TMI System

A Manufacturing Test Framework in Python

Ver06

What does it do?

- Provides a framework to develop production test suites
  - Test developers write python code that is called by the framework. For example,
    - load a "manufacturing firmware image" (MFI) to the DUT
    - Send commands to the MFI so that it can test itself
    - Measure external signals, analog/digital
    - Excite the DUT with external input
- A Results Server presenting a Dashboard perspective

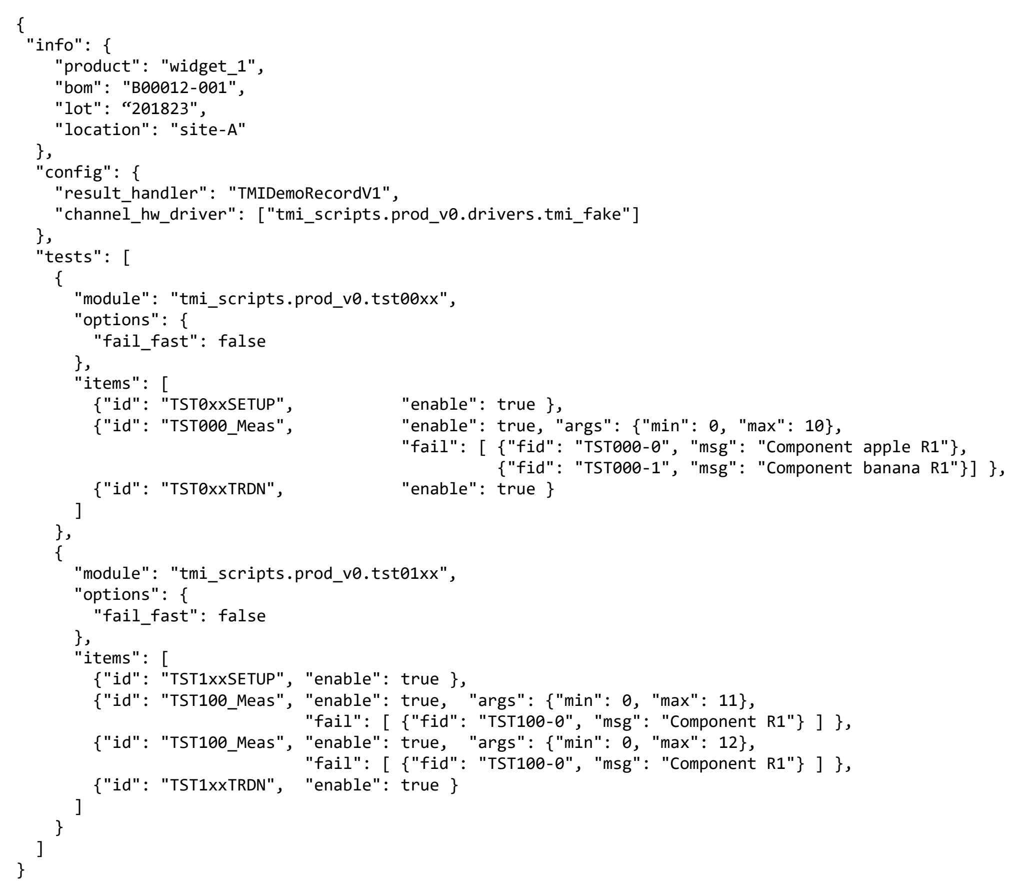
- Capabilities with MicroPython board:
  - Load firmware on the DUT
  - Access the DUT thru I2C, SPI, UART
  - Measure static analog voltages
  - Read/Write GPIOs

TMI System Features

- Low Computer Cost
  - One Linux PC can operate up to 4 DUT Jigs, ~$300/ea
  - Uses single board computer (MicroPython Board) for each Jig, ~$40/Jig
- JSON Test Scripts
  - Human readable, enable/disable tests, change limits
  - Non-programmer can make changes
- JSON Results
  - Human readable, easy to post process
- Result SQL Database
  - Dashboards and queries
- Python Codebase
  - Popular/easy programming
  - Multi-threaded for concurrency
- Traceability
  - Capture serial numbers, lot numbers, and any other identifier information from the DUT
  - All these identifiers go into SQL DB for query later
- Results stored in postgres SQL database which can be located anywhere (local, cloud, etc)

JSON Test Scripts

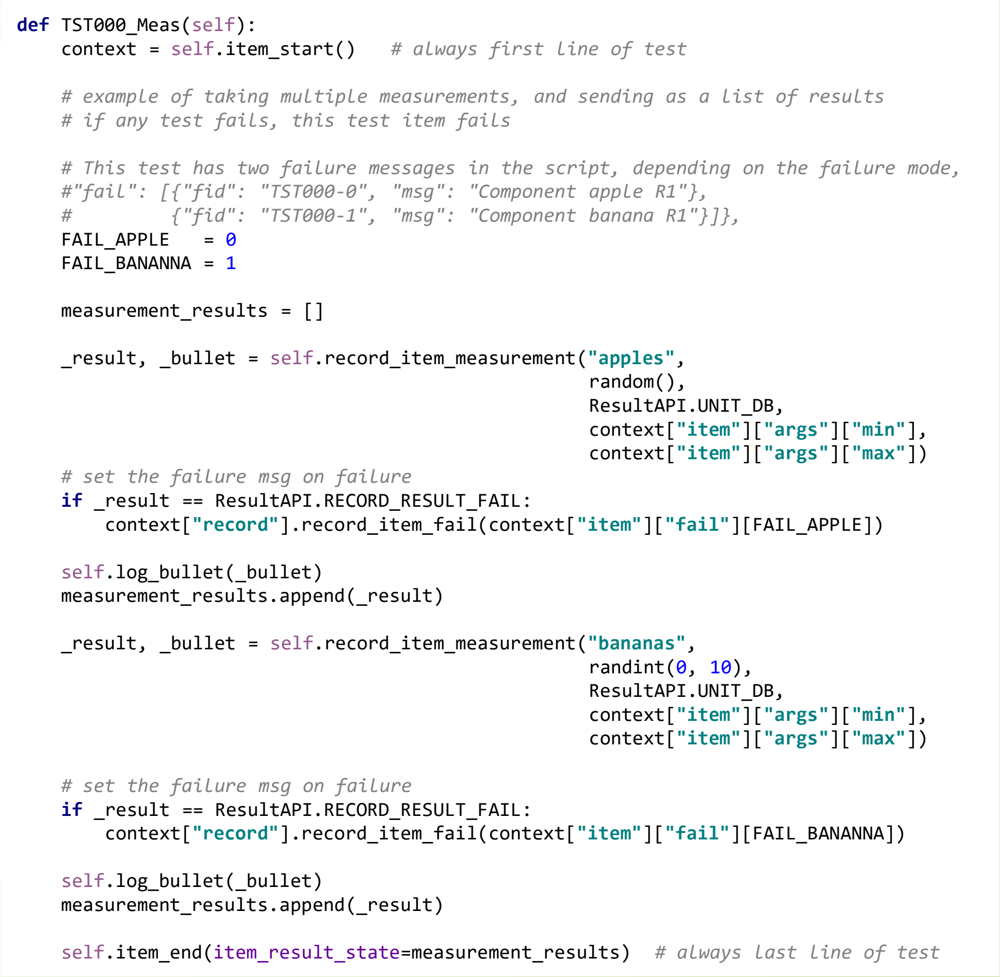
- Human readable
- Drives the test bench
- Each test item as an “id”, which corresponds to python function that implements the test
- Non-programmer can read this file and make changes



```
{
 "info": {
    "product": "widget_1",
    "bom": "B00012-001",
    "lot": “201823",
    "location": "site-A"
  },
  "config": {
    "result_handler": "TMIDemoRecordV1",
    "channel_hw_driver": ["tmi_scripts.prod_v0.drivers.tmi_fake"]
  },
  "tests": [
    {
      "module": "tmi_scripts.prod_v0.tst00xx",
      "options": {
        "fail_fast": false
      },
      "items": [
        {"id": "TST0xxSETUP",           "enable": true },
        {"id": "TST000_Meas",           "enable": true, "args": {"min": 0, "max": 10},
                                        "fail": [ {"fid": "TST000-0", "msg": "Component apple R1"},
                                                  {"fid": "TST000-1", "msg": "Component banana R1"}] },
        {"id": "TST0xxTRDN",            "enable": true }
      ]
    },
    {
      "module": "tmi_scripts.prod_v0.tst01xx",
      "options": {
        "fail_fast": false
      },
      "items": [
        {"id": "TST1xxSETUP", "enable": true },
        {"id": "TST100_Meas", "enable": true,  "args": {"min": 0, "max": 11},
                              "fail": [ {"fid": "TST100-0", "msg": "Component R1"} ] },
        {"id": "TST100_Meas", "enable": true,  "args": {"min": 0, "max": 12},
                              "fail": [ {"fid": "TST100-0", "msg": "Component R1"} ] },
        {"id": "TST1xxTRDN",  "enable": true }
      ]
    }
  ]
}
```

