Entwicklung eines Konzepts zur digitalen Erfassung von Zählerständen an ATESTEO-Prüfständen

Yannick Zgodda (Matr. Nr: 3567665)

Seminar

ATESTEO GmbH & Co. KG

Dezember 2024

1 Erklärung

Hiermit versichere ich, dass ich die Seminararbeit mit dem Thema „Entwicklung eines Konzepts zur digitalen Erfassung von Zählerständen an ATESTEO-Prüfständen“ selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel bentutzt habe, alle Ausführungen die anderen Schriften wörtlich oder sinngemä0 entnommen wurden, kenntlich gemacht sind und die Arbeit in gleicher oder ähnlicher Fassung noch nicht Bestandteil einer Studien- oder Prüfungsleistung war.
Ich verpflichte mich, ein Exemplar der Seminararbeit fünf Jahre aufzubewahren und auf Verlangen dem Prüfsungsamt des Fachbereiches Medizintechnik und Technomathematik auszuhändigen.

Aachen, 12.12.2024 

Ort, Datum, Unterschrift Student

Inhaltsverzeichnis

2 Einleitung

2.1 Themenvorstellung

Die Firma ATESTEO GmbH & Co. KG ist ein weltweit führender Anbieter von Prüfstandsdienstleistungen für die Automobilindustrie mit sieben Standorten weltweit und insgesamt über 600 Mitarbeitern. Die Dienstleistungen umfassen Prüfläufe von Motoren, Getrieben, Elektromotoren, Akkus und Antriebssträngen sowie Abgasanalysen, Lautstärkemessungen und Fahrzeugerprobungen. Als Beispiel eines Getriebetests wird bei einem Prüflauf der Prüfling mit Elektromaschinen verbunden, die den Luft-, Trägheits- und Reifenwiderstand simulieren und damit Last für den Prüfling erzeugen.
Das Angebot von ATESTEO umfasst außerdem die Entwicklung und den Verkauf von Messwellen für kundeneigene Prüfstände, sowie die Kalibrierung von Messwellen und elektrischen Leistungsmessgeräten. Die Softwareabteilung (intern PS - Product Software) entwickelt Software zur internen Nutzung. So ist die Abteilung für die Automatisierungssoftware PDES (Prüfstandsdatenerfassung und Steuerung) verantwortlich, mit der die Prüfstände betrieben werden. Neben diesem Hauptprodukt betreut die Abteilung diverse kleinere und mittlere Softwareprojekte zur internen Nutzung.

2.2 Motivation

Bei einem, wie in der Einleitung beschriebenen, Prülauf muss der Stromverbrauch erfasst und protokolliert werden, da dieser nach Beendigung des jeweiligen Projekts in Rechnung gestellt und für die Kostenplanung überwacht wird. Neben den oben genannten Elektromaschinen wird z.B. auch der Stromverbrauch des Schaltkastens oder des Prüflings überwacht. All diese Geräte sind mit entsprechenden Verbrauchszählern ausgerüstet. Da viele Zähler keine Schnittstelle zum automatischen Auslesen besitzen, werden alle Zähler von Mitarbeitern in regelmäßigen Abständen abgelesen und die Werte auf Papier oder in Exceltabellen festgehalten. Mithilfe dieser Daten wird dem Kunden der entsprechende Stromverbrauch in Rechnung gestellt. Das händische Ablesen und Pflegen der Zählerstände erfolgt zurzeit dezentral. Zum Monatsende müssen alle Papierlisten und Exceltabellen eingesammelt und die Summen berechnet werden. Endet ein Projekt nicht am letzten Tag eines Monats, muss der Stromverbrauch von Hand bis zu diesem Datum berechnet werden. Dass die einzelnen ATESTEO-Standorte die Erfassung der Zählerstände unterschiedlich organisieren, erschwert eine konzernweite Auswertung. Da ATESTEO nach ISO 50001 zertifiziert ist, ist die Firma verpflichtet, 90% ihres Stromverbrauchs zu identifizieren. Das heißt, es muss ersichtlich sein, wo welcher Anteil des vom Energieversorger berechneten Gesamtverbrauchs verursacht wurde. Um die Erfassung der Zählerstände zentral und einheitlich zu organisieren, wurde die Abteilung PS gebeten, ein Programmkonzept zu erarbeiten. Die gefundenen Konzeptionsideen werden im Folgenden erläutert, bewertet und verglichen. Im Anschluss wird dargelegt, welche Gründe den Ausschlag für die finale Lösung gegeben haben. Danach folgt ein Ausblick auf die weitere Entwicklung des Projekts.

