Hannah Tomio
August 18, 2015

# MCP Scrubber Monitoring

## 1 Introduction

The scrubber is a device that bombards microchannel plates (MCP's) with electrons, thus "scrubbing" them. This is done both to stabilize the gain of the MCPs and to, as the name would suggest, clean them of any residue. As detailed by Tyler Lutz in his collection of design documents[1], a mercury vapor bulb and a nichrome layer on a fixed MCP serve as the scrubber's electron source. The number of electrons is then amplified by the fixed MCP. The MCPs that are being scrubbed (the scrubbees) are situated between the fixed MCP and an anode. A voltage is applied between the nichrome layer and the anode to accelerate the stream of electrons towards the scrubbees. A uniform magnetic field is created with two pairs of Helholtz coils, placed orthogonally to each other at the sides of the MCP's in order to ensure a uniform spread of electrons.

## 2 Measurements

Over the course of the scrubber's scrubbing cycle, we would like to take several measurements: temperature, pressure, and the residual gases present. This lab currently has several thermocouples, a Pfeiffer Compact FullRange Gauge PKR 251[2], and an SRS Residual Gas Analyzer[3] available to take these measurements, as well as a Raspberry Pi Model B[4] to control and log these devices and their outputs. Instead of a Gertboard[5], however, we will be using a LabJack U6 Pro[6] to extend the analog inputs of the Pi.

## 3 Components

We will be using the following:

**Pfeiffer Compact FullRange Gauge PKR 251**
This is a gauge designed for vacuum measurement in the pressure range of $5x10^{-9}$ to 1000 mbarr. We will use it to measure the pressure inside the scrubber.

**Raspberry Pi Model B**
The Raspberry Pi is a low cost, low power computer running a Debian-based Linux distribution. It can be remotely accessed via ssh, and will run the monitoring program.

[1] psec.uchicago.edu/library/doclib/documents/230
[2] www.pfeiffer-vacuum.com/en/products/measurement/activeline/activeline-gauges/?detailPdoId=16099
[3] www.thinksrs.com/products/RGA.htm
[4] https://www.raspberrypi.org/documentation
[5] https://www.sparkfun.com/products/retired/11773
[6] http://labjack.com/u6

**LabJack U6 Pro**

This is a multifunction DAQ with 14 analog inputs with ranges of ±10, ±1, ±0.1, and ±0.01 volts and a 22-bit effective resolution. Thus it can directly measure raw thermocouple signals. While LabJack provides Windows software to easily log inputs, it also provides a Linux driver and Python module (LabJackPython) which we will be using to communicate with the U6 from the Raspberry Pi.

**CB37 Terminal Board**

The CB37 board[7] provides screw terminals for the DB37 connector on the LabJack U6, thus providing screw terminals for all 14 analog inputs.

**5 K-Type Surface-mount Thermocouples SA1XL**

These thermocouples[8] have response times of less than 0.15 seconds and temperature ranges of −73 degrees Celsius to 315 degrees Celsius. We will use them to measure the temperature of the scrubber.

**OMEGABOND 400 Cement**

A high temperature, air set cement[9] rated to 1427°C, for adhering the thermocouples to the scrubber.

**ACDC Converter**

This AC to DC wall mounted adapter[10] provides 24 volts (at 0.5 amps) and will be used to power the Pfeiffer Compact FullRange Gauge.

# 4 Infrastructure and Maintenance

## 4.1 Setup

The setup of this system is very simple – only minor soldering will be necessary.

1. Connect the CB37 to the Labjack via the DB37 connector on the Labjack.
2. Connect the LabJack to the Raspberry Pi via a USB A-to-B cord.
3. Connect the thermocouples to the screw terminals of the CB37, with their probes cemented to the scrubber where needed. Make sure they are in the correct orientation!
4. Cut the connector end off of the Pfeiffer gauge sensor cable[11] and strip to reveal the wires within.
5. Pins 4 and 5 (yellow wire and grey wire, respectively) must go to the ACDC wall adapter, with pin 4 going to the supply and pin 5 going to the ground. Pins 2 (brown wire) and 3 (green wire) must connect to the CB37, with pin 2 going to the analog input and pin 3 going to the gound. Pin 1 (white wire) and the shield should go to ground as well. Soldering a PCB board to help connect these wires is a good idea.