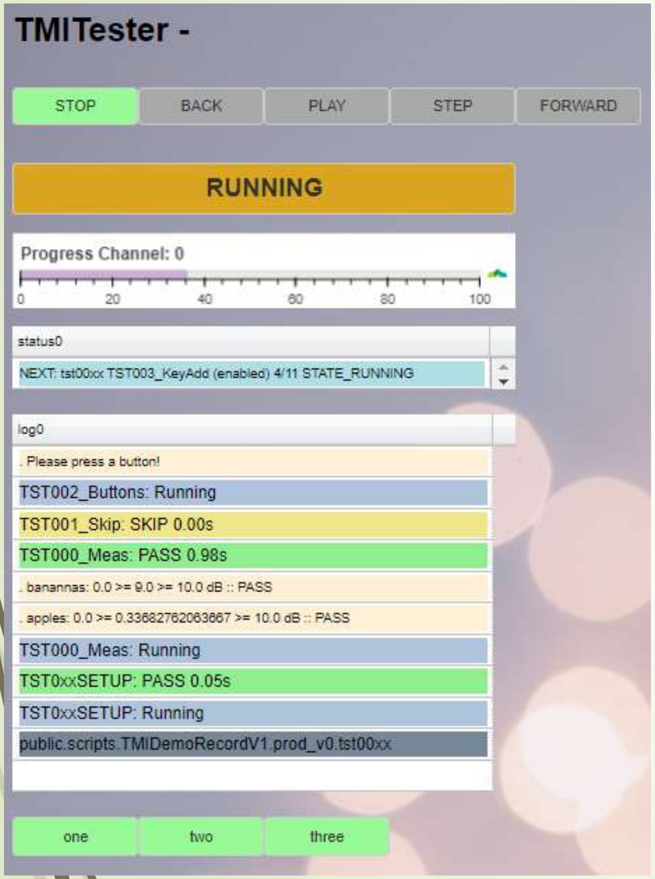
Python Test Code

- Each test item from the JSON script (previous slide), is a python coded function
- APIs to make test driver code easy
  - Save any measurement
  - Get user input (buttons, text entry)
  - Set product keys (ex serial number)
  - Add logs
- NOTE: Not shown in the code snippet is code related to controlling your hardware to make measurements.



```python
def TST000_Meas(self):
    context = self.item_start()   # always first line of test

    # example of taking multiple measurements, and sending as a list of results
    # if any test fails, this test item fails

    # This test has two failure messages in the script, depending on the failure mode,
    #"fail": [{"fid": "TST000-0", "msg": "Component apple R1"},
    #         {"fid": "TST000-1", "msg": "Component banana R1"}]},
    FAIL_APPLE   = 0
    FAIL_BANANNA = 1

    measurement_results = []

    _result, _bullet = self.record_item_measurement("apples",
                                                    random(),
                                                    ResultAPI.UNIT_DB,
                                                    context["item"]["args"]["min"],
                                                    context["item"]["args"]["max"])
    # set the failure msg on failure
    if _result == ResultAPI.RECORD_RESULT_FAIL:
        context["record"].record_item_fail(context["item"]["fail"][FAIL_APPLE])

    self.log_bullet(_bullet)
    measurement_results.append(_result)

    _result, _bullet = self.record_item_measurement("bananas",
                                                    randint(0, 10),
                                                    ResultAPI.UNIT_DB,
                                                    context["item"]["args"]["min"],
                                                    context["item"]["args"]["max"])

    # set the failure msg on failure
    if _result == ResultAPI.RECORD_RESULT_FAIL:
        context["record"].record_item_fail(context["item"]["fail"][FAIL_BANANNA])

    self.log_bullet(_bullet)
    measurement_results.append(_result)

    self.item_end(item_result_state=measurement_results)  # always last line of test
```

