PROJECT REPORT ON

AUTOMATED RF DEVICE TESTING SOFTWARE

Submitted by ***Ganesh Maharaj Kamatham (22BDS0168)***

Under the Supervision of

Mr. SANJAY K TOMAR

&

Mr. SUDHIR KUMAR

DRDO, SOLID STATE PHYSICS LAB

For the award of the Industrial Internship credits

Of

B.TECH Computer Science Engineering, specialization in DATA SCIENCE

SCHOOL OF COMPUTER SCIENCE

VELLORE INSTITUTE OF TECHNOLOGY

VELLORE – 632014

(MAY – JULY 2025)

CERTIFICATE

This is to certify that Ganesh Maharaj Kamatham of B.Tech 3rd Year, have Presented a project titled **"AUTOMATED RF DEVICE TESTING SOFTWARE"** in fulfilment for the award of the Industrial Internship Credits under **DRDO, SSPL** DELHI.

DATE : 14/07/2025

Mr. SANJAY K TOMAR

Mr. SUDHIR KUMAR

MENTOR

CANDIDATE'S DECLARATION

I hereby declare that the work, which is being presented in the project, entitled **"AUTOMATED RF DEVICE TESTING SOFTWARE"** in fulfilment for the award of the Industrial Internship Credits under **DRDO, SSPL** DELHI submitted to the Department of Microwave Engineering **DRDO, SSPL** DELHI is a record of our own investigations carried under the Guidance of Mr. Sanjay K Tomar & Mr. Sudhir Kumar, Department of Microwave Engg. **DRDO, SSPL** DELHI. I have not submitted the matter presented in this report anywhere for the award of any other Degree.

GANESH MAHARAJ KAMATHAM

INTERN

ABSTRACT

The Solid State Physics Laboratory (SSPL), a key establishment under the Defence Research and Development Organisation (DRDO), is engaged in the design and validation of cutting-edge semiconductor and RF devices for strategic applications. A vital stage in this process involves wafer-level device characterization, where electrical and RF performance parameters are extracted to validate fabrication quality and functional behavior. Traditionally, these measurements—such as DC I-V characterization and RF power sweeps—were performed manually using discrete instruments, resulting in extended test durations (often several hours per device), inconsistent measurements, and significant dependency on operator skill.

During my two-month internship at SSPL, I designed and implemented two fully automated systems:

1. **I-V Characterization Automation**, and
2. **RF Power Sweep Automation**,
   tailored specifically to SSPL's in-house device testing workflow.

The **I-V Characterization** tool controls the Rohde & Schwarz NGP800 dual-channel power supply to execute configurable voltage sweeps on wafer-level devices across gate (Vg) and drain (Vd) terminals. It captures corresponding current readings, calculates derived electrical parameters, and renders real-time plots of Id-Vd and Id-Vg characteristics. A custom GUI allows parameter tuning, sweep control, emergency shutdown, and CSV export, greatly enhancing usability and repeatability.

The **RF Power Sweep** module orchestrates the Keysight EXG Analog Signal Generator and R&S NRX Power Meter to perform programmable frequency and power sweeps. It captures forward and reverse power readings, computes gain, output power, and power-added efficiency (PAE), and visualizes results in real time. Support for pulsed and CW mode, batch measurement capability, and error handling was integrated to match real-world test scenarios.

Both systems were initially prototyped on Ubuntu using Python, PyQt5, and PyVISA-Py, and were later ported to Windows with NI-VISA support to align with SSPL's lab infrastructure. Special attention was given to USB communication stability, SCPI command synchronization, and multithreaded UI responsiveness.

These automation solutions have reduced test time from hours to under 10 minutes, eliminated manual errors, enabled batch processing, and significantly improved data reliability. The developed tools are now ready for deployment in SSPL's wafer characterization pipeline, offering a robust, cross-platform, and operator-friendly testing framework aligned with defence-grade R&D workflows.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 TASK

The Solid State Physics Laboratory (SSPL), a constituent lab under the Defence Research and Development Organisation (DRDO), plays a pivotal role in the design, development, and evaluation of semiconductor and microwave devices used in advanced defense systems. A critical aspect of their R&D pipeline is the characterization of these devices at the **wafer level**, which involves highly repetitive, precise, and time-consuming measurement procedures. Traditionally, two core processes **I-V (Current vs Voltage) Characterization** and **RF Power Sweep Testing** were conducted manually. These tasks often required skilled operators to configure instruments, set voltage and frequency levels, collect data, and plot the required curves. The manual method was not only time-intensive, typically taking several hours per wafer, but also prone to operator fatigue and human error, affecting reproducibility and throughput.

During my two-month internship at SSPL, I was tasked with addressing this challenge by designing, developing, and deploying **Automated software tools** for these processes. The goal was to significantly reduce the test time, increase data accuracy, and enable repeatable measurements across devices while integrating these tools with existing lab equipment. After understanding the existing test protocols and working closely with the technical staff, I developed two separate yet interrelated automation systems: **RF Power Sweep Automation Tool** and **I-V Characterization Automation Tool**.

1.2 Project 1: I-V Characterization Automation Tool

The **I-V Characterization Tool** was developed to automate the sweeping of gate voltage (Vg) and drain voltage (Vd) using the **Rohde & Schwarz NGP800 power supply**, while simultaneously measuring the corresponding gate and drain currents. The system is capable of performing nested sweeps across a defined voltage grid, plotting the resulting output characteristics in real-time using PyQtGraph. I implemented a graphical user interface (GUI) using **PyQt5**, where users can configure voltage ranges, step sizes, delays, and output file paths.

To interact with the NGP800 over USB, I used **PyVISA**, a Python library that communicates with instruments using SCPI commands over VISA interfaces. The application was initially developed and tested on **Ubuntu**, where permission issues related to USB devices were resolved using udev rules and chmod. Later, the software was successfully ported to **Windows**, which required the installation of **NI-VISA** and adjustments to USB/COM port recognition logic.

The tool includes features like **pause/resume**, **emergency stop**, and **live plotting**, with all readings saved to .csv files for further analysis. This automation reduced I-V test time from

~2 hours per wafer to under 10 minutes and produced consistent and high-resolution graphs with zero operator intervention.

1.3 Project 2: RF Power Sweep Automation Tool

The **RF Power Sweep Tool** was designed to automate the testing of wafer devices under varying RF power and frequency conditions. The process involves sweeping the frequency and/or power levels using a **Keysight EXG Analog Signal Generator (N5173B)**, while measuring the power input and output using a **Rohde & Schwarz NRX power meter**. Based on these measurements, the system computes crucial performance parameters such as **gain**, **power-added efficiency (PAE)**, and **insertion loss**.

The software interfaces with both instruments using **SCPI commands** via USB. The challenge here was to synchronize the signal generator and power meter reads in real-time, with minimal latency. The GUI allows users to define the frequency range, power range, sweep direction, number of averaging points, and device under test (DUT) configuration. Real-time plots show power in vs. power out, gain vs. frequency, and PAE curves, helping engineers instantly validate DUT performance.

Instrument communication and plot updates are handled using multithreading, ensuring that the GUI remains responsive during long test cycles. The output data is saved in structured .csv files, with logs for timestamped metadata. The automation has drastically improved throughput and eliminated manual reconfiguration and reading steps, especially during pulse mode operation where precision and timing are critical.

1.4 Platform Transition: Ubuntu to Windows

Both applications were initially developed on **Ubuntu 22.04 LTS** due to its powerful command-line tooling and ease of use for debugging hardware interfaces. Libraries and dependencies were managed using **Anaconda** and pip, with care taken to ensure all installations were reproducible via environment files.

However, due to operational requirements at SSPL, the tools had to be made compatible with **Windows 10 systems** where instrument drivers and USB interfaces are more commonly used. This required:

- Installing **NI-VISA** for Windows
- Handling USB COM ports using PyVISA backends
- Adapting file path syntax (/ vs \\)
- Creating **.bat launchers** for non-technical users
- Handling admin permission issues (especially for COM port communication)
- Packaging the software using pyinstaller for Windows executables

All platform-specific adjustments were documented, and a detailed installation manual was created for both Linux and Windows users to ensure seamless deployment across machines.

1.5 Tools, Libraries, and Hardware

Languages & Libraries:

- Python 3.10
- PyQt5 (for GUI)
- PyVISA (instrument communication)
- PyQtGraph & Matplotlib (for real-time plotting)
- Numpy, Pandas (data handling)
- Threading and time libraries for concurrency

Hardware:

- Rohde & Schwarz NGP800 Power Supply
- Keysight EXG Analog Signal Generator (N5173B)
- Rohde & Schwarz NRX Power Meter
- Probe Station & Wafer Fixtures
- Windows 10 and Ubuntu 22.04 LTS workstations

1.6 Outcomes and Impact

The outcome of this project was highly successful, both technically and operationally. The automation tools resulted in:

- **>90% reduction in test time** for I-V and RF sweep operations
- **Elimination of manual errors** and enhanced data consistency
- **Real-time graphical feedback** to enable immediate insights
- **Seamless logging** for future analysis and documentation
- **Cross-platform compatibility** for deployment flexibility

Engineers at SSPL can now test multiple devices in a batch with minimal setup, freeing up time for analysis and research rather than routine measurements. The GUI was designed to be intuitive, enabling even non-programmers to run full test sequences with a few clicks.

The software has also been built to allow future enhancements such as **remote monitoring**, **automatic DUT identification**, and **integration with databases** for centralized result tracking.

1.7 Conclusion

This internship offered a unique opportunity to work on real-world defense applications where automation plays a crucial role in enhancing productivity and data quality. By applying software development skills in hardware-interfacing scenarios, I was able to contribute meaningfully to ongoing research and streamline existing workflows at SSPL. The successful implementation of these tools demonstrates how **software-defined testing frameworks** can revolutionize traditional laboratory operations and significantly accelerate innovation in semiconductor and RF research.

2. Project Overview

2.1 Overview of Device Testing Workflow

Manual Process Overview

In the wafer testing domain at DRDO SSPL, precise electrical and RF measurements are critical to validate the behavior and quality of semiconductor devices. Traditionally, two key types of tests were conducted manually:

- **I-V (Current vs Voltage) Characterization**:
    - Engineers manually configured voltage sources (for Gate and Drain terminals).
    - Step-wise voltage sweeps were performed by incrementing values via front-panel controls or external knobs.
    - Current values were either read from digital displays or recorded using external meters.
    - Data was logged manually on paper or in spreadsheets.
    - Plots (Id vs Vd for various Vg, etc.) were generated post-experiment using tools like Excel or MATLAB.
- **RF Power Sweep Testing**:
    - The RF Signal Generator (EXG N5173B) was configured manually to sweep frequencies or power levels.
    - Power meter (NRX) readings were taken at each step.
    - Calculations for Gain and Power-Added Efficiency (PAE) were done offline after measurements.
    - RF pulsing had to be manually enabled and aligned.
    - Each test cycle required constant operator supervision and adjustment.

This manual workflow was highly time-consuming (1.5–3 hours per device), error-prone, and difficult to reproduce due to human involvement and inconsistencies in timing, delays, or logging.