[7]http://labjack.com/catalog/cb37-terminal-board-rev-21

[8]http://www.omega.com/pptst/SA1XL.html

[9] http://www.omega.com/pptst/SA1XL.html

[10]http://www.digikey.com/product-detail/en/SWI12-24-N-SC/T1284-ND/5052006

[11]https://www.pfeiffer-vacuum.com/en/products/measurement/activeline/activeline-accessories/?detailPdoId=16130

6. Connect the Raspberry Pi to the internet via the ethernet port and plug in the Raspberry Pi's power supply.

Currently the system is not capable of running the RGA, however configuring it to do so shouldn't be too difficult. You must connect the pi to the RGA with a USB to RS232 cable (provided in the electronics box) and adapt Eric Spieglan's "rgadev.py" (in the Margherita-Code folder) code.

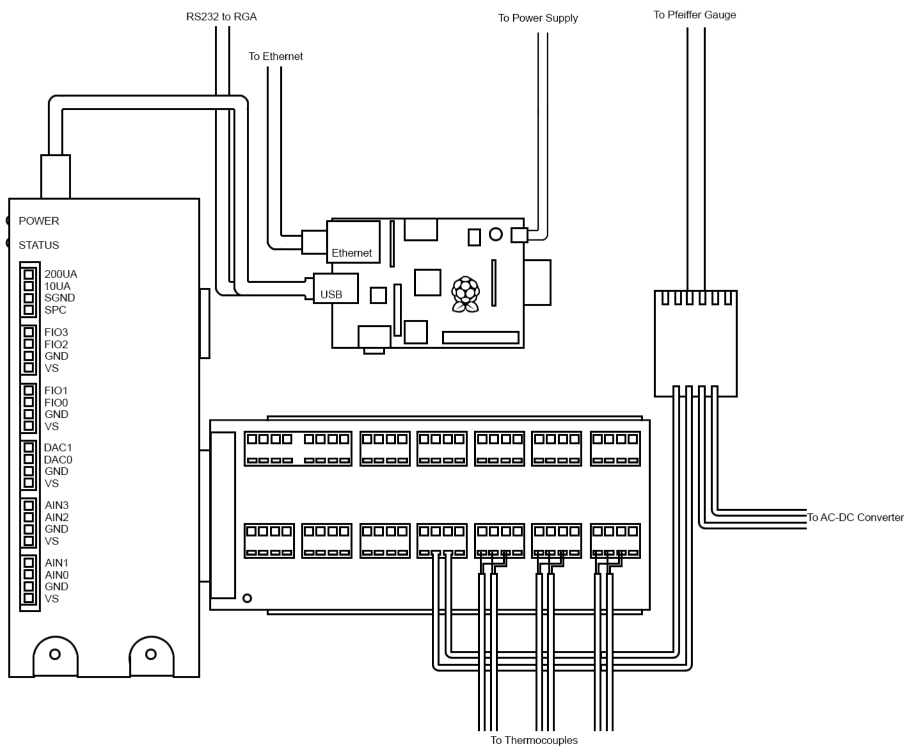


Figure 1: Diagram of the setup inside the electronics box

## 4.2 Logging

The Raspberry Pi runs a Python program called "scrubberpi.py" to read voltages from the LabJack, convert to temperature or pressure measurements, and record these measurements in a text file. Conveniently, LabJack provides a Python module to connect to the U6, which our logging program uses. The conversion equations and their constants for K-type thermocouples can be found at the National Institute of Standards and Technology's website `srdata.nist.gov/its90/main/`. The code for this program can be found in the appendices of this manual.

1. To start the logging program, first ssh into the Raspberry Pi. For this particular pi,

the ip address is 205.208.20.42, the username is “pi”, and the password is “forceevanidmegapolisconversantly”.