Python Test Code – User Buttons



Progress Channel: 0

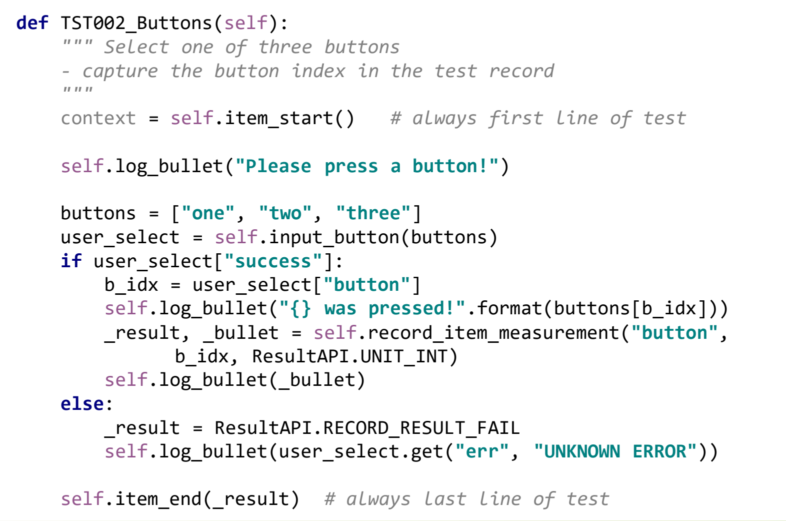
status0

NEXT: tst00xx TST003_KeyAdd (enabled) 4/11 STATE_RUNNING

log0

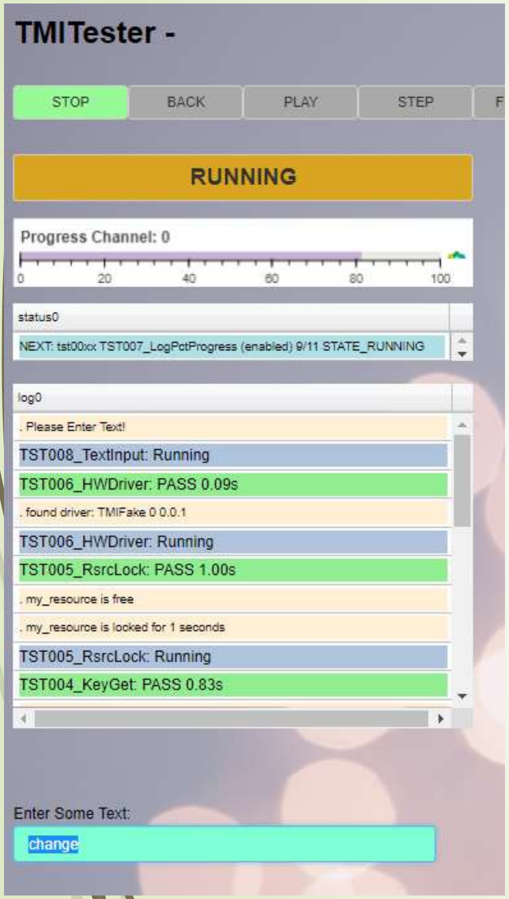
```
. Please press a button!
TST002_Buttons: Running
TST001_Skip: SKIP 0.00s
TST000_Meas: PASS 0.98s
. banannas: 0.0 >= 9.0 >= 10.0 dB :: PASS
. apples: 0.0 >= 0.33682762063667 >= 10.0 dB :: PASS
TST000_Meas: Running
TST0xxSETUP: PASS 0.05s
TST0xxSETUP: Running
public.scripts.TMIDemoRecordV1.prod_v0.tst00xx
```

- When a test requires Operator input, one option is buttons
- Shows is the Test panel presenting three buttons to the Operator
- The code for this button example ->



```python
def TST002_Buttons(self):
    """ Select one of three buttons
    - capture the button index in the test record
    """
    context = self.item_start()   # always first line of test

    self.log_bullet("Please press a button!")

    buttons = ["one", "two", "three"]
    user_select = self.input_button(buttons)
    if user_select["success"]:
        b_idx = user_select["button"]
        self.log_bullet("{} was pressed!".format(buttons[b_idx]))
        _result, _bullet = self.record_item_measurement("button",
              b_idx, ResultAPI.UNIT_INT)
        self.log_bullet(_bullet)
    else:
        _result = ResultAPI.RECORD_RESULT_FAIL
        self.log_bullet(user_select.get("err", "UNKNOWN ERROR"))

    self.item_end(_result)  # always last line of test
```

Python Test Code – Text Input

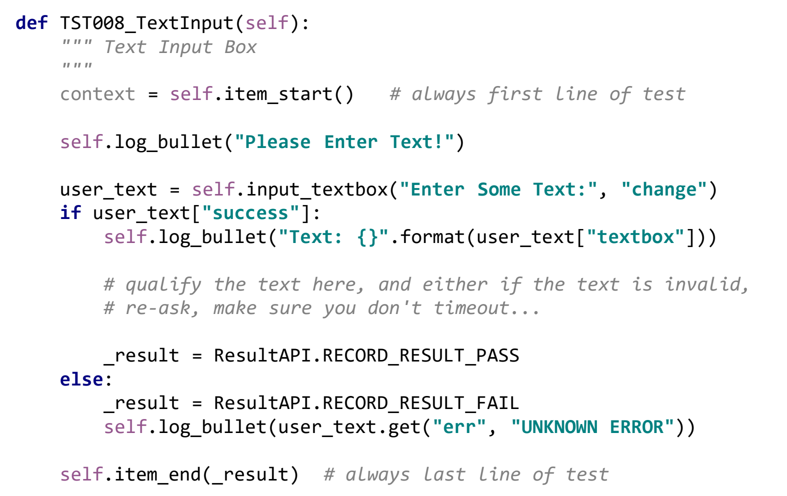


```
NEXT: tst00xx TST007_LogPctProgress (enabled) 9/11 STATE_RUNNING
```

```
. Please Enter Text!
TST008_TextInput: Running
TST006_HWDriver: PASS 0.09s
. found driver: TMIFake 0 0.0.1
TST006_HWDriver: Running
TST005_RsrcLock: PASS 1.00s
. my_resource is free
. my_resource is locked for 1 seconds
TST005_RsrcLock: Running
TST004_KeyGet: PASS 0.83s
```

Enter Some Text: change

- For the case when text input is needed
  - Text input is NEVER a good thing for production environment, too slow and error prone
  - However this input is meant for **Barcode Scanners**, which can output text like a keyboard.
    - For example, scanning lot codes of parts used on a DUT
- The code used for Text Input ->



