TEAMNAME MAY1733

Testing Environment for Accessing and Monitoring Networked Automation and Measurement Equipment

Antonio Montoya
Christian Hurst
Braden Rosengren
Ben Wiggins
Chris Little

Advisor/Client
Dr. Geiger

Slides Available on our website at: http://may1733.sd.ece.iastate.edu/

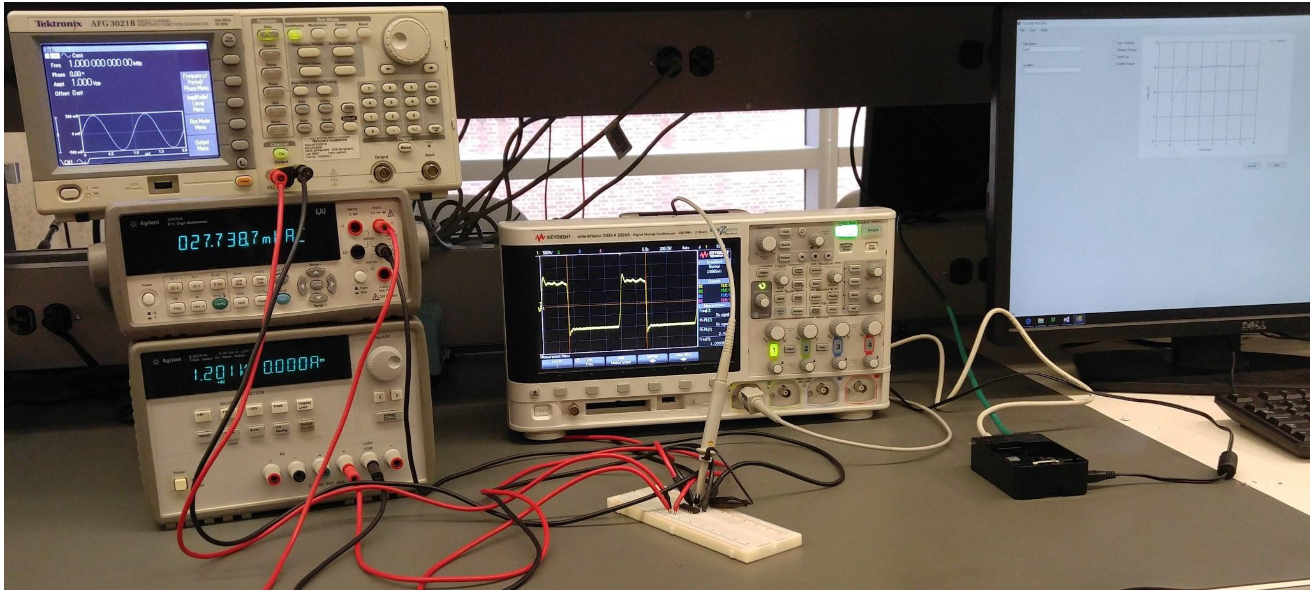
Project Background

Purpose/Problem Statement

Our purpose in this project is to:

- Develop remote hardware test automation with monitoring capabilities
- Develop an API (Application Program Interface) for user-expandable hardware support
- Provide inexpensive alternative to more costly automation solutions
- Demonstrate conceptual viability of standardized system

Project Vision



For illustration purposes only

Market Survey & Usage Target

- Saw a need for remote test hardware automation in both academia and industry
- Wanted a testing platform usable independent of the user environment
  - Prevent driver/PC hardware compatibility issues
  - Prevent updates from changing user’s environment
- Current GPIB interfaces may be too expensive for those with smaller budgets