Why Automation is Needed

The need for automation at SSPL arises from multiple practical and strategic concerns:

- **Time Efficiency**: Manual testing took several hours per device. Automation reduces this to a few minutes, improving throughput.
- **Repeatability**: Human error in voltage steps, reading delays, or measurement mismatches can be eliminated by precise, software-driven control.
- **Real-time Visualization**: Engineers benefit from instant plots of I-V curves, Gain, and PAE during execution — enabling faster decision-making.
- **Batch Testing**: Automation enables sequential testing of multiple devices with minimal reconfiguration.
- **Data Logging**: Software ensures structured logging of all measurements (CSV), timestamps, metadata, and configuration parameters for traceability.
- **Cross-Platform Usability**: By designing the system to run on both Linux and Windows, deployment is simplified across lab environments.
- **Operator Independence**: Once set up, even non-technical users can run tests with minimal training through a GUI.

In high-volume or research-critical environments like DRDO labs, where precision and reproducibility are paramount, these benefits become essential to maintain quality, efficiency, and pace of innovation.

2.2 Objectives

The main objectives of this internship project were defined as follows:

1. **Automate I-V Characterization Process**
    - Develop a Python-based tool to interface with the NGP800 power supply.
    - Enable user-defined voltage sweep ranges for gate (Vg) and drain (Vd).
    - Measure and record drain and gate currents.
    - Plot and store output/input characteristics (Id–Vd curves) in real-time.
    - Provide safety interlocks, GUI controls, and pause/resume functionality.
2. **Automate RF Power Sweep Testing**
    - Interface with the EXG signal generator and NRX power meter.
    - Automate power and/or frequency sweeps.

- Capture power input/output readings.
- Calculate performance parameters such as Gain and PAE dynamically.
- Plot these parameters live and save structured output files.

3. **Reduce Test Time**
    - Decrease the test duration per wafer device from hours to minutes.
    - Eliminate manual measurement overhead and data re-entry.
4. **Ensure Reliability and Repeatability**
    - Maintain consistent step sizes, timing delays, and command sequences using software control.
    - Avoid variability introduced by human operations.
5. **Design a User-Friendly System**
    - Build intuitive graphical user interfaces for each tool using PyQt5.
    - Allow flexible control of test parameters without requiring programming knowledge.
    - Offer real-time status, visual feedback, and error handling for better usability.
6. **Cross-Platform Compatibility**
    - Initially develop the system on Ubuntu.
    - Port and validate functionality on Windows systems used in the lab.
    - Document installation and runtime instructions for both platforms.
7. **Enable Future Scalability**
    - Design the architecture to allow integration with additional instruments.
    - Make the tools extensible to include database logging, remote control, or probe station integration

3. Tools and Technologies Used

This section outlines the hardware, software platforms, and Python libraries used in the development of the two automation systems during the internship at DRDO SSPL. The integration of these components enabled the successful automation of I-V characterization and RF power sweep processes.

3.1 Hardware Components

Instrument	Description	Purpose
Rohde & Schwarz NGP800 Power Supply	Programmable dual-channel power supply with SCPI interface	Used for voltage sweeps in I-V characterization (Vg and Vd control)
Keysight EXG Analog Signal Generator (N5173B)	High-performance RF signal generator	Used to generate frequency and power sweeps during RF testing
Rohde & Schwarz NRX Power Meter	Precision RF power meter with USB/SCPI support	Measures input and output power during RF sweep
Wafer Probe Station *(if used)*	Manual probe system for contacting wafer pads	Used for making electrical contact with on-wafer test structures

These instruments were controlled through USB interfaces using SCPI (Standard Commands for Programmable Instruments) commands issued from Python scripts.

3.2 Software Environment

Software	Purpose
Python 3.10	Primary programming language for all automation logic
Anaconda Distribution	Simplified Python environment and package management
VS Code	Code development and debugging environment

Software	Purpose
PyInstaller	Packaging of scripts into standalone .exe executables (for Windows deployment)
Windows Terminal / Command Prompt	Running scripts and launching executables on Windows
Ubuntu Terminal (22.04)	Initial development and testing environment for Linux-based automation
NI-VISA *(Windows)*	Interface driver required for instrument communication on Windows
pyvisa-py *(Linux)*	Python backend for VISA protocol on Ubuntu systems
Device Manager (Windows) / lsusb, dmesg (Ubuntu)	For verifying USB device detection and troubleshooting connection issues

The software stack was designed to be cross-platform compatible, ensuring consistent behaviour across both Linux and Windows environments.

3.3 Python Libraries and Frameworks

Library	Description	Usage
pyvisa	VISA library for Python	Core library for communicating with USB-connected instruments using SCPI commands
pyqt5	Python bindings for Qt	Used to build user-friendly graphical interfaces for both I-V and RF sweep tools
pyqtgraph	High-performance plotting library	Real-time plotting of voltage-current curves, gain, power output, and PAE graphs
numpy	Numerical computing package	Used for sweep calculations, data handling, and array operations

Library	Description	Usage
pandas	Data manipulation and storage	Stores tabular test results and outputs .csv files for each session
matplotlib	Plotting and visualization	Used for static plots and report generation (if required)
threading / time *(Python standard libraries)*	Multithreaded execution and timing control	Ensures responsive GUI during long test runs and controls sweep delays precisely

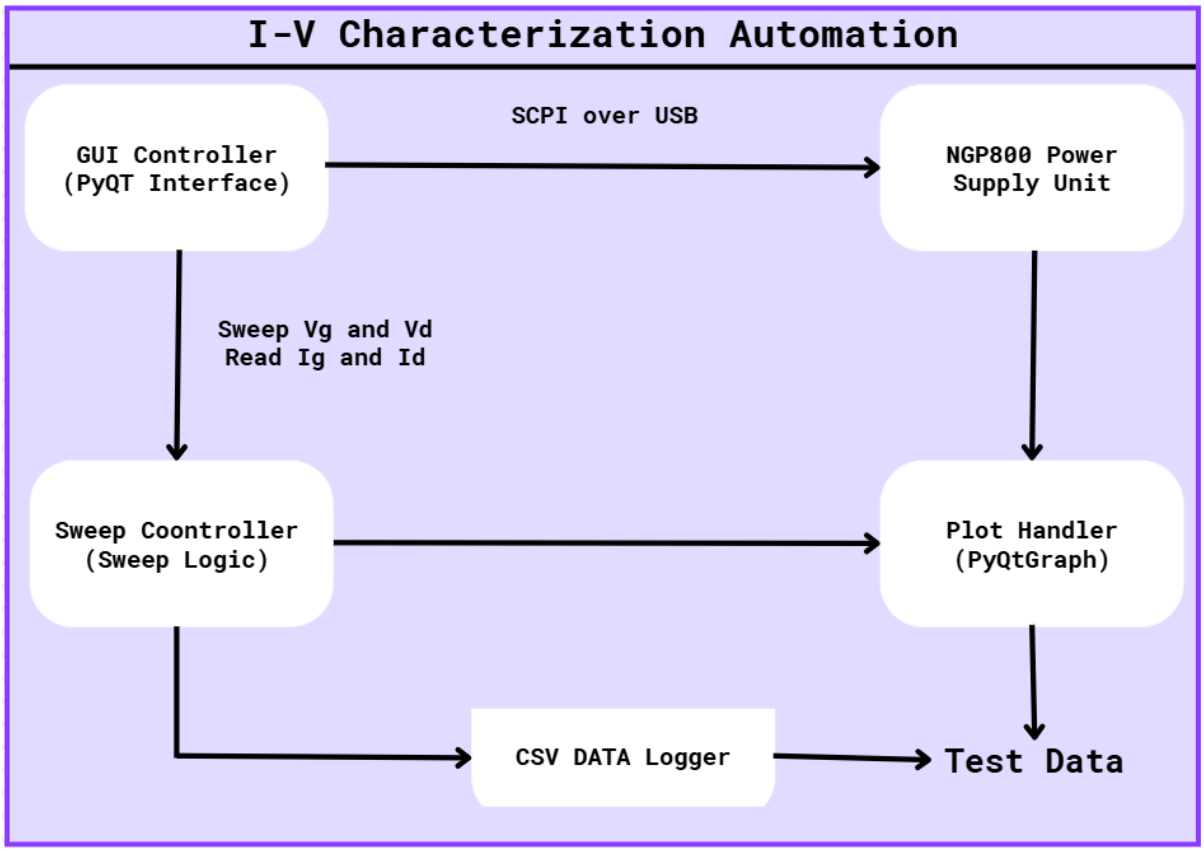
These libraries formed the core of the automation logic, enabling robust and efficient interaction with laboratory instruments while maintaining a clean and intuitive front-end interface.

4. System Architecture / Design

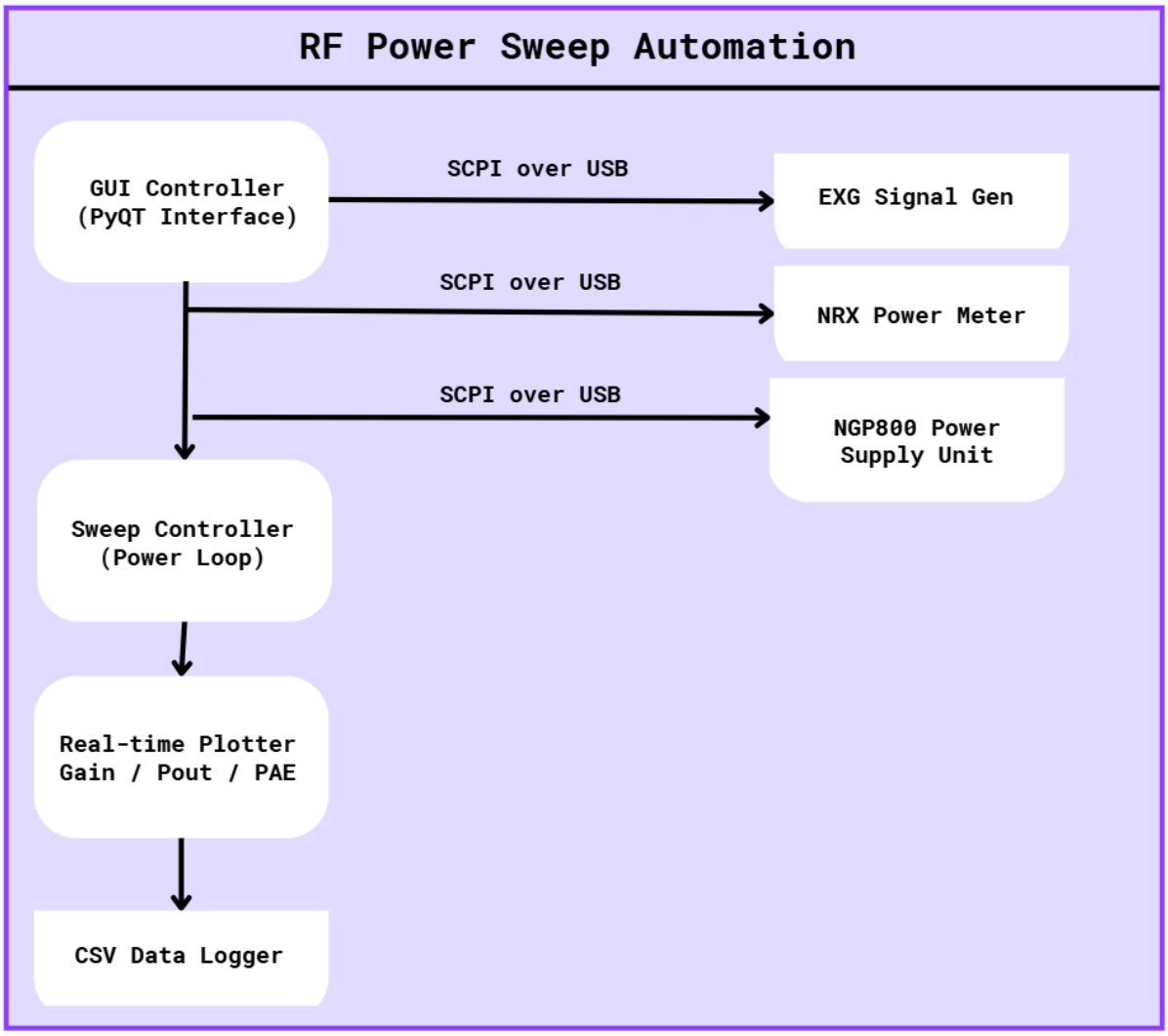
This section describes the high-level architecture of both automation systems—**I-V Characterization** and **RF Power Sweep**—including data flow, GUI layout, threading model, and instrument communication structure.

