INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY
CAMPUS MONTERREY

ESCUELA DE INGENIERÍA Y CIENCIAS

INGENIERÍA EN ELECTRÓNICA (IE PLAN 2019)

Tec21 - TE3004B.501 - Desarrollo de Telecomunicaciones y Sistemas Energéticos (Gpo 501)

FINAL REPORT: PLUGIN DEVELOPMENT FOR BT2202A

Ricardo Islas Guerra	A01568531
Germán Alvarado De los Santos	A01721018
Cheyenne Rigel De Jesús González	A01285156

Index:

Introduction:

The following report is being written with the motive of justifying and going over the process of the development of a plugin for Keysight's Test Automation Platform (TAP), built on OpenTap. The plugin will allow the user to develop test sequences for Keysight's BT2202A Charge-Discharge, Li-ion Cell Formation, and Test Solution.

The plugin will enable engineers throughout Latin America and the world to leverage the BT2202A and expand their testing capabilities on energy storage cells.

Objectives:

The project is set to achieve the following objectives:

- Initial method that allows TAP to detect the BT2202A instrument.
- Enable drag and drop like buttons for various test operations, such as: charge, discharge, etc…
- Enable inputs to define the cells on the device to use for testing.
- Enable users to run parallel test sequences on different cells.

Energy Storage:

With the surge of industrialization and modernization across the globe, our demand for energy rises alongside it. From our daily commutes and transporting goods, to powering the latest tech, we rely on electricity as the enabler of modern life.

Fossil fuels release harmful gasses, polluting the environment and accelerating climate change. With rising fuel costs and environmental urgency, both governments and industries are pushing towards electrification and clean energy sources.

However, renewable sources like wind and solar have their caveats: energy generation is inconsistent and doesn't always match up with the times we need it most. Battery Energy Storage Systems (BESS) provide the answer. By storing excess energy when production is high, BESS allows us to use it when demand peaks, making the grid more efficient and bringing us closer to an electric-powered way of life.

The ability to store energy brings significant benefits for both producers and consumers, cutting operational costs, eliminating fuel consumption, and slashing carbon emissions.

Advances and innovation in energy storage systems are creating a path toward mass electromobility, offering a cleaner and smarter way to move around. Longer autonomy, faster charger times, lower prices, higher safety, and reliability in batteries are making electromobility and real electrification reality. Yet, the journey has its own challenges.

The heart of these energy storage systems—the Li-Ion cells—demands a meticulous manufacturing process. Each cell module undergoes rigorous testing to ensure it meets the highest standards for performance and safety, during both production and operation. The pressures on cell manufacturers are intense: reducing unit costs, dealing with material constraints, ensuring return on investment, and upholding safety at scale.

As demand rises, manufacturers must expand capacity swiftly. This starts with prototype or pilot production lines, where new cell designs are first tested and refined. Unlike massive production lines tailored for high volumes, these smaller lines need versatile, adaptable equipment to handle frequent reconfigurations.

BT2202A Device:

Keysight's BT2200 Charge-Discharge, Li-ion Cell Formation, and Test Solution is a cost-effective and easily configurable for Li-Ion cell formation and lifetime cell cycling. A modular design supports cells requiring maximum currents ranging from ± 6 A to ± 800 A, with up to 256 cells or channels per chassis. It can quickly deploy different channel configurations as cell requirements change and capacities grow [1].

The team will work with the Keysight BT2202A, which is an 8-slot mainframe for 400 VAC 3-phase power with support for up to 256 user channels per chassis. It is an attractive platform due to cost and modular design, which brings the possibility of scalability in the future.

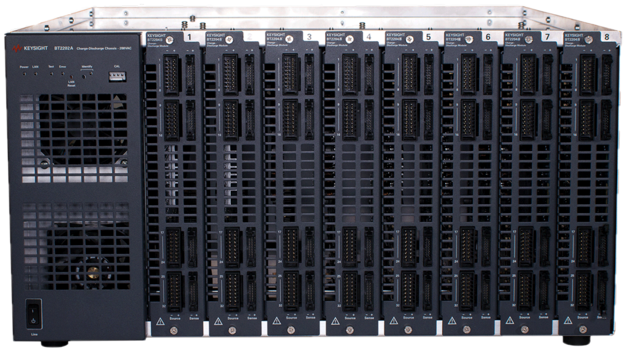

Fig. 1: BT2202A

Test Automation Platform (TAP):

Keysight's Test Automation Platform (TAP) is a powerful software framework designed to streamline and enhance the testing and automation process. Its architecture supports integration with a wide variety of SCPI instruments and offers features such as Pass/Fail logs, detailed reporting, and easy customization of test procedures.