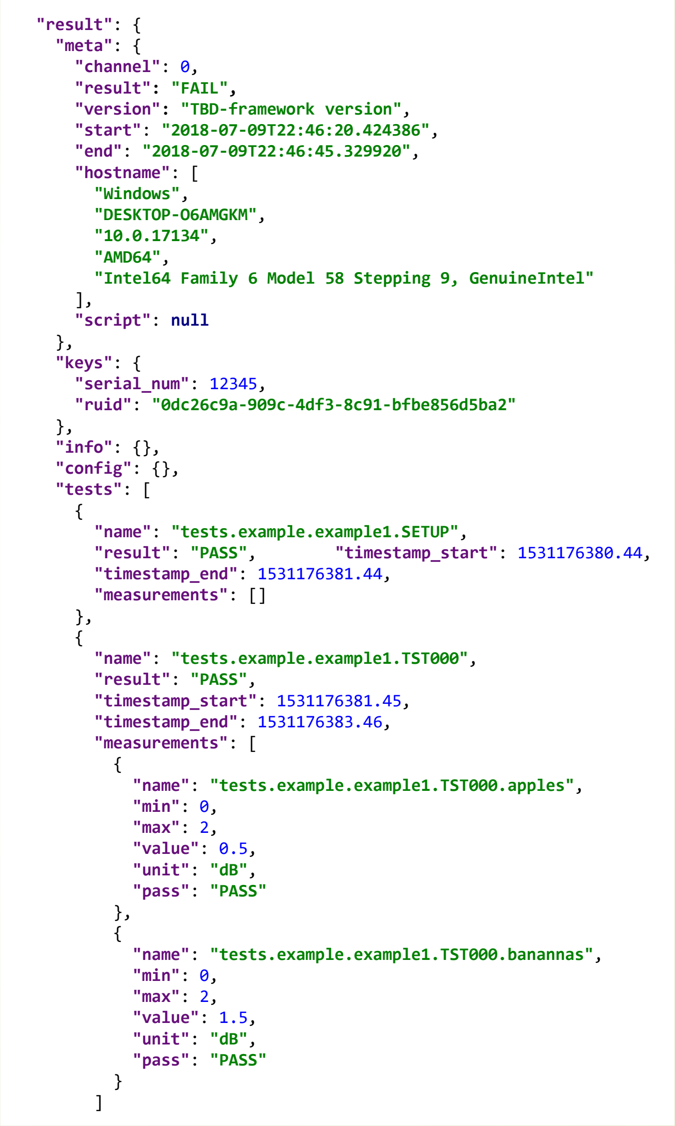
```python
def TST008_TextInput(self):
    """ Text Input Box
    """
    context = self.item_start()   # always first line of test

    self.log_bullet("Please Enter Text!")

    user_text = self.input_textbox("Enter Some Text:", "change")
    if user_text["success"]:
        self.log_bullet("Text: {}".format(user_text["textbox"]))

        # qualify the text here, and either if the text is invalid,
        # re-ask, make sure you don't timeout...

        _result = ResultAPI.RECORD_RESULT_PASS
    else:
        _result = ResultAPI.RECORD_RESULT_FAIL
        self.log_bullet(user_text.get("err", "UNKNOWN ERROR"))

    self.item_end(_result)  # always last line of test
```

JSON Results

- Human readable
- Normalized, all results have the same structure, making it easier to process in a standard way

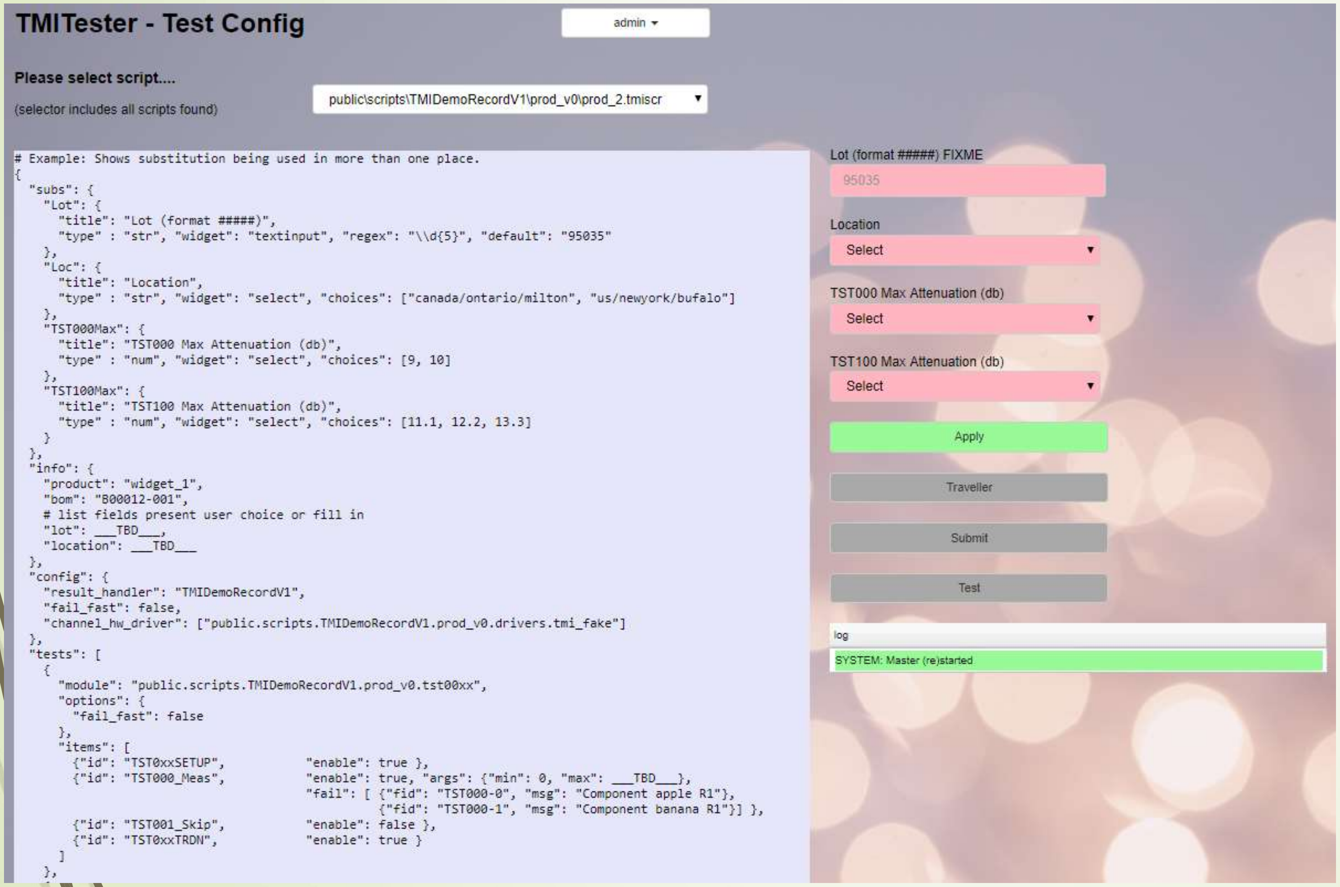
- Each Result has unique RUID
- Measurement data “name” is a full path to the test

- NOTE: Results are encrypted at the test station and sent to the results server. The results server decrypts the results and keeps them stored as backup



```
"result": {
  "meta": {
    "channel": 0,
    "result": "FAIL",
    "version": "TBD-framework version",
    "start": "2018-07-09T22:46:20.424386",
    "end": "2018-07-09T22:46:45.329920",
    "hostname": [
      "Windows",
      "DESKTOP-O6AMGKM",
      "10.0.17134",
      "AMD64",
      "Intel64 Family 6 Model 58 Stepping 9, GenuineIntel"
    ],
    "script": null
  },
  "keys": {
    "serial_num": 12345,
    "ruid": "0dc26c9a-909c-4df3-8c91-bfbe856d5ba2"
  },
  "info": {},
  "config": {},
  "tests": [
    {
      "name": "tests.example.example1.SETUP",
      "result": "PASS",         "timestamp_start": 1531176380.44,
      "timestamp_end": 1531176381.44,
      "measurements": []
    },
    {
      "name": "tests.example.example1.TST000",
      "result": "PASS",
      "timestamp_start": 1531176381.45,
      "timestamp_end": 1531176383.46,
      "measurements": [
        {
          "name": "tests.example.example1.TST000.apples",
          "min": 0,
          "max": 2,
          "value": 0.5,
          "unit": "dB",
          "pass": "PASS"
        },
        {
          "name": "tests.example.example1.TST000.banannas",
          "min": 0,
          "max": 2,
          "value": 1.5,
          "unit": "dB",
          "pass": "PASS"
        }
      ]
```

GUI: Test Configuration (optional)



Please select script....