4.1 Block Diagrams and Data Flow

A. I-V Characterization Automation – Block Diagram

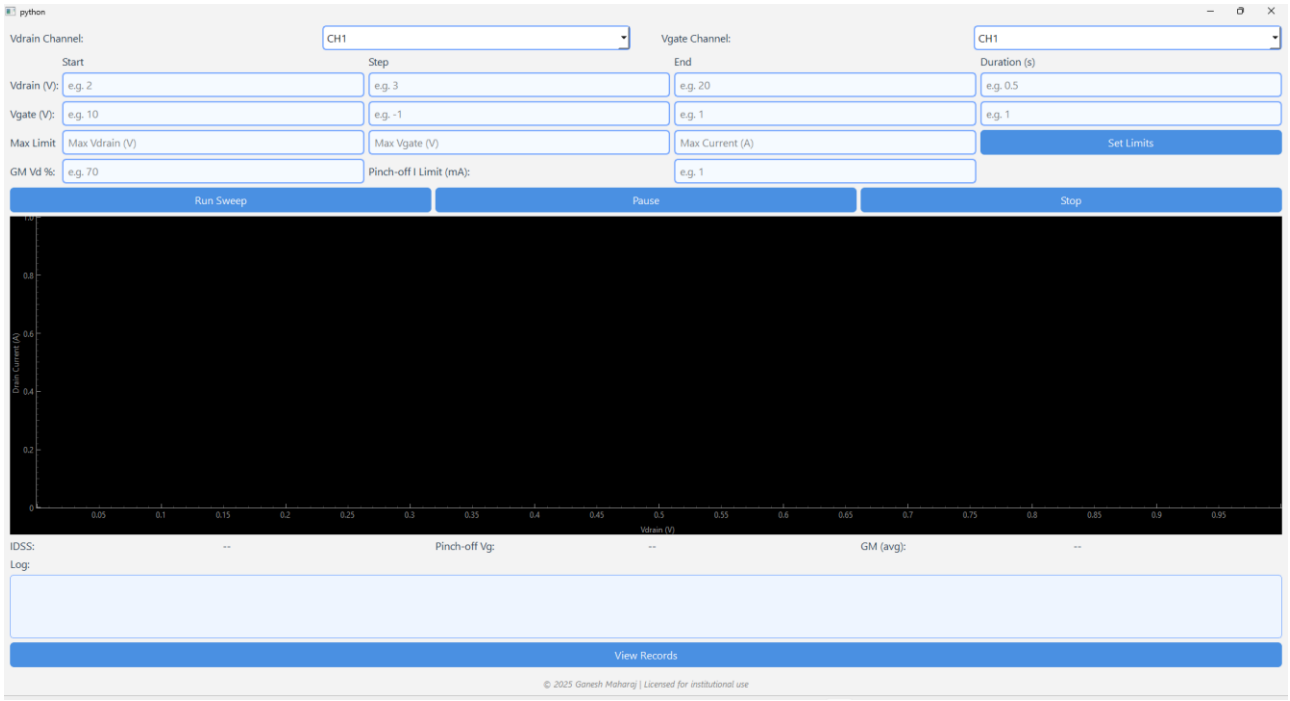


B. RF Power Sweep Automation – Block Diagram

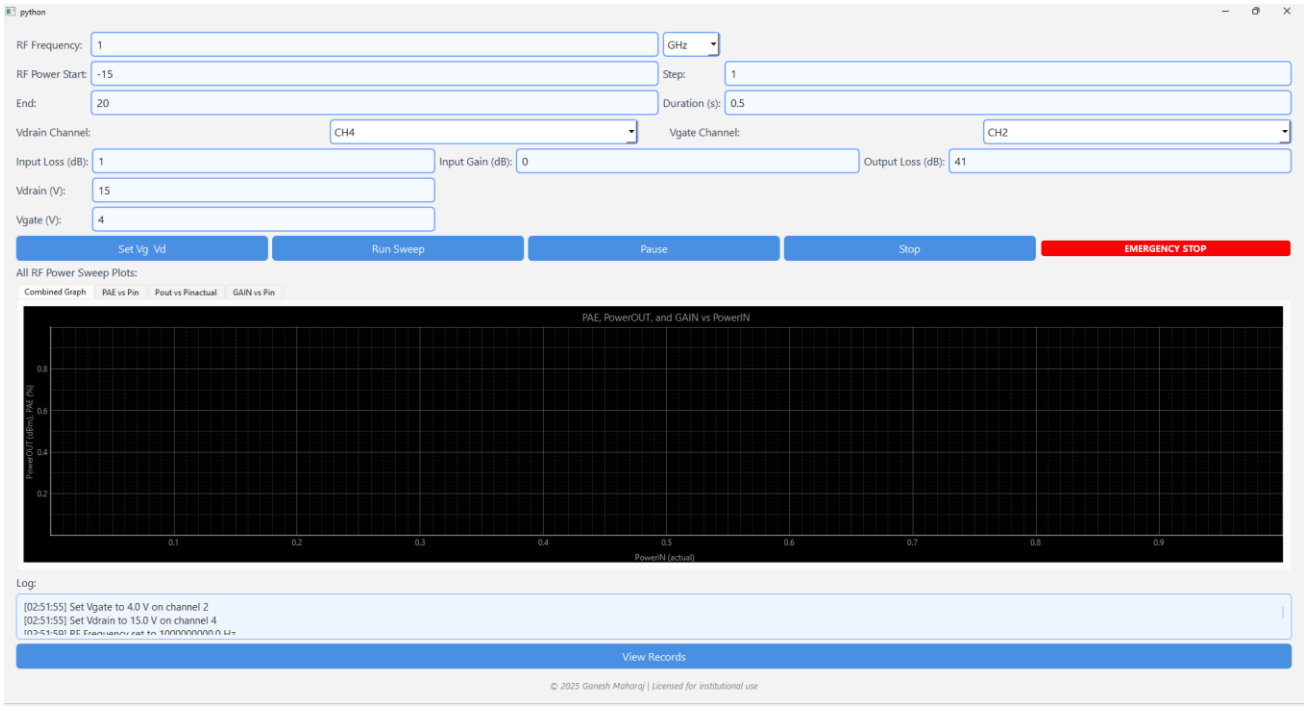


4.2 GUI Design Snapshots

I-V Characterization GUI Layout



RF Power Sweep GUI Layout



4.3 Threading Model

To maintain GUI responsiveness during long-running sweeps, **Python's threading module** was used.

Multithreading Use Case

- **Main Thread**: Handles PyQt GUI event loop and rendering.
- **Worker Thread 1**: Executes sweep logic (looping over Vg/Vd or Freq/Power).
- **Worker Thread 2**: Handles plotting and CSV logging if decoupled.

Design Considerations

- PyQt signals and slots used to update plots and logs from worker threads.
- Thread-safe flags used for **Pause**, **Resume**, and **Abort** controls.
- Ensures no UI freezing during high-speed measurement cycles.**8.4 Instrument Communication Flow**

I-V Characterization

1. **Connection Phase**
    - VISA resource manager detects connected USB instruments.
    - Device ID query (*IDN?) used to verify NGP800.
2. **Sweep Phase**
    - Vg set via INSTr:SEL CH1; VOLT {value}; OUTP ON
    - Nested loop over Vd via INSTr:SEL CH2; VOLT {value}
    - Delay introduced between steps.
    - Current measured via MEAS:CURR?
3. **Logging**
    - Ig, Id, Vg, Vd recorded at each point and sent to plotter and logger.

RF Power Sweep

1. **Connection Phase**
    - EXG and NRX identified via *IDN?
    - VISA addresses stored and validated.
2. **Sweep Phase**
    - EXG programmed with frequency and power using:
        - FREQ {value} and POW:LEV {value}
    - Output enabled: OUTP ON

- NRX queried for power readings via MEAS:POW:AC? or FETCH?

3. **Calculation**
    - Gain = Pout – Pin
    - PAE = (Pout – Pin) / DC Input Power × 100
4. **Live Plotting and CSV Logging**

5. Project 1 – I-V Characterization Automation

5.1 Problem Statement

Manual I-V characterization of wafer-level semiconductor devices involves performing nested sweeps of gate voltage (Vg) and drain voltage (Vd), while recording the corresponding gate and drain currents (Ig and Id). This process, when done manually using front-panel instrument controls and multimeters, is extremely time-consuming and susceptible to human error.

On average, a single device took over **1.5 to 2 hours** to fully characterize due to manual configuration, measurement, logging, and graph plotting. Moreover, any missed steps or operator inconsistencies could result in unusable or unreliable data. There was a strong need to **automate** the process for faster, repeatable, and more accurate characterization.

5.2 Description

The **I-V Characterization Automation Tool** was designed to automate this entire workflow using the **Rohde & Schwarz NGP800 dual-channel programmable power supply**. It allows users to define sweep parameters for Vg and Vd, executes the nested sweeps, measures the corresponding currents, and plots the **I-V characteristics in real time**.

Voltage Sweep of Vg and Vd

- **Outer loop**: Gate voltage (Vg) is swept across a specified range (e.g., 0 V to 3 V) with a configurable step size.
- **Inner loop**: For each Vg, the drain voltage (Vd) is swept (e.g., 0 V to 10 V) in steps.
- Voltages are set via SCPI commands sent to the NGP800 channels:
    - INSTr:SEL CH1; VOLT {Vg}
    - INSTr:SEL CH2; VOLT {Vd}
- Output for both channels is turned on before sweep starts: OUTP ON

Current Measurement and Plotting

- After each Vd set point, the drain current (Id) and optionally gate current (Ig) are measured via:
    - MEAS:CURR?
- The readings are stored in a matrix for plotting:
    - **X-axis**: Vd
    - **Y-axis**: Id

- **Multiple curves** for different Vg values

- Plots are updated in real time using PyQtGraph.