3 Konzeptvorstellung

3.1 Anforderungen

Der Standort Kassel wurde als Pilotstandort ausgewählt und sollte daher von Beginn an in das Projekt eingebunden werden. Im Vorfeld der Konzeption traf ich mich daher mit den Kollegen des ATESTEO-Standortes in Kassel, dem Chef des Testing-Bereichs sowie meinem Bereichsleiter, um die Anforderungen an die Lösung zu besprechen. Folgende Anforderungen haben sich hier herausgestellt:

- Es soll möglich sein, einen Datumsbereich festzulegen und für diesen Bereich die Zählerwerte der Stromzähler im Prüfstand sowie deren jeweilige Summe angezeigt zu bekommen.
- Für die Identifikation des Stromverbrauchs nach ISO 50001 soll man den in Rechnung gestellten Stromverbrauch eingeben können und anschließend eine Tabelle angezeigt bekommen, die die Verbräuche aller Räume sowie deren prozentualen Anteil am Gesamtverbrauch enthält
- Bereits eingegebene Werte sollen im Falle eines Fehlers im Nachhinein korrigiert werden können.
- Eine Korrektur darf nur mit entsprechenden Rechten möglich sein und muss mit einer Begründung protokolliert werden.
- Zur Speicherung der Zählerstände soll die bereits vorhandene Datenbank des Programms AIM genutzt werden. (s.u.)

AIM steht für ATESTEO-Inventory-Management und ist ein Programm zur Verwaltung von Geräten und ihren Aufenthaltsorten. Es bietet die Möglichkeit, Geräte in die jeweiligen Prüfstandsräume „einzubuchen“,um sie über das Programm schnell lokalisierbar zu machen. Mit AIM können darüber hinaus auch Wartungen und Reparaturen sowie Kennwerte, wie z.B. das maximale Drehmoment oder Betriebsstunden, eingesehen werden. AIM kann bis jetzt aber immer nur einen Wert zu einem Kennwert speichern. Für die Digitalisierung der Zählerstandserfassung ist es aber nötig, dass auch Historien von Daten gespeichert werden. Für diese Funktion muss die AIM-Datenbank um eine weitere Tabelle ergänzt werden.

Abbildung 3.1: Beispiel eines Smartmeters mit serieller Schnittstelle
[7]

ATESTEO
Excellence in drivetrain testing

Stromerfassung Prüfstand [redacted]

Pos.	KW	Datum	Tag	Prüfstand	Betriebsstunden [redacted]	Zähler-ID - 2P6 Messschrank und Steckdosen im PST [kWh]	Zähler-ID - 3P2 Frequenzumrichter [kWh]	Verbrauch Kilowattstunden [kWh]	Bemerkungen
23	13	[redacted]	Mo	[redacted]	29804,11	144782,6	1090088,3	189,82	0,00
24			Di		29804,11	144797,2	1090140,0	66,29	0,00
25			Mi		29804,11	144797,2	1090140,0	0,00	0,00
26			Do		29812,22	144848,2	1090239,9	150,87	8,11
27			Fr		29836,32	144848,2	1090239,9	0,00	24,10
28			Sa		29860,38	144848,2	1090239,9	0,00	24,06
29			So		29883,39	144848,2	1090239,9	0,00	23,01
30	14		Mo		29907,39	144848,2	1090239,9	0,00	24,00
31			Di		29923,20	145103,7	1095084,8	5100,42	15,81
				Monatsabschluss:	185,69	1158,17	8856,35	10497,70	