2. Using the console of the Raspberry pi, go to the folder “ScrubberCode” and call “sudo python scrubberpi.py”. Any errors that occur will be printed to the console. This program will run indefinitely, unless you do a keyboard interrupt, kill the process directly, or the LabJack becomes disconnected.

3. If you would like to read the measurements as they are logged, you should run the program in the background instead (call “sudo python scrubberpi.py &”), then go to the logs folder and call “tail -f nameoffile”. The files will be named according to this format: “temp-mm-dd-yy.txt” and “pressure-mm-dd-yy.txt” with the “mm” being the month, “dd” being the day, and “yy” being the year.

4. To graph the data, there is a basic matplotlib plotter program included in the appendicies. The data can also be graphed in many other ways (MATLAB, etc). To use the included matplotlib program, you must first combine the data files into one file (if it is not already one file). Use the combine_files program, but change the names of the files in the code to the names of the files you would like to combine. Then you may use the pressure_plotter or temp_plotter functions to plot your graphs.

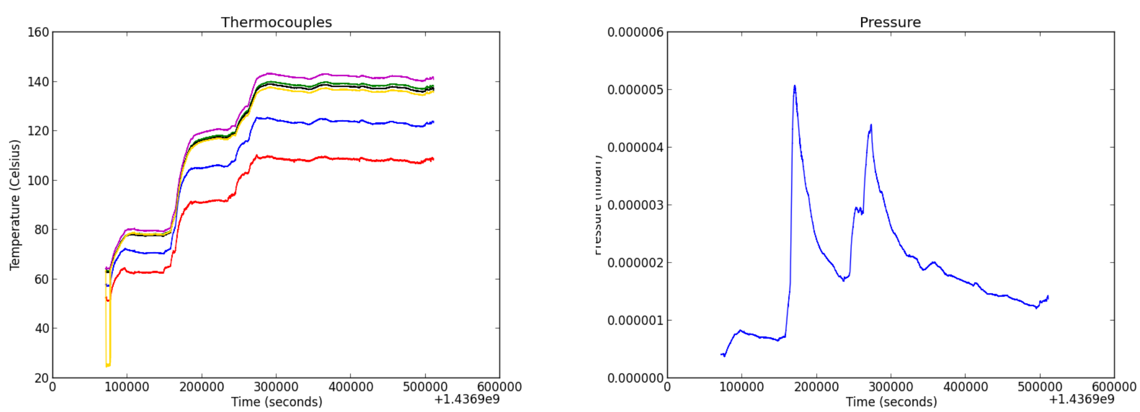


Figure 2: Plots of temperature and pressure from a vacuum bakeout

### 4.3 Maintenance

After the setup of this system is complete and the final configuration is established, the system will be housed in an aluminum enclosure with the wires properly strain relieved. Except for checking the secure connections of the wires, there is little maintenance that needs to be done. The code and any necessary modules can be found on the project website, at `psec.uchicago.edu/Code`, and of course on the Raspberry Pi itself.

# Appendices

Listed here is all of the code used to run the monitoring system.