All results are logged to .csv files along with metadata like timestamp, sweep parameters, and instrument ID.

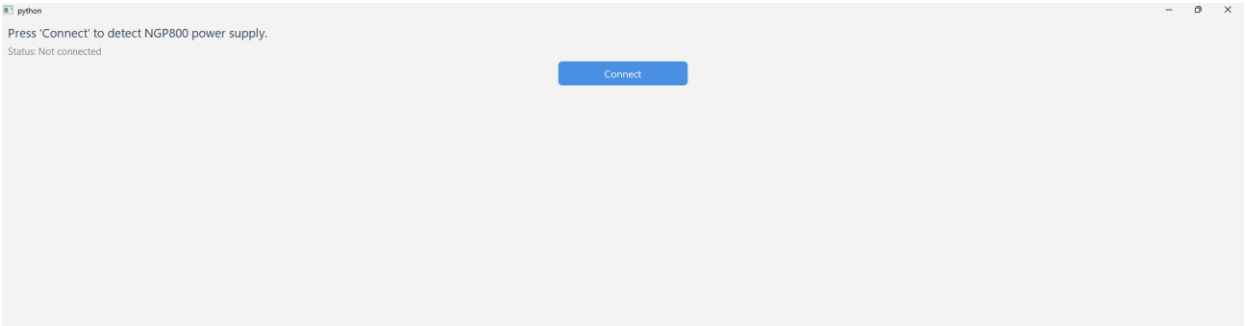
5.3 Implementation Details

GUI Elements

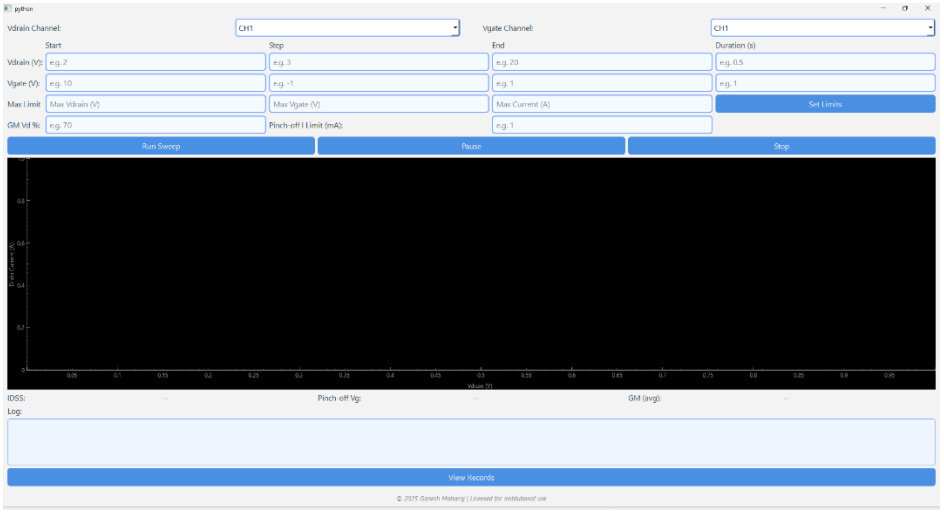
The GUI was developed using **PyQt5** and organized into logical components:

- **Intro Screen :**

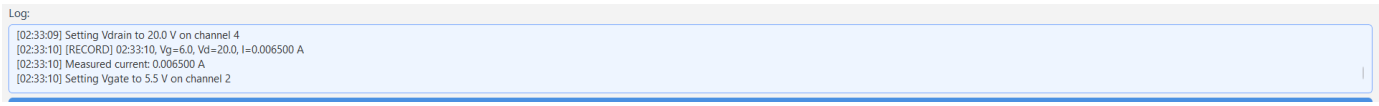
- **Connect Screen :**



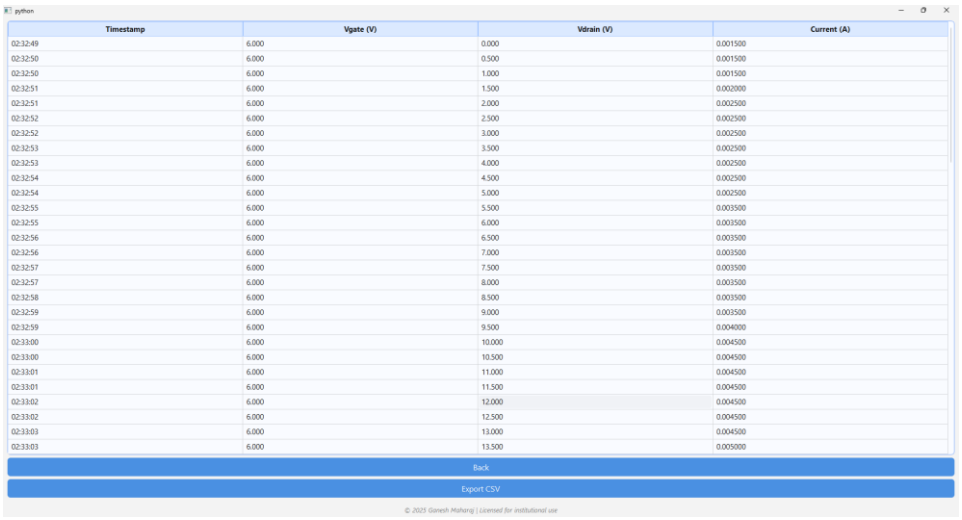
- **Control Screen :**



- **Logs :**



- **Records Screen :**



Safety / Emergency Handling

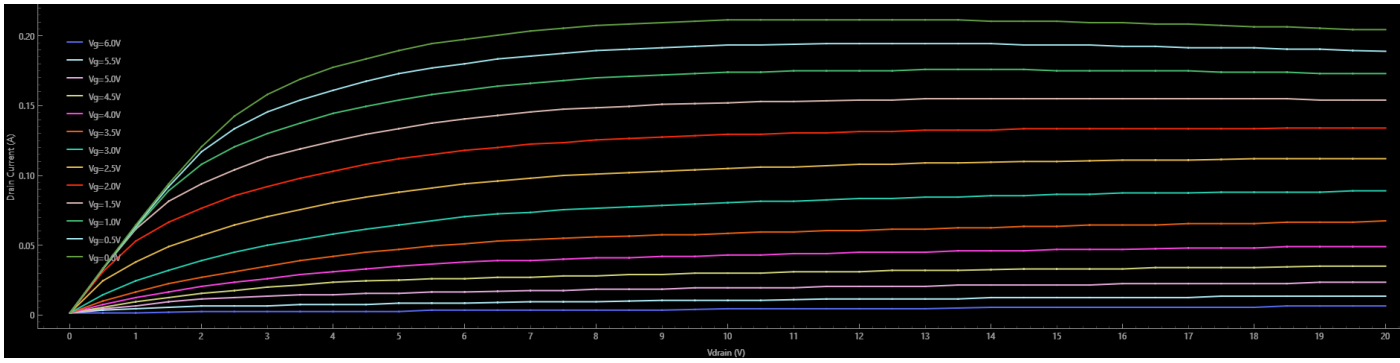
- **Emergency Stop Button**: Sets a flag that breaks out of all loops and immediately sets voltage to 0 V on both channels
- **Overcurrent Handling**: Optional threshold checking, disables output if exceeded
- **Device Disconnect Handling**: Detects if NGP800 is unexpectedly disconnected and alerts user
- **Reset on Exit**: Voltages and outputs are set to 0/off upon normal or abnormal termination

5.4 Sample Output

The following are typical outputs produced by the tool for the given Device details:

Wafer : CN 42 |.| **Location** : x=5 y=3 |.| **Device** : LN4

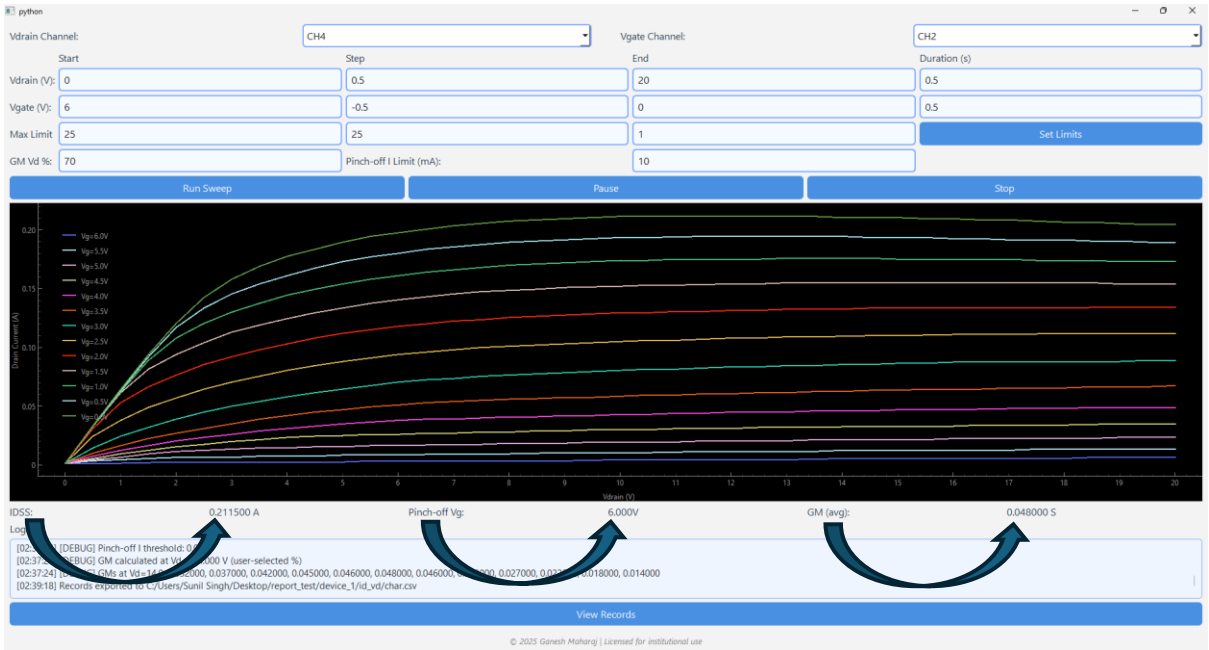
A. I-V Characteristic Plots



Graph features:

- Smooth curves with clearly separated Vg levels
- Real-time updates as sweep progresses
- Legends with Vg values
- Grid and axis markers for better analysis

