UC SANTA CRUZ | Baskin Engineering

KEYSIGHT

Keysight A1 Plugin Generation

Team Members:
Shaunveer Gill
Ahmad Joseph
Huy Nguyen
Philip Xie
Madeline Miller

Sponsors and Stakeholders:
Maxim Pletner (Mentor)
Alan Copeland (Mentor)
Brennen DiRenzo (Mentor)
Jeff Dralla (Director)
Ivan Diep (Mentor)

Who is Keysight?

- The leading manufacturer of electronics testing and measurement equipment
- Provides software and hardware to perform measurements and tests

Terminology

- **OpenTAP** - Keysight's open source test automation project; automated testing for devices
- **Plugin** - Software used to communicate between OpenTAP & the instrument being tested.
- **LLM** - Large Language Model; AI that can recognize/generate text; e.g. ChatGPT
- **RAG Approach** - Retrieval Augmented Generation; uses external data sources to provide context that the LLM uses to base its answer on

Current State Of Plugin Generation

- To be able to code a plugin, you need to parse lots of documentation for devices and have a solid understanding of hardware
- Test Engineers lack software experience
- Requires experience with developing plugins and the OpenTAP Software Development Kit
- Time saved creating plugins can be spent testing

KEYSIGHT Test Automation - DemoTestplan1

File Settings Tools View Help

9.20.0

Test Steps

Search...

- Basic Steps
- Flow Control
  - Legacy
    - If Verdict — Add / Add Child
    - Lock — Add / Add Child
    - Parallel — Add / Add Child
    - Repeat — Add / Add Child
    - Sequence — Add / Add Child
    - Sweep Parameter — Add / Add Child
    - Sweep Parameter Range — Add / Add Child
    - Test Plan Reference — Add / Add Child
- OTE demo
  - Access UUT ID

Runs its child steps in parallel in separate threads.

Test Plan *DemoTestplan1*

Test Plan: Completed in 499 ms

Name	Verdict	Duration	Flow	Type
Powerup	Pass	75.8 ms		OTE demo \ Numeric Teststep
Firmware check	Pass	30.6 ms		OTE demo \ String Teststep
Output power	Pass	327 ms		Flow Control \ Sequence
Power Channel 1	Pass	100 ms		OTE demo \ Numeric Teststep
Power Channel 2	Pass	75.7 ms		OTE demo \ Numeric Teststep
Power Channel 3	Pass	90.4 ms		OTE demo \ Numeric Teststep
Power Channel 4	Pass	60.3 ms		OTE demo \ Numeric Teststep
Frequency error	Pass	50.3 ms		OTE demo \ Numeric Teststep

Operator Panel | Test Plan

Test Step Settings

Scale Result	0.1
Step time (mS)	100
OTE	
Test info	
Test class	Normal
Unit	dBm
Format	F1
Test condition	
Limits	
Limit checking	LowerUpper
Upper Limit	25.000
Lower Limit	22.000
Step Result	24.081
Step Verdict	Pass
Common	
Enabled	☑
Step Name	Power Channel 1
Break Conditions	☐ Break on E

Log

☑ Errors 0 ☑ Warnings 0 ☑ Information 29 ☐ Debug 18 Sources Search Filter ✓ Auto Scroll

```
17:50:12.804  Summary   -------- Summary of test plan started 07/19/2023 17:50:12 --------
17:50:12.804  Summary    Powerup                                   75.8 ms  Pass
17:50:12.804  Summary    Firmware check                            30.6 ms  Pass
17:50:12.804  Summary    Output power                               327 ms  Pass
17:50:12.804  Summary      Power Channel 1                          100 ms  Pass
17:50:12.804  Summary      Power Channel 2                         75.7 ms  Pass
17:50:12.804  Summary      Power Channel 3                         90.3 ms  Pass
17:50:12.804  Summary      Power Channel 4                         60.2 ms  Pass
17:50:12.804  Summary    Frequency error                           50.3 ms  Pass
```