(selector includes all scripts found)

public\scripts\TMIDemoRecordV1\prod_v0\prod_2.tmiscr ▼

```
# Example: Shows substitution being used in more than one place.
{
  "subs": {
    "Lot": {
      "title": "Lot (format #####)",
      "type" : "str", "widget": "textinput", "regex": "\\d{5}", "default": "95035"
    },
    "Loc": {
      "title": "Location",
      "type" : "str", "widget": "select", "choices": ["canada/ontario/milton", "us/newyork/bufalo"]
    },
    "TST000Max": {
      "title": "TST000 Max Attenuation (db)",
      "type" : "num", "widget": "select", "choices": [9, 10]
    },
    "TST100Max": {
      "title": "TST100 Max Attenuation (db)",
      "type" : "num", "widget": "select", "choices": [11.1, 12.2, 13.3]
    }
  },
  "info": {
    "product": "widget_1",
    "bom": "B00012-001",
    # list fields present user choice or fill in
    "lot": ___TBD___,
    "location": ___TBD___
  },
  "config": {
    "result_handler": "TMIDemoRecordV1",
    "fail_fast": false,
    "channel_hw_driver": ["public.scripts.TMIDemoRecordV1.prod_v0.drivers.tmi_fake"]
  },
  "tests": [
    {
      "module": "public.scripts.TMIDemoRecordV1.prod_v0.tst00xx",
      "options": {
        "fail_fast": false
      },
      "items": [
        {"id": "TST0xxSETUP",          "enable": true },
        {"id": "TST000_Meas",          "enable": true, "args": {"min": 0, "max": ___TBD___},
                                       "fail": [ {"fid": "TST000-0", "msg": "Component apple R1"},
                                                 {"fid": "TST000-1", "msg": "Component banana R1"}] },
        {"id": "TST001_Skip",          "enable": false },
        {"id": "TST0xxTRDN",           "enable": true }
      ]
    },
```

Lot (format #####) FIXME: 95035

Location: Select ▾

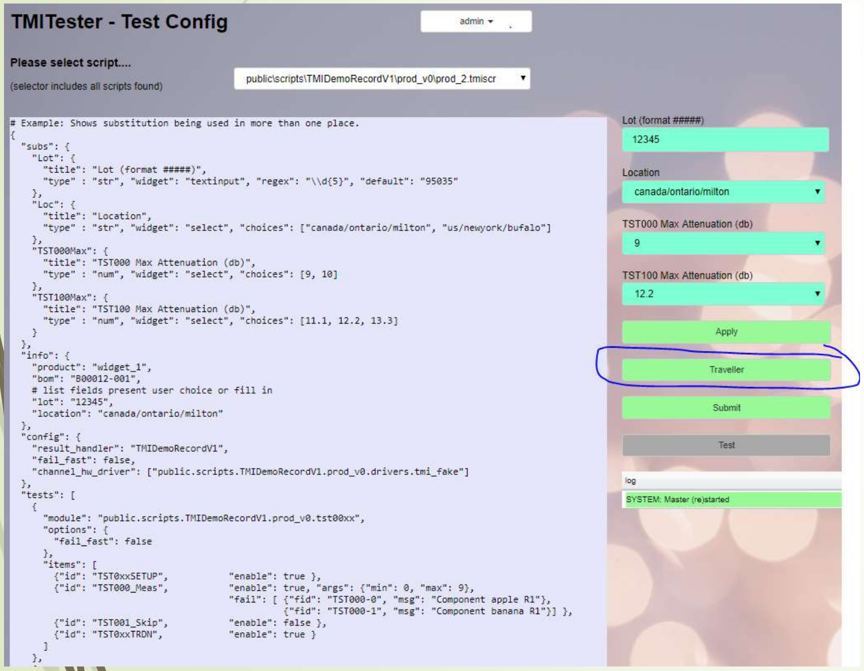
TST000 Max Attenuation (db): Select ▾

TST100 Max Attenuation (db): Select ▾

SYSTEM: Master (re)started

- Scripts "subs" section can define items that are to be set at test time
- For example,
  - Lot Number
  - Location
  - Measurement limits
- Definition controls user options
- Regex patterns are also supported for text entry fields

GUI: Test Configuration Create Barcode (optional)



TMITester - Test Config

admin ▾

Please select script....

(selector includes all scripts found)

public\scripts\TMIDemoRecordV1\prod_v0\prod_2.tmiscr ▾

```
# Example: Shows substitution being used in more than one place.
{
  "subs": {
    "Lot": {
      "title": "Lot (format #####)",
      "type" : "str", "widget": "textinput", "regex": "\\d{5}", "default": "95035"
    },
    "Loc": {
      "title": "Location",
      "type" : "str", "widget": "select", "choices": ["canada/ontario/milton", "us/newyork/bufalo"]
    },
    "TST000Max": {
      "title": "TST000 Max Attenuation (db)",
      "type" : "num", "widget": "select", "choices": [9, 10]
    },
    "TST100Max": {
      "title": "TST100 Max Attenuation (db)",
      "type" : "num", "widget": "select", "choices": [11.1, 12.2, 13.3]
    }
  },
  "info": {
    "product": "widget_1",
    "bom": "B00012-001",
    # list fields present user choice or fill in
    "lot": "12345",
    "location": "canada/ontario/milton"
  },
  "config": {
    "result_handler": "TMIDemoRecordV1",
    "fail_fast": false,
    "channel_hw_driver": ["public.scripts.TMIDemoRecordV1.prod_v0.drivers.tmi_fake"]
  },
  "tests": [
    {
      "module": "public.scripts.TMIDemoRecordV1.prod_v0.tst00xx",
      "options": {
        "fail_fast": false
      },
      "items": [
        {"id": "TST0xxSETUP",           "enable": true },
        {"id": "TST000_Meas",           "enable": true, "args": {"min": 0, "max": 9},
                                        "fail": [ {"fid": "TST000-0", "msg": "Component apple R1"},
                                                  {"fid": "TST000-1", "msg": "Component banana R1"}] },
        {"id": "TST001_Skip",           "enable": false },
        {"id": "TST0xxTRDN",            "enable": true }
      ]
    },
```

Lot (format #####)

12345

Location

canada/ontario/milton ▾

TST000 Max Attenuation (db)