B. IDSS, Pinch-off, GM



C. CSV Data Format

	A	B	C	D
1	Timestamp	Vgate (V)	Vdrain (V)	Current (A)
2	02:32:49	6	0	0.0015
3	02:32:50	6	0.5	0.0015
4	02:32:50	6	1	0.0015
5	02:32:51	6	1.5	0.002
6	02:32:51	6	2	0.0025
7	02:32:52	6	2.5	0.0025
8	02:32:52	6	3	0.0025
9	02:32:53	6	3.5	0.0025
10	02:32:53	6	4	0.0025
11	02:32:54	6	4.5	0.0025
12	02:32:54	6	5	0.0025
13	02:32:55	6	5.5	0.0035
14	02:32:55	6	6	0.0035
15	02:32:56	6	6.5	0.0035
16	02:32:56	6	7	0.0035
17	02:32:57	6	7.5	0.0035
18	02:32:57	6	8	0.0035
19	02:32:58	6	8.5	0.0035
20	02:32:59	6	9	0.0035
21	02:32:59	6	9.5	0.004
22	02:33:00	6	10	0.0045
23	02:33:00	6	10.5	0.0045
24	02:33:01	6	11	0.0045
25	02:33:01	6	11.5	0.0045
26	02:33:02	6	12	0.0045
27	02:33:02	6	12.5	0.0045
28	02:33:03	6	13	0.0045
29	02:33:03	6	13.5	0.005
30	02:33:04	6	14	0.0055
31	02:33:04	6	14.5	0.0055
32	02:33:05	6	15	0.0055
33	02:33:05	6	15.5	0.0055
34	02:33:06	6	16	0.0055

5.5 Challenges and Solutions

Challenge	**Solution**
USB device detection failure on Ubuntu	Added udev rules and used lsusb/dmesg to debug. Switched backend to pyvisa-py.
NGP800 not responding to SCPI commands	Verified command syntax with instrument manual. Added delays between commands.
GUI freezing during sweep	Moved sweep logic to a separate thread using Python threading.Thread. Used PyQt signals for plot updates.
Sudden disconnects or power fluctuations	Implemented try/except wrappers and retry mechanisms. Logged error messages in status panel.
Cross-platform issues between Ubuntu and Windows	Used OS-specific path handling and tested on both systems. Ensured ni-visa installed on Windows.
Plot lag on large sweeps	Switched from matplotlib to pyqtgraph for high-performance, real-time rendering.

6. Project 2 – RF Power Sweep Automation

6.1 Problem Statement

At DRDO SSPL Lab, RF wafer devices such as amplifiers, switches, and detectors need to be evaluated for performance under varying input power levels. The traditional method of conducting RF power sweeps involved manually adjusting power levels on the **EXG Analog Signal Generator**, reading the input and output power levels from the **NRX Power Meter**, and calculating critical parameters like **Gain** and **Power Added Efficiency (PAE)** externally after test completion.

This manual approach was time-consuming, required constant operator involvement, especially during **pulse operation** where timing was critical. Any delay or misalignment between setting the power and reading measurements could yield inaccurate results.

To streamline this, an **automated power sweep tool** was developed to perform the full test cycle — from configuring power levels to logging and plotting results — via software control, eliminating manual intervention and enabling rapid, repeatable, and accurate testing.

6.2 Description

The **RF Power Sweep Automation Tool** interfaces with two key instruments:

- **Keysight EXG Analog Signal Generator (N5173B)** for generating RF signals at a fixed frequency
- **Rohde & Schwarz NRX Power Meter** for measuring RF power input and output

The frequency is **fixed** throughout the test session. The tool performs a **linear power sweep**, increasing the RF output power from a start level to a stop level in user-defined steps (e.g., from -20 dBm to +20 dBm with 1 dB step).

At each input power level (**Pin**):

- The **EXG** sets the power level and outputs a continuous or pulsed RF signal.
- The **NRX** reads the **input power (Pin)** and **output power (Pout)** via connected sensors.
- The software calculates:
    - **Gain** = Pout – Pin (in dB)
    - **PAE** = [(Pout – Pin) / DC Input Power] × 100

All values are plotted in real-time and logged to .csv files. The GUI also supports **pulse enable/disable control**, and automatically pauses between steps to ensure accurate and stable measurements.

6.3 Implementation Details

Pulse Control

- The tool includes a GUI toggle to enable or disable **pulse mode**.
- When pulse mode is ON, the EXG is configured to output RF pulses using SCPI commands:
    - Pulse ON: *PULM:STAT ON*
    - Set period, width, and delay as required using:
        - *PULM:PER <val>, PULM:WIDT <val>*
- This is particularly useful for testing power-sensitive devices that should not be exposed to continuous waveforms.

Power Sweep Logic

- **EXG is configured once** at a fixed frequency (manually or through GUI).
- The sweep then increments the **RF output power (Pin)** in dB steps using:

POW:LEV <value>

- After each power setting:
    - Wait for a short **settling delay** (e.g., 500 ms to 1 s)
    - **Read measurements** from the NRX using SCPI:

FETCH:POW? CH2 → Pout

Gain and PAE Calculation

- **Gain (dB)** = Pout – Pin
- **PAE (%)** = ((Pout – Pin) / DC Input Power) * 100
- These values are dynamically calculated and updated on the plots and logs.

Record Logging

- Each reading is saved in a structured .csv file with timestamp and user-defined filename.
- Example file format:

Timestamp, Pin (dBm), Pout (dBm), Gain (dB), PAE (%)

2025-07-11 14:20:05, -10, -6.3, 3.7, 25.4

- **Live plots** (using PyQtGraph) are updated after each measurement to show:

- Pout vs Pin
- Gain vs Pin
- PAE vs Pin

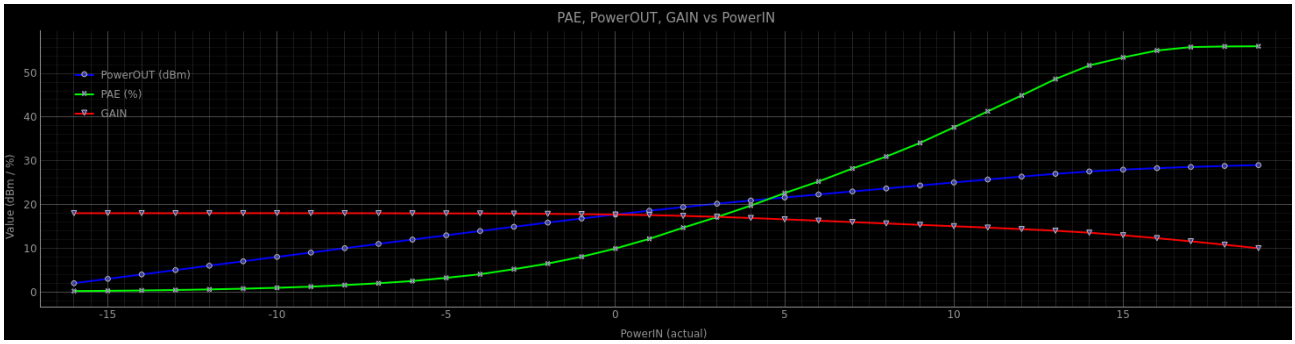
6.4 Sample Output

The following are typical outputs produced by the tool for the given Device details:

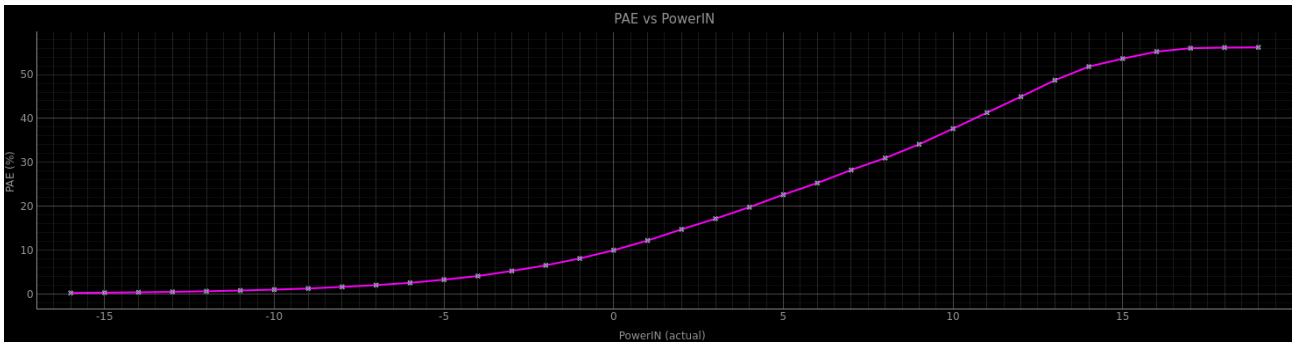
Wafer : CN 42 |.| **Location** : x=5 y=3 |.| **Device** : LN4 and Frequency : 1.5GHZ

