SUBINITIAL™

stacks

Automated Test, Simplified.

Overview

What We Are Covering

- Part I – Introduction to Subinitial
- Part II – Introduction to Testing
  - Basics of electronics test
  - Basic test implementations
- Part III – Stacks Platform
  - Overview & Features
  - Examples & Demos
- Part IV – Subinitial Python Library
  - Test Framework + Drivers

Part I – Introduction

Who We Are

- About the Presenters:
  - Kyle Howen
    - EE: Embedded Architecture, Software User Experience
  - Scott McClusky
    - EE: High Voltage, Hardware, Power Systems, CAD
- Subinitial, LLC
  - Established 2012
  - 25 years combined Electrical Engineering experience

Part I – Introduction

Who We Are

- What we believe:
  - Automation is awesome
    - It also saves time/money
  - User interface is extremely important to minimize time and effort needed to do a task
  - Quality design pays off
- What we do:
  - Electronics design
  - Automate processes, both software and hardware
  - Produce intuitive and attractive user interfaces

Part I – Introduction

What We Have Made – Quick Look

- Stacks, a modular electronics test and automation platform with features that provide:
  - DMM Measurements
  - Power Switching
  - Serial/Digital Interfacing
  - Web Browser Debug Interface
  - Procedures Scripted in Python


```
stacksConnection = Stacks.NetworkConnection()
aDeck = Analog.Deck(stacksConnection, 2)
rDeck = Relay.Deck(stacksConnection, 4)
cDeck = Core.Deck(stacksConnection, 1)

deck = aDeck
stacksConnection.connect()

aDeck.write('DIO_CONFIG',  0b00000000) #11111100)
aDeck.write('DIO_SET', 0b11111111)
aDeck.printRead('DIO_CONFIG')
aDeck.printRead("DIO_COMMAND")
```

