Testing Environment for Accessing and Monitoring Networked Automation and Measurement Equipment

Antonio Montoya
Christian Hurst
Braden Rosengren
Ben Wiggins
Chris Little

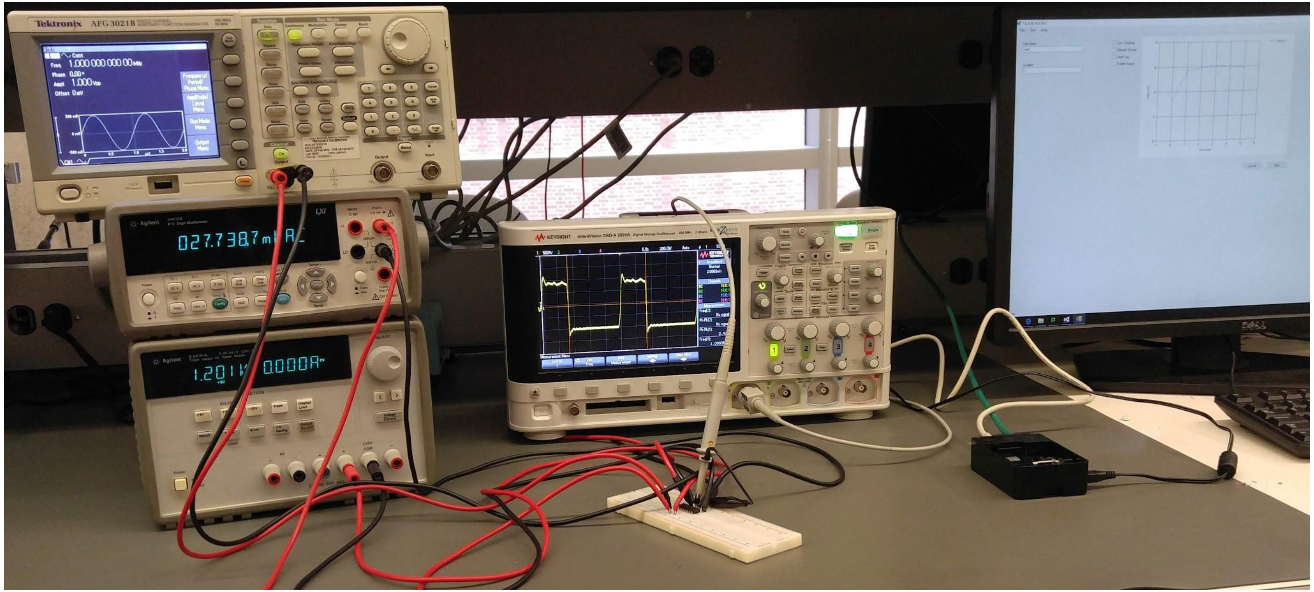
Project Background

Purpose/Problem Statement

In this project, we aim to:

- Develop remote hardware test automation and monitoring capabilities
- Develop an API for user-expandable hardware support
- Provide inexpensive alternative to more costly automation solutions
- Demonstrate conceptual viability of standardized system
- Present plan for creating an open source platform

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 economically infeasible for those with smaller budgets