A. Graphs

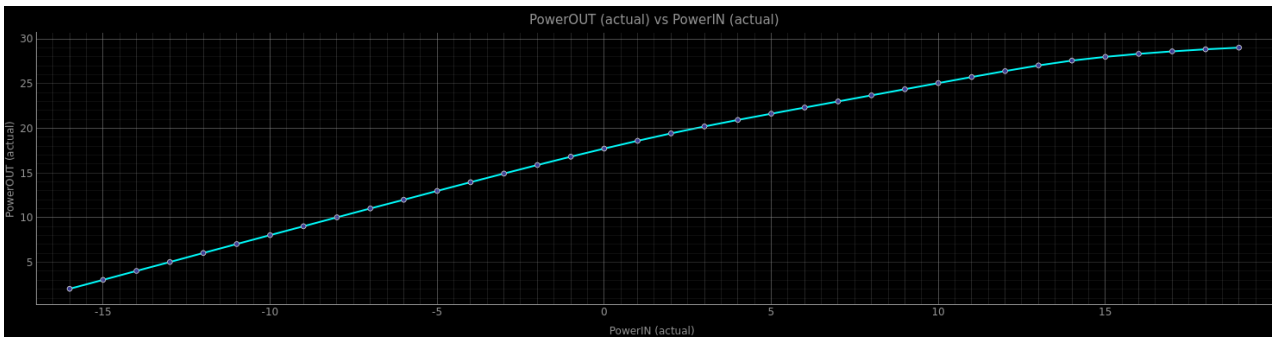
1. **Combined Graph**



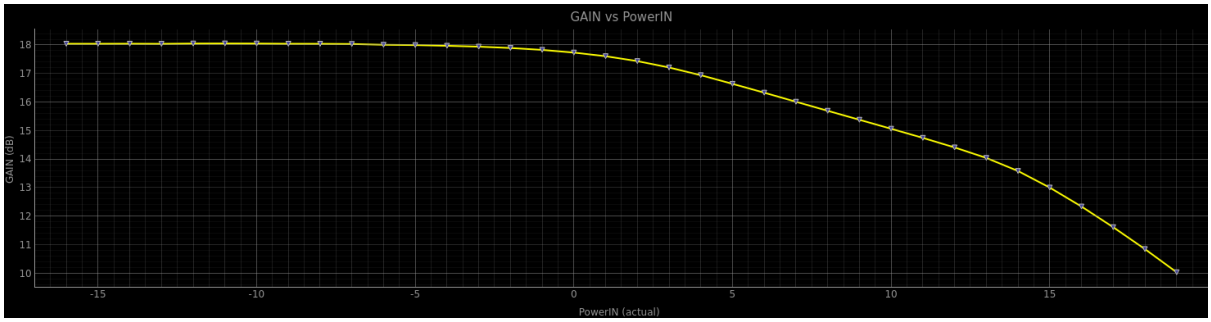
2. **PAE vs Pin**



3. **Pout vs Pin**



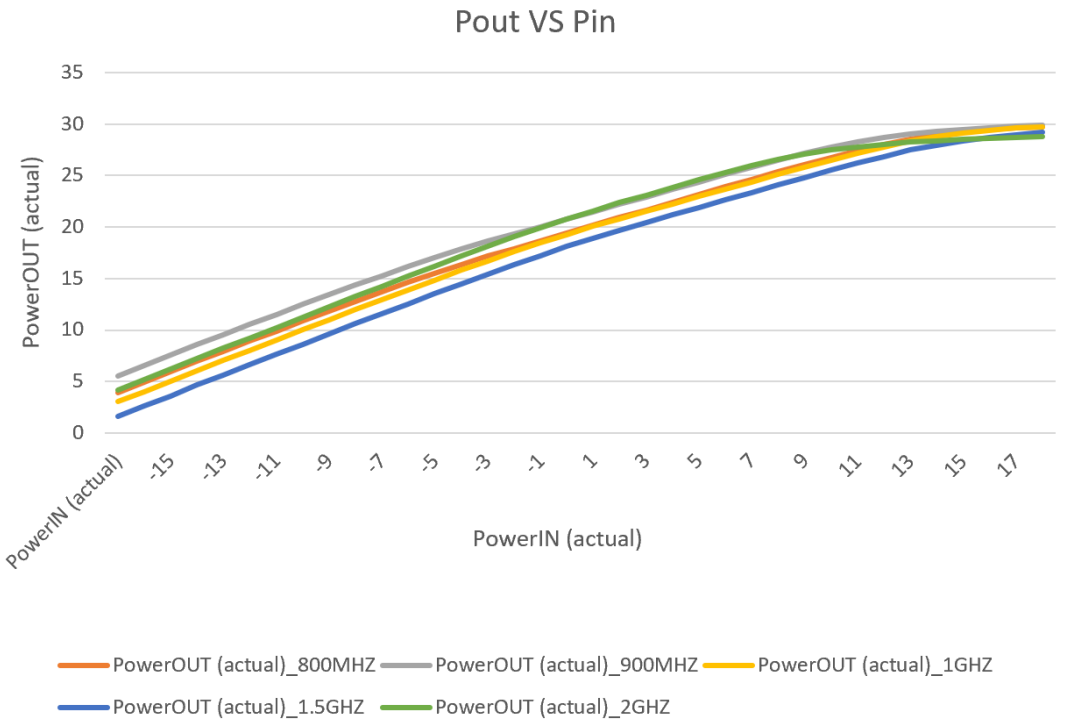
4. **GAIN vs Pin**

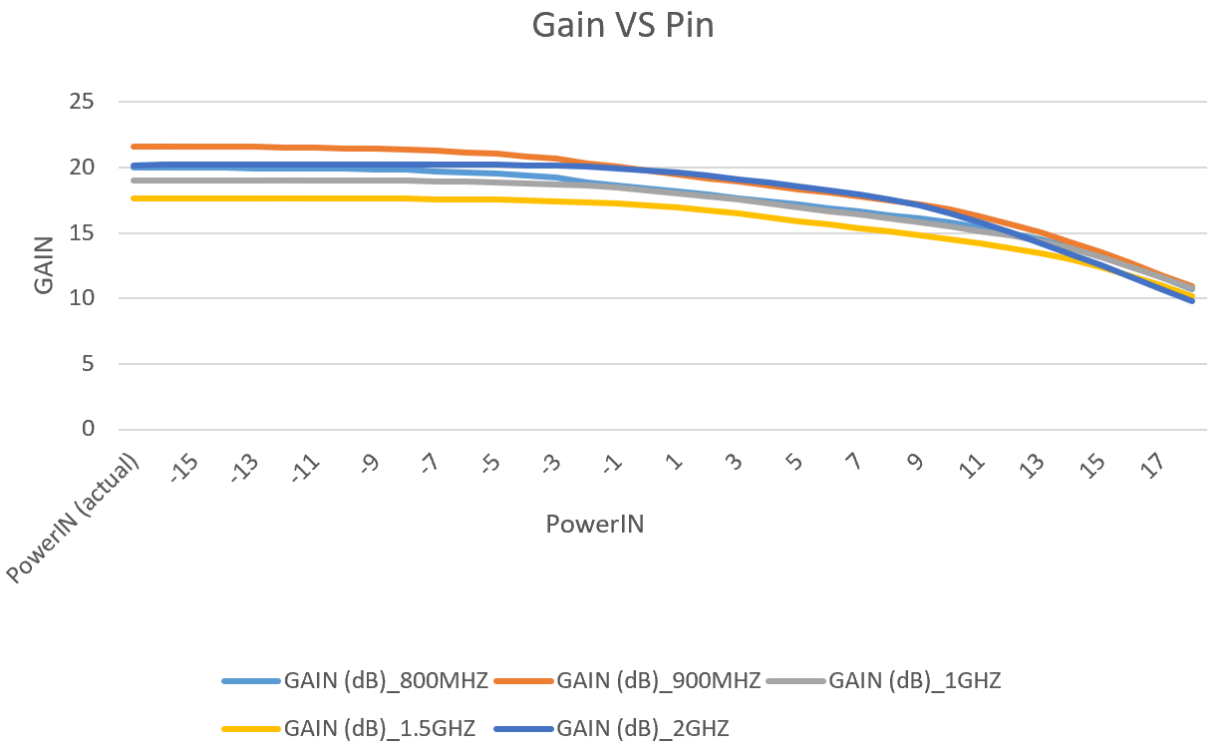


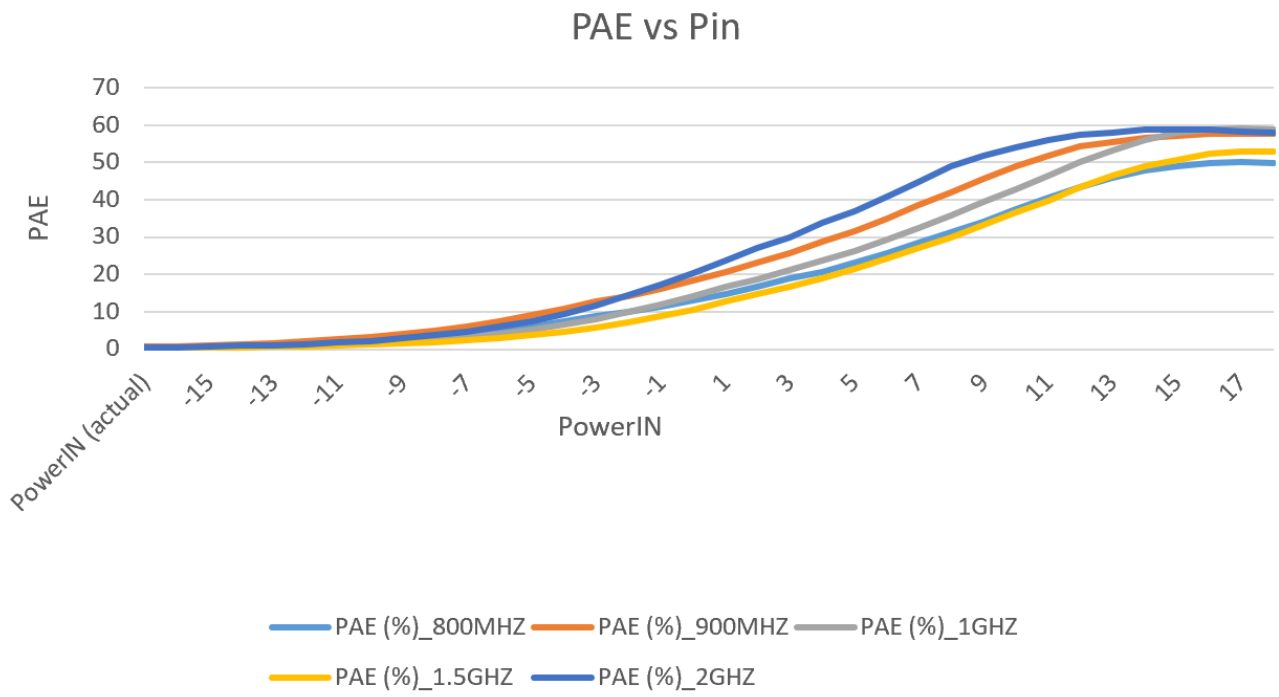
B. Data Table Format

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Timestamp	Vgate (V)	Vdrain (V)	Current (A)	RF Freq (Hz)	PowerIN (dBm)	PowerOUT (dBm)	PowerIN (actual)	PowerOUT (actual)	Pin_actual (mW)	Pout_actual (mW)	GAIN (dB)	Compression	PAE (%)
2	11:41:42	4	15	0.043	1.5E+09	-15	-38.95792	-16	2.04208	0.02511886	1.6003243	18.04208	0	0.244218
3	11:41:45	4	15	0.043	1.5E+09	-14	-37.95892	-15	3.04108	0.03162278	2.01422508	18.04108	0.001	0.30738
4	11:41:49	4	15	0.043	1.5E+09	-13	-36.95646	-14	4.04354	0.03981072	2.53719589	18.04354	-0.00146	0.387192
5	11:41:52	4	15	0.043	1.5E+09	-12	-35.96067	-13	5.03933	0.05011872	3.19104552	18.03933	0.00275	0.486965
6	11:41:55	4	15	0.0425	1.5E+09	-11	-34.95028	-12	6.04972	0.06309573	4.02691071	18.04972	-0.00764	0.621775
7	11:41:58	4	15	0.042	1.5E+09	-10	-33.94833	-11	7.05167	0.07943282	5.071857	18.05167	-0.00959	0.792448
8	11:42:01	4	15	0.042	1.5E+09	-9	-32.95084	-10	8.04916	0.1	6.38140047	18.04916	-0.00708	0.997048
9	11:42:05	4	15	0.042	1.5E+09	-8	-31.95818	-9	9.04182	0.12589254	8.02014093	18.04182	0.00026	1.253055
10	11:42:08	4	15	0.041	1.5E+09	-7	-30.96012	-8	10.03988	0.15848932	10.09225	18.03988	0.0022	1.615246
11	11:42:10	4	15	0.041	1.5E+09	-6	-29.96708	-7	11.03292	0.19952623	12.68504464	18.03292	0.00916	2.030166
12	11:42:12	4	15	0.041	1.5E+09	-5	-28.99443	-6	12.00557	0.25118864	15.86927185	18.00557	0.03651	2.539526
13	11:42:14	4	15	0.04	1.5E+09	-4	-28.00458	-5	12.99542	0.31622777	19.93159252	17.99542	0.04666	3.269227
14	11:42:16	4	15	0.04	1.5E+09	-3	-27.02954	-4	13.97046	0.39810717	24.94858966	17.97046	0.07162	4.091747
15	11:42:18	4	15	0.039	1.5E+09	-2	-26.0594	-3	14.9406	0.50118723	31.19320504	17.9406	0.10148	5.246499
16	11:42:20	4	15	0.039	1.5E+09	-1	-25.10624	-2	15.89376	0.63095734	38.84865612	17.89376	0.14832	6.53294
17	11:42:22	4	15	0.039	1.5E+09	0	-24.17308	-1	16.82692	0.79432823	48.16061232	17.82692	0.21516	8.096801
18	11:42:24	4	15	0.039	1.5E+09	1	-23.26723	0	17.73277	1	59.33036225	17.73277	0.30931	9.971002
19	11:42:25	4	15	0.039	1.5E+09	2	-22.39493	1	18.60507	1.25892541	72.52821682	17.60507	0.43701	12.18278
20	11:42:27	4	15	0.039	1.5E+09	3	-21.56637	2	19.43363	1.58489319	87.77341585	17.43363	0.60845	14.73308