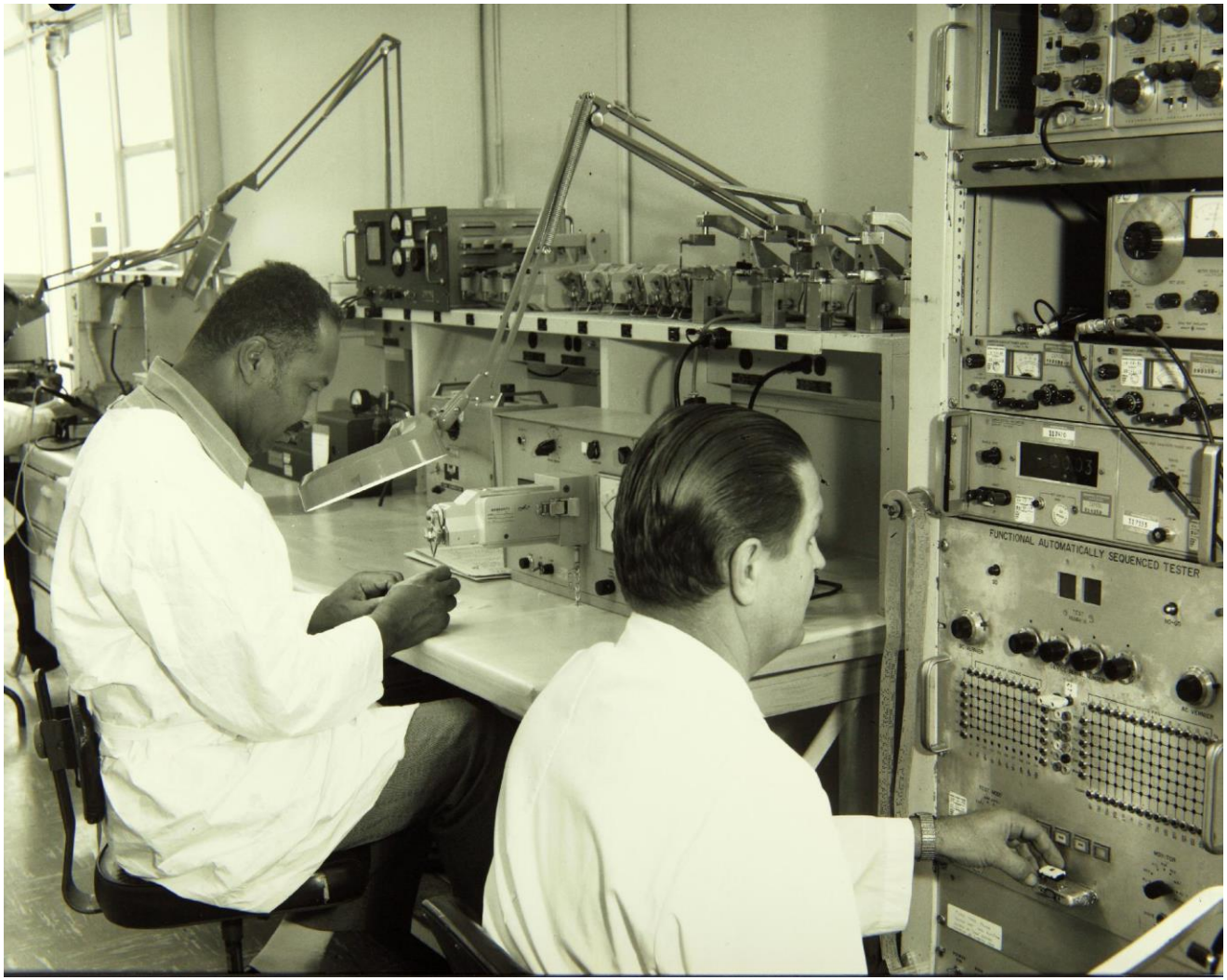
Part I – Introduction

Why We Have Made Stacks

- We are not happy with the current electronics test solutions because they take too much time.
  - Both designing and running the test

- For those of you not familiar with electronics test we'll spend a brief few slides covering it.

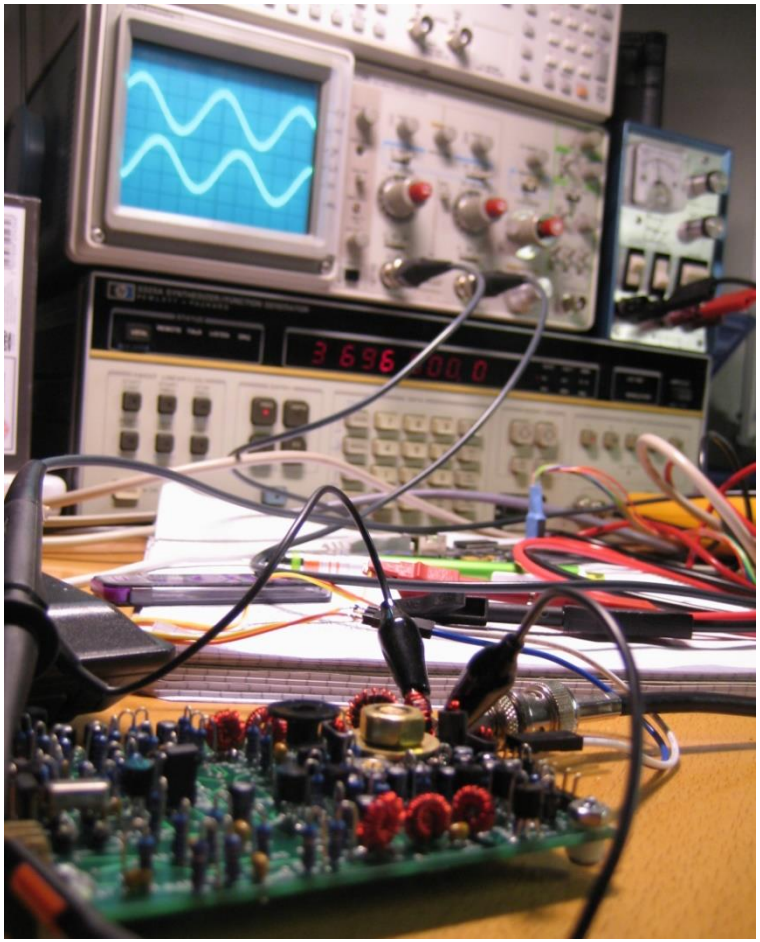
Part II – Intro to Testing



Part II – Intro to Testing

What is Design Verification Test?

- The kind any electrical designer is familiar with
- Used to verify design requirements
- Done once per design, typically by hand
- Before production



Part II – Intro to Testing

What is Production Test?

- Test performed on every unit produced
- Used to verify manufacturing process

Part II – Intro to Testing

Why Test in Production?

- Testing improves **quality** of shipped product
  - Increase Sales through Improved Reputation and Customer Satisfaction
  - Minimize Cost due to Reduced Support and Returns
- How it's done:
  - Test each unit before shipping
  - More thorough tests catch more problems

Part II – Intro to Testing

Why Test in Production?

- What it potentially catches:
  - Process issues
    - i.e. Old solder paste yielding poor wetting, bridging, or incomplete solder joints
  - Incorrect parts
    - i.e. Wrong resistors installed during pick-n-place
    - i.e. Counterfeit parts don't perform up to specs
  - PCB issues
    - i.e. "Open" in PCB trace
  - Design errors
    - i.e. Design didn't account for all component tolerances

Part II – Intro to Testing

Test Options

- Manual Test
  - Technicians / Training
  - Human Errors / Insight
  - Slow manual operation of test equipment
- Automated Test
  - Computer controlled test procedures
    - Using LabVIEW™ or custom software
  - Custom Interface PCBs and test hardware
    - uC / Switches / Arduino
  - Automated test data and reports

Part III – Stacks Platform

Core

Analog

Relay

stacks *Automated Test, Simplified.*

Wireless

Motor

Battery

Part III – Stacks Platform

What Is Stacks?