# A Main Code

```
# This code measures and logs temperature from 5 thermocouples and pressure from a Pfeiffer PK-251 gauge. The output files will be located in the logs folder. The logging program can be stopped with a keyboard interrupt, but will catch other errors and print them to console.

import u6
import time
import pfeifferGauge
import rgadev
import typeKthermocouple


# These describe the input channels on the labjack in use. Channels 0-5 are reading different thermocouples
# and channel 6 is reading from the Pfeiffer gauge.
pgaugeChannel = 6
thermoChannel= [0, 1, 2, 3, 4, 5]

# Declare a labjack U6 object from the u6 module
labJack = u6.U6()


# This code will run indefinitely, it can be quit with a keyboard interrupt (Ctrl C)
while True:
  try:
    # Open the files the measurements will be logged in. These are located in the logs folder
    temp = open('./logs/temp-{}.txt'.format(time.strftime('%m-%d-%y')), 'a+')
    pressure = open('./logs/pressure-{}.txt'.format(time.strftime('%m-%d-%y')),'a+')

    # The cold junction temperature in celsius, compensating for the screw junction temperature
    coldJunctionTemp = labJack.getTemperature() + 2.5 - 273.15

    temps = []
    for a in thermoChannel:
      # For each thermocouple, get the it's voltage in millivolts
      tvolt = ((labJack.getAIN(a, resolutionIndex = 12, gainIndex = 3) * 1000) +
          typeKthermocouple.tempToVolts(coldJunctionTemp))
      temps.append(str(typeKthermocouple.voltsToTemp(tvolt)))
    entry = (','.join(temps) + ',' + str(time.time()) + '\n')
    temp.write(entry)

    # Get the pressure gauge's voltage
    pvolt = labJack.getAIN(pgaugeChannel, resolutionIndex = 12, gainIndex = 0)
    current_pressure = pfeifferGauge.voltsToPressure(pvolt)
    pressure.write('{},{}\n'.format(current_pressure, time.time()))
    time.sleep(1)
  except Exception as e:
    # This will catch all exceptions and simply print them to console, allowing
```

```
    # the program to indefinitely
    print e
    pass
  finally:
    # Upon exiting, the file descriptors must be closed
    temp.close()
    pressure.close()
```

# B Thermocouple Module

```
# This module describes the approximate reference functions and inverse functions for type K thermocouples.The reference functions give the thermoelectric voltage, E, as a function of temperature, T, where E isin mV and T is in celsius.The inverse functions give temperature, T, as a function of the thermoelectric voltage, E, where E is in mV and T is in celsius.

# All coefficients are from http://srdata.nist.gov/its90/main/


import math

# Coefficieints for reference functions

# For -270 degrees C to 0 degrees C
tempToVoltsCoeff1 = [
0.0E0,
0.394501280250E-1,
0.236223735980E-4,
-0.328589067840E-6,
-0.499048287770E-8,
-0.675090591730E-10,
-0.574103274280E-12,
-0.310888728940E-14,
-0.104516093650E-16,
-0.198892668780E-19,
-0.163226974860E-22
]

# For 0 degrees C to 1372 degrees C
tempToVoltsCoeff2 = [
-0.176004136860E-1,
0.389212049750E-1,
0.185587700320E-4,
-0.994575928740E-7,
0.318409457190E-9,
-0.560728448890E-12,
0.560750590590E-15,
-0.320207200030E-18,
0.971511471520E-22,
-0.121047212750E-25
]

# For 0 degrees C to 1372 degrees C
```

```
tempToVoltsCoeff3 = [
0.118597600000E0,
-0.118343200000E-3,
0.126968600000E3
]

# Coefficients for the inverse functions

# For -200 degrees C to 0 degrees C
# For -5.891 mV to 0 mV
voltsToTempCoeff1 = [
0.0E0,
2.5173462E1,
-1.1662878E0,
-1.0833638E0,
-8.977354E-1,
-3.7342377E-1,
-8.6632643E-2,
-1.0450598E-2,
 -5.1920577E-4
]

# For 0 degrees C to 500 degrees C
# For 0 mV to 20.644 mV
voltsToTempCoeff2 = [
0.0E0,
2.508355E1,
7.860106E-2,
-2.503131E-1,
8.31527E-2,
-1.228034E-2,
9.804036E-4,
-4.41303E-5,
1.057734E-6,
-1.052755E-8
]

# For 500 degrees C to 1372 degrees C
# For 20.644 mV to 54.886 mV
voltsToTempCoeff3 = [
-1.318058E2,
4.830222E1,
-1.646031E0,
5.464731E-2,
-9.650715E-4,
8.802193E-6,
-3.11081E-8
]

def voltsToTemp(volts):
  if volts < -5.891:
    raise Exception("Undefined thermocouple voltage: voltage underrange")
  if volts > 54.886:
    raise Exception("Undefined thermocouple voltage: voltage overrange")
  if -5.891 <= volts < 0:
    coefficients = voltsToTempCoeff1
  elif 0 <= volts < 20.644:
```

```
    coefficients = voltsToTempCoeff2
  else:
    coefficients = voltsToTempCoeff3
  temp = 0
  power = 0
  for d in coefficients:
    temp += d * (volts**power)
    power += 1
  return temp

def tempToVolts(temp):
  if temp < -270:
    raise Exception("Undefined temperature: temperature underrange")
  if temp > 1372:
    raise Exception("Undefined temperature: temperature overrange")
  if -270 <= temp < 0:
    volts = 0
    power = 0
    coefficients = tempToVoltsCoeff1
    for c in coefficients:
      volts += c * (temp ** power)
      power += 1
    return volts
  else:
    volts = 0
    power = 0
    constant = 0
    coefficients = tempToVoltsCoeff2
    for c in coefficients:
      volts += (c * (temp ** power))
      power += 1

    extended = (tempToVoltsCoeff3[0] * math.exp(tempToVoltsCoeff3[1] *
                  (temp-tempToVoltsCoeff3[2])**2))
    return volts + extended
```