Abbildung 3.2: Beispiel einer bisher genutzte Exceltabelle

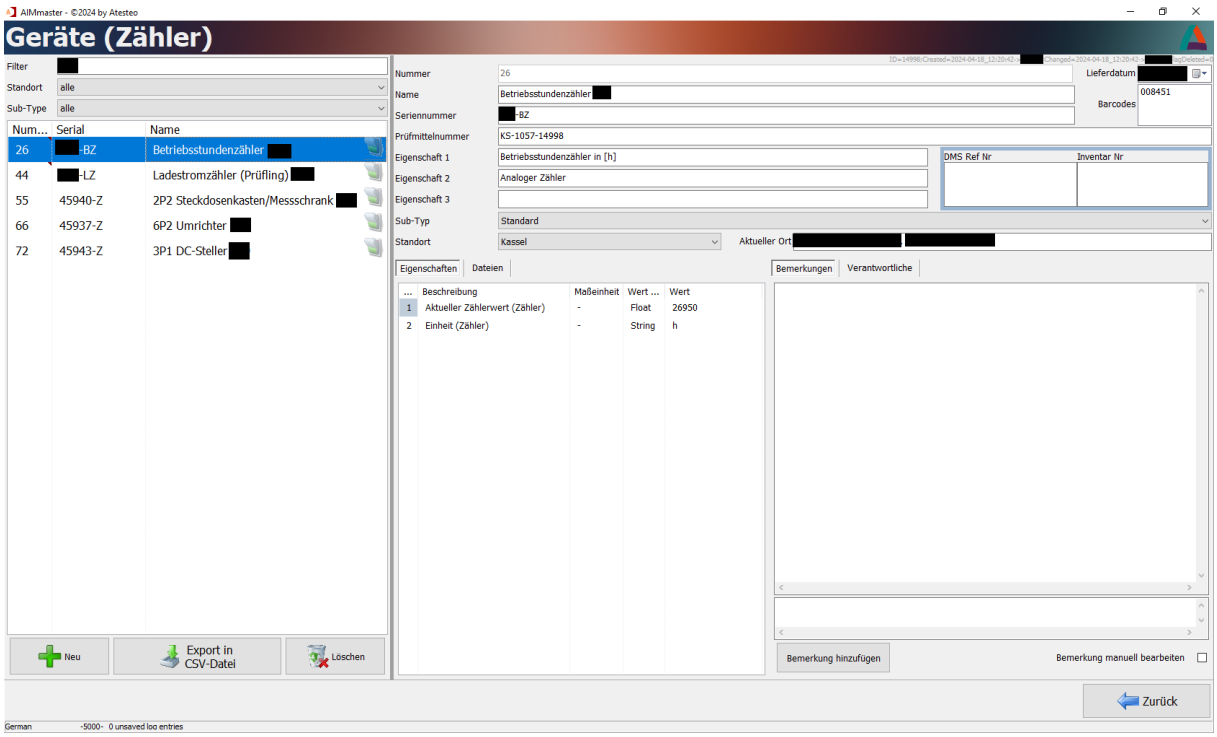

Abbildung 3.3: Ansicht der Zähler eines Prüfstandes in AIM

3.2 Konzeptionsideen

Basierend auf den Anforderungen aus dem vorherigen Abschnitt habe ich drei verschiedene Konzeptionsideen entwickelt, die ich im folgenden Abschnitt genauer erläutern, bewerten und vergleichen werde. Die erste Idee ist eine Erfassung mit einem Mikrocontroller in jedem Prüfstand. Die zweite Idee umfasst eine webbasierte Lösung zur Erfassung und Auswertung der Zählerstände. Die dritte Idee ist die Entwicklung eines Delphi-Programms zum selben Zweck. Dabei dient für alle Konzepte die AIM-Datenbank als Speicherort.