DUT

- Stacks is a modular electronics test, measurement, and control platform.
  - Tests and measures your designs/DUT (Device Under Test).
  - Controls your test procedure, scripts, and debug panels.
  - Stacks contains measurement devices with many different types of I/O and features.
  - Modules (“Decks”) add more features to stack as needed.
  - Connectivity and interfacing your DUT is made simple with external accessories, such as Tracks and breakout boards.
  - Integrates with your existing test equipment.

Part III – Stacks Platform

Stacks Makes Test Faster

- **Easy Learning Curve** - Intuitive interface that doesn't require extensive training like LabVIEW™, yet still has advanced features for users who need them
- **Writing Tests is Fast** - Prebuilt Python libraries makes scripting a custom test fast
- **No Interface PCBs Required** - Breakout boards are supplied so you don't have to make any interface PCBs
- **Minimized Test Design Time** – Designers and Test Engineers can rapidly develop and change a test even as testing requirements and scope are adjusted.

Part III – Stacks Platform

Stacks Makes Test Cheaper

- **Lower Hardware Costs** - Stacks hardware costs less than comparable test equipment (i.e. DMM, function generator, etc)
- **No Licensing Fees** - all software comes included with the hardware
- **Reduced Labor** - Faster implementations mean labor required to create a test is significantly reduced
- **Reduced Size** - Smaller area required for test equipment

Part III – Stacks Platform

Stacks Replaces Benchtop Test Equipment

- Stacks provides the major functionality of calibrated DMMs, Power Supplies, Arbitrary Wave Generators, and DIO breakouts
- Existing equipment can easily be used in conjunction with Stacks

Part III – Stacks Platform

A Stack consists of a Stacks Core, Decks, and external Accessories

- **Stacks Core:**
  - The Stacks Core provides base functionality, controls all Decks and Accessories with monitoring, and provides power.
  - It is the heart of the system and is required for any Stacks setup.

- **Decks:**
  - Provide extra functionality to the Stack to suit your needs.
  - Typically have a wide array of features.
  - Connect by stacking together with a Core.
  - Any number of Decks may be added, in any order.

- **Accessories:**
  - Are economical components that can be embedded into a test setup to add extra functionality.
  - Typically have narrow feature sets.
  - Communicate with the Core via an external accessory RS485 bus.
  - Can optionally be powered by the Core.

stacks Feature Matrix

Core

16-72Vin or PoE

100Mbps Ethernet

Analog

Relay

Power

Provide

DUT Power Supplies User on/off Ctrl

- **+12V** x1 — 38W. Programmable Current limit up to 3.2A
- **+5V** x1 — 7.5W. Programmable Current limit up to 1.5A
- **+3.3V** x1 — 5W. Programmable Current limit up to 1.5A
- **-12V** x1 — 100mA

- **Power DAC Source/Meter** x1 — 0 to 10V, 1.5A, 3.5 digit
- **Neg Power DAC** x1 — -10 to 0V, 100mA, 12-bit

Switch

- **Relay** x2 — 45V, 100mA, Solid-state
- **Solenoid Drive** x4 — Low-side Switches, 35V, 350mA
- **Relay** x2 — 8A, 250VAC, Fuse Detect, SPDT, w/Current Meter
- **Relay** x2 — 4A, 250VAC, Fuse Detect, DPDT
- **Relay** x1 — 1A, 250VAC, AC only, Solid-state
- **Relay** x1 — 48VDC, 750mA, DC/AC Solid-state

Signal

Sense

- **Power Supply Telemetry** x1 — Core Internal Supply Voltages, Powers, Temps Currents
- **ADC** x4 — 12-bit ENOB, 0 to 2.5V
- **DMM** — 6 digit, 14kSPS, Multichannel; Voltage: ±250V x6; Current: ±200mA x1
- **Power Meter Voltage • Current** — 3.5 digit, ±45V Common Mode; Voltage: ±13.5V x2; Current: ±1.5A x1; Current: ±150mA x1
- **Current Meter** x2 — ±50A, 3.5 digit, 10kSPS, On SPDT relays

Generate

- **±3V DAC** x2 — 12-bit, -3.3 to +3.3V 20mA
- **RGB LED** x1 — User Programmable
- **RGB LED** x1 — User Programmable
- **Waveform Generator** x1 — 16-bit, ±5V Differential DC/Arb Waveform, 1MSPS
- **+5V Ref** x1 — 0.05% Accuracy
- **RGB LED** x1 — User Programmable

Digital

Discrete

- **DIO** x11 — 3.3V CMOS Tolerant to ±20V, MUX'd
- **PWM** x3 — 1.2MHz, +1 Complementary PWM
- **Clocks** x2 — Programmable 32kHz, 1.9MHz to 30MHz
- **Frequency Counter Event Timer** x1 — 32-bit, Max 30MHz
- **DIO** x8 — 5V or 3.3V CMOS Programmable
- **Isolated Input** x2 — Isolation ±100V, Vth = 5V, Max 32V
- **Isolated Input** x2 — Isolation 250VAC, Vth = 3.3V, Max = 250VAC