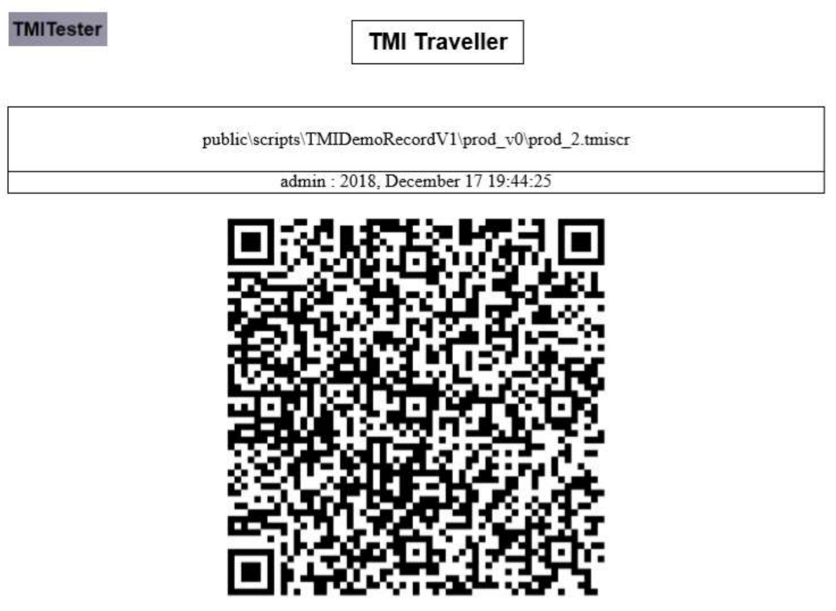
9 ▾

TST100 Max Attenuation (db)

12.2 ▾

log

SYSTEM: Master (re)started



TMITester

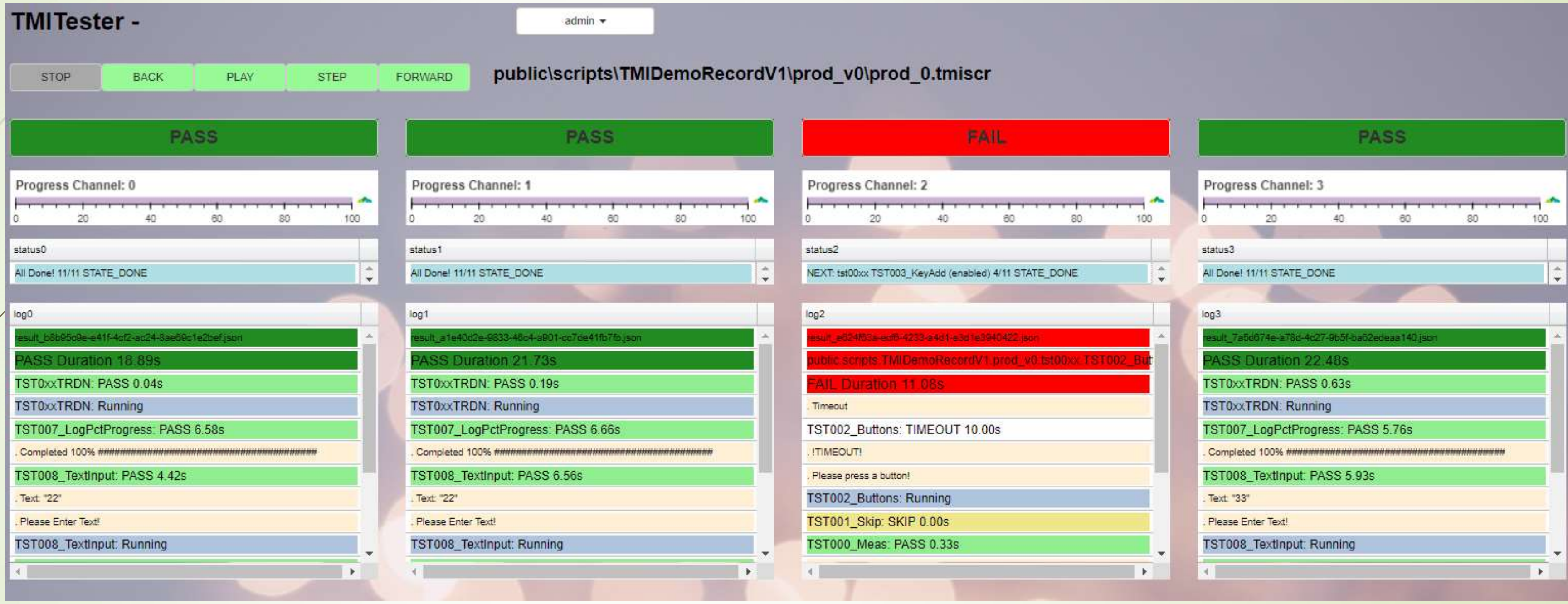
TMI Traveller

public\scripts\TMIDemoRecordV1\prod_v0\prod_2.tmiscr

admin : 2018, December 17 19:44:25

- Travellers can be made to capture all subs variables, and then scanned on the production floor
  - No manual entry by test operators

GUI: Testing Panel



public\scripts\TMIDemoRecordV1\prod_v0\prod_0.tmiscr

status0

```
All Done! 11/11 STATE_DONE
```

log0

```
result_b8b95c9e-e41f-4cf2-ac24-8ae69c1e2bef.json
PASS Duration 18.89s
TST0xxTRDN: PASS 0.04s
TST0xxTRDN: Running
TST007_LogPctProgress: PASS 6.58s
. Completed 100% ##########################################
TST008_TextInput: PASS 4.42s
. Text: "22"
. Please Enter Text!
TST008_TextInput: Running
```

status1

```
All Done! 11/11 STATE_DONE
```

log1

```
result_a1e40d2e-9833-46c4-a901-cc7de41fb7fb.json
PASS Duration 21.73s
TST0xxTRDN: PASS 0.19s
TST0xxTRDN: Running
TST007_LogPctProgress: PASS 6.66s
. Completed 100% ##########################################
TST008_TextInput: PASS 6.56s
. Text: "22"
. Please Enter Text!
TST008_TextInput: Running
```

status2

```
NEXT: tst00xx TST003_KeyAdd (enabled) 4/11 STATE_DONE
```

log2

```
result_e624f63a-edf8-4233-a4d1-e3d1e3940422.json
public.scripts.TMIDemoRecordV1.prod_v0.tst00xx.TST002_But
FAIL Duration 11.08s
. Timeout
TST002_Buttons: TIMEOUT 10.00s
. !TIMEOUT!
. Please press a button!
TST002_Buttons: Running
TST001_Skip: SKIP 0.00s
TST000_Meas: PASS 0.33s
```