Bei den ersten beiden Lösungen sollen die Auswertung und Korrektur webbasiert umgesetzt werden. Dabei soll es eine Seite zur Auswertung geben auf der, neben dem gewünschten Standort und Prüfstand, auch ein Datumsbereich festgelegt werden kann. Wird dieses Formular abgeschickt, soll die Webseite eine Tabelle erzeugen, in der für jedes Gerät im Prüfstand die jeweiligen Tagesstände ersichtlich sind. Ebenfalls soll die Differenz des Stromverbrauchs zum Vortag enthalten sein. Die letzte Zeile der Tabelle soll die Summen der jeweiligen Geräte und den gesamten Stromverbrauch über den ausgewählten Datumsbereich anzeigen. Im Eingabedialog soll dazu noch die Möglichkeit bestehen, optional einen Strom-Monatsverbrauch einzutragen, um die, wie in den Anforderungen erklärt, prozentuale Aufteilung auf die jeweiligen Prüfstände zu berechnen.

Die Korrektur soll auf einer eigenen Seite umgesetzt werden. Es ist durchaus möglich, dass es zu einem Tippfehler bei der Eingabe kommt oder ein Wert für das falsche Gerät eingetragen wird. Um dies zu beheben, soll diese Seite eine Art Fusion aus Erfassung und Auswertung sein. Einerseits soll sie einen Eingabebereich für die Geräte besitzen, der noch um eine Datumsauswahl für den Tag des Nachtrags erweitert ist. Gleichzeitig soll sie aber auch noch eine Tabelle der bisherigen Werte anzeigen, damit der Tag der Fehleingabe identifiziert werden kann. Dass diese Funktionalität auf einer zusätzlichen Seite abgebildet wird, liegt daran, dass sie nicht von jedem genutzt werden darf und mit einer Zugangskontrolle versehen werden soll. Eingaben und Auswertungen sind grundsätzlich von jedem Mitarbeiter nach erfolgreichem Login möglich. Eine Korrektur stellt aber einen Ausnahmefall dar und kann im Zweifel auch missbraucht werden, weshalb sie nur von ausgewählten Mitarbeitern gemacht werden darf. Dazu müssen der Benutzername sowie die geänderten Werte erfasst und eine Begründung angegeben werden.

3.2.1 Erfassung mittels Mikrocontroller

Eine Lösungsidee ist, die Erfassung mit einem Mikrocontroller umzusetzen, wie beispielsweise einem Rasberry Pi [8]. Mikrocontroller sind eigenständige, sehr kleine Computer, die auch im Alltag an diversen Stellen für Mess- und Regelaufgaben eingesetzt werden (wie z.B. beim Betrieb von Waschmaschinen). Sie haben den großen Vorteil, dass sie sehr kompakt in ihrer Bauform sind und nur wenig Strom verbrauchen. Viele moderne Verbrauchszähler, sogenannte Smartmeter, bieten die Möglichkeit einer Schnittstelle, über die der Zählerstand abgefragt werden kann. Wobei hier unter Umständen einige Neubeschaffungen nötig sind, da nicht alle Geräte mit modernen Smartmetern ausgestattet sind.

Für diese Lösung würde der Controller im Prüfstand angebracht und könnte über entsprechende Schnittstellen die Werte der Zähler erfassen und den erfassten Wert autonom für den Prüfstand in die Datenbank eintragen. Die angeschlossenen Geräte und der Standort des Controllers können über eine Konfigurationsdatei festgelegt werden, welche vom Programm auf dem Controller ausgelesen wird. So muss bei einem Geräte- oder Standortwechsel nur die Datei und nicht das Programm auf dem Controller angepasst werden. Kommt es zu einem Netzwerkausfall, könnten die Controller die erfassten Werte für eine gewisse Zeit puffern und, wenn die Verbindung wiederhergestellt ist, in die Datenbank schreiben. Die Auswertung und Korrektur sollen über einen firmenintern gehosteten Server als Webseite erreichbar sein. Die genaue Funktionalität wurde bereits im Abschnitt „Anforderungen“behandelt.

Da wir Node-Red [1] bei uns in der Firma bereits an verschiedenen Stellen eingesetzt ha-

ben, verfügen wir über entsprechende Erfahrung mit der Umgebung. Daher würde sich für die Programmierung des Mikrocontrollers und der oben erklärten Webseiten zur Auswertung und Korrektur Node-Red anbieten. Node-Red ist ein auf der Programmiersprache NodeJS basierendes grafisches Entwicklungswerkzeug, bei dem Knoten Informationen über Nachrichten untereinander austauschen. Die Knoten können aneinandergereiht in sogenannten „Flows“verbunden werden. Die Nachricht läuft dann von einem Startknoten zum nächsten Knoten, wird dort auf verschiedene Art und Weise verarbeitet und dann zum nächsten Knoten geschickt.