Project Requirements

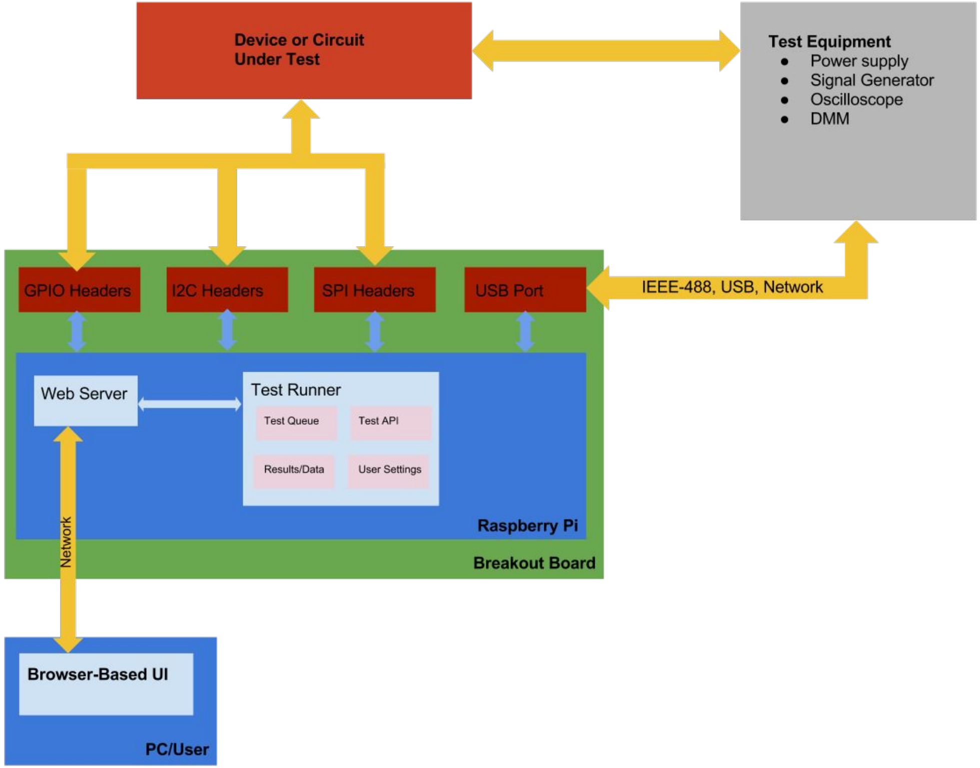
- Web server and browser-based UI/UX
- Raspberry Pi must interface with lab equipment
- Raspberry Pi must initialize all required processes on startup
- PCB breakout board for Pi to chip communication
- Creation of a variable logic level shifter for SPI, I2C and GPIO busses
- Create resources for writing tests and supporting additional test equipment
- Demonstrate proof of concept and potential usage cases

Constraints & Considerations

- We need to create a cost-effective platform
- The remote interface should be cross-platform compatible
- We need to support GPIB commands over IEEE-488, USB, and Ethernet
- Enable users to create and execute custom tests
  - Using a simple widely known scripting language (Python)
- Make the design system to be easily extensible
  - Add previously unsupported test equipment to the API
  - Easily write new test scripts

Project Design

Functional Description



Detailed Design (Modular design specifics)

PC UI/UX

- Web Server with Python backend
- Browser-based design ensures cross-platform compatibility