status3

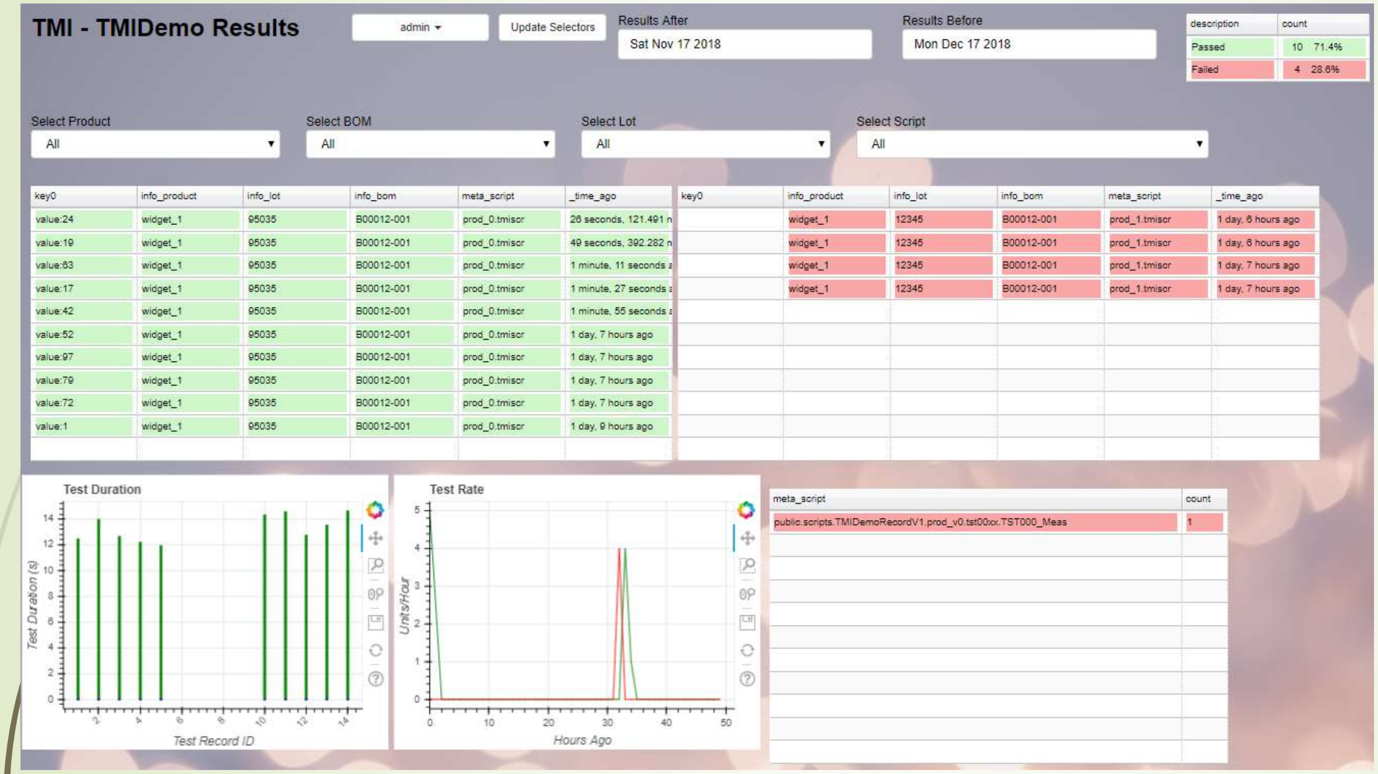
```
All Done! 11/11 STATE_DONE
```

log3

```
result_7a6d674e-a78d-4c27-9b5f-ba62edeaa140.json
PASS Duration 22.48s
TST0xxTRDN: PASS 0.63s
TST0xxTRDN: Running
TST007_LogPctProgress: PASS 5.76s
. Completed 100% ##########################################
TST008_TextInput: PASS 5.93s
. Text: "33"
. Please Enter Text!
TST008_TextInput: Running
```

- 4 Channels are shown
- Each channel is an independent thread

Result Server Dashboard



TMI - TMIDemo Results

admin ▾ | Update Selectors

Results After: Sat Nov 17 2018

Results Before: Mon Dec 17 2018

description	count
Passed	10 71.4%
Failed	4 28.6%

Select Product: All ▾

Select BOM: All ▾

Select Lot: All ▾

Select Script: All ▾

key0	info_product	info_lot	info_bom	meta_script	_time_ago
value:24	widget_1	95035	B00012-001	prod_0.tmiscr	26 seconds, 121.491 n
value:19	widget_1	95035	B00012-001	prod_0.tmiscr	49 seconds, 392.282 n
value:63	widget_1	95035	B00012-001	prod_0.tmiscr	1 minute, 11 seconds a
value:17	widget_1	95035	B00012-001	prod_0.tmiscr	1 minute, 27 seconds a
value:42	widget_1	95035	B00012-001	prod_0.tmiscr	1 minute, 55 seconds a
value:52	widget_1	95035	B00012-001	prod_0.tmiscr	1 day, 7 hours ago
value:97	widget_1	95035	B00012-001	prod_0.tmiscr	1 day, 7 hours ago
value:79	widget_1	95035	B00012-001	prod_0.tmiscr	1 day, 7 hours ago
value:72	widget_1	95035	B00012-001	prod_0.tmiscr	1 day, 7 hours ago
value:1	widget_1	95035	B00012-001	prod_0.tmiscr	1 day, 9 hours ago

key0	info_product	info_lot	info_bom	meta_script	_time_ago
	widget_1	12345	B00012-001	prod_1.tmiscr	1 day, 6 hours ago
	widget_1	12345	B00012-001	prod_1.tmiscr	1 day, 6 hours ago
	widget_1	12345	B00012-001	prod_1.tmiscr	1 day, 7 hours ago
	widget_1	12345	B00012-001	prod_1.tmiscr	1 day, 7 hours ago

meta_script	count
public.scripts.TMIDemoRecordV1.prod_v0.tst00xx.TST000_Meas	1

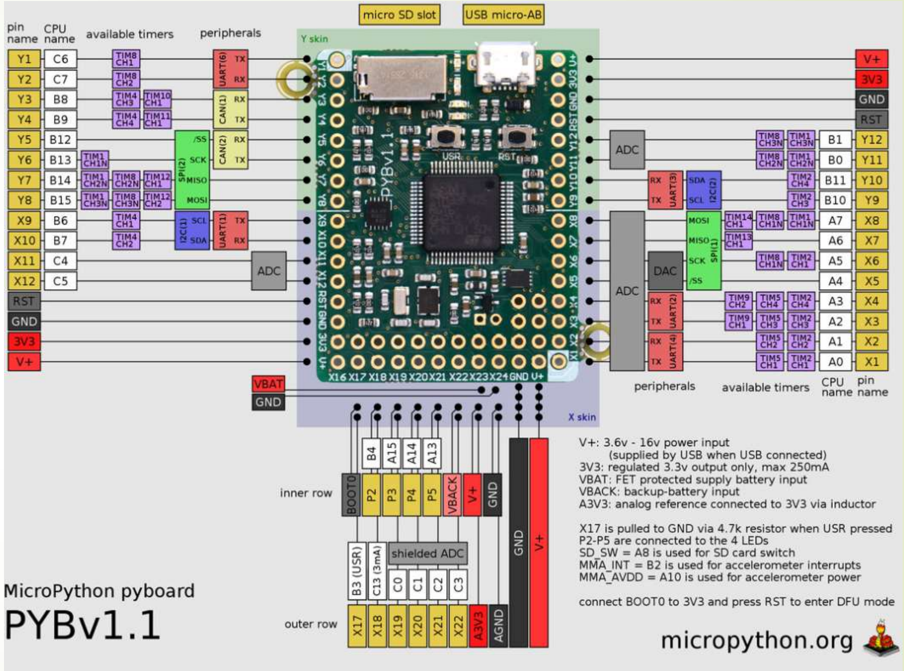
- Top Row selectors to sort data
  - Can view data from a single LOT#
- Select row of Pass/Fail tables to bring up the test record details
- Summary Tables
  - Results reflect Selector settings
  - Pass/Fail Counts
  - Top failed tests and counts

- Graphs to show trends
  - Test Duration and Pass/Fail Hourly Rate

MicroPython Test Instrument

- Low Cost
- Read/Write GPIOs
- ADC
- Proxy for Serial, I2C, SPI commands

- NOTE: The MicroPython board may not be suitable for your application. This board is used to demonstrate and develop the features of the framework.