Comms

- **USB** x1 — USB 2.0 Device
- **CAN** x1 — 1Mbps, PHY included
- **I²C** x1 — 400kbps, master or slave
- **RS485** x1 — 15Mbps Max
- **UART** x1 — 7.5Mbaud Max
- **SPI** x1 — 40Mhz, Master, 3 CS
- **SD Card** x1 — Micro-SD
- **USB** x1 — USB 2.0 Device
- **USB** x1 — USB 2.0 Device

Tracks

Accessory

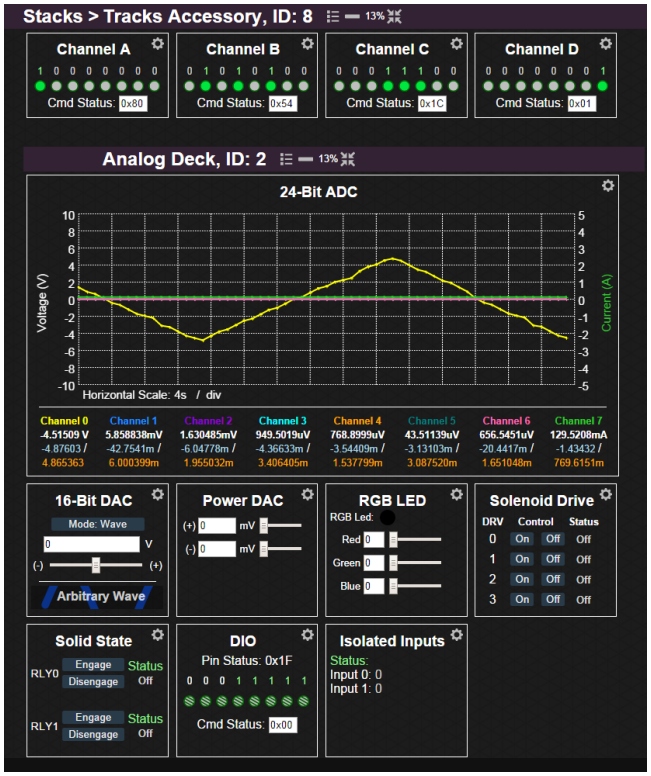
Power/Signal Switch

- **Relay MUX** — 32 Channel, Full isolation. 200V, 1A Signal and Power Routing

Part III – Stacks Platform

Web Browser Control / Debug Panel

- Each stack hosts it's own website.
- Each deck has a collection of feature cards.
- Each feature card hosts its own control and status.
- All decks, and feature cards have a "Details" icon that provides descriptions, scripting examples, configuration controls, and more.



Web Browser Screenshot

Part III – Stacks Platform

Test Procedures & Programming Interface

- The Stacks Python library neatly wraps up TCP Modbus register requests to all Decks.
- For Example:

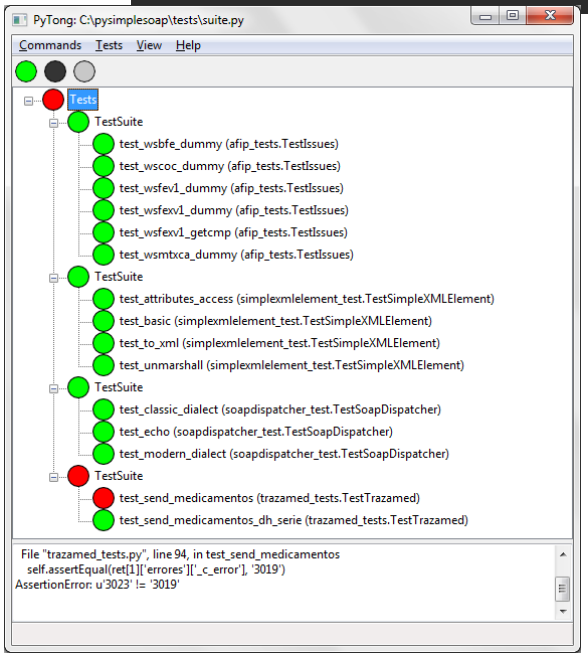
```
results = analogdeck0.read(DMM_VCH0)
analogDeck1.write(WAVEFORM_DC, 0x1234)
```

- Tests can be written as simple python programs, tailored to individual needs, run on virtually any platform, and output data in a wide variety of formats (Excel included)

```
stacksConnection = Stacks.NetworkConnection()
aDeck = Analog.Deck(stacksConnection, 2)
rDeck = Relay.Deck(stacksConnection, 4)
cDeck = Core.Deck(stacksConnection, 1)

deck = aDeck
stacksConnection.connect()

aDeck.write('DIO_CONFIG',  0b00000000) #11111100)
aDeck.write('DIO_SET', 0b11111111)
aDeck.printRead('DIO_CONFIG')
aDeck.printRead("DIO_COMMAND")
```

