Stefan

**Projektarbeitname:** Fahrzeugdatenanalysesoftware mit Serviceschnittstelle und Simulationsplattform

**Zweckbeschreibung:**

Im Sinne von big data und predictive maintanence werden aus Sicht eines fiktiven KFZ-Herstellers Fahrzeugdaten gesammelt und analysiert um frühzeitig defekte zu erkennen , zu vermeiden , Langzeitauswertungen aufstellen zu können und eventuelle Garantieansprüche nachvollziehen zu können.

Um dies umzusetzen müssen als erstes alle Daten vorhanden sein und an eine Zentrale Stelle übertragen werden. Dies ist durch SimKarten und alle nötigen Sensoren zur Datenerfassung in den Fahrzeugen sichergestellt.

Die Daten werden dann von Zentraler Stelle ausgewertet, gespeichert und Dargestellt. Eine Serviceschnittstelle zum Kundensupport und Defektfrüherkennung sollen ebenso vorhanden sein.

**Projektbeschreibung:**

In unserem Projekt werden eine Anzahl an Fahrzeugen Mithilfe von Node Red Simuliert um Daten wie GPS-Koordinaten, Motortemperatur, Drehzahl, Geschwindigkeit, KM-Stand, Bremsverschleiss, Motorölstand zu generieren.

Die Simulation erfolgt vollautomatisch (es kann aber durch ein Userinterface auf den Simulationsverlauf Einfluss genommen werden) und deckt alle Testfälle, wie z.B.:

* Drehzahl wiederholt über kritischem Schwellwert -> Serviceintervall automatisch durch den Controller(Zentrale Stelle) herabgesetzt -> info an Bordcomputer.
* Kundendienst nicht gemacht, Launchcontroll starts, Änderung an Motorsteuerung erkannt -> Gerantieanspruch verloren.
* Unfall mir oder ohne auslösen des Airbags, dementsprechend nur Eintrag in Datenbank, Meldung an Fahrzeug oder sofortiges verständigen von Krankenwagen und Pannendienst.
* Keine Ereignisse, nur Datenauswertung und Daten in Datenbank ablegen
* Bremsenverschleissmeldung -> Rückmeldung Werkstatt aufsuchen auf den Bordcomputer
* Motorölstand/km -> Rückmeldung Werkstatt aufsuchen

Die Kommunikation zwischen Fahrzeug und Controller, der die Zentrale Stelle zur Auswertung darstellt, wird über das machine to machine Protocol MQTT realisiert. Alle Daten die generiert, gespeichert, dargestellt und analysiert werden, laufen über den Controller, welcher auch die Mustererkennungsverfahren (Pattern-Analysen) durchführt.

Eine MySQL Datenbank wird zur Datenspeicherung und als Quelle für Langzeitanalysen eingesetzt.

Die Darstellung der Fahrzeugspezifischen Livedaten, Langzeitauswertungen und Serviceinformationen (Garantie, Unfälle, nächster Service) erfolgt durch eine GUI oder Website(wäre Plattformunabhängig).