Breakout Board

- Level shifters allow communication with extended range of DUTs

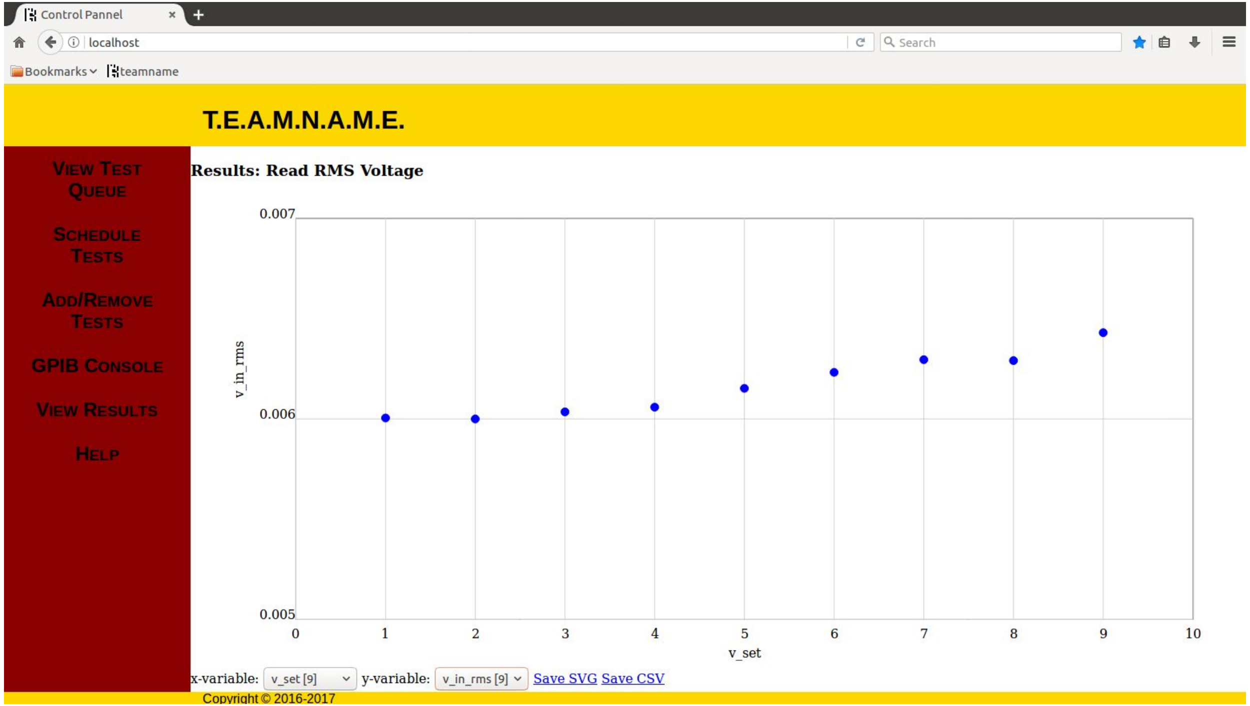
USB to GPIB Connector

- Allows control of test equipment over IEEE 488 bus

Test Writing API

- Allows users to easily define and execute their own tests
- Streamlines storage and retrieval of results

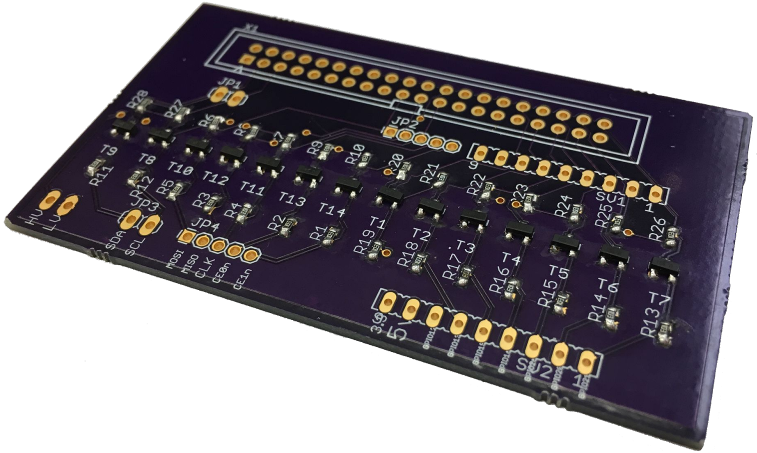
Browser-Based Interface



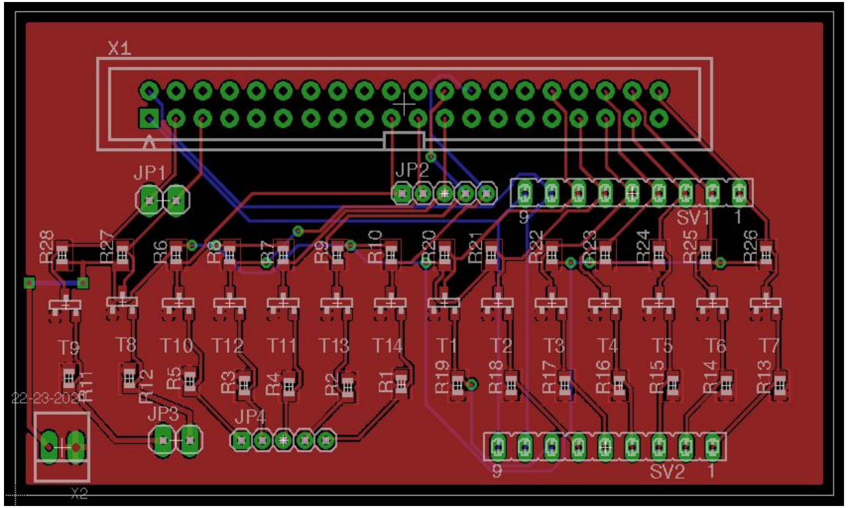
Interface showing the in-browser result viewer demonstrates the web server's functionality.