Similarly, we have Tested this device at different frequencies and This is its Behavior:




Gain VS Pin
25
20
15
10
5
0
GAIN
PowerIN (actual)
-15
-13
-11
-9
-7
-5
-3
-1
1
3
5
7
9
11
13
15
17
PowerIN
GAIN (dB)_800MHZ
GAIN (dB)_900MHZ
GAIN (dB)_1GHZ
GAIN (dB)_1.5GHZ
GAIN (dB)_2GHZ


PAE vs Pin
70
60
50
40
30
20
10
0
PAE
PowerIN (actual)
-15
-13
-11
-9
-7
-5
-3
-1
1
3
5
7
9
11
13
15
17
PowerIN
PAE (%)_800MHZ
PAE (%)_900MHZ
PAE (%)_1GHZ
PAE (%)_1.5GHZ
PAE (%)_2GHZ

6.5 Challenges and Solutions

Challenge	Solution
Pulse timing misalignment between EXG and NRX	Introduced delay after setting power level and before measurement to allow stable RF pulse
NRX not responding reliably to FETCH?	Implemented retry logic with timeout; added delays between SCPI calls
GUI freezing during sweep	Used Python threading.Thread to run sweep logic in the background, with PyQt signals for plot and log updates
Instrument detection issues on Windows	Installed and configured **NI-VISA**; used visa.ResourceManager() to list and match device IDs
Incorrect gain calculation due to mismatched sensors	Verified calibration and ensured correct mapping of NRX channels to DUT input/output
Platform differences in SCPI handling	Ensured code used platform-neutral VISA strings and tested thoroughly on both Ubuntu and Windows
Handling missing data or zero readings	Added exception handling, sanity checks (e.g., skip or warn on Pout < Pin or zero readings)

7. Platform Transition (Ubuntu to Windows)

To ensure maximum usability across the lab's systems, both automation tools were designed for **cross-platform compatibility**, starting with initial development on **Ubuntu 22.04 LTS** and later migrated to **Windows 10**. This transition required adjustments in package management, USB driver handling, VISA communication, and application launch mechanisms.

7.1 Initial Development on Ubuntu

Package Installation via pip and Anaconda

- The development environment was set up using **Anaconda** for simplified dependency management.
- Required Python libraries (e.g., pyvisa, pyqt5, pyqtgraph, numpy, pandas, matplotlib) were installed using:
- A custom Conda environment was created to isolate and manage project dependencies.

pip install -r requirements.txt

USB Permissions and Device Access

- USB instruments were initially inaccessible due to permission issues. These were resolved by:
    - Adding **udev rules** in /etc/udev/rules.d/99-instruments.rules:

SUBSYSTEM=="usb", ATTR{idVendor}=="XXXX", ATTR{idProduct}=="YYYY", MODE="0666"

    - Reloading rules:

sudo udevadm control --reload-rules
sudo udevadm trigger

- Verified connectivity using:
    - lsusb: List USB devices
    - dmesg: Inspect connection logs and kernel messages
    - sudo chmod 666 /dev/ttyUSB0 (for temporary access during debugging)

Device Identification and PyVISA Setup

- PyVISA was configured to use the **pyvisa-py backend** on Ubuntu (no NI-VISA required):

rm = visa.ResourceManager('@py') # Forces use of pyvisa-py backend

- Devices were discovered using:

instruments = rm.list_resources()

print(instruments)

- Instrument response verified via *IDN? SCPI command to ensure correct resource mapping.

7.2 Migration to Windows

NI-VISA Installation

- On Windows, **NI-VISA** is required for USB and GPIB communication.
- The following steps were followed:
    - Installed **NI-VISA Runtime 21.x** or latest stable version.
    - Verified connectivity via **NI Measurement & Automation Explorer (NI-MAX)**.
    - Used standard PyVISA:

rm = visa.ResourceManager()

COM Port Access and Permission Issues

- On Windows, USB instruments often appear as virtual COM ports.
- Required steps:
    - Identify COM port using **Device Manager**.
    - Ensure the script runs with **Admin privileges** to access certain ports.
    - Used longer VISA timeout to handle delayed responses:

instrument.timeout = 5000 # in milliseconds

- Anti-virus/firewall occasionally blocked USB/VISA access. These were disabled during testing.

Batch File Creation for Ease of Launching

- To simplify tool usage for non-technical staff, a **.bat file** was created to launch the application:

@echo off

cd C:\Users\YourUsername\Documents\IVTool

call activate iv-env

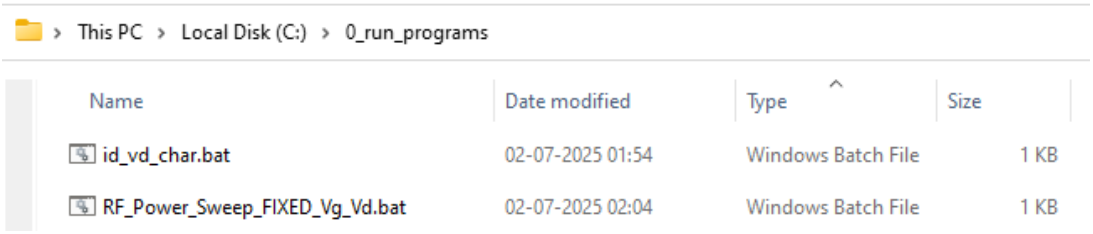
python main.py

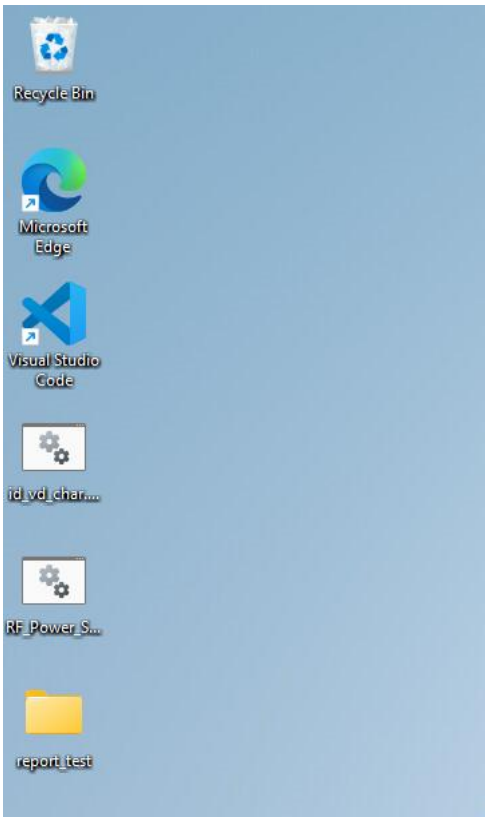
pause

- For compiled .exe versions, this was adjusted to:

start IV_Tool.exe

- Icons and desktop shortcuts were added for direct launching.





Cross-Platform Code Compatibility Fixes

Issue	Fix
File path differences (/ vs \\)	Used os.path.join() for all path operations
Backend differences in PyVISA	Explicitly handled @py vs default backend
Case sensitivity in filenames	Avoided mixed-case imports and filenames
Serial port naming	Used list_resources() and partial ID matching (e.g., USB, COM, ASRL)

7.3 Summary of OS-Specific Hurdles and Resolutions

Issue	Ubuntu	Windows	Resolution
VISA backend	pyvisa-py	NI-VISA required	Used platform detection to select backend
USB Permissions	udev rules, chmod	Not required	Added Linux permission fix documentation
COM Port Detection	/dev/ttyUSBx	COMx	Used rm.list_resources() dynamically
Plot Lag	None	Slight lag with large datasets	Switched to PyQtGraph for performance
Launching Script	.sh from terminal	.bat file	Provided launch scripts for both platforms
Packaging	Not required	Needed .exe for distribution	Used PyInstaller on Windows

The tools now run **consistently on both platforms**, with full instrument communication, GUI functionality, and logging intact.

8. Results and Performance

The automation of **I-V Characterization** and **RF Power Sweep** at DRDO SSPL Lab delivered substantial improvements in testing speed, data accuracy, and operator usability. The new system replaced a tedious, manual process with a robust, GUI-driven solution that integrated seamlessly with existing lab instruments.

8.1 Time Saved Compared to Manual Process

Prior to automation, both I-V and RF power sweep procedures were conducted manually, requiring significant human involvement for setting voltage/power levels, recording measurements, and generating plots.

Task	Manual Time	Automated Time	Reduction
I-V Characterization (1 Device)	~1.5–2 hours	~8–10 minutes	~90%
RF Power Sweep (1 Device)	~1.5–2 hours	~5–7 minutes	~90–95%

Parallel Benefits:

- No operator supervision required after launch
- Batch testing of multiple DUTs possible
- Instant availability of plots and CSV logs

The significant time reduction enabled faster wafer-level validation and allowed researchers to test more devices per session.

8.2 Repeatability and Accuracy Improvements

Automation eliminated the variability introduced by manual operation, improving data integrity and reproducibility across devices and test sessions.

Manual Issues	Resolved By Automation
Inconsistent voltage/power step sizes	Software-defined precise sweep resolution
Human delay between setting & measuring	Programmed delays with millisecond-level accuracy

Manual Issues	Resolved By Automation
Errors in manual data logging	Automatic, timestamped, and structured CSV export
Plotting errors (scale mismatch, typos)	Real-time standardized graph rendering
Missed safety conditions (e.g., overcurrent)	Emergency abort logic and interlock checks

Furthermore, instruments were queried directly using SCPI, removing any subjective reading or timing issues, and ensuring every test followed a strictly repeatable protocol.

8.3 Feedback from Lab Personnel

The automation tools were demonstrated to and used by RF and device engineers at SSPL. Based on their usage and feedback, the following points were noted:

Positive Feedback

- **"This saves us hours every day — just set and forget."**
- **"The I-V plot updates are clean, and we can capture saturation regions instantly."**
- **"Very easy to use. Even interns can now run the tests."**
- **"No more struggling with Excel plots from multimeter readings."**

Adoption Impact

- The tools were adopted for use on both Windows and Ubuntu-based lab systems.
- Engineers appreciated the cross-platform compatibility and GUI simplicity.
- Lab staff acknowledged improved **workflow consistency** and **data traceability** across multiple test benches.