**Realisierung:**

1. Raspberry 1 (einzelner Raspberry; läuft immer)
   1. Controller
      1. Bekommt die Simulationsdaten, verarbeitet diese und übergibt sie an die Datenbank
      2. Übertragungsprotokoll MQTT
   2. Bereitstellen einer Webanwendung (HTML, CSS)
      1. dadurch plattformunabhängig
      2. Login über Benutzerprofil (Aus Datenbank)
      3. Karte (Google Maps) mit Livedaten über Fahrzeuge
2. Datenbank (Laptop)
   1. Logintabelle (Benutzername, Passwort (gespeichert als Hashcode))
   2. Tabelle aller Fahrzeuge (ID, Kennzeichen, Hersteller)
   3. Tabelle eines einzelnen Fahrzeugs
      1. ID
      2. Fahrgestellnummer
      3. ~~Hersteller~~
      4. Gefahrene KM
      5. Tankfüllung
      6. Motortemperatur
      7. Servicedatum (wann war das Auto schon beim Service? 🡪 01.05.19, 30.06.19)
      8. Unfälle (Unfall am 02.02.19, 10.05.19)
      9. ~~Fahrten (05.05.19 Erlangen 🡪 Nürnberg)~~
   4. Unfalltabelle
      1. Fahrgestellnummer
      2. GPS Koordinate
      3. Zeitstempel
      4. Wer hat die Reparatur durchgeführt (BMW Werkstatt Erlangen, Günther-Scharowsky-Straße)
      5. Reparaturzeit (02.02.19, 10:00:00 – 10:00:10)
3. Raspberry 2 (Simulation, Fahrzeugcockpits)
   1. Simulation
      1. Node Red
         1. Browser Oberfläche zum Starten einzelner Fahrzeuge über Buttons
         2. Fahrsimulation (Startpunkt, Endpunkt, Aktuelle GPS Position)
         3. Jedes Fahrzeug hat eigenen /Topic wird aber zusätzlich durch Fahrgestellnummer Identifiziert
         4. Fehlersimulation auch über Buttons möglich
            1. Unfall mit oder ohne Airbag auslösen (button)
            2. Motortemperatur über Slider einstellbar
            3. Bremsverschleiß über Slider einstellbar um plötzlichen, unerwarteten Defekt zu simulieren)
            4. Pro gefahrenen KM z. B. Bremsverschleiß – 1 %
            5. Motorölstand slider
            6. Launch control start (button)
      2. MQTT Broker (Mosquitto)
         1. Wäre in der Praxis in jedem einzelnen Fahrzeug
         2. Erfasst sämtliche Daten sämtlicher Sensoren des Fahrzeugs
         3. Sendet diese ungefiltert zur Weiterverarbeitung an den Controller (Raspberry 1)
      3. Fahrzeugcockpit
         1. Oberfläche Node-Red
         2. ~~Jedes Fahrzeug hat einen einzelnen "Benutzer"~~
         3. ~~Dieser kann sich über die Node Red Oberfläche anmelden~~
         4. Bordcomputerdarstellung
            1. KM Stand
            2. Motortemperatur
            3. Servicemeldungen (erhalten durch Webanwendung; müssen von hier bestätigt, bzw. ausgeführt werden)
            4. Unfallmeldungen (Reparaturauftrag muss bestätigt werden)

Im "Hauptmenü" Auswahl zwischen

1. Unfallmeldungen (Alle Fahrzeuge; neue Unfallmeldungen ploppen auf zur Bestätigung)
2. Fahrzeugauswahl (Auswahl eines einzelnen Fahrzeugs anhand von z. B. Marke, Kennzeichen, ID)
   1. Unfallmeldungen des jeweiligen Fahrzeugs (Abruf aus Datenbank)
   2. Livedaten (Fahrzeug fährt gerade, Tankfüllung, Motortemperatur) 🡪 Abruf von Livedaten erhalten vom Controller
   3. Serviceinformationen (Nächster Kundendienst in … km, kritische Situationen (Motordrehzahl mehrmals in kurzer Zeit über bestimmtem Wert, kann auf einen Defekt hindeuten)) 🡪 Abruf aus Datenbank
   4. Langzeitauswertungen (Gefahrene KM in der letzten Woche, Motortemperaturdiagramm, etc) 🡪 Abruf aus Datenbank
3. (Erfolgt automatisch wenn neues Auto Hersgestellt meldet es sich durch neuen topic und erscheint autmatisch in der GUI)Neue Fahrzeuge können angelegt werden, dadurch wird dieses in die Tabelle aller Fahrzeuge eingetragen und eine eigene Tabelle für dieses Fahrzeug wird angelegt und nach Aktionen des Fahrzeugs mit Werten befüllt.
4. Damit wird dieses Fahrzeug auch in der Fahrzeugauswahl auf der HTML Seite auswählbar.