However, one notable limitation of TAP is its lack of native support for the BT2202A. While TAP can interface with various SCPI-compatible instruments, the BT2202A's unique capabilities and requirements demand the creation of a specialized plugin to fully leverage its potential.

Background:

During the first half of 2024, a group of Electronics Engineering students at Tec de Monterrey started the development of a BT2202A plug-in for TAP. The project had the objective to enable the user to create test sequences using a custom GUI and drag-and-drop like blocks, making an easy to use and user-friendly interface.

This plug-in leverages on CSV to set and send all types of parameters, commands, and queries to the instrument. This presents a couple of limitations, the use of csv files make a difficult setup for new users and a hard learning curve, as the CSV and code need to be modified if different test sequences need to be implemented.

However, this early version of the plugin gave the team a head start and helped them understand the OpenTAP development process.

Development:

Github: https://github.com/ricardo1slas/BT2202A_Plugin/tree/main

Basic Plugin:

To develop a basic plugin it is necessary to have the following files.

- **<Instrument>.csproj File:**
  This file establishes the framework for OpenTAP and facilitates the plugin development process. It defines the project structure and dependencies.

- **<Instrument>.cs File:**
  This file declares the new instrument and enables communication with it. Within this file, the VISA address and the instrument's name are specified, ensuring proper identification and connection.

- **package.xml File:**
  A configuration file that defines the plugin's metadata and versioning, making it compatible with OpenTAP.

- **<step>.cs Files:**
  These files define the test steps for the plugin. Each test operation, such as charge, discharge, or measure, is represented by a separate step file. Test steps can be configured as Parent (main) steps or Child (sub) steps, allowing for modular and hierarchical test sequence design. These steps handle the execution of SCPI commands and define how TAP interacts with the instrument during testing.

The basic plugin the team developed used these principles to develop a basic plugin that can perform charge, discharge, test, and measure steps with a Pass/Fail log as shown in figure 2. This plugin is available on the github link at the start of the Development section.

Name	Last commit message	Last commit date
..		
.vs	corrected stuff	2 weeks ago
.vscode	new	last month
Properties	managed to measure, tried to charge ;-;	last month
bin	corrected stuff	2 weeks ago
obj	corrected stuff	2 weeks ago
BT2202.cs	managed to measure, tried to charge ;-;	last month
BT2202A.csproj	managed to measure, tried to charge ;-;	last month
DurationTestStep.cs	yo	3 weeks ago
bt2202a.sln	managed to measure, tried to charge ;-;	last month
charge.cs	corrected stuff	2 weeks ago
discharge.cs	corrected stuff	2 weeks ago
package.xml	added discharge	last month

Fig 2. Github Repo.

Theory of Operation:

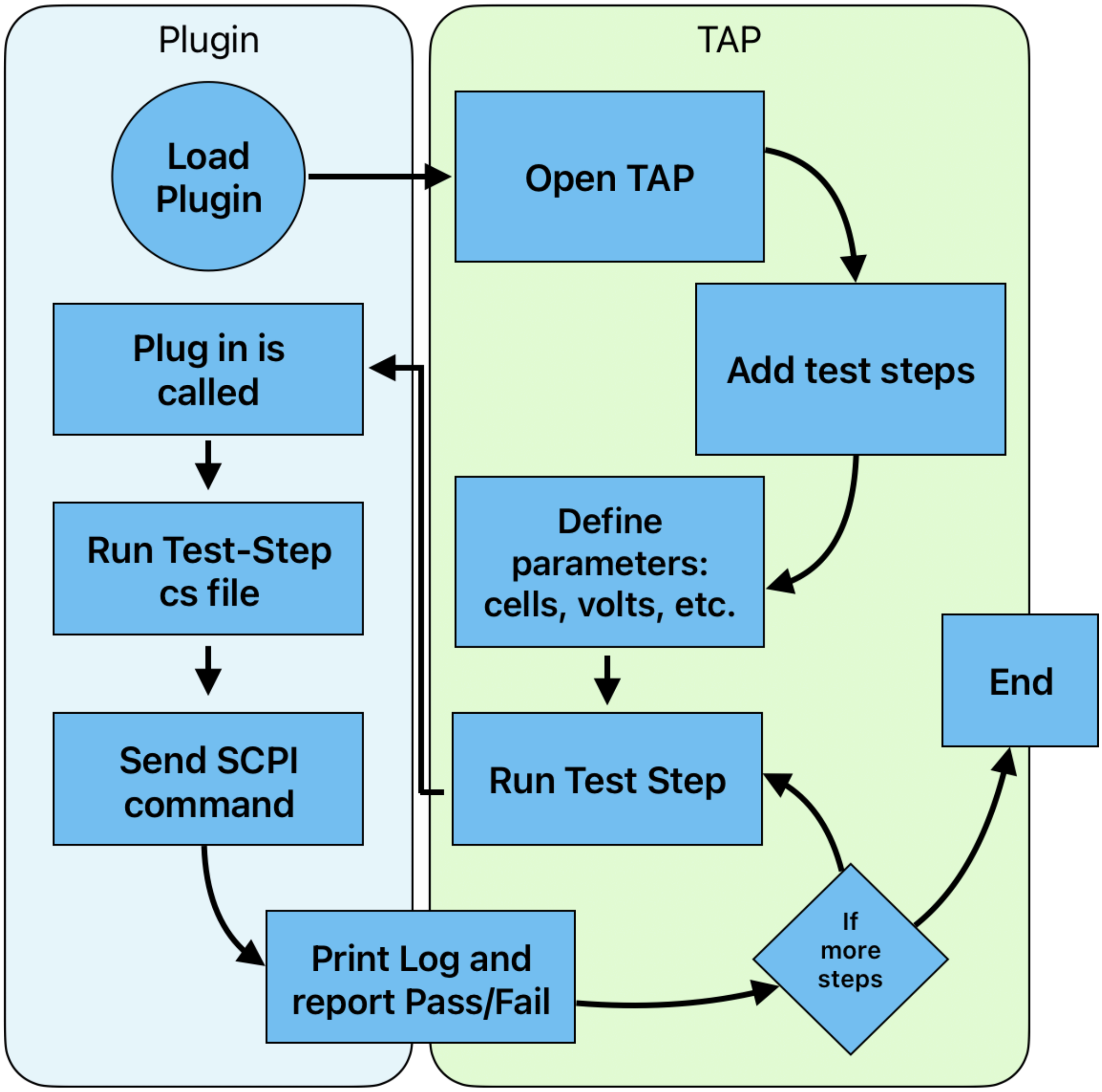

Fig 3.Operation Flow Chart of Basic Plugin.

Running Tests in Parallel:

Running sequences in parallel on the BT2202A presents a challenge due to a design limitation of the instrument: it cannot process more than one SCPI command at a time. Attempting to send multiple commands simultaneously results in errors. This limitation is particularly problematic given that the BT2202A supports up to 256 channels but restricts them to executing only one type of test at any given time.

This is an issue when running a Measure loop, because it is constantly running SCPI queries to the machine. To address this issue and enable the execution of different sequences while measuring channels simultaneously, the team is running two instances of TAP. One instance through the terminal, using the tap command to run a TapPlan that only runs a measure, and the other one running on the TAP software with all the specific commands.

By running a measure loop on all channels through the terminal, TAP can efficiently manage the timing of SCPI commands. This approach ensures that commands are dispatched to different channels sequentially and rapidly enough to simulate parallel operations. As a result, the plugin can effectively bypass the instrument's inherent limitations, enabling true parallel testing and monitoring across multiple channels.

This method not only maximizes the BT2202A's channel capabilities but also enhances the efficiency of testing workflows, making it a critical improvement for scalable energy storage testing, however to enable this method, modifications had to be made on the plugin script. Native measuring features were disabled on each of the test steps, flags were added to signal each of the steps running whether they could send messages to the device, as well as some other minor changes.

