

FALLSTRICKE VERMEIDEN

Fünf häufige Fehler im HMI-Design

Ein robustes HMI für den Industrieinsatz scheitert oft an Grundlagen, wenn Zeitdruck herrscht: unwartbarer Spaghetti-Code, fehlende Abfangmechanismen für Sensorfehler oder schlechte Bedienlogik. Eine Analyse der fünf häufigsten Fallstricke.

Embedded-HMI: HMIs, die unter Zeitdruck entwickelt wurden, sind oft schwer bedienbar. Unstrukturierter Code macht die Wartung zum Albtraum und fehlendes Error-Handling legt bei Sensorausfällen die gesamte Anlage lahm.

Ein Human-Machine-Interface (HMI) bildet die kritische Schnittstelle zwischen dem Bediener und der Anlage. Es ermöglicht dem Personal, Prozesse zu überwachen, zu steuern und Störungen frühzeitig zu erkennen. Ein funktionales HMI steigert die Produktivität, erhöht die Anlagenverfügbarkeit und reduziert durch intuitive Führung Bedienfehler und Sicherheitsrisiken. In der Praxis wiederholen sich jedoch oft dieselben grundlegenden Fehler. Das ist insbesondere bei Projekten unter hohem Zeitdruck oder fehlenden Entwicklungsstandards der Fall. Die Folgen reichen von langwierigen Nachent-

wicklungen bis hin zu Produktionsausfällen durch fehlinterpretierte Anzeigen. Dieser Beitrag beleuchtet fünf häufige Fehler im HMI-Design und gibt Empfehlungen, wie sich diese vermeiden lassen.

1. Unvollständige Planung und fehlende Anforderungsdefinition

Ein Hauptfehler entsteht oft vor der ersten Codezeile: der Mangel an strukturierter Planung. Häufig beginnt die Visualisierung, ohne dass Anforderungen, Benutzerrollen



```

/** Pumpe *****
#VRI ID{R_Pos_X_temp}, (objX(Pos_H002_
#MFS ((float(R_H002_RV_101_rWert*10))=
#PPP Pos_H002_RV_101,"Dreiwegventil_1
#MFS 1,1
Pos_H002_RV_101,"Dreiwegventil_1
iv ?
RV_101_bSMSA==1),2
RV_101
ekt mit normaler Da
Dreiwegventil_1
Objekt blinl
*****
_X_temp}, (objX(Pos_H002_
(float(R_H002_PW_101_rWert*10))=
#PPP Pos_H002_PW_101,"Pumpe_Aktiv_60x
#MFS 1,1
#PPP Pos_H002_PW_101,"Pumpe_Normal_60x
/** bAktiv ?
#MFS (R_H002_PW_101_bSMSA==1),2
#OVI 1,Pos_H002_PW_101

```



Bild 1: Ein erfolgreiches HMI entsteht durch ansprechende und aus der Anwendersicht durchdachte Visualisierung, ausgewogene Kombination aus Planung, Softwarequalität und Benutzerfreundlichkeit.

oder der Informationsfluss hinreichend definiert sind. Die Konsequenzen zeigen sich im Projektverlauf: Funktionen müssen mehrfach angepasst werden, die Screens geraten inkonsistent und neue Anforderungen lassen sich nur schwer integrieren. Die Abstimmung zwischen Automatisierungstechnik, Softwareentwicklung und Anlagenbetreibern wird erschwert. Eine sorgfältige Planung muss folgende Punkte umfassen:

- Definition der Benutzergruppen: Welche Informationen und Einstellmöglichkeiten benötigt der jeweilige Bediener (Operator, Instandhalter, Administrator)?
- Strukturierung der Hierarchie: Thematische Gliederung der Navigation und Aufteilung der Bildschirmseiten.
- Festlegung eines Designstandards: Vorgaben für eine einheitliche, auf das Wesentliche reduzierte Darstellung.
- Wireframes/Mock-ups: Frühzeitige Tests der Bedienlogik außerhalb der Entwicklungsumgebung.
- Lastenheft: Dokumentation der Anforderungen und Funktionen.

Je früher diese Grundlagen stehen, desto effizienter verläuft die spätere Implementierung.

2. Fehlende Software-Architektur und unwartbarer Spaghetti-Code

HMI-Projekte wachsen oft über Jahre. Ohne klare Programmierstandards entsteht unwartbarer Spaghetti-Code. Funktionen werden redundant implementiert, globale Variablen unkontrolliert eingesetzt und Bildobjekte ohne Struktur erstellt. Das führt zu gravierenden Problemen bei Änderungen, da unerwartete Seiteneffekte auftreten. Fehlersuche und Einarbeitung neuer Entwickler werden extrem zeitaufwändig; oft endet dies in einer kompletten Neuprogrammierung. Ein professionelles HMI-Projekt erfordert eine modulare Architektur. Wiederkehrende Funktionen gehören in zentrale Bibliotheken. Klare Ordnerstrukturen und Coding-Guidelines sichern die langfristige Wartbarkeit. Je komplexer die Anlage, desto ent-

Reinhardt System- und Messelectronic GmbH

00406179-005

94.0 mm x 134.0 mm (Format: 14 2)

Displays, die bleiben... Partnerschaften, die sich vertiefen!

Von der Idee bis zur Serie – über 40 Jahre Erfahrung



KI generiert

Automatisierung, Messtechnik und moderne Bedienkonzepte stellen hohe Anforderungen an Displays und Tastaturen.

Display Elektronik begleitet anspruchsvolle Projekte, von der Auswahl über die Entwicklung bis zur Serienversorgung – **persönlich, kompetent und partnerschaftlich** ...