# C Pressure Module

```
# This module contains the function that converts the Pfeiffer gauge voltage
    to pressure, as described the Pfeiffer Gauge PKR-251 manual, as found here
    :
# http://edg.uchicago.edu/tutorials/pumps/BG5155BEN.pdf in appendix A.
# If an error occurs, this function raises an exception which will be caught
    by the main program and printed to console.


# turns gauge voltage into a pressure in mbarr
def voltsToPressure(volts):
  if 8.6 < volts <= 9.5:
    raise Exception("Undefined pgauge voltage: voltage overrange")
  elif volts > 9.5:
    raise Exception("Pgauge sensor error: Pirani defective")
  elif 0.5 <= volts < 1.82:
    raise Exception("Undefined pgauge voltage: voltage underrange")
```

```
    elif volts < 0.5:
      raise Exception("Pgauge sensor error")
    pressure = 10**((1.667*volts) - 11.33)
    return pressure
```

# D Plotting

```
# Basic plotter using pyplot

import numpy as np
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt

def temp_plotter(filename):
  thermo0, thermo1, thermo2, thermo3, thermo4, thermo5, time = np.loadtxt(filename,
    dtype=float, delimiter = ',', unpack = True)
  plt.plot(time, thermo0, color = 'r')
  plt.plot(time, thermo1, color = 'b')
  plt.plot(time, thermo2, color = 'g')
  plt.plot(time, thermo3, color = 'm')
  plt.plot(time, thermo4, color = 'k')
  plt.plot(time, thermo5, color = 'gold')
  plt.title('Thermocouples')
  plt.xlabel('Time (seconds)')
  plt.ylabel('Temperature (Celsius)')
  plt.savefig('Thermocouples')

def pressure_plotter(filename):
  pressure, time = np.loadtxt(filename,
   dtype=float, delimiter = ',', unpack = True)
  plt.plot(time, 1/pressure, color = 'blue')
  #plt.ylim((0, 0.000006))
  plt.title('Inverse Pressure')
  plt.xlabel('Time (seconds)')
  plt.ylabel('Pressure (mbarr)')
  plt.savefig('Inverse Pressure')

#use this to combine files of multiple days to plot a full run
#you'll have to change the file names directly here
def combine_files():
  #this opens in append mode, so make sure you delete any files with the same name before running this code
  fout = open('totalpressure.txt', 'a') #be careful!
  for fname in ('./logs/pressure-07-15-15.txt', './logs/pressure-07-16-15.txt', './logs/pressure-07-17-15.txt',
                './logs/pressure-07-18-15.txt', './logs/pressure-07-19-15.txt', './logs/pressure-07-20-15.txt'):
    f = open(fname)
    for line in f:
      fout.write(line)
    f.close()
  fout.close()
```

```

#combine_files()
temp_plotter('totaltemps.txt')
pressure_plotter('totalpressure.txt')
```