Diese Lösung bietet den Vorteil, dass die Erfassung der Zählerwerte automatisiert erfolgt und kein weiteres Eingreifen der Mitarbeiter erfordert, wenn die Zähler die entsprechenden Schnittstellen besitzen. Aufgrund der bereits vorhandenen Erfahrung mit Node-Red würden wir keine Einarbeitungszeit benötigen und könnten schnell in die Entwicklung einsteigen. Dazu bietet Node-Red viele bereits vorgefertigte Knotentypen, z.B. für Datenbankabfragen oder zur Kommunikation mit Smartmetern, sodass der Entwicklungsaufwand geringer ist und nur weniger Code geschrieben werden muss.

Diese Lösung bringt jedoch auch einige Nachteile mit sich. So würde eine flächendeckende Einführung bedeuten, dass für jeden Prüfstand ein eigener Mikrocontroller beschafft, eingerichtet und montiert werden müsste. Außerdem müsste bei einem Update des Node-Red-Flows jeder Mikrocontroller einzeln mit dem neuesten Flow ausgestattet werden sowie Betriebssystem- und Node-Red-Updates auf den Controllern verteilt werden. Nicht alle Zähler verfügen über eine Schnittstelle zum Auslesen des Zählerstandes, sodass auch hier einige Neubeschaffungen vonnöten wären. Dies würde jedoch einen erheblichen Kostenaufwand verursachen, da es alleine am Standort Kassel 18 Prüfstände mit teilweise fünf zu überwachenden Zählern gibt.

Das Thema Informationssicherheit ist bei ATESTEO extrem wichtig, weshalb es nötig sein könnte, die Mikrocontroller in ein eigenes Netzwerk einzubinden, was den Inbetriebnahme- und Wartungsaufwand zusätzlich erhöhen würde.

Betrachtet man nun die Möglichkeiten zur Entwicklung der Webseiten, so zeigen sich auch hier Nachteile auf. Zwar wäre eine Entwicklung mittels Node-Red möglich, aber das Tool ist nicht für größere und komplexere Entwicklungen gedacht. Der Code wird aufgrund fehlender Strukturierung in Dateien und dem kleinen Eingabefenster schnell unübersichtlich und Debugging ist, wie in einer klassischen Entwicklungsumgebung, nicht möglich. Stattdessen müssen Ausgaben in die Debug-Konsole geschrieben werden, um Variablenwerte auszulesen oder zu prüfen, welche Verzweigung das Programm betreten hat. Das macht die Entwicklung eines größeren Programms zwar nicht unmöglich, aber ungleich ineffizienter, als bei einer klassischen Entwicklungsumgebung. Nicht unbedingt ein Nachteil, aber zusätzlicher Aufwand entsteht auch, weil die Webseiten bei der hausinternen IT auf einem Server mit entsprechender Ausfallsicherheit laufen müssen, da auf einem System, das möglichst durchgängig erreichbar sein muss, nicht einfach zu jeder Zeit Updates eingespielt werden können. Ein Ausfall oder ein Update zu einer ungünstigen Zeit, könnte eine erhebliche Auswirkung auf den Betrieb haben.

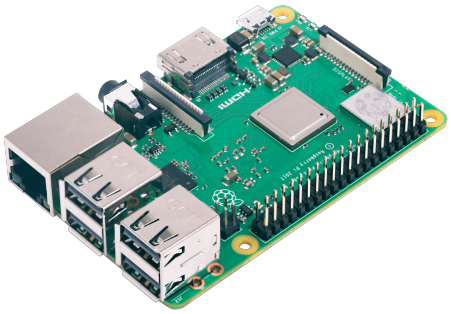

Abbildung 3.4: Beispiel eines Rasberry Pi Mikrocontrollers
[3]