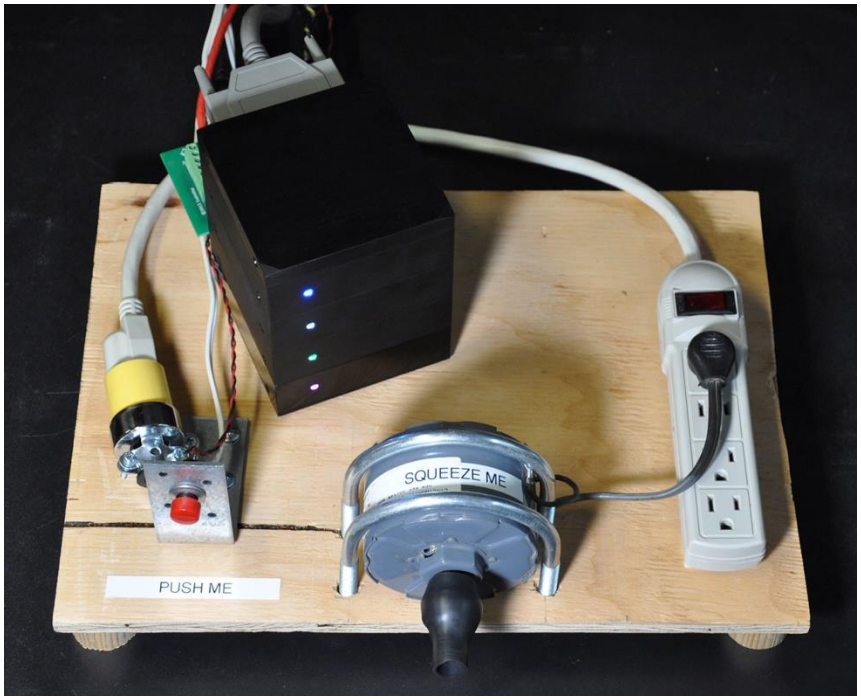


Part III – Stacks Platform

Relay Tactile Control Demo

(Live Demo)

- Detects button press
- Engages on-board relay to power 120VAC motor
- Motor shuts off instantly whenever stalled (i.e. person holding the shaft)
  - Achieved via on-board galvanically isolated relay current sense and programmable trip level