Project Requirements

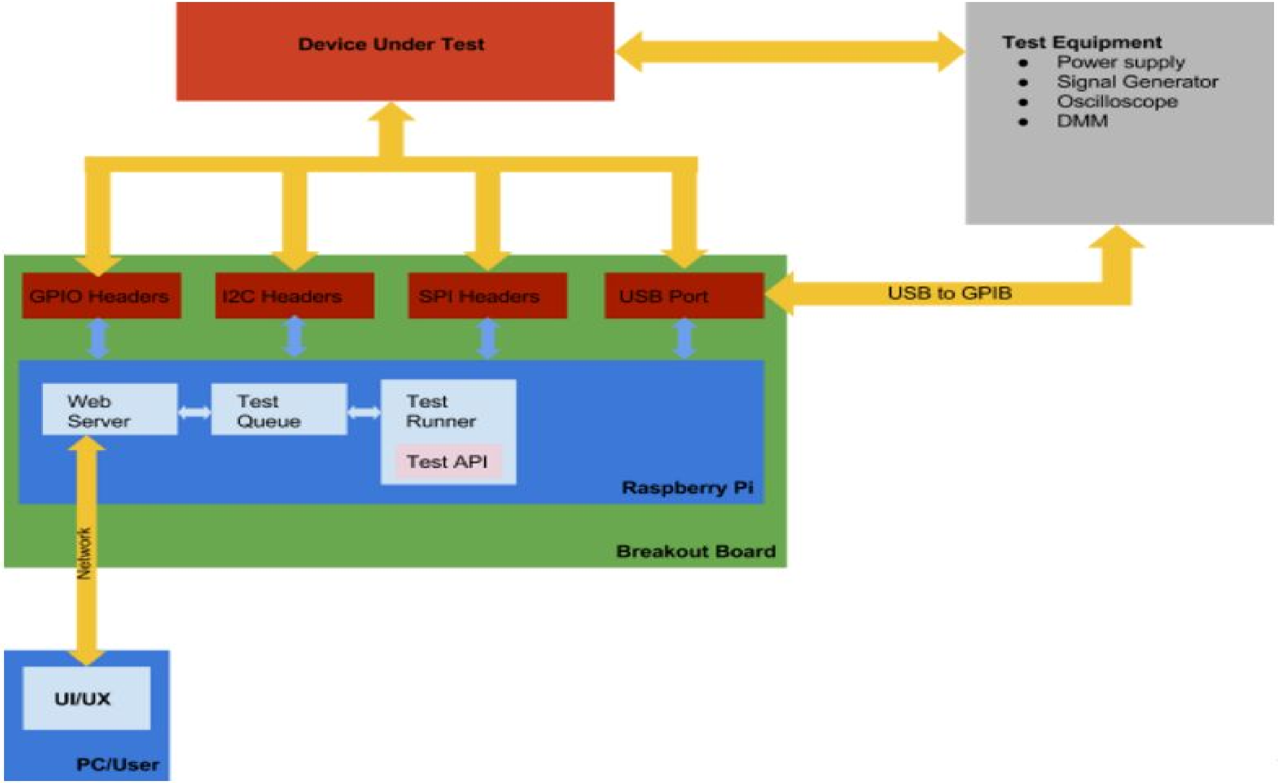
- Web server and browser-based UI/UX
- Raspberry Pi must interface with lab equipment
- Raspberry Pi must initialize all required processes on startup
- PCB breakout board for Pi to chip communication
- Creation of a variable logic level shifter for SPI, I2C and GPIO busses
- Create a guide for writing 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

Project Design

Functional Description



Detailed Design (Modular design specifics)