3.2.2 Erfassung über Website

Eine weitere Konzeptidee besteht darin, die Erfassung über drei PHP [4] oder JavaScript [9] basierte Webseiten zu organisieren. Eine für die Erfassung, eine für die Auswertung und eine zur nachträglichen Bearbeitung vorheriger Werte für den Fall eines Fehlers. Auf der Seite zur Erfassung sollen der Standort und ein Prüfstand auswählbar sein. Für den ausgewählten Prüfstand werden dann die eingebuchten Geräte in der AIM-Datenbank abgefragt und für jedes Gerät ein Eingabefeld erstellt, in dem dann der Zählerwert eingetragen werden kann. Wird dieses Formular abgeschickt, werden die neuen Werte in die entsprechende Tabelle in der Datenbank geschrieben. Die Webseiten zur Auswertung bzw. Korrektur wurden bereits im Abschnitt „Anforderungen“ erklärt. Der einzige Unterschied zur vorherigen Lösung besteht hier in der Umsetzung mit PHP oder JavaScript.

Diese Lösung hat den großen Vorteil, dass sie von jedem beliebigen Gerät im Firmennetzwerk erreicht werden kann und keine zusätzliche Hardware beschafft und installiert werden muss. Deshalb wären auch keine Arbeiten außerhalb unserer Abteilung nötig. Die Lösung ist außerdem flexibler, da sie die eingebuchten Geräte direkt bei der AIM-Datenbank abfragen kann und nicht aus einer Konfigurationsdatei lesen muss, die bei einem Gerätewechsel angepasst werden müsste. Dafür müssen die Werte aber händisch gepflegt werden.

Problematisch würde sich jedoch die Entwicklung dieser Lösung gestalten, da in unserer Abteilung nur wenig Expertise zur Webentwicklung existiert. Hier wäre aus Gründen der Redundanz eine Schulung und Einarbeitung von mehreren Entwicklern nötig. Die Problematik der Server-Ausfallsicherheit besteht bei dieser Lösung analog zur Ersten, wobei ein Ausfall schlimmer wäre als in Konzept eins, da die Mikrocontroller die Daten puffern können, was eine Webseite nicht kann. Außerdem besteht durch die manuelle Auswahl des Prüfstandes das Risiko, dass Werte für ein falsches Gerät eingetragen werden.

3.2.3 Erfassung mit Delphi-Programm

Die dritte Idee zur Umsetzung des Projekts ist eine Windows-Anwendung in der Programmiersprache Delphi [5]. Auch hier sollen die drei Hauptfunktionen Darstellung der bisherigen Zählerwerte, Erfassung neuer Werte und Korrektur fehlerhafter Werte sein. Für jede Funktion sollte das Programm über eine eigene Ansicht verfügen.

Die Ansicht zum Einfügen neuer Zählerwerte soll die Möglichkeit bieten, den Standort und den Prüfstand auszuwählen und anschließend die dort eingebuchten Geräte anzeigen. Es wäre sogar denkbar, falls das Programm auf einem Prüfstandsrechner ausgeführt wird, den Namen des Prüfstands aus dem USB-Dongle des Prüfstandsrechners auszulesen. Wir nutzen USB-Dongle zum Kopier- und Lizenzschutz unserer Software. Zusätzlich sind dort prüfstandsspezifische Informationen wie die Prüfstandsnummer abgelegt.

Die zweite Ansicht soll, wie zuvor auch, die Auswertungen für einen Datumsbereich darstellen und soll ebenfalls für die Identifikation der jeweiligen Anteile am Stromverbrauch dienen. Zur Darstellung empfiehlt sich die Komponente „Virtual Treeview“ [2] für Delphi, da diese viele Optionen zur Anpassung des Designs bietet und sehr performant ist.