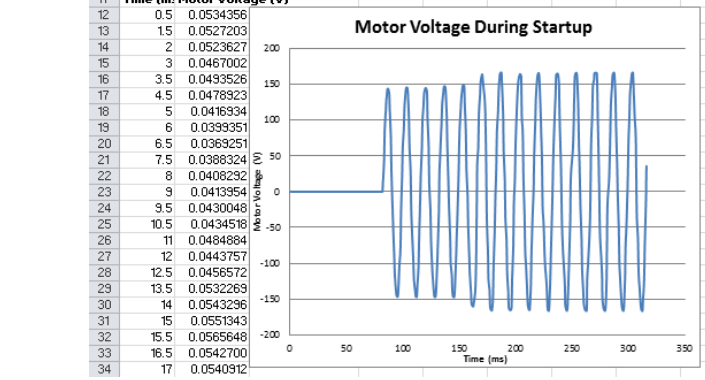


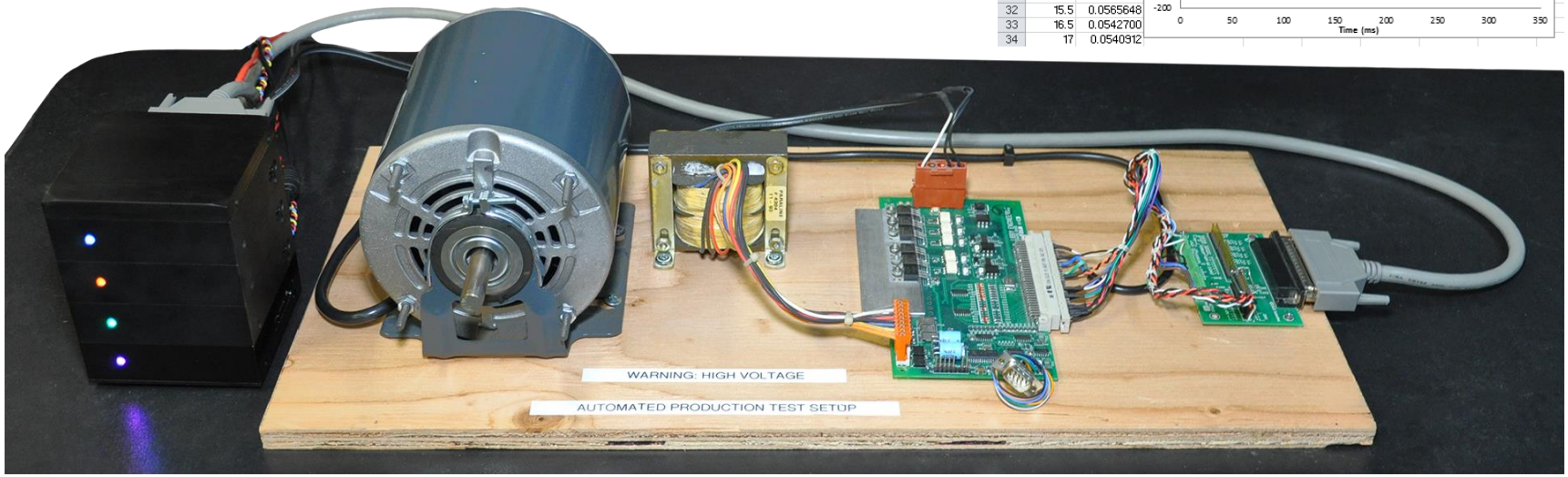
Part III – Stacks Platform

Production Test Demo (Live Demo)

- Functional test of a water center control system
- Automatic spreadsheet output with graph
- Real-time test control and status

	A	B	C
1	**Water Center Motor Driver Test**		
2			
3	**1) DUT Current (mA)**		
4	270		
5	**2) Motor Current (mA)**		
6	6552		
7	**3) Motor Voltage Plot**		
8	**V min**	-166.34	Volts
9	**V max**	166.50	Volts
10	**V pp**	332.83	Volts
11	**Time (ms)**	**Motor Voltage (V)**	
12	0.5	0.0534356	
13	1.5	0.0527203	
14	2	0.0523627	
15	3	0.0467002	
16	3.5	0.0493526	
17	4.5	0.0478923	
18	5	0.0416934	
19	6	0.0399351	
20	6.5	0.0369251	
21	7.5	0.0388324	
22	8	0.0408292	
23	9	0.0413954	
24	9.5	0.0430048	
25	10.5	0.0434518	
26	11	0.0484884	
27	12	0.0443757	
28	12.5	0.0456572	
29	13.5	0.0532269	
30	14	0.0543296	
31	15	0.0551343	
32	15.5	0.0565648	
33	16.5	0.0542700	
34	17	0.0540912	

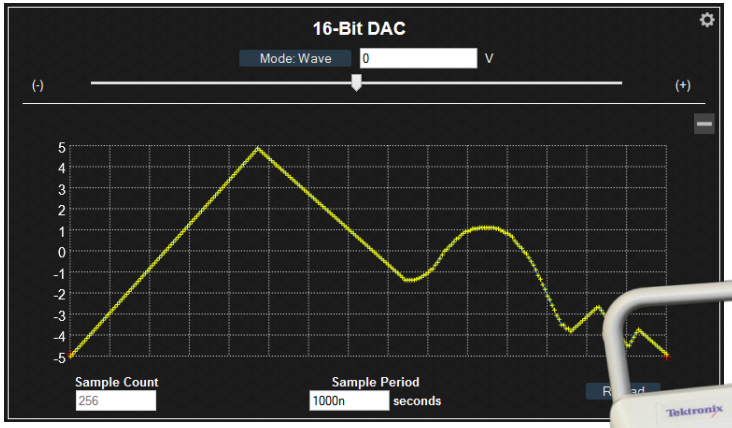




Part III – Stacks Platform

DAC Arbitrary Waveform Demo

(Live Demo)



Web Browser Screenshot

- Draw a waveform in the browser with touchscreen or mouse on-the-fly

- Stacks sources the waveform live

Part III – Stacks Platform

Custom Linux Deck

- Integrates Raspberry Pi™ into the Stack as a standalone PC
- Can run Subinitial Test Framework code, custom software, SSH, all headless
- Can host USB-TMC test equipment
- Enables secure HTTPS connection to Stacks over the internet

Part IV – Subinitial Python Library

Subinitial

python™

Library

Part IV – Subinitial Python Library

The Subinitial Python Library…

- Is an open-source cross-platform software library that runs on a PC workstation
- Facilitates controlling Stacks & other 3rd party test equipment
- Provides a Test Framework for writing and deploying production test procedures

Controlling Test Equipment

- Drivers provided for controlling Stacks
  - Easy to use classes with intuitive methods
    &
  - Low-level register read write access

- 3rd Party Test Equipment Integration
  - Drivers and base-classes provided for LXI and USB-TMC compatible equipment (Tektronix, Agilent, Rigol)