Display Elektronik GmbH
www.display-elektronik.de
T. +49 (0) 6043 / 98888-0



LCD · TFT · LED · OLED · TOUCH PANELS · TASTATUREN

Variablen	
Ungünstig	Besser
Temp1	MotorTemperature
Start	ConveyorStartCommand
M100	EmergencyStopActive
X1	TankLevelPercent

Tabelle:
Unterschied zwischen schlecht wartbaren Variablennamen und strukturierten, selbsterklärenden Namen.

scheidender ist eine saubere Softwarestruktur für den wirtschaftlichen Lebenszyklus des Projekts.

3. Kryptische, inkonsistente oder nichtssagende Variablennamen

Variablennamen wie DB1_VAR12, Temp1 oder WertNeu sind während der initialen Entwicklung vielleicht noch verständlich, führen später aber zu Problemen. Kryptische Bezeichnungen erschweren die Wartung und vor allem die Fehlersuche, also das Debugging. Damit steigt auch die Gefahr, unbeabsichtigt Fehler einzubauen. Die Zusammenarbeit im Team oder die spätere Übergabe des Projekts wird ohne klare Namenskonventionen massiv behindert. Aussagekräftige Namen dokumentieren den Code „on the fly“ und reduzieren den Bedarf an Kommentaren.

Es wird empfohlen, verbindliche Konventionen einzuführen. Dazu zählen beispielsweise Präfixe für Datentypen oder Funktionsgruppen. Im *HMIdesigner* von Display Visions werden Register beispielsweise standardmäßig mit dem Präfix „R_“ vorbelegt, Buttons mit „bt_“ oder Ähnlichem. Was anfangs pedantisch wirken mag, zahlt sich bei langfristigen Projekten aus.

4. Mangelhafte Fehlerbehandlung und fehlende Robustheit im Feld



VERFASST VON
Ralph Tischer
Entwicklungsleiter
Display Visions

oder Systemblockaden durch endlose Retry-Versuche ohne Abbruchkriterien führen.

Ein robustes HMI muss Kommunikationsfehler eindeutig visualisieren, wie zum Beispiel Messwert „---“ statt „0“. Fehlermeldungen müssen verständlich sein und die Quelle im Klartext benennen. Guter Code führt Plausibilitätsprüfungen durch und fängt kritische Zustände ab. Eine saubere Fehlerbehandlung reduziert Stillstandszeiten und ermöglicht in vielen Fällen einen Notbetrieb bei Teilausfällen. Hardwarenahe Diagnosemeldungen unterstützen Instandhalter bei der schnellen Ursachenanalyse.

5. Unübersichtliche Menüstrukturen und mangelnde Usability

Selbst technisch einwandfreie HMIs scheitern, wenn Darstellung und Bedienung unübersichtlich sind. Typische Usability-Probleme sind tief verschachtelte Menüs, inkonsistente Navigationselemente (Wechsel zwischen Buttons, Icons und Textlinks) oder überladene Bildschirme mit zu vielen Farbcodes. Das lenkt ab und provoziert Fehlbedienungen. Wichtige Informationen müssen priorisiert und permanent sichtbar sein. Wenn Bediener in kritischen Situationen nach Funktionen suchen müssen, kostet dies wertvolle Zeit.

Ein modernes HMI orientiert sich an Usability-Prinzipien und Ansätzen wie High-Performance HMI. Dazu gehören:

- Flache Menüstrukturen (max. zwei Ebenen ab Hauptmenü).
- Konsistente Navigation und klare Hierarchien.
- Reduzierte Darstellung: Relevante Informationen werden größer und an fester Position gezeigt.
- Wenige, standardisierte Farben und Symbole (Farbe primär zur Signalisierung von Anomalien).
- Eindeutige Rückmeldungen auf Benutzeraktionen.

Ein erfolgreiches HMI entsteht aus der Kombination von vorausschauender Planung, hoher Softwarequalität (Struktur, Benennung, Error Handling) und konsequenter Benutzerorientierung (Usability). Die Vernachlässigung dieser Punkte führt zu hohen Folgekosten in Wartung und Service bei gleichzeitig schlechterer Bedienbarkeit. Unternehmen, die von Beginn an auf Standards und modulares Design setzen, profitieren von höherer Verfügbarkeit und Akzeptanz. Display Visions unterstützt hierbei mit entsprechenden Softwareservices bei Strukturierung und Umsetzung. (heh)

EP-INFO

Kurzcheck: HMI-Qualität in fünf Fragen

- Der 30-Sekunden-Test: Kann ein neuer Bediener die drei wichtigsten Hauptfunktionen innerhalb von 30 Sekunden finden, ohne eine Einweisung zu erhalten?
 - Der Stecker-Zieh-Test: Was passiert auf dem Display, wenn Sie das Sensorkabel oder die Busverbindung im laufenden Betrieb trennen? Wird der Fehler klar und verständlich angezeigt oder frieren die Werte ein?
 - Der Remote-Eye-Test: Können Sie den Zustand der Anlage auf einen Blick erfassen, wenn Sie sich drei Meter vom Display entfernt befinden?
- Der Wartungs-Test: Sind wiederkehrende Funktionen, wie beispielsweise eine Zehnertastatur zur Eingabe, als zentrales Bibliothekselement implementiert? Oder wurde der Code dafür zehnmal kopiert?
 - Der Nachfolger-Test: Kann ein fremder Programmierer allein anhand der Namen die Bedeutung der Kern-Variablen erkennen, ohne den Code zu debuggen?

Wenn Sie diese fünf Fragen alle mit *Ja* beantworten können, verfügt Ihr entwickeltes HMI bereits über ein solides Qualitätsniveau.