Die letzte Ansicht dient der Korrektur fehlerhafter Werte. Hier soll neben einem Eingabedialog für Standort, Prüfstand, Zählerstand und Nachtragsdatum auch eine Tabelle vorhanden sein, die die letzten Werte anzeigt, damit der Tag der Fehleingabe identifiziert werden kann. Die Kommunikation mit der Datenbank kann mit der in Delphi eingebauten Datenbankkomponente FireDAC [6] umgesetzt werden. Um die verschiedenen Programme, die in unserer Abteilung entwickelt werden, für die Anwender verfügbar zu machen, werden diese in einem dafür bestimmten Ordner auf einem Netzlaufwerk abgelegt. Das könnte für diese Delphi-Anwendung analog gemacht werden. Somit wäre ein Update sofort bei jedem Benutzer aktiv.

Der größte Vorteil dieser Lösung ist die Erfahrung in unserer Abteilung in der Arbeit mit Delphi. So wäre jeder Entwickler nach einer kurzen Einarbeitung in der Lage, das Programm zu

warten und zu erweitern, was bei den anderen Lösungen nicht bzw. nur bedingt der Fall ist. Da Delphi, anders als Node-Red, eine vollständige Entwicklungsumgebung mit modernen Debug-Techniken besitzt, ist die Fehlersuche erheblich einfacher und die Entwicklung dadurch schneller. Darüber hinaus kann die Anwendung, wenn sie auf einem Netzlaufwerk abgelegt wird, von jedem Rechner im Firmennetzwerk aus genutzt und leicht aktualisiert werden. Da die Information, an welchem Prüfstand das Programm ausgeführt wird, aus dem USB-Dongle ausgelesen werden kann, ist es nicht erforderlich, für jeden Rechner eine eigene Konfigurationsdatei anzulegen, was den Arbeitsaufwand ebenfalls verringert und das Risiko einer Eingabe für das falsche Gerät verringert.

4 Fazit und Ausblick

4.1 Fazit

Nach gründlicher Abwägung der drei möglichen Konzepte nach Entwicklungsaufwand, Wartbarkeit und Flexibilität entschied ich, dass die Lösung mittels Delphi-Anwendung die beste Möglichkeit für uns darstellt. Sie erfordert keine Anschaffung weiterer Hardware, wie bei der Lösung mittels Mikrocontroller, welche ebenfalls noch die Montage der Controller an jedem Prüfstand inklusive entsprechender Kabelverbindungen nötig machen würde. All das ist für die Delphi-Anwendung nicht nötig. Die Installation muss ebenfalls nicht auf diversen Geräten durchgeführt werden, da die Verteilung des Programms über ein Netzlaufwerk, auf das alle Rechner Zugriff haben, bewerkstelligt werden kann. Genauso müssen keine Betriebssystemupdates verteilt werden, da keine weitere Hardware nötig ist. Darüber hinaus muss nicht für jeden Prüfstand eine Konfigurationsdatei angelegt werden, wie es etwa beim Mikrocontroller der Fall gewesen wäre, da alle Informationen aus dem USB-Dongle des Prüfstandsrechners gelesen oder vom Nutzer manuell eingegeben werden können. Die Lösung über eine Webseite würde einen größeren Schulungs- und Einarbeitungsaufwand bei den Entwicklern erzeugen, da erst eine entsprechende Wissensbasis aufgebaut werden muss. Node-Red, wozu bereits entsprechende Erfahrung vorhanden ist, kommt, aufgrund der unzureichenden Debugging-Möglichkeiten und schnell entstehenden Unübersichtlichkeit, auch nicht als Entwicklungswerkzeug in Frage. Eine webbasierte Lösung würde auch keinen wirklichen Vorteil bezüglich Flexibilität bringen, da sowohl an den Prüfständen (Eingabe) als auch im Büro (Auswertung) Windows-Computer eingesetzt werden. Dazu besteht auf der Webseite auch immer das Risiko, dass Werte für ein falsches Gerät eingetragen werden, falls bei der Auswahl des Prüfstandes ein Fehler passiert. Das kann, aufgrund der automatischen Auswahl des Prüfstandes mithilfe des USB-Dongles bei der Delphi-Anwendung nicht passieren. Stark ins Gewicht fällt auch die große Erfahrung in der Entwicklung von Delphi-Anwendungen bei uns in der Abteilung. So kann das Programm von jedem Kollegen der Abteilung gewartet und erweitert werden. Außerdem existiert bereits eine große Menge fertiger Code für diverse Anwendungsfälle, der in diesem Programm wiederverwendet werden kann. All diese Aspekte zusammen haben für mich den Ausschlag gegeben, dass die Entwicklung einer Delphi-Anwendung für die Anforderungen die beste Lösung ist.