PC UI/UX

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

Breakout Board

- Level shifters allow communication with extended range of DUTs

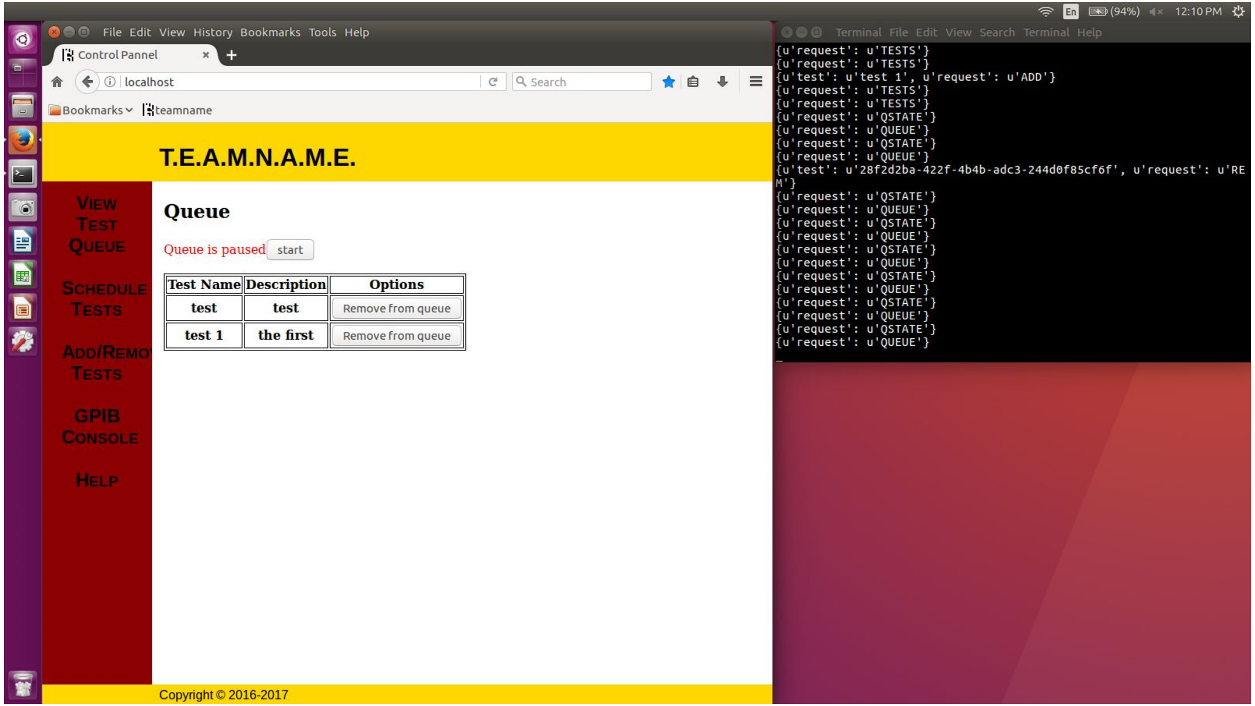
USB to GPIB Connector

- Allows control of test equipment over IEEE 488 bus

Test API

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

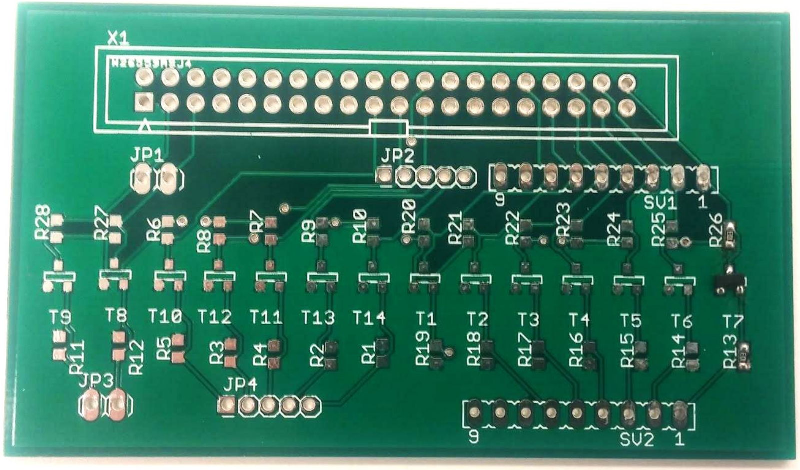
Browser-Based Interface



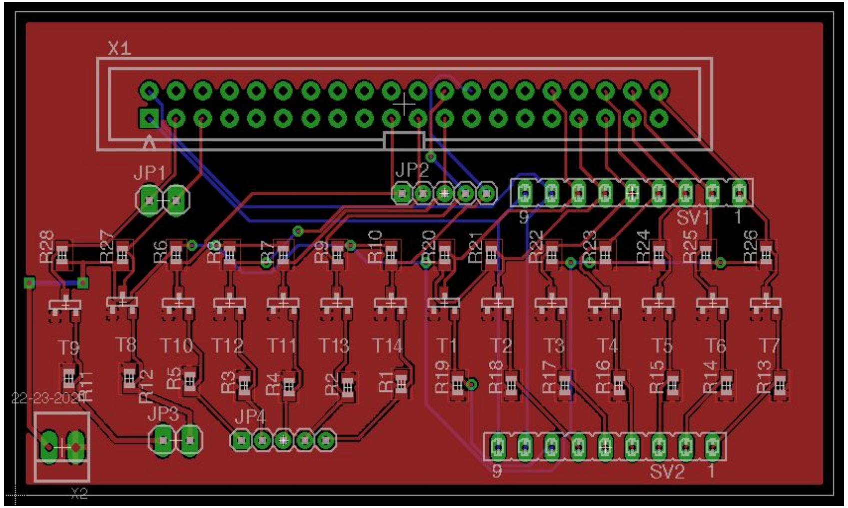
Interface showing test queue (left) and server log (right) demonstrate the web server.

Variable Level Shifter Breakout

Rev. 2 Breakout

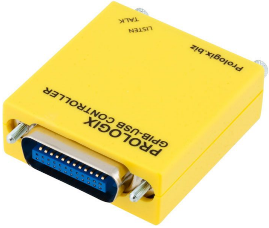


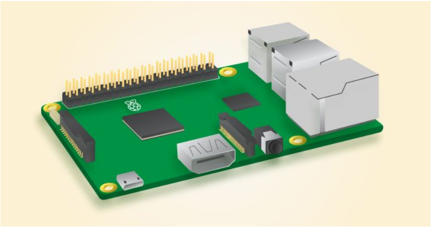
Rev. 3 Design



Resource & Cost Projection

- Raspberry Pi 3 Model B - $35.69 (Amazon.com)
- Prologix GPIB-USB Controller - 149.95 (Prologix.biz)
- Custom breakout board - TBD
- 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

Design Process

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

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 :