Theory of Operation:

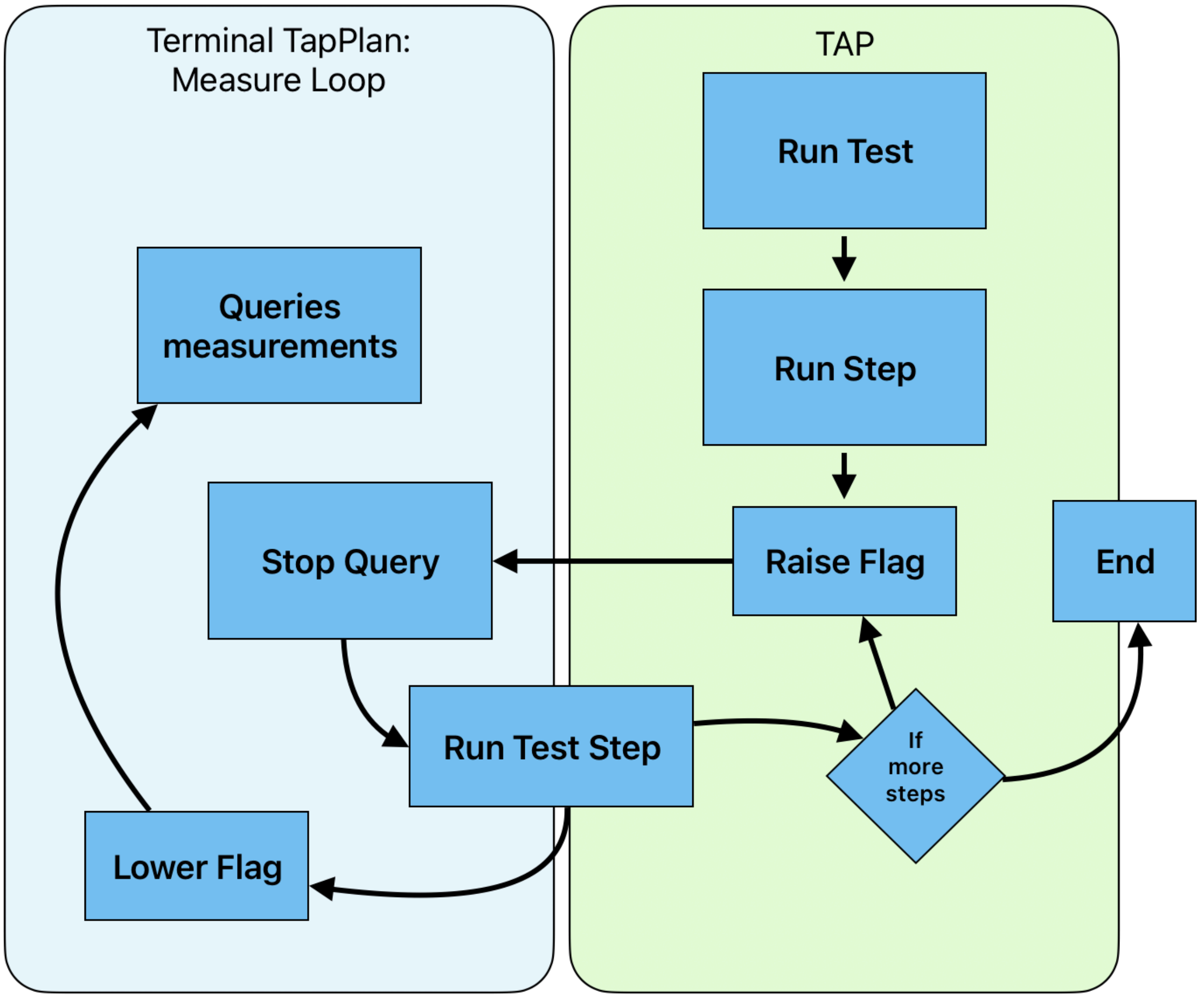

Fig 4.Operation Flow Chart of Parallel Testing.