4.2 Ausblick

Eine erste Version soll an einem ausgewählten ATESTEO-Standort erprobt werden. Als Standort wäre Kassel am besten geeignet, da die Kollegen dort auch zu Ihren Anforderungen befragt wurden. Diese Version soll einen vollständigen Monat an diesem Standort erprobt werden, um einen vollständigen Zyklus inklusive Auswertung zu durchlaufen. Währenddessen soll Feedback der Anwender gesammelt werden. Gleichzeitig sollen die Kollegen die Änderungen am Arbeitsablauf vor Ort erproben. Nach der Einarbeitung des Feedbacks kann mit dem Ausrollen des Programms an alle Standorte begonnen werden.

Im weiteren Verlauf des Projekts wäre es auch denkbar, dass man die Möglichkeit schafft, Werte von ausgewählten Smartmetern automatisiert abzufragen. Das würde den Arbeitsaufwand der Kollegen verringern und gleichzeitig das Fehlerrisiko senken. Eine Auswahl von verschiedenen Schnittstellenprogrammen haben wir bereits. Falls nötig, könnten auch noch weitere implementiert werden. Hierfür ist vorher mit den entsprechenden ATESTEO-Abteilungen zu

klären, welche Zählermodelle hierfür in Frage kämen und wie ein Wechsel der Zähler an ausgewählten Standorten umgesetzt werden kann. Es bietet sich an, einen einheitlichen Stromzähler zu beschaffen, damit das Programm nicht für eine Vielzahl von Modellen angepasst werden muss. Darüber hinaus wäre eine Funktion hilfreich, die automatisch Berichte zu den Zählerständen per Mail verschicken kann, die bereits Übersichten für die einzelnen Prüfstände und den gesamten Firmenstandort enthalten. Hier könnte man dann auch die Identifikation des Verbrauchs nach ISO 50001 integrieren. Auch eine automatische Zuordnung des Stromverbrauchs in das Rechnungssystem wäre denkbar, um den Abrechnungsprozess zu vereinfachen und zu automatisieren. Zukünftig könnte die Erfassung dann auf weitere Messgrößen erweitert werden, wie Wasser, Gas oder Treibstoff. Bei Gas und Wasser dient dies vor allem dem firmeninternen Energiemanagement. Der Treibstoffverbrauch muss bei Prüfläufen mit Verbrennungsmotoren erfasst werden, damit er im Anschluss an das Projekt in Rechnung gestellt werden kann.

Literatur

[1] OpenJS Foundation. *Node-Red*. 29. Okt. 2024. URL: https://nodered.org/.

[2] JAM Software GmbH. *Virtual Treeview*. 29. Okt. 2024. URL: https://www.jam-software.com/virtual-treeview.

[3] pi3g GmbH & Co. KG. *buyzero.de*. Abgerufen am 29.10.2024. 29. Okt. 2024. URL: https://buyzero.de/en/products/raspberry-pi-3b-plus.

[4] The PHP Group. *PHP*. 29. Okt. 2024. URL: https://www.php.net/.

[5] Embarcadero Inc. *Delphi*. 29. Okt. 2024. URL: https://www.embarcadero.com/de/products/delphi.

[6] Embarcadero Inc. *FireDAC*. 29. Okt. 2024. URL: https://www.embarcadero.com/de/products/rad-studio/firedac.

[7] Alexander Kabza. *Meine SmartHome-Projekt SmartMeter*. Abgerufen am 29.10.2024. 29. Okt. 2024. URL: https://www.kabza.de/MyHome/SmartMeter/SmartMeter.php.

[8] Raspberry Pi Ltd. *Rasberry Pi*. 29. Okt. 2024. URL: https://www.raspberrypi.com/.

[9] selfhtml. *JavaScript-Dokumentation*. 29. Okt. 2024. URL: https://wiki.selfhtml.org/wiki/JavaScript.