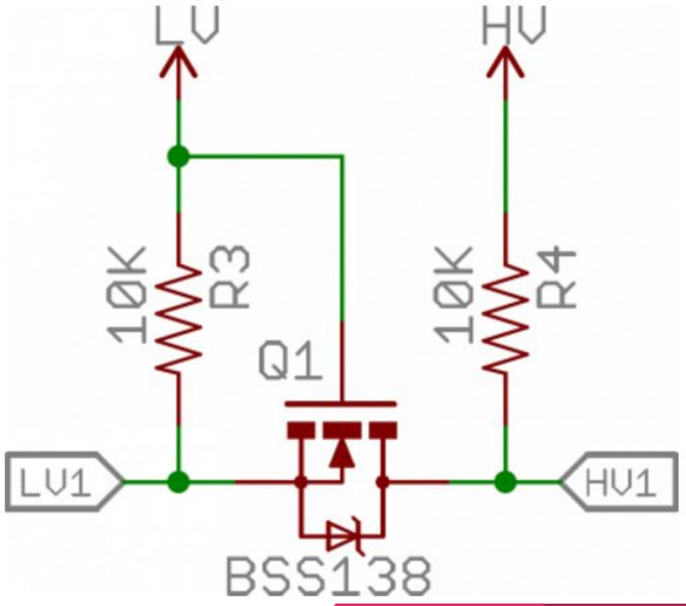
- Sold by Prologix
- $149.95
- Presents itself as a simple USB serial port
- Support is built into the kernel
- Support for high level abstraction of functionality needs to be built out

Designing the Server

- Chose Apache server as the web server
  - CGI support allows dynamic content generation with Python
  - Easy to install/use
  - 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
- Process must communicate with web server to allow user to see tests
  - Named pipes vs. Unix sockets
  - Unix sockets are nonblocking and bidirectional, unlike named pipes
  - Unix sockets can implement client/server design using TCP or UDP

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 API

- API must be extensible to allow users to add support for new test equipment
  - Goal is to launch an open source project for 3rd party collaboration on building out library of supported devices compatible with a standard API.
- Need a standardized way to interact with hardware
- Interpreted markup helps to mitigate software malfunctions

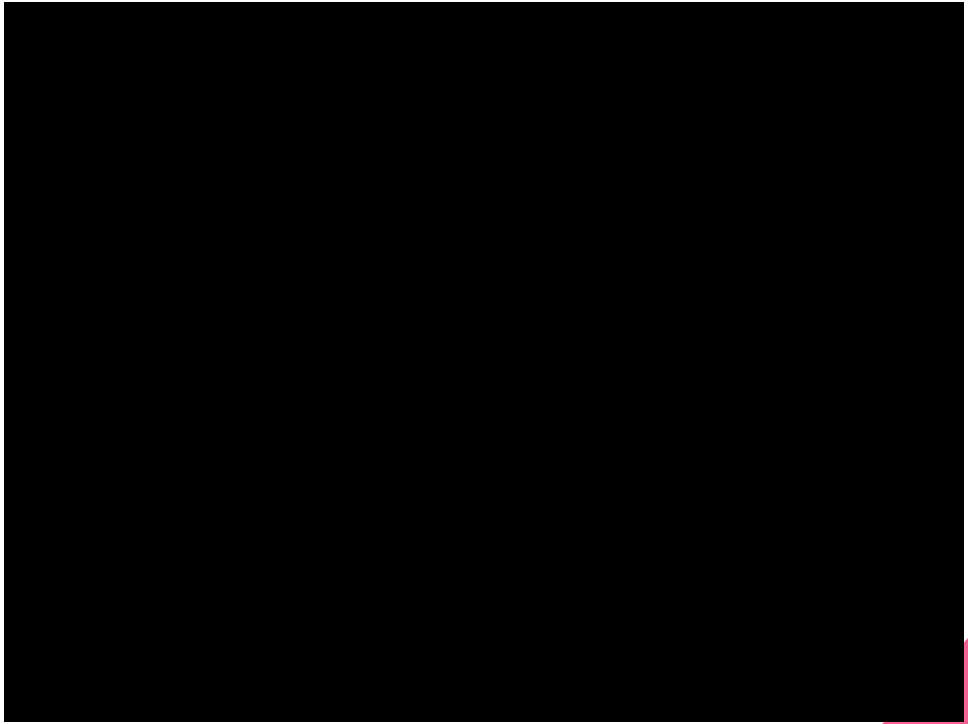
Project Testing

Module P.O.C. Tests

- Verified GPIB communication from Raspberry Pi to signal generator
- Verified dynamic content generation with Python CGI scripts
- Verified proposed level shifter schematic
- Verified PCB functionality
- Verify I2C communication
- Implement EE 230 lab test case

Software Verification: GPIB Hardware Control

Hardware Verification: Level Shifter



Planned Test Cases

- 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

Questions?