Info aus dem INTERNETZ 😊:

**BMW: Predictive Maintenance-System sieht viele Fahrzeugausfälle vorher**

Neben der direkten Analyse von Schwingungs- oder Verschleißdaten kommen die neuen Analysemethoden aber auch anderweitig zum Einsatz. Beim Automobilhersteller BMW nutzt man z. B. die Text- und Datamining-Lösung IBM-SPSS, welche die vorausschauende Analyse unterstützt. Ging es ursprünglich nur darum, die Häufigkeit von teuren Reparaturen zu erkennen, nutzen heute umfangreiche Mustererkennungsverfahren (Pattern-Analysen) alle verfügbaren Daten. Hierzu gehören sowohl technische Daten aus der eigenen Produkt- und Fahrzeugdatenbank, die maschinellen Informationen, die ein Fahrzeug selbst speichert, sowie die Rückmeldungen der Händler.

Die Systeme zur vorausschauenden Instandhaltung sind damit in der Lage, teure Reparaturen oder gravierende Fahrzeugausfälle vorherzusehen und vorbeugende Aktivitäten einzuleiten, bevor größere Schäden entstehen. „Wir können heute mit hoher Wahrscheinlichkeit vorhersagen, ob und wann ein bestimmtes Fahrzeug ausfallen wird. Meistens sind die Ursachen dafür vermeidbar, indem beim nächsten Wartungsintervall die entsprechenden Teile ausgetauscht werden. Im Extremfall können wir den Halter auch anrufen und in eine Werkstatt bitten“, erläuterte ein Sprecher vom BMW-Forschungszentrum den Einsatz dieser Analytics-Software.

## Das Ziel: Komplexe Systeme zu beherrschen und die Kundenerfahrung zu verbessern

Auch in anderen Bereichen, in denen durch Artificial Intelligence komplexe Systeme beherrschbar werden, ist noch Entwicklungsarbeit zu leisten. In einigen Fällen sind Fortschritte aber nur eine Frage der Zeit: Eines der wesentlichen Merkmale von intelligenten Algorithmen ist ihre **Lernfähigkeit**.

Um die komplexen Zusammenhänge der Gesamtverkehrslage zu interpretieren, ist es wie bei „Shimon“, der Musik komponieren und spielen kann, zunächst nötig, die Voraussetzungen zu lernen. Dazu gehört es, die Verkehrsregeln zu verstehen, das Verhalten der Verkehrsteilnehmer zu analysieren und bis zu einem gewissen Grad vorherzusagen sowie zahlreiche Lösungsansätze zur Optimierung durchzuprobieren.

Für Autohersteller lohnt es darüber hinaus eine automatisierte Auswertung von Fahrzeugdaten anzustreben. Daten, wie die zur Abnutzung von Bremsbelägen und Filtern oder zum Ölverbrauch, sind wertvolle Informationen. Sie lassen Schlüsse über das verwendete Material zu, dienen zur Prognose von möglichen Schäden und Ausfällen, und stellen nicht zuletzt die Grundlage zur automatisierten Planung von Werkstattterminen über den Bordcomputer dar. Denn genau dies ist das übergeordnete Ziel, das all die genannten Entwicklungschancen letztlich verfolgen: Die **konsequente Verbesserung** der Kundenerfahrung und die Erhöhung von Komfort und Sicherheit im Straßenverkehr insgesamt.

Patrick

Programm zur Verwaltung eines Fuhrparks. Oberfläche für den Benutzer: HTML Seite auf einem Raspberry Pi. Auf der Seite wird nach einem Login gefragt (Benutzer (wahrscheinlich nur einer) in einer Datenbank gespeichert). Abgleich des Passworthashes mit dem in der Datenbank gespeicherten Hashcode (verschlüsselt).

3 Teilgebiete:

1. Raspberry 1 (einzelner Raspberry; läuft immer)
   1. Controller
      1. Bekommt die Simulationsdaten, verarbeitet diese und übergibt sie an die Datenbank
      2. Übertragungsprotokoll MQTT
   2. Bereitstellen einer Webanwendung (HTML, CSS)
      1. dadurch plattformunabhängig
      2. Login über Benutzerprofil (Aus Datenbank)
      3. Karte (Google Maps) mit Livedaten über Fahrzeuge
2. Datenbank (könnte auf Raspberry 1 laufen)
   1. Logintabelle (Benutzername, Passwort (gespeichert als Hashcode))
   2. Tabelle aller Fahrzeuge (ID, Kennzeichen, Hersteller)
   3. Tabelle eines einzelnen Fahrzeugs
      1. ID
      2. Kennzeichen
      3. Hersteller
      4. Gefahrene KM
      5. Tankfüllung
      6. Motortemperatur
      7. Servicedatum (wann war das Auto schon beim Service? 🡪 01.05.19, 30.06.19)
      8. Unfälle (Unfall am 02.02.19, 10.05.19)
      9. Fahrten (05.05.19 Erlangen 🡪 Nürnberg)
   4. Unfalltabelle
      1. Welches Fahrzeug?
      2. Wo war der Unfall?
      3. Wann war er?
      4. Wer hat die Reparatur durchgeführt (BMW Werkstatt Erlangen, Günther-Scharowsky-Straße)
      5. Reparaturzeit (02.02.19, 10:00:00 – 10:00:10)
3. Raspberry 2 (Simulation, Fahrzeugcockpits)
   1. Simulation
      1. Node Red
         1. Browser Oberfläche zum Starten einzelner Fahrzeuge über Buttons
         2. Fahrsimulation (Startpunkt, Endpunkt, Aktuelle GPS Position)
         3. Fehlersimulation über Buttons
            1. Unfall
            2. Motortemperatur über Slider einstellbar
            3. Bremsverschleiß über Slider einstellbar um plötzlichen, unerwarteten Defekt zu simulieren)
            4. Pro gefahrenen KM z. B. Bremsverschleiß – 1 %
            5. Reifendruck über Slider einstellbar
      2. MQTT Broker (Mosquitto)
         1. Wäre in der Praxis in jedem einzelnen Fahrzeug
         2. Erfasst sämtliche Daten sämtlicher Sensoren des Fahrzeugs
         3. Sendet diese ungefiltert zur Weiterverarbeitung an den Controller (Raspberry 1)
      3. Fahrzeugcockpit
         1. Oberfläche Node-Red
         2. Jedes Fahrzeug hat einen einzelnen "Benutzer"
         3. Dieser kann sich über die Node Red Oberfläche anmelden
         4. Befindet sich dann im Fahrzeugcockpit und erhält teilweise Informationen über das eigene Fahrzeug
            1. KM Stand
            2. Tankfüllung
            3. Servicemeldungen (erhalten durch Webanwendung; müssen von hier bestätigt, bzw. ausgeführt werden)
            4. Unfallmeldungen (Reparaturauftrag muss bestätigt werden)

Im "Hauptmenü" Auswahl zwischen

* Unfallmeldungen (Alle Fahrzeuge; neue Unfallmeldungen ploppen auf zur Bestätigung)
* Fahrzeugauswahl (Auswahl eines einzelnen Fahrzeugs anhand von z. B. Marke, Kennzeichen, ID)
  + Unfallmeldungen des jeweiligen Fahrzeugs (Abruf aus Datenbank)
  + Livedaten (Fahrzeug fährt gerade, Tankfüllung, Motortemperatur) 🡪 Abruf von Livedaten erhalten vom Controller
  + Serviceinformationen (Nächster Kundendienst in … km, kritische Situationen (Motordrehzahl mehrmals in kurzer Zeit über bestimmtem Wert, kann auf einen Defekt hindeuten)) 🡪 Abruf aus Datenbank
  + Langzeitauswertungen (Gefahrene KM in der letzten Woche, Motortemperaturdiagramm, etc) 🡪 Abruf aus Datenbank
* Neue Fahrzeuge können angelegt werden, dadurch wird dieses in die Tabelle aller Fahrzeuge eingetragen und eine eigene Tabelle für dieses Fahrzeug wird angelegt und nach Aktionen des Fahrzeugs mit Werten befüllt.
* Damit wird dieses Fahrzeug auch in der Fahrzeugauswahl auf der HTML Seite auswählbar.

Beispielsituation:

Ein Unternehmen hat 200 Fahrzeuge. Ein Fahrzeug bekommt einen Auftrag und fährt los von Startpunkt (bzw. aktuellem Standort) zum Ziel. Dieses Fahrzeug wird in der HTML Seite als "aktiv, im Dienst, sonst irgendwas) angezeigt. Wenn dieses Fahrzeug ausgewählt wurde, ("Haupt"Seite dieses Fahrzeugs) wird die aktuelle Position auf einer Karte angezeigt. Wenn das Fahrzeug fährt, wird in den Livedaten z. B. der KM Stand erhöht. Durch das Erhöhen des KM Standes rückt der nächste Servicetermin näher, die Tankfüllung sinkt und der Bremsenverschleiß steigt. Wenn diese Situation eintritt (Fahrzeug muss in 1000 km zum Service) erscheint eine gelbe Mitteilung am Bildschirm. Werkstätten in der Nähe des Fahrzeugs werden angezeigt. Diese Informationen können, müssen aber nicht, ans Fahrzeug weitergeschickt werden. Wenn das Fahrzeug noch z. B. 100 km bis zum Service hat, erscheint eine rote Mitteilung am Bildschirm (unabhängig davon, wo man sich gerade in der HTML Seite befindet), "Fahrzeug MUSS zum Service" Werkstatt in der Nähe muss ausgewählt werden und an das Fahrzeug geschickt werden. Im Cockpit des Fahrzeugs erscheint diese Meldung und muss bestätigt werden. Das Fahrzeug unterbricht seine aktuelle Fahrt und fährt zur ausgewählten Werkstatt, um den Service durchführen zu lassen. Während dieser Zeit steht das Fahrzeug nicht zur Verfügung. Das wird in der Serviceoberfläche angezeigt (Service wird gerade durchgeführt). Zeit des Services wird in Datenbank geschrieben und ist somit im Fahrzeugmenü unter "Serviceinformationen" sichtbar. Sobald der Service abgeschlossen ist, setzt das Fahrzeug seinen Weg fort. Wird ein Unfall simuliert, ploppt ein rotes Fenster in der HTML Seite (Unabhängig von der aktuellen Position auf der Seite) auf und es muss eine Werkstatt in der Nähe ausgewählt werden, die die Reparatur durchführt. Es gibt 2 Arten von Unfällen: Unfälle mit und Unfälle ohne Personenschaden.

Bei Unfällen ohne Personenschaden wird nur die Werkstatt informiert, bei Unfällen mit Personenschaden wird Werkstatt und Krankenwagen alarmiert. Im Fahrzeugcockpit wird angezeigt, dass ein Unfall stattgefunden hat. Wenn von der HTML Seite die Pannenhelfer geschickt wurden, wird angezeigt, dass diese da sind und das Fahrzeug repariert wird. (Dauer z. B. 10 Sekunden). Wenn das Fahrzeug repariert wurde, setzt es seinen Weg fort. Wie beim Service werden Informationen über den Unfall (Ort, Zeit, Reparaturdauer, Personenschaden ja / nein) in die Datenbank in die Fahrzeug- und Unfalltabelle eingetragen.

Wenn ein Motorschaden oder ähnliches auftritt, wird auch das in der HTML Oberfläche angezeigt und die nächstgelegene Werkstatt wird alarmiert. Wieder muss das vom Fahrzeug bestätigt werden und die Reparatur wird durchgeführt. Auch hier werden Daten über den Defekt in die jeweils passende Datenbank geschrieben und können im Nachhinein wieder in der Oberfläche abgerufen werden.