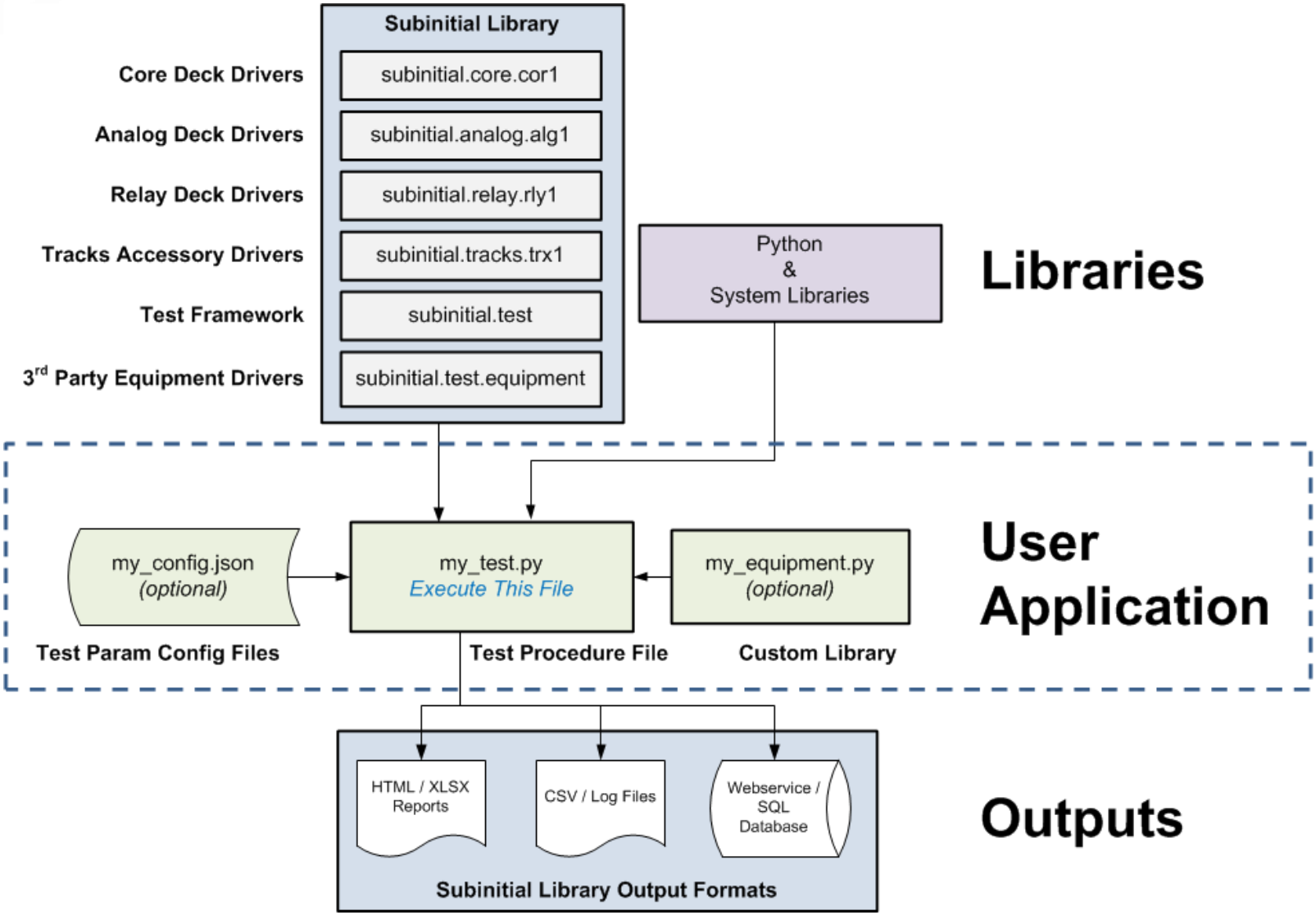
Part IV – Subinitial Python Library

Subinitial Test Framework:

Designing Production Test Procedures

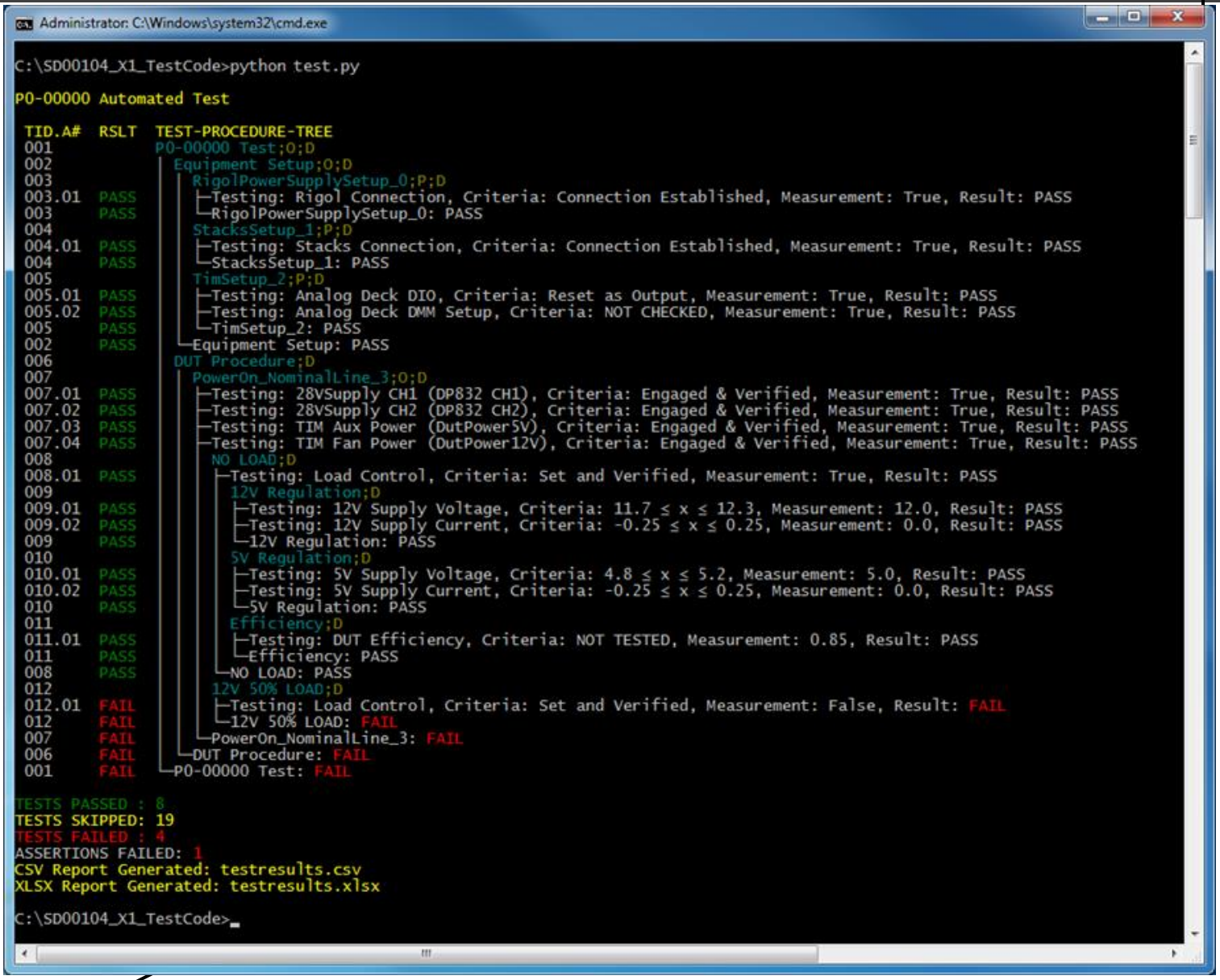
- **PASS/FAIL Tests:**
  - The Subinitial Test Framework hosts a set of tools to develop modular & configurable *TestNodes*
  - *TestNodes* in the Subinitial Test Framework perform a procedure, record results, and return criteria-based PASS/FAIL
- **Test Procedure:**
  - Multiple *TestNodes* can be assembled into a production test procedure with optional hierarchy and contextual dependencies
  - Generic *TestNodes* allow code re-use / reliability
- **Web-Browser or `Command-Line` Interface:**
  - The test technician runs the test procedure via command-line or web-browser interface
  - The test framework facilitates reporting test data and results

Part IV – Subinitial Python Library



Part IV – Subinitial Python Library

Subinitial Test Framework Command-Line Interface



Future Ideas

Future Accessories

Electronic Load

- Provides programmable dynamic and static loading to test power supplies

Isolated Digital Potentiometer

- 4 x isolated digital precision pots for instrumentation

DIO Breakout

- 32-bit bi-directional 3.3V / 5V digital Input / Output

Serial Communications Breakout

- Configurable UART, RS232, RS422, RS485, I2C, SPI for interface testing

Future Decks

Wireless

- Wireless mesh network hub to connect personal area network (PAN) devices and other Stacks decks

Motor

- Equipped with 4 channels of high power output for driving brushless, brushed, AC or DC, and stepper motors

Battery

- 4 lithium ion batteries with charging circuits, diagnostics, and can power an entire stack

Contact

stacks by SUBINITIAL™

More information at subinitial.com

Jimmy Libby – (619) 504-1352
jimmy.libby@subinitial.com

Kyle Howen – (858) 376-7164
kyle.howen@subinitial.com

Scott McClusky – (619) 356-0198
scott.mcclusky@subinitial.com

Attribution/Legal

- **stacks**, the Subinitial logo, and the phrase “Automated Test, Simplified.“ are trademarks of Subinitial LLC.
- LabVIEW™ is a trademark of National Instruments. Neither Subinitial LLC, nor any software programs or other goods or services offered by Subinitial LLC, are affiliated with, endorsed by, or sponsored by National Instruments.
- Raspberry Pi is a trademark of the Raspberry Pi Foundation
- The Python logo is a trademark of the Python Software Foundation
- Images shown below, used in this document, are licensed under Creative Commons Attribution 2.0 Generic
  - Images Retrieved October 2014
  - http://creativecommons.org/licenses/by-sa/2.0/legalcode

By U.S. Naval Forces Central Command/U.S. Fifth Fleet

By Steve

By Juan

By Karl-Martin Skontorp