Suggestions Received

- Add option for **real-time gain markers** or slope estimation in I-V plots.
- Add a feature for **frequency** Sweep
- Add **current compliance limit control** via GUI.

These suggestions have been recorded for future enhancements.

In summary, the automation systems significantly enhanced the productivity, accuracy, and usability of wafer-level characterization tasks at SSPL, aligning with the lab's goal of modernizing and scaling semiconductor testing infrastructure.

9. Conclusion

9.1 Summary of Achievements

During the two-month internship at **Solid State Physics Laboratory (SSPL), DRDO**, two critical wafer-level test processes—**I-V Characterization** and **RF Power Sweep**—were successfully automated using Python-based solutions. These systems replaced the existing manual workflow, which was time-consuming and error-prone, with robust, GUI-driven tools that significantly improved the **speed**, **repeatability**, and **accuracy** of device testing.

Key accomplishments include:

- Developed two full-fledged automation tools with **real-time plotting**, **SCPI-based instrument control**, and **CSV data logging**.
- Achieved **90%+ reduction in testing time**, improving daily throughput and enabling batch testing.
- Designed intuitive **cross-platform GUIs** using PyQt5, suitable for both technical and non-technical users.
- Integrated with **Rohde & Schwarz NGP800**, **Keysight EXG**, and **R&S NRX** using **PyVISA** on both **Ubuntu** and **Windows**.
- Ensured safe operation via features like **emergency stop**, **pulse mode control**, and **threaded execution** to maintain GUI responsiveness.
- Packaged tools with .bat/.sh launchers and documented full installation steps for lab deployment.

The tools were tested in the lab environment and received positive feedback from SSPL engineers for their **usability**, **reliability**, and **time-saving impact**.

9.2 How the System Can Be Extended or Improved

While the current version of the automation tools meets the core requirements, several enhancements are possible to further improve functionality and flexibility:

Suggested Improvements

Area	**Potential Extension**
Measurement Accuracy	Add averaging or statistical filtering over multiple samples per point
Multi-DUT Testing	Add support for batch or sequential testing of multiple devices automatically

Area	Potential Extension
Database Logging	Store results in structured database (e.g., SQLite or MongoDB) for long-term tracking
Remote Access	Web-based dashboard or remote control via Flask/HTTP API
Probe Station Integration	Combine with automated probe station for fully unattended wafer characterization
Compiled Executable	Provide .exe/.AppImage versions for plug-and-play deployment without requiring Python installation

Implementing these features would allow the tools to scale into a **more modular, extensible, and production-grade testing framework**, suitable not just for lab R&D, but also for small-volume manufacturing or prototyping environments.

10. Future Work

While the current system provides a robust foundation for automating I-V Characterization and RF Power Sweep processes, there are several opportunities to extend its functionality. These improvements aim to enhance **usability**, **scalability**, and **integration** with broader lab automation workflows.

10.1 Remote Control and Monitoring

To increase flexibility and reduce physical dependence on test benches, a **remote access interface** can be developed:

- **Approach**:
    - Implement a **Flask or FastAPI server** that exposes key operations (start, stop, status, get plot) over HTTP.
    - Use a **web-based dashboard** for remote monitoring of sweep progress and real-time plots.
- **Benefits**:
    - Engineers can monitor long sweeps remotely.
    - Tests can be initiated from a centralized control room or remotely accessed lab terminal.
    - Error messages or plots can be sent via email/Slack notifications for unattended testing.

10.2 Auto Device Recognition and SCPI Detection

To simplify initial setup and eliminate manual VISA resource selection:

- **Approach**:
    - Implement a **device discovery module** using rm.list_resources() and *IDN? queries.
    - Automatically map known instrument IDs to their roles (e.g., "NGP800" → Power Supply).
- **Benefits**:
    - Easier for new users or interns to launch the tools.
    - Prevents configuration errors due to incorrect resource selection.
    - Supports plug-and-play behavior across benches.

10.3 Data Export to Centralized Database

To enable historical tracking, trend analysis, and searchability:

- **Approach**:
    - Replace or augment CSV logging with **database export**, using:
        - SQLite (local)
        - MongoDB / PostgreSQL (networked)
    - Include test metadata, sweep settings, timestamps, and results.
- **Benefits**:
    - Centralized repository of all test data
    - Enables post-processing, statistical analysis, and reporting
    - Supports filtering by DUT ID, date, operator, etc.

10.4. Integration with Probe Station Automation

To complete the loop for **fully unattended wafer testing**, integrate the software with a motorized or semi-automatic **probe station**:

- **Approach**:
    - Interface with the probe station's stage controller via serial or GPIB/USB (if available)
    - Move to the next die location automatically after each test
- **Benefits**:
    - Enables sequential testing of all dies on a wafer
    - No operator involvement once the wafer is loaded

10.5 Additional Enhancements (Optional)

Feature	**Description**
Live Graph Export	Enable one-click export of plots as PNG or PDF
Test Templates	Save/load sweep configurations to reduce operator setup time
Self-diagnostics	Include a startup hardware check to validate instrument communication
Calibration Mode	Add support for sensor calibration or offset correction

11. References

This section lists the technical documents, software libraries, and manuals referred to during the design, development, and deployment of the I-V Characterization and RF Power Sweep Automation Tools.

11.1 Instrument Manuals and Documentation

1. **Rohde & Schwarz NGP800 Series User Manual**
    - Document for SCPI command set and dual-channel voltage control
    - Source: https://www.rohde-schwarz.com
2. **Rohde & Schwarz NRX Power Meter Operating Manual**
    - Covers measurement modes, SCPI commands, and pulse configuration
    - Source: https://www.rohde-schwarz.com
3. **Keysight EXG Analog Signal Generator (N5173B) Programmer's Guide**
    - SCPI syntax for RF frequency, power, pulse control, and modulation
    - Source: https://www.keysight.com

11.2 Python Library Documentation

1. **PyVISA – Python VISA Interface Library**
    - https://pyvisa.readthedocs.io
2. **PyQt5 – Python Bindings for Qt**
    - https://www.riverbankcomputing.com/software/pyqt/intro
3. **PyQtGraph – Real-Time Data Plotting Library**
    - http://www.pyqtgraph.org/documentation/
4. **NumPy – Scientific Computing Tools**
    - https://numpy.org/doc/
5. **Pandas – DataFrame and CSV Handling**
    - https://pandas.pydata.org/docs/
6. **Matplotlib – Plotting and Visualization**
    - https://matplotlib.org/stable/contents.html
7. **NI-VISA – National Instruments VISA Library**
    - https://www.ni.com/en/support/downloads/drivers/download.ni-visa.html

11.3 DRDO and SSPL-Specific References

- **Internal Test Procedures and Lab Standards**
  (Confidential – Referenced during lab onboarding, not publicly available)
    - Includes wafer testing protocols, safety guidelines, and result validation norms
    - Provided under SSPL supervision during internship
- **SSPL Equipment SOPs and Compliance Requirements**
    - For probe station operation, RF shielding protocols, and instrument handling
    - Discussed with lab mentors and technical team

17. Appendix

This section contains supporting technical artifacts used in the implementation of the I-V Characterization and RF Power Sweep automation tools. These resources serve as a reference for future development, debugging, or training purposes.

17.1 Sample Code Snippets

I-V Characterization: Voltage Sweep Logic

```
for i, vd in enumerate(self.vd_values):
    while self.pause_event.is_set():
        time.sleep(0.1)
    if self.stop_event.is_set():
        break

    self.log_msg.emit(f"Setting Vdrain to {vd} V on channel {self.vd_chan}")
    self.instrument.write(f"INST:NSEL {self.vd_chan}")
    self.instrument.write(f"VOLT {vd}")
    self.instrument.write(f"OUTP ON")
    time.sleep(self.vd_dur)

    self.instrument.write("MEAS:CURR?")
    current = float(self.instrument.read())
    timestamp = time.strftime("%H:%M:%S")
    # Check limits
    if self.vd_max is not None and vd > self.vd_max:
        self.log_msg.emit(f"Vdrain limit exceeded: {vd} > {self.vd_max}. Stopping sweep.")
        self.stop_event.set()
        break
    if self.curr_max is not None and current > self.curr_max:
        self.log_msg.emit(f"Current limit exceeded: {current} > {self.curr_max}. Stopping sweep.")
        self.stop_event.set()
        break

    self.log_msg.emit(f"[RECORD] {timestamp}, Vg={vg}, Vd={vd}, I={current:.6f} A")
    # Signal used for record logging - handled via log_msg for simplicity

    record_signal = pyqtSignal(str, float, float, float)
    self.log_msg.emit(f"Measured current: {current:.6f} A")
    currents.append(current)
    self.data_points.append((vg, vd, current))

    self.update_plot.emit(plot_data, self.vd_values[:i+1], currents)
```

RF Power Sweep: Gain and PAE Calculation

```
try:
    self.instrument.write(f"INST:NSEL {self.vd_chan}")
    self.instrument.write("MEAS:CURR?")
    current = float(self.instrument.read())

    rf_freq = None
    power_in = None
    live_power = None

    if self.rf_instr:
        try:
            rf_freq = float(self.rf_instr.query("FREQ?").strip())
            power_in = float(self.rf_instr.query("POW?").strip())
        except Exception as e:
            self.log_msg.emit(f"[WARNING] RF read failed: {e}")

    if self.pm_instr:
        try:
            time.sleep(0.5)
            if hasattr(self.pm_instr, "device_type") and self.pm_instr.device_type == "NRP2":
                response = self.pm_instr.query("READ?").strip()
            else:
                response = self.pm_instr.query("MEAS:POW?").strip()
            live_power = float(response)
        except Exception as e:
            self.log_msg.emit(f"[WARNING] Power meter read failed: {e}")

    timestamp = time.strftime("%H:%M:%S")
    self.log_msg.emit(
        f"[RECORD] {timestamp}, Vg={self.app.latest_vg}, Vd={self.app.latest_vd}, I={current:.8f} A, "
        f"Freq={rf_freq}, PowerIN={power_in}, Power={live_power}"
    )

    self.log_msg.emit(f"Measured current: {current:.8f} A")

except Exception as e:
    self.log_msg.emit(f"[ERROR] Exception in SweepWorker: {e}")
```

17.2 Instrument SCPI Command Lists

NGP800 Power Supply

Command	Description
*IDN?	Query device identity
INSTr:SEL CHx	Select channel x
VOLT <value>	Set voltage level
MEAS:CURR? CHx	Measure current on channel x
OUTP ON/OFF	Enable or disable output

Keysight EXG Signal Generator

Command	Description
*IDN?	Query identity

Command	Description
FREQ <val>	Set frequency (manual)
POW:LEV <val>	Set output power
OUTP ON/OFF	Enable/disable RF output
PULM:STAT ON/OFF	Enable/disable pulse modulation

R&S NRX Power Meter

Command	Description
*IDN?	Identify device
FETCH:POW? CH1	Measure input power
FETCH:POW? CH2	Measure output power