Results:

The development process concluded on the development of two TAP plugins to enable the user to develop test sequences for the BT2202A instrument. The first one is a basic plugin with a couple of test steps, simple enough, open source, and modular in nature to allow future development. The basic plugin completes most of the usual tasks with ease with the caveat of not being able to run different tests on different cells at the same time. Below the figure shows the two cells voltage changing when being programmed different steps:

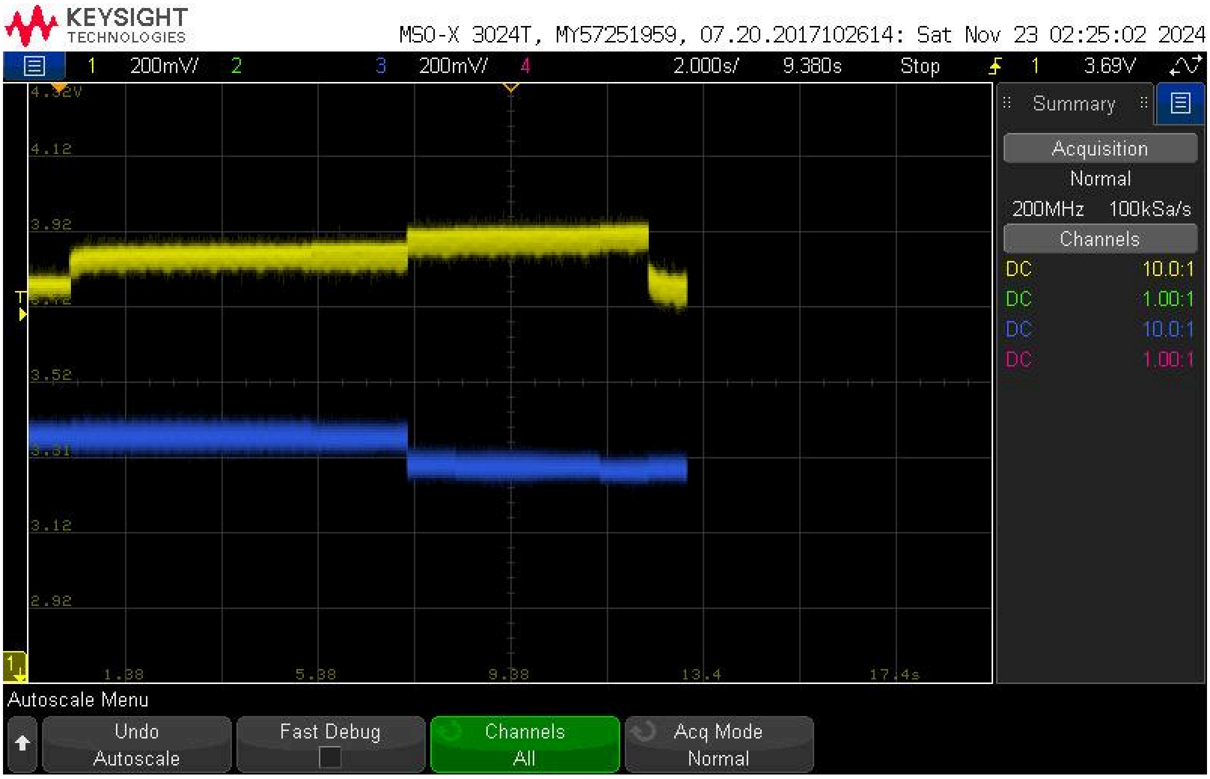

Fig 5. Cell running charge (yellow) and other cell running discharge (blue).

The second plugin was developed on top of the first one, adding a couple of modifications for test control; such as raising flags, removing loops, etc. The nature of the instrument and TAP presents design limitations that can be worked around utilizing the second plug in along with a separate program running in parallel through the terminal. In the figure below, how both TAP instances are running in parallel, but also how they are communicating with each other through a JSON file. When one step is programmed in the main terminal, it calls upon a stop action to the measure loop.