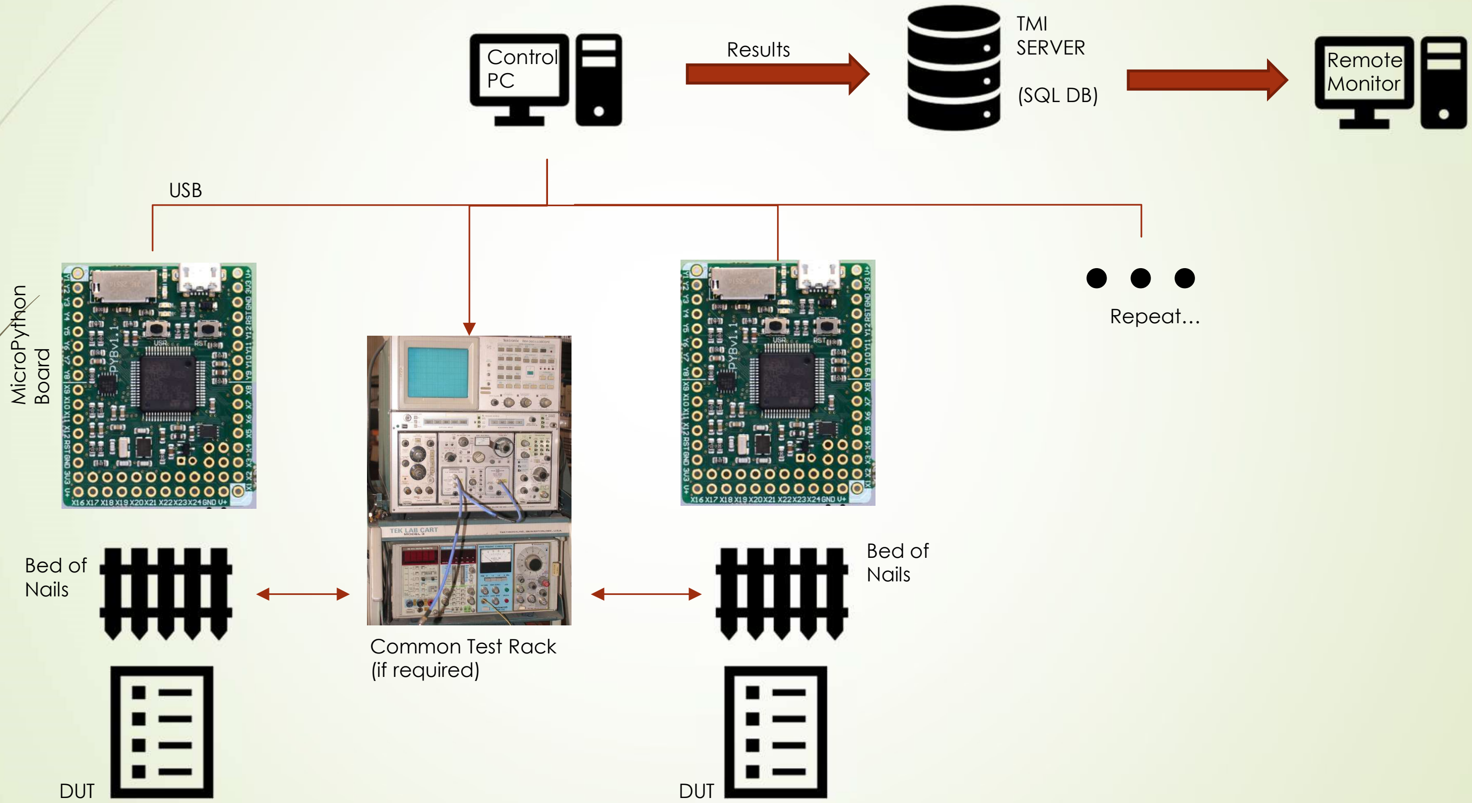
V+: 3.6v - 16v power input
(supplied by USB when USB connected)
3V3: regulated 3.3v output only, max 250mA
VBAT: FET protected supply battery input
VBACK: backup-battery input
A3V3: analog reference connected to 3V3 via inductor

X17 is pulled to GND via 4.7k resistor when USR pressed
P2-P5 are connected to the 4 LEDs
SD_SW = A8 is used for SD card switch
MMA_INT = B2 is used for accelerometer interrupts
MMA_AVDD = A10 is used for accelerometer power

connect BOOT0 to 3V3 and press RST to enter DFU mode

micropython.org

System Block Diagram



DUT Design

- Add test points for the bed of nails jig
- Understand the MicroPython Board IO pin capabilities
  - Or create your own "interface board"
- Create PCB to interface MicroPython Board to the DUT
- Determine what external test equipment is required to test things that can't be tested with MicroPython Board

- Write (python) Software within this framework…
  - Results will be normalized, stored in SQL DB
  - Logging
  - Results Dashboard

Cost

Tier	Cost/Month/Site (CAN$)	Units/Month
1	$500	0 – 2k
2	$750	2k – 10k
3	$1250	10k – 50k
4	$2000	50k – unlimited

- Subscription Based Model
  - First 3 months are billed up front
    - This is typically "ramp up" phase thus quantities are low, therefore $1500 up front.
  - Second 3 months billed after (at 6 months)
    - Tier is determined by the peak month within the last 3
- Customer Other Costs
  - All the Hardware
  - One person familiar with Python to operate the system
  - Cloud costs
    - The system is designed to work on AWS Free Tier level, assuming your data storage needs are not >5GB
  - Contracting/Training
    - $250/hour

Project Current Status

- Ready to Demo

- Next Features in Development
  - Result Encryption on the test station
  - Deployment/Upgrade model
  - Other security features

Other

- DUT measurements
  - Considering the Diligent Analog Discover 2
    - 2CH (100Msample/sec) scope, GPIO, Power sources, Digital Bus Analyzers (SPI, I²C, UART, Parallel)
    - Python drivers
    - CON
      - $400 each!
      - No (serial, i2c, etc.) ports