

# Garnering Temperature Sensor Data to Display on a GCC Base Diagram Illustration



**Sirius Ben-Judah**  
**Alabama Agricultural and Mechanical University**

## **Abstract**

This paper describes a project to implement a web-based monitoring of temperatures within the Grid Computing Center. Through the use of a Python program, the temperatures are read out from thermocouple sensors and superimposed on a floormap of GCC in the proper locations of the sensors.

## **Introduction.**

In 1967, the United States Atomic Energy Commission authorized the construction and operation of what was then known as the National Accelerator Laboratory under a bill signed by President Lyndon Johnson. The Universities Research Association (URA), formed earlier by National Academy of Sciences president Frederick Seitz and headed by Norman Ramsey assumed the task of building and operating the new laboratory. Robert R. Wilson was selected as its founding director. Since then, the laboratory has proven to be a vital research institution engaged in the study of high-energy particle physics. The facility was dedicated and renamed in honor of the Italian Nobel Prize winning physicist Enrico Fermi in May of 1974. Fermilab is home to the largest high-energy physics laboratory in the United States, as well as the world's highest energy accelerator.

Fermilab conducts fundamental research into particle physics. Scientists investigate the smallest building blocks of matter separated by the smallest distances known to science. They constantly seek to understand more about these fundamental particles, and understand the forces that hold them together or force them apart. This knowledge is gained by conducting experiments in which subatomic particles are forced to collide at extremely high speeds, with the fragments of the collision then scattering into particle detectors. This process is much akin to how human vision functions in which photons bounce off of objects in our environment and are scattered into our retina. In a similar manner, Fermilab scientists follow a similar approach to "see" into the structure of matter by scattering subatomic particles off of similar particles. It is the results of these collisions that are the focus of the experiments. From the discoveries of the bottom and top quarks in 1977 and 1995, respectively, to the first direct observation of the tau neutrino in 2000, Fermilab has produced a number of results that have advanced science.

### Necessity of Computers

The vast amounts of data that are collected daily through experiments and observation would be of little use without the aid of computer technology. Extensive information about the state of the experiment's detector and the detection of the fragments of the collisions from experiments such as D0 and CDF are sent to various computers on the Fermilab site, such as the Feynman and Grid Computing Centers. Here, the data is analyzed and summarized as results which are then published for the benefit of scientists at both Fermilab and around the world.

### My Program's Niche

In order to maintain the functionality in the Grid Computing Center, which will soon house some 2800 computers, it is necessary to monitor the computing center environment for proper temperature, humidity, and the like. To provide temperature observation, a total of twenty temperature sensors have been installed in various locations within the GCC data center. As additional computer units and computer racks are purchased and installed, there will be a need for a greater number of temperature sensors. In order to monitor all of the computers, the Computing Division will likely to purchase a minimum of 60 total thermocouples. By monitoring the temperatures, the proper functioning of the computer room air conditioners (CRAC'S) can be verified. A drastic change in temperature is a sign of a malfunction of a CRAC or other GCC component. With the amount of vital physics data that is analyzed at the GCC, a breakdown of this sort could prove catastrophic to the progress of an experiment. Of course, there are a number of paging and alert systems, such as Metasys and NGOP, that guard against this sort of event. However, there is not, up until now, an easy to access, quick to interpret, display that shows the floor layout and the temperatures at the specific places on the layout where the readings are measured. Further, the distribution of temperature sensors in Metasys is too course-grained (only measured at the input of the CRACs) or, as in the case of NGOP, is too fine-grained (measured at every CPU chip in GCC—soon to be 2800 chips to be precise). My program is designed to measure in an intermediate way at 20 easily-expandable points on the front and back of selected racks within the GCC data center.

### Background

Information gathered from many of Fermilab's most important experiments are sent here for analysis, including data from the D0 and CDF experiments. In addition, data from simulations of the CMS experiment are analyzed. The computers in GCC run constantly, and are stopped only when maintenance procedures require it.

The computers, which generally run Linux, are powered via a 1000 KVA uninterruptible power supply (UPS) system that obtains its power from the laboratory electrical feeders supplied by commercial power companies via a dedicated electrical substation.<sup>5</sup> The UPS insures that in the case of an outage of commercial power, the computers have time to shutdown in a controlled manner.