Example of Power Supply & OpenTAP

Example Plugin

```python
from opentap import *
from System import Double, String
import OpenTap
import time

@attribute(OpenTap.Display("Infiniium", "A basic example of a SCPI instrument driver.", "Infiniium"))
class Scope(OpenTap.ScpiInstrument):

    def __init__(self):
        super(Scope, self).__init__()
        self.log = Trace(self)
        self.Name = "Infiniium"

    def GetIdnString(self):
        a = self.ScpiQuery[String]("*IDN?")
        return a

    def reset(self):
        self.normalSCPI(":SYSTem:PRESet FACTory")

    def Setup(self, WfmPosPath, WfmNegPath ):
        self.normalSCPI(":CHANnel1:DISPlay OFF")
        self.normalSCPI(f':DISK:LOAD "{WfmPosPath}", WMEMory1, INT16')
        self.normalSCPI(":WMEMory1:DISPlay ON")
        # self.normalSCPI(f':DISK:DELete "{WfmPosPath}"')
        self.normalSCPI(f':DISK:LOAD "{WfmNegPath}", WMEMory2, INT16')
        self.normalSCPI(":WMEMory2:DISPlay ON")
        # self.normalSCPI(f':DISK:DELete "{WfmNegPath}"')
        self.normalSCPI(":TIMebase:REFerence:PERCent 25")
        self.normalSCPI(":TIMebase:RANGe 1e-05")
        self.normalSCPI(":FUNCtion1:SUBTract WMEMory1, WMEMory2")
        self.normalSCPI(":DISPlay:MAIN OFF, WMEMory1")
        self.normalSCPI(":DISPlay:MAIN OFF, WMEMory2")
```

Check for OpenTAP syntax

Inherit from the OpenTAP Instrument class

Controlling the instrument, such as turning off the display

Demo

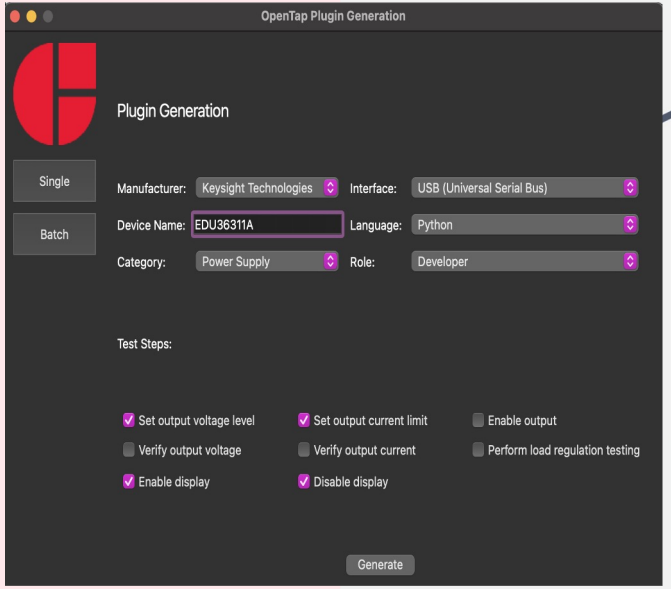
Project Goals

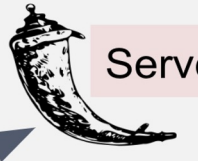
- Reduce development time for OpenTAP plugins
- Automate testing of generated plugins to ensure minimum quality standards are met

OpenTAP

Architecture

Step 1 - User inputs their device specifications





Server

Step 2 - Request sent to server. Context is pulled from database, and LLMs are called to produce and verify the plugin.

Step 3 - AI-produced plugin is integrated with OpenTAP.

Step 4 - The plugin enables interaction between the device and computer through OpenTAP allowing the user to control the power supply, such as turning it on or off.

1st LLM generates plugin and 2nd LLM verifies plugin

Both LLMs access a database with PDF docs and pull relevant content to generate its response.

Step 1 - 1st LLM generates plugin code.

Step 2 - The 2nd LLM verifies the code generated by the 1st LLM.

If plugin is **incorrect**

Step 3 - If the 2nd LLM deems the plugin is incorrect, repeat steps 1 and 2, and then go to step 4.

If plugin is **correct**

Step 4 -

If the 2nd LLM deems the plugin as correct, it's returned to the user.

If it is the second attempt and still incorrect, it's returned to the user with a description of why the plugin is incorrect.

Challenges

- **Prompt Engineering:** Creating LLM prompts to return quality plugin was incredibly difficult
- **Gathering Data:** Relevant instrument documentation related to Python was hard to come by
- **Automated testing:** A wide range of tests & use case scenarios needed to be accounted for
- **Updating Technologies**: We had to switch to updated versions of technologies many times

Results

- **Plugin Generation:** Successfully generates plugins for a wide range of Keysight tools and decreased development time from ~48 hours to 8-10 hours, decreasing development time by at least 75%,

- **Streamlined Verification**: Plugins are verified before they are sent back to the user, ensuring only quality plugins are returned (e.g. can compile and can connect to an instrument)

- **Compatibility Assurance:** Plugins produced can be used directly with various Keysight softwares without any additional modifications

Next Steps

- **Expanding C# Plugin Support**: Plan to add support for generating plugins in C# to broaden our capabilities.
- **Fine-Tuning Prompts**: Aim to further refine the prompts used with the LLMs to enhance accuracy and consistency.
- **Enhancing Database Resources**: Expand the LLMs' accessible database with more extensive documentation to improve the quality of the generated plugins.

Acknowledgements

Thank you to our Keysight sponsors!

Maxim Pletner

Alan Copeland

Ivan Diep

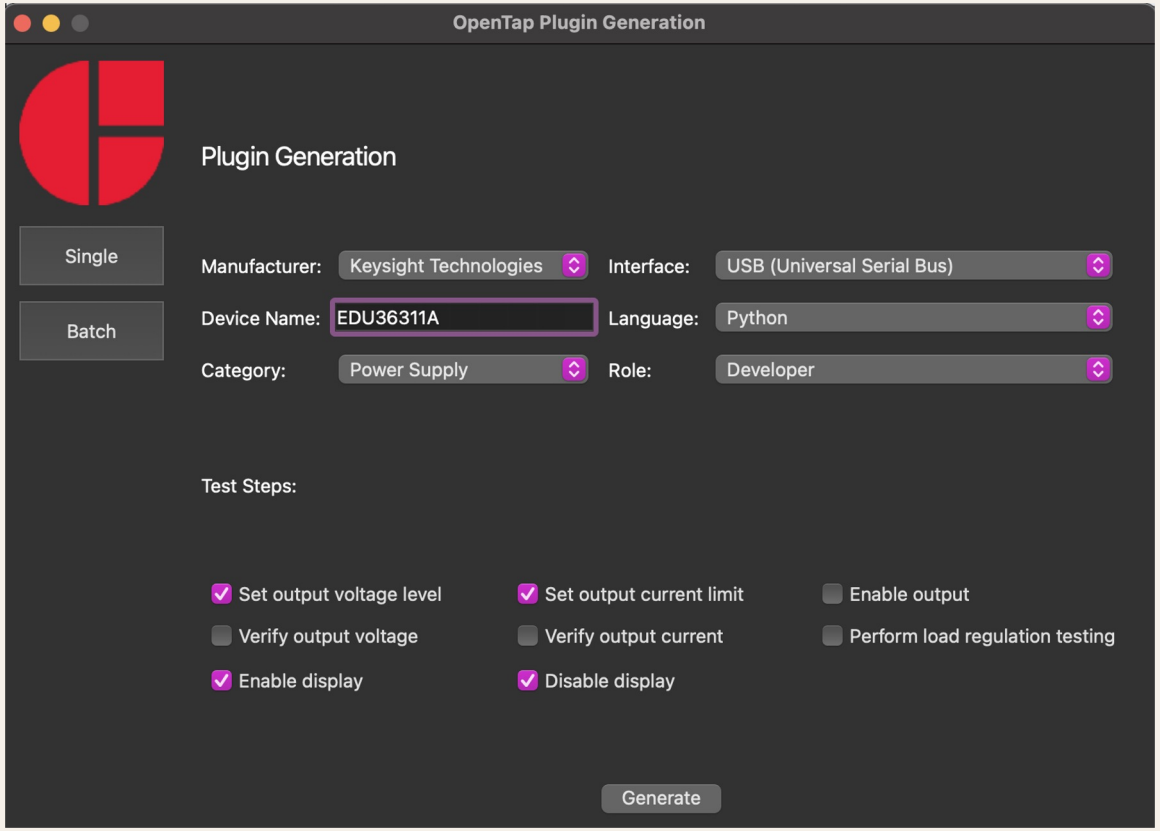
Brennen Direnzo

Jeff Dralla

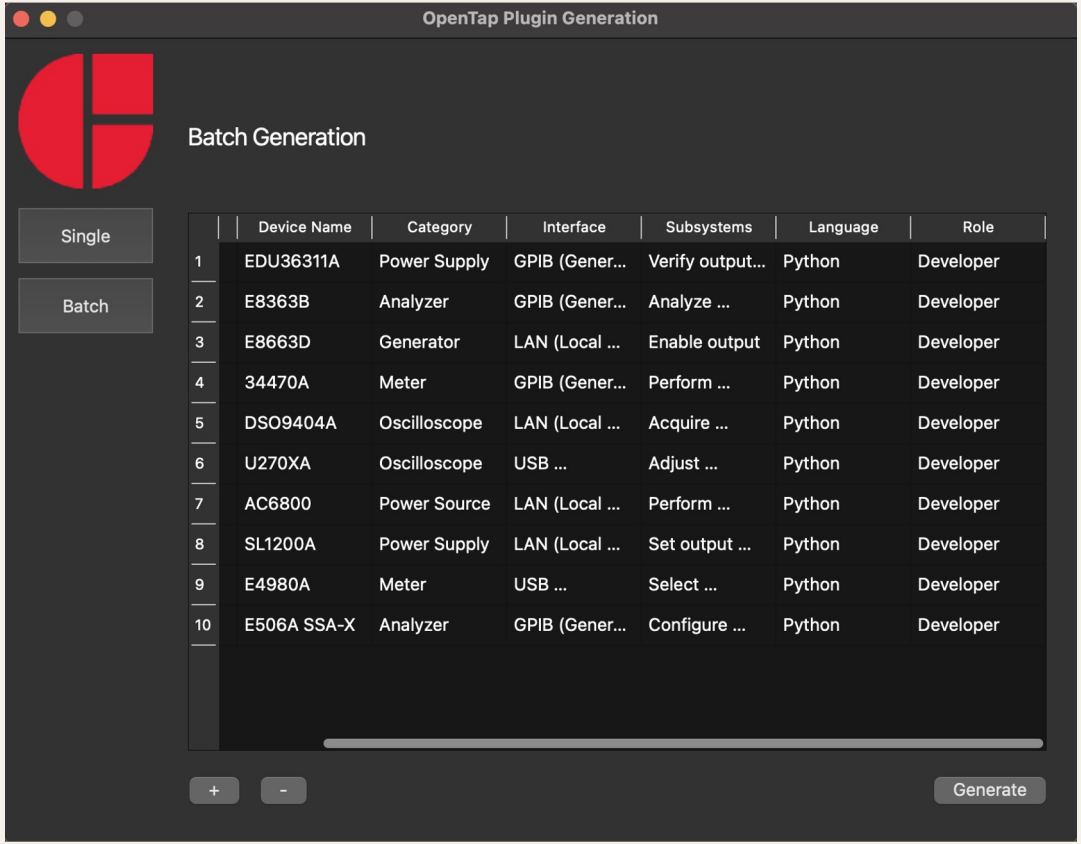
Thank You

Any Questions?

Single Production

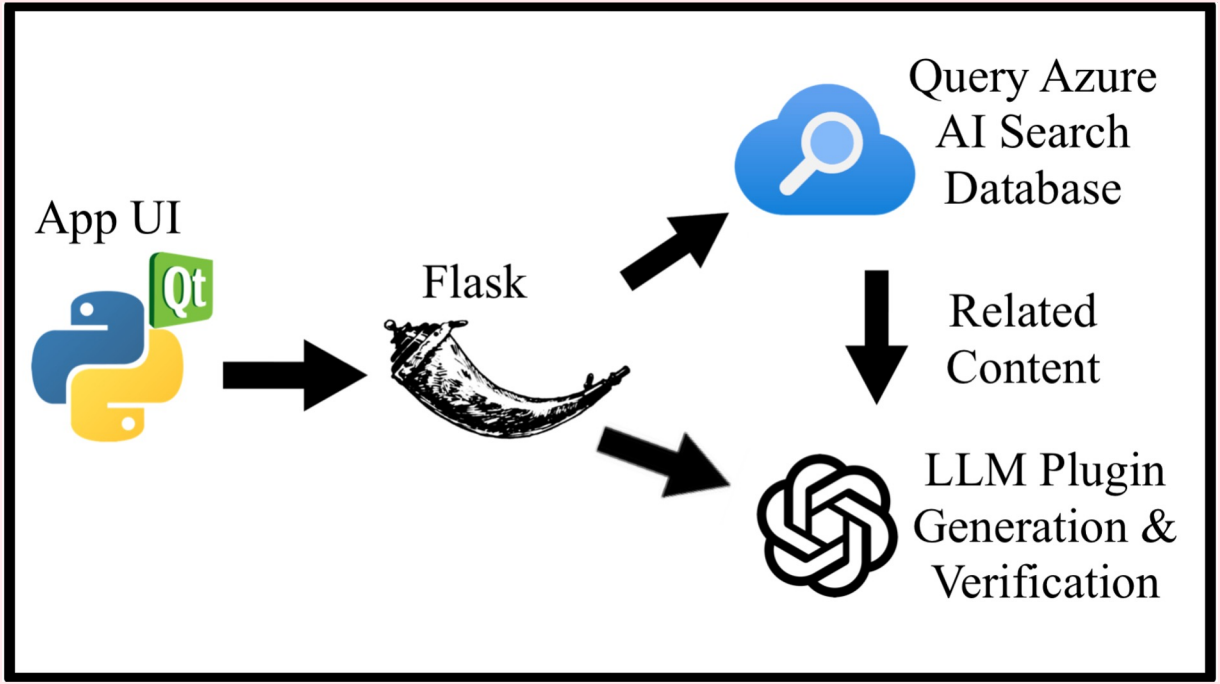


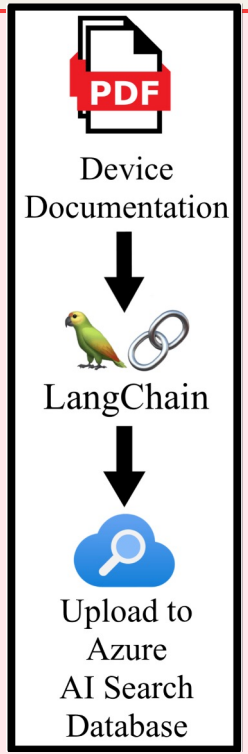
Batch Production



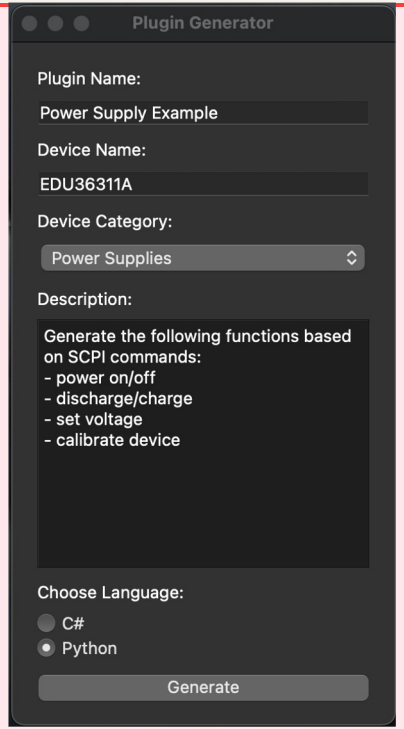
	Device Name	Category	Interface	Subsystems	Language	Role
1	EDU36311A	Power Supply	GPIB (Gener...	Verify output...	Python	Developer
2	E8363B	Analyzer	GPIB (Gener...	Analyze ...	Python	Developer
3	E8663D	Generator	LAN (Local ...	Enable output	Python	Developer
4	34470A	Meter	GPIB (Gener...	Perform ...	Python	Developer
5	DSO9404A	Oscilloscope	LAN (Local ...	Acquire ...	Python	Developer
6	U270XA	Oscilloscope	USB ...	Adjust ...	Python	Developer
7	AC6800	Power Source	LAN (Local ...	Perform ...	Python	Developer
8	SL1200A	Power Supply	LAN (Local ...	Set output ...	Python	Developer
9	E4980A	Meter	USB ...	Select ...	Python	Developer
10	E506A SSA-X	Analyzer	GPIB (Gener...	Configure ...	Python	Developer

Technology Architecture





Technologies Used



Python PyQt5 Frontend

OpenAI's GPT-4

Flask - Runs a server that makes calls to Azure AI Search and LLM

Technologies Used (continued)

Azure AI Search - Queries the database based on keywords found in user's input

LangChain

LangChain - Chunks the PDFs, uploads them to AI Search's database