Variable Level Shifter Breakout

Rev. 3 Breakout



Rev. 3 Design



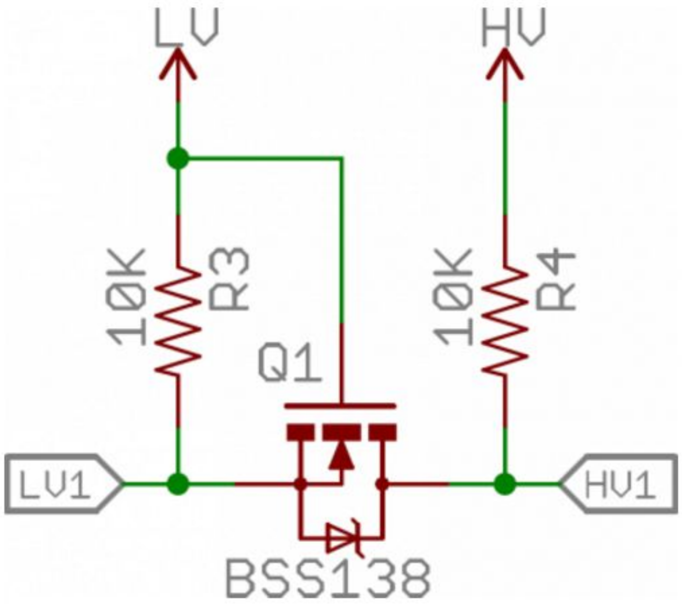
Design Process

Designing the Server

- Chose lighttpd server as the web server
  - CGI support allows dynamic content generation with Python
  - Easy to install/use (supported in Buildroot)
  - Requires CGI scripts to complete execution before user is sent content
- Chose to implement Python-based application for test runner
  - Need separate process to manage test execution
  - Sends data to CGI scripts for user interface

Designing the Level Shifter

Utilized BSS138 to achieve bi-directional level shifting of GPIO pins on Pi-board from 3.3V to an externally supplied reference voltage, selected to match DUT's VDD.



Designing the Hardware Driver/API Scheme

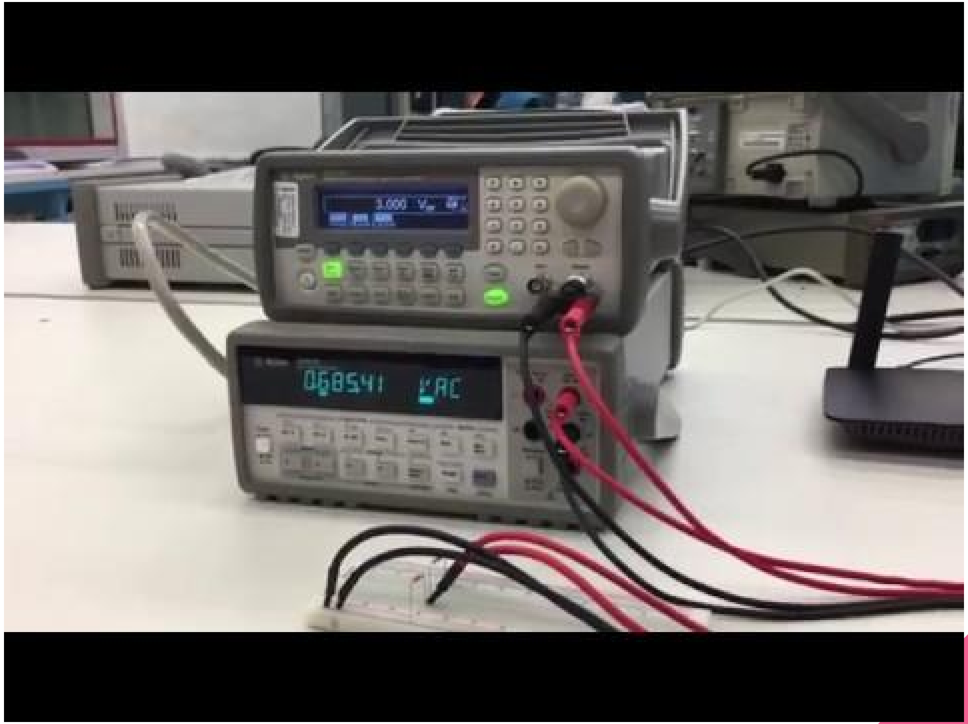
- API must be extensible to allow users to add support for new test equipment
- Provided tools allow streamlined development of 3rd party drivers
- User generated content limited to IEEE standard GPIB strings for device
- Newly generated drivers can be comprehensive for public use or custom made for a single use