### Learning Perl

When I arrived at Fermilab, I had to learn the technology and languages required to program proficiently. The first of these languages was PERL (Practical Extraction and Report Language). Larry Wall invented this computer programming language in 1987. The language is available for Unix, Windows NT, Mac, and DOS computer systems. Perl is fast, complete, concise, flexible, portable, and is available free of charge. To learn Perl, I wrote several smaller programs in the Perl language that made use of "for", "foreach", and "while" loops, as well as "if-then-else" statements and subroutines. This exercise culminated in my writing a simple

“adventure game” using the Perl language, in which many loops provided a series of consequences based upon decisions made by the player.

There were several purposes to my learning Perl. First, it is a good language to learn for my future career, and a handy background for future programming tasks. Second, it was a way (though I did not know at the time) for my supervisor to evaluate my capabilities. Had I failed to show that I was capable of learning Perl, this paper and my summer project would have been entirely different. Third, and finally, it might have been the computer language that I would have used for the temperature-monitoring project. But as it turned out, my supervisor decided that a different language, Python, would be better suited for the task. For that reason, I took on the charge of learning Python.

### **Learning Python**

Guido van Rossum invented this computing language in February 1991. Prior experiences and difficulties in working with the ABC, Modula-3, and Amoeba computer languages were the inspiration for his work during the winter of 1989. After a year of development, Python was released.<sup>7</sup> Since then, it has been modified and adapted by other programmers as to ensure that the language can always remain up-to-date and serve current software needs.

Python is a very adaptable language, and has many advantages. The important ones for my project were:

- Python uses an elegant syntax for readable programs.
- Python is an agile language that makes it easy to get a program working. This makes Python an ideal language for prototype development and other ad-hoc programming tasks, without compromising maintainability.
- A variety of basic data types are available: numbers (floating point, complex, and unlimited-length long integers), strings (both ASCII and Unicode), lists, and dictionaries.
- Python supports object-oriented programming with classes and multiple inheritance.
- Code can be grouped into modules and packages.
- The language supports raising and catching exceptions, resulting in cleaner error handling.
- Data types are strongly but dynamically typed. Mixing incompatible types (e.g. attempting to add a string and a number) causes an exception to be raised.
- The large standard library supports many common programming tasks such as connecting to web servers, regular expressions, and file handling.
- Python's interactive mode makes it easy to test short snippets of code. There's also a bundled development environment called IDLE (Integrated Development Environment for Python).
- Python runs on many different computers and operating systems: Windows, MacOS, many brands of Unix, OS/2, and Amiga.
- Python is copyrighted but placed under an open source license meaning that Python can be freely distributed

## **Tools Required**

All of my programming this summer was accomplished through a PC. The machine is an old one and for that reason not very fast, but it was adequate for the job. The PC uses Windows 2000. I downloaded Perl from [www.perl.org](http://www.perl.org) and Python from [www.python.org](http://www.python.org).

There were also several times where I found it necessary to download several libraries and modules from various websites in order to make certain aspects of my program function as intended. One such site was [www.pythonware.com](http://www.pythonware.com) from where I found very useful information for using imaging functions and methods. Another resource was <http://wiki.wxpython.org>, from where I obtained more useful information concerning the proper way to display text on a JPEG image of the GCC data center. From <http://effbot.org/downloads/> I was able retrieve a list of the fonts I would use to display the temperature data in a text format on the map.