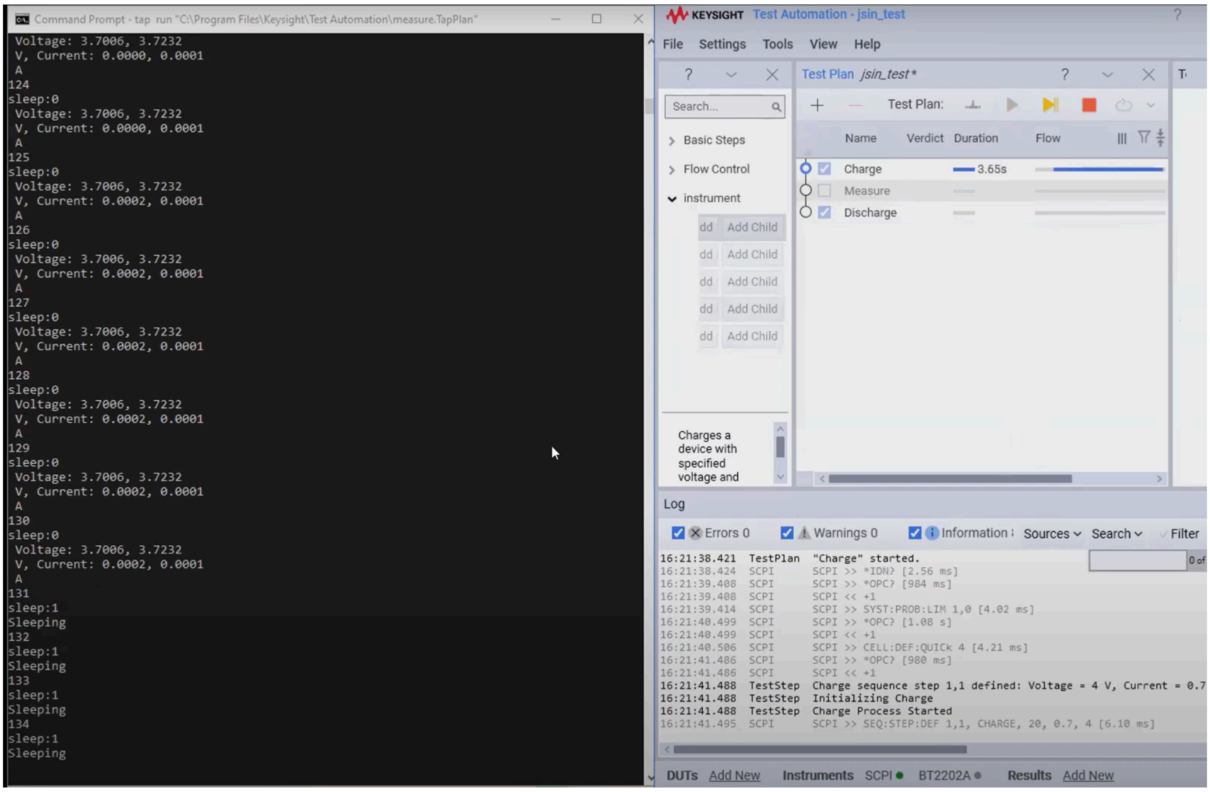

Fig 6. Parallel TAP plans running simultaneously.

Future Steps and Constraints:

As of right now, there is no way of telling the code which cells are active or inactive. Measuring all cells is not feasible due to the device not being able to measure disabled cells, doing so will cause an error. There needs to be a way to reference and keep track of active and inactive cells.

This constraint goes further. A problem the team was facing was that whenever running tests in parallel, the report of the Pass/Fail was happening on the measure loop and not on TAP. Even if a test failed or passed, the report of the value was not directly linked to the test sequence running. This is a problem because, for example: a test sequence is to be aborted if a step is failed, this constraint would not allow TAP to know when to abort the procedure.

Implementation of an extra configuration file such as a JSON would be greatly beneficial for future plugin development. Having a configuration file set up at the beginning of each sequence would allow the steps to perform a consultation before executing to adjust their parameters, as for determining Pass/Fail verdicts. It will also streamline the control of the operations, having a centralized file with all the flags.

Conclusions:

Finally, we can conclude that the team was successful in building a solid base of what in the future can be a full-fledged TAP plugin for the BT2202A; pioneering on the development of this device, researching the device limitations, and how to work around them.

The comments in the code, alongside this report will streamline the development of the plugin and future plugins for seasoned developers, as well as beginners.

Although not ideal, the team managed to overcome the instrument's hardware limitations through control techniques embedded in the plugin development. This has opened a way for Keysight’s team or other future TAP developers to continue the development of the plugin.