Pogo-Pin Test System Development Plan

# Introduction

This project will create an open-source, reusable software platform for bed-of-nails (pogo-pin) test fixtures. This will allow Sensit to (1) quickly develop bed-of-nails test fixtures without requiring software development, (2) increase productivity by automating manual labor for testing (3) log results of production tests in a database. This will ultimately decrease time-to-market for new products and provide data for production enhancements.

Your mission, should you choose to accept it, will be to enhance the *Sensit Test Suite*, an automation application and SDK, adding features to enable control of bed-of-nails (pogo pin) test fixtures. Source code is here:

<https://github.com/SensitTechnologies/TestSuite/>

User Stories

The following user stories define the application we wish to develop:

* As an operator…
  + I want to start, stop, or pause a bed-of-nails functional test, so I can work quickly.
  + If a test passes, I want to see a simple indication of that, so I can minimize confusion.
  + If a test fails, I want to see a simple indication that the test failed, along with a picture and instructions about what should be checked, so I can inform a repair technician.
  + I want all test fixtures to use the same application, so I can understand what to do.
* As a technician…
  + I want all test fixtures to use the same application, so I can easily set up a new test PC.
  + I want to be able to configure a test, so I can make test fixtures without writing code.
  + I want to save test fixture settings in a file, so I don’t have to repeat any setup work.
* As an engineer…
  + I want to record test results in a database, so I can calculate first pass yield.
  + I want to record test results in a CSV file, so I can collect data if a database fails.
  + I want to be able to add support for new types of devices by adding a new class to an SDK and rebuilding the application, so I can quickly add support for new test equipment.

# Configuration Files

Configuration file defines a series of actions that comprise a test. Each action includes:

* a command string (which will be sent via a serial port to a test device),
* An expected result (can be a string received in response to a command, or an acceptable range for a numeric value),
* A picture displayed to the user if the action fails,
* A string displayed to the user if the action fails.
* User must be able to set serial port and device type (pick from devices in the SDK).

After the initial application is delivered, support for additional hardware such as multimeter, temperature chamber, etc. can be added to the SDK as needed, and will be automatically supported in the application.

# Development Plan

The application will be developed in C# using Visual Studio. Sensit is familiar with these technologies and can easily support them. If needed, a future project could upgrade these applications to run on a variety of platforms, including Android, Linux, or as web applications.

The application is be organized into three projects:

* A Software Development Kit (SDK) which contains utilities for each of the deliverables and can be reused in future applications.
* A WinForms application which utilizes the SDK to present the deliverables to the user.
* A collection of unit-tests for the SDK to aid development and maintenance.

To facilitate code reuse, the SDK, its unit-tests, and the application will be stored in git repositories.

The deliverables do not require intellectual protection and will be released under an open-source MIT license. All proprietary information will be contained in application configuration files.

This test could be accomplished using the following Arduino-compatible relay board and DC/DC converter:

* [KTA-223 USB/RS485 Relay IO Board](https://www.sparkfun.com/products/9526) ($144.95, via SparkFun)
* [SparkFun Buck-Boost Converter](https://www.sparkfun.com/products/15208) ($9.95)

The KTA-223 has eight relay outputs, three analog inputs, and 4 digital inputs, as well as a small 5V power supply, so it can control LEDs, control power to a device, and measure voltages. It can also be programmed using the Arduino IDE or controlled remotely via serial port. The Buck-Boost Converter can be configured to output any voltage between 2.5 – 9V, so can be used to simulate a lithium battery cell’s 3.6V output.

# Revision History

|  |  |  |
| --- | --- | --- |
| **Date** | **By** | **Change Description** |
| 4/5/2022 | Adam Johnson | Initial revision. |