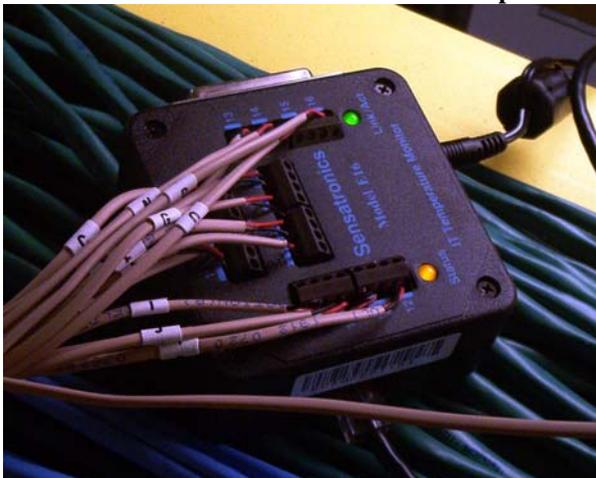
### **Temperature Recording Devices**

The process of recording the temperatures in GCC was accomplished through the use of Sensatronic Environmental Sensing Thermocouples ([www.sensatronics.com](http://www.sensatronics.com)) and the E4 temperature module that had been previously obtained. I initially began working with a quantity of four. When it became clear that my project was going to be a success, my supervisor ordered 16 additional thermocouples and an E16 unit.

The E4 and E16 are small microcomputers with no disks and only a fixed program in read-only memory. Besides a connection to a power supply similar to the power cube one typically sees providing power to a cell phone charger, these microcomputers have both a connection to the Ethernet and connections to cables which lead to thermocouples. The E4 and E16 collect data from four and sixteen thermocouples, respectively, hence the name of each microcomputer webserver. In addition, they have a serial connector that is used to initially configure the units via the serial port on a PC and through the use of the Hyperterminal program.

This configuration step is done only once and is the way the E4 and E16 is told what their IP addresses are, as well as the name designations for the sensors. After that, one reads the webservers at specified IP addresses, with a web browser, each microcomputer possessing a separate IP address. The E4 and E16 then return a web page (composed of basic HTML) with the sensor names and current temperature readings. Of course, one can use any program to read the information provided as long as that program can use the web rules (http and html protocols) for reading the information.

**The E16 Webserver Box with 16 thermocouple connections**



**The 50-foot thermocouple wires installed on Aisle F.**



**A thermocouple positioned on a rack. The white probe records the temperature.**



## **Objective**

The sensors all send information to the Sensatronics E4 or E16 to which they are connected. My supervisor applied to the Net Manager in the Computing Division to obtain an IP address for both web servers. In doing so, he supplied a unique Machine address (MAC address) for both the Sensatronics E4 and E16. The Net Manager assigned the IP address  $w.x.y.z$  to the E4 microcomputer, while the E16 was assigned  $w.x.y.z$ , and registered these associations between the IP addresses and their respective MAC addresses in the Fermilab servers. I assigned each IP address to a variable in my program, so that the program can be made to work for different and multiple IP addresses. (**Note:** the actual IP addresses have been removed per local policy requirements.) My Python program uses the `urllib` module of the Python Standard Library to read the E4 and E16 web servers that have the MAC addresses provided by the name server when the Python program provides the IP addresses. The result of the read done on these servers is a sequence of ASCII characters—just like what I would get from a read of a data file. My Python program then uses a series of regular expressions to extract the text strings that contain the name and temperature value of each sensor from the sequence of ASCII characters that is provided.

Because the E4 and E16 provide the characters with HTML tags (so it can be read by an ordinary web browser) the regular expressions used initially filter out all the HTML, leaving only the characters of data (sensor names and temperatures). My supervisor and I desired to make the program as flexible as possible to allow for the easy accommodation of future additions or removals of thermocouples from the GCC. This required the use of regular expressions in order to avoid having to use any predetermined variables as coordinates. These coordinates are the blueprints for the placement of the data values that are read from the IP address. It became necessary to name each sensor in a manner that would enable the program's regular expressions to retrieve the names, coordinates and temperature values, despite the placement or modification of thermocouples.