Project Testing

Module P.O.C. Tests

- Verified GPIB communication from Raspberry Pi to signal generator
- Verified dynamic content generation with Python scripts
- Verified proposed level shifter schematic
- Verified populated PCB functionality
- Implement EE 230 lab test case

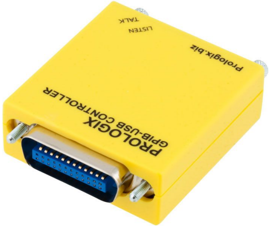
Project Demonstration

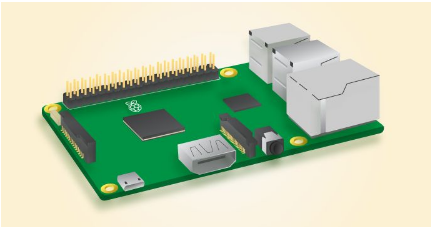


Questions?

Resource & Cost Projection

- Raspberry Pi 3 Model B - $35.69 (Amazon.com)
- Prologix GPIB-USB Controller - $149.95 (Prologix.biz)
    - While this is the largest cost component, new equipment has USB interfaces that make this part obsolete
- Custom breakout board - Approx. $5
- Free and open source software





http://prologix.biz/images/detailed/0/GPIB-USB-front.jpg
https://www.raspberrypi.org/wp-content/uploads/2016/02/Pi_3_Model_B.png

Selecting a GPIB (IEEE 488) to USB Adapter

NI GPIB-USB-HS+ :

- Sold by National Instruments
- $611.00
- Complicated process to support on linux
- Full feature support requires Expensive software
- High Level abstraction is supported through expensive software

PROLOGIX GPIB-USB Controller :

- Sold by Prologix
- $149.95
- Presents itself as a simple USB serial port
- Support is built into the kernel

- Support for high level abstraction of functionality needs to be built out

Early Platform Design

- Wanted platform to be consistent between users
  - Web server and browser-based interface allows cross-platform compatibility
  - Raspberry Pi standardizes computer hardware and software to prevent incompatibilities
- Wanted platform to be remotely accessible
  - Web-based design means server can be accessed from anywhere on the network
- Needed to be able to control testing equipment
  - GPIB over the IEEE 488 bus is common and appeared a good place to begin
- DUTs may operate at voltages other than that of the Raspberry Pi
  - DUTs may require digital configuration inputs
  - Variable level shifter for GPIO allows for a greater range of functionality

Driver Generation: User Written File

```
Keysight20xx                            #DEVICE_NAME
Keysight 20xx Family of Oscilloscopes   #DEVICE_DESCRIPTION
Oscilloscope                            #DEVICE_TYPE
IEEE-488                                #DEVICE_CONNECTION
Keysight20xx.py                         #TARGET_FILE


#END_HEADER
#NOTES SECTION
#
#
#
##START_DEFINES
_clear_status            {clear_device_status}                    *CLS
_event_status_enable  Q  {enable_event_status {check_ESE_state}}  *ESE    <mask_argument>
_check_device_id      QO {get_device_id check_device_id}          *IDN?
##END

//_example_function    {desired_user alias_functions {query_alias}} :GPIB:COMMand:FOR:FUNCtion <required_input_values> [Optional Arguments {available_inputs_to_required_arg}]
///                  ^  place a 'Q' here to also generate a query version of the function
///                              use 'QO' for query only
```

Driver Generation: Command List & Python Code

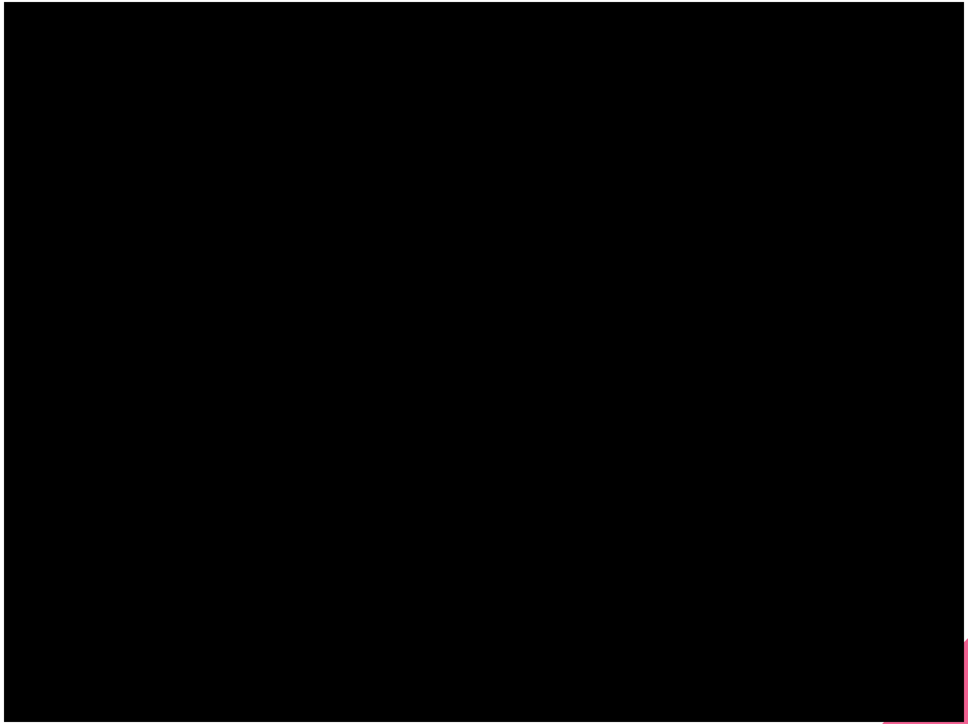
```
##START_DEFINES
_clear_status                {clear_device_status}                        *CLS
_event_status_enable    Q  {enable_event_status {check_ESE_state}}    *ESE    <mask_argument>
_check_device_id        QO {get_device_id check_device_id}            *IDN?
##END
```

```
def Keysight20xx_clear_status(GPIB_Context):
    command = '*CLS'
    GPIB_Context.send(device_data, command)
    return 0

def Keysight20xx_event_status_enable(GPIB_Context):
    command = '*ESE'
    GPIB_Context.send(device_data, command)
    return 0

def Keysight20xx_event_status_enable_query(GPIB_Context):
    command = '*ESE?'
    GPIB_Context.send(device_data, command)
    return GPIB_Context.read(device_data)

def Keysight20xx_check_device_id(GPIB_Context):
    command = '*IDN?'
    GPIB_Context.send(device_data, command)
    return GPIB_Context.read(device_data)
```

```
'clear_device_status' : Keysight20xx_clear_status,
'check_ESE_state' : Keysight20xx_event_status_enable_query,
'enable_event_status' : Keysight20xx_event_status_enable,
'get_device_id' : Keysight20xx_check_device_id,
'check_device_id' : Keysight20xx_check_device_id
```

Software Verification: GPIB Hardware Control

Hardware Verification: Level Shifter



Future Test Cases: Project Continuation

- Demonstrate level shifter and bus support with I2C and SPI compatible chips
- Use a previously verified DAC voltage drift test on an I2C DAC IC and monitor the results with the system
- Demonstrate data acquisition in the context of a EE 230 lab

http://www.ti.com/graphics/folders/partimages/DAC6574.jpg