Testsequenzer als Webapplikation

Studiengang:

Bei Bruker werden alle Elektronikbaugruppen Funktions- und Performanceprüfungen unterzogen, bevor sie verbaut werden. Dafür werden intern HIL-Testsysteme gebaut und die Tests mittels C#-Desktop-Prüfsoftware, basierend auf einem eigenen Testsequenzer, automatisiert. Die Verteilung einer Prüfsoftware ist mit dem heutigen Ansatz aufwändig und langsam. Ein neuer Testsequenzer, als Webapplikation mit Plug-In-Mechanismus, soll dies vereinfachen und beschleunigen.

Ausgangslage

Aktuell werden etwa 35 Testsysteme mit ungefähr 75 verschiedenen Prüfsoftwaren für über 4500 Prüflingsvarianten betrieben.
Der bestehende Testsequenzer wird als Bibliothek angeboten und eine spezifische Version davon in eine Prüfsoftware eingebunden. Zusammen werden sie zu einer MSI-Installationsdatei verpackt und manuell oder automatisiert, mit Hilfe des „SCCM"-Tools von Microsoft, auf die HIL-Testsysteme verteilt. Während manuelle Installationen allgemein fehleranfällig sind (ein Testsystem kann vergessen gehen), ist für die automatische Verteilung eine zusätzliche Konfiguration seitens der IT-Abteilung notwendig und das Ausrollen dauert bestenfalls Stunden (die „SCCM"-Skripte zur Detektion von neuen Software-Releases laufen in der Nacht). Durch die direkte Einbindung des Testsequenzers in der Prüfsoftware muss diese zudem für eine Aktualisierung des Testsequenzers jedes Mal angefasst werden.

Zielsetzung

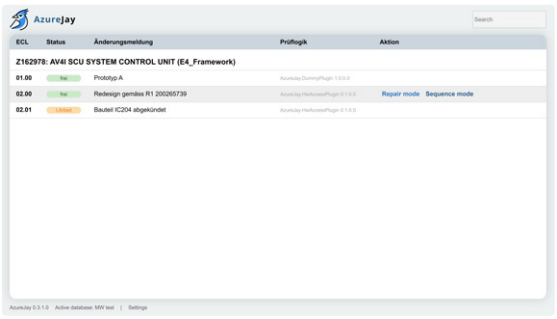
Um die oben genannten Probleme anzugehen, sollte im Rahmen dieser Arbeit eine Machbarkeitsstudie durchgeführt werden, den Testsequenzer als Webapplikation zu implementieren, der die Testlogik mittels Plug-In-Mechanismus bei Bedarf nachlädt. Um die Entwicklung einer Prüflogik weiterhin in .NET/C# zu ermöglichen, sollte der neue Testsequenzer mit dem ASP.NET Core Blazor Frontend-Framework umgesetzt werden. Zudem musste sich die Lösung nahtlos in die bestehende Umgebung mit den Prüflings- und Resultate-Datenbanken einfügen.

Ergebnis

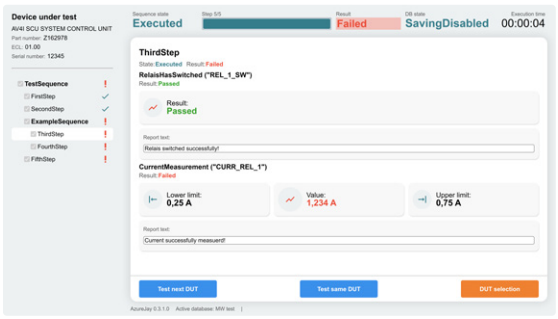
Mit dem Testsequenzer-Framework „AzureJay" konnte ein PoC umgesetzt werden, das den Stakeholdern mittels eines Akzeptanztests vorgeführt und erste Rückmeldungen dazu eingeholt werden konnte. Die Prüflogik für einen Prüfling kann anhand des Plug-In-Schnittstellenkontrakts separat entwickelt und innert Minuten deployt werden. Die richtige DLL wird aufgrund der Prüflingsauswahl des Benutzers im Browser nachgeladen und ausgeführt. Auch ein Framework-Update kann dank Container-Technologie innert Minuten für alle Prüfsysteme ausgerollt werden.

Eine Herausforderung stellte die Ansteuerung der Hardware in den HIL-Testsystemen aus dem Browser dar. Dafür wurden Adapter für die Ansteuerung eines Netzteils mittels der „WebUSB API" und für die serielle Kommunikation mit einem Prüfling mittels der „Web Serial API" implementiert. Zudem musste gewährleistet werden, dass diese Hardware-Services konfigurierbar sind und den Prüflogik-Plug-Ins mittels Dependency Injection zur Verfügung gestellt werden können.

Marco Stadler
MAS Information Technology
marco.stadler@bruker.com



Prüflings- und Testmodusauswahl für Prüflogik aus unterschiedlichen Plug-Ins



Ausgeführte Testsequenz mit Messdaten und Resultaten