### **Naming the Thermocouples**

In order for the regular expressions to be able to extract the name information from the other non HTML data (such as the company name "Sensatronics", which was included in the E4 and E16 readout), I had to design an easily recognized pattern for the desired information. The configuration chosen for the sensor names was `': [H|C] - [A-F] \d\d? - [T|M]'`. The colon is literal. The `'[H|C]'` portion was designed to quickly identify a thermocouple as being located in either the hot or cold aisles of GCC, where hot or cold designates whether the computers exhaust their air into the aisle or intake air from the aisle. The two hyphens outside of the square braces were also literal. `'[A-F]'` is configured to identify what aisle of computers the thermocouples are located on. It is written as a series, so that as additional aisles of computers are installed, different letters can be assigned to each probe to accommodate its location. `'\d\d?'` tells what rack the probe is on in a particular aisle, as `'\d'` represents a digit. There was a need to add a question mark on at the end of the second digit, in case the probe was on any of the first nine racks of an aisle. If there is an additional digit, as in the case of racks 10-16, the second digit will also be matched. The final `'[T|M]'` informs a user of whether the thermocouple is at the top or middle position on a particular rack. For instance, the name `":H-F11-T"` would apply to a thermocouple that is at the top of the 11<sup>th</sup> rack of the hot side of the 'F' computer aisle in GCC, whereas `"C-D4-M"` identifies a thermocouple that is on the cold side of the 'D' computer aisle. It has been installed on the fourth rack of that aisle, at the middle of the rack.

To fulfill the task of obtaining the temperature values, I used the pattern `'. * (\d\d\.\d) &deg;F. *'`. The `'. *'` portion of my pattern served the purpose of matching, but not returning each, any and all characters that precede the pattern `'(\d\d\.\d)'`. Each `'\d'` of the pattern would match one numeral. Each temperature value in the E4 HTML readout was in the form of two digits, followed by a decimal point, and then another digit. A period, however, has a special meaning when used in regular expressions, so it became necessary to use a backslash with it to match a literal period. I matched the HTML code segment `'&deg;F'` that is used to draw the degree circle, but did not have this match returned. In this manner, I was able to extract each of the temperature values without accidentally matching any other numerals in the HTML source code.

My program cannot function to its full extent unless the thermocouples are named with coordinates representing a close approximation of its location on a GCC map. In order to determine these coordinates, I needed another pattern to extract the correct coordinate values. The pattern of characters that proved most beneficial was `'(\d\d\d\d\d\d)'`. In this pattern, the `'\d'` serves the same purpose as in the last regular expression pattern. In this manner, the three digits that make up both the x- and y-coordinates of the position of the thermocouple will be found. With the letter x in the middle, I am assured that resulting matches will always be a point rather than just a series of irrelevant numbers.

HTML Output from E4 and E16 Web servers	Regular Expressions needed to Extract Vital Information	Resulting Strings and Lists
<pre>&lt;!DOCTYPE HTML PUBLIC "-//W3C// DTD HTML 4.0 Transitional//EN"&gt; &lt;html&gt;&lt;head&gt;&lt;title&gt;IT Temperature Monitor: GCCa01&lt;/title&gt;&lt;meta http-equiv ="REFRESH" content="60;url=index. html"&gt;&lt;/head&gt;&lt;body bgcolor="#C0C0C0 " text="#000000" vlink="#800080" link= "#0000FF"&gt;&lt;table border=2 cell padding =2&gt; &lt;tbody&gt;&lt;tr&gt;&lt;td align=center&gt;&lt;table cellpadding=2 cellspacing=0 border=1&gt; &lt;TR&gt; &lt;TD&gt;Model:&lt;/TD&gt;&lt;TD width=10 rowspan=3&gt;&lt;BR&gt;&lt;/TD&gt;&lt;TD&gt;E4&lt;/TD&gt;&lt;T D width=20rowspan=3&gt;&lt;BR&gt; &lt;/TD&gt;&lt;TD&gt; Firmware Version:&lt;/TD&gt;&lt;TD width=10 rowspan=3&gt;&lt;BR&gt;&lt;/TD&gt;&lt;TD&gt;3.4&lt;/TD&gt; &lt;/TR&gt;&lt;tr&gt;&lt;TD&gt;Manufacturer:&lt;/TD&gt; &lt;TD&gt; Sensatronics&lt;/TD&gt;&lt;TD&gt;Release Date:&lt;/TD&gt;&lt;TD&gt;July 7, 2004 &lt;/TD&gt; &lt;/TR&gt;&lt;TR&gt; &lt;TD&gt;Website:&lt;/TD&gt;&lt;TD&gt;&lt;a href="http:// www.sensatronics.com"&gt; http :// www.sensatronics.com&lt;/a&gt; &lt;/TD&gt; &lt;TD&gt;Serial Number:&lt;/TD&gt;&lt;TD&gt;EA1G5 C0T143&lt;/TD&gt; &lt;/TR&gt;&lt;/table&gt;&lt;TR&gt;&lt;TD align=center&gt;Unit name: GCCa01 &lt;/TD&gt; &lt;/TR&gt;&lt;tr&gt;&lt;td align=center&gt;&lt;table border =0&gt;&lt;tbody&gt;&lt;tr&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;table border=2&gt;&lt;tbody&gt; &lt;tr&gt;&lt;td&gt;140x530:H- A4-T&lt;/td&gt;&lt;td&gt;165x545:H-A4-M&lt;/td&gt; &lt;td&gt; 140x455:C-A4-T &lt;/td&gt;&lt;td&gt;165x470:C-A4- M&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;90.2 &amp;deg;F&lt;/td&gt; &lt;td&gt;85.3 &amp;deg;F &lt;/td&gt;&lt;td&gt;64.5 &amp;deg;F &lt;/td&gt;&lt;td&gt;64.4&amp;deg;F&lt;/td&gt;&lt;/tr&gt;&lt;/tbody&gt;&lt;/ table&gt; &lt;/td&gt;&lt;/tr&gt;&lt;tr&gt; &lt;td&gt; &lt;/td&gt;&lt;tr&gt;&lt;td align=center&gt; &lt;b&gt; Sensatronics &lt;/b&gt;&lt;br&gt; Environmental Sensing&lt;br&gt;&lt;br&gt;Model E4&lt;br&gt;&lt;br&gt;IT Temperature Monitor &lt;/td&gt; &lt;td&gt; &lt;/td&gt; &lt;/tr&gt;&lt;/tbody&gt;&lt;/table&gt;&lt;/td&gt;&lt;/table&gt;&lt;/body &gt;&lt;/html&gt;</pre>	<pre>page = re.sub('&lt;([a-z]   [A- Z]) +&gt; &lt;/([a-z]   [A-Z]) +&gt;', '', page)  page = re.sub('&lt;([&gt;]+)&gt;', '', page)  temps = re.split('\d\d\d.\d &amp;deg;F', page)  coors = re.split('\d\d\d\dx \d\d\d)', page)  names = re.split('(:[H C] - [A-F] \d\d\d?[T M]) ', page)</pre>	<pre>IT Temperature Monitor:GCCa01Model: E4FirmwareVersion:3.4Manufacturer:Sen satronicsRelease Date:July 7, 2004 Web site:http://www.sensatronics.comSerial Number:EA1G5C0T143Unit name: GCC a01140x525:H-A4-T180x525:H-A4-M140x 475:C-A4-T180x475:C-A4-M79.1 &amp;deg; F84.5 &amp;deg;F65.1 &amp;deg;F65.3 &amp;deg;F Sen satronicsEnvironmental SensingModel E4 IT Temperature Monitor  [IT Temperature Monitor: GCCa01 Model: E4Firmware Version:3.4 Manufacturer:Sen satronicsRelease Date:July 7, 2004 Website: http:// www.sensatronics.com Serial Number: EA1G5C0T143Unit name: GCCa01 140x525 :H-A4-T180 x525 :H-A4-M140x475:C-A4-T 80x475:C-A4-M, '79.1', ', '84.5', ', '65.1', ', '65.3', 'SensatronicsEnvironmental SensingModel E4IT Temperature Monitor']  [IT Temperature Monitor: GCCa01 Model: E4Firmware Version:3.4 Manufacturer: SensatronicsReleaseDate:July 7, 2004 Website: http://www.sensatronics.comSerialNumber: EA1G5C0T143Unit name: GCCa01', '140x525', ':H-A4-T', '180 x525', ':H-A4- M', '140x475', ':C-A4-T', '180x475', ':C- A4-M 79.1 &amp;deg;F84.5&amp;deg; F65.1 &amp;deg; F65.3&amp; deg;FsensatronicsEnvi ronmental Sensing Model E4IT Tempera ