual Pin V0
175 delay(1000);
176 int h = dht.readHumidity(); // Read from GPIO
177 Blynk.virtualWrite(V1, h); // Send data to Virtual Pin V1
178 delay(1000); // Update every second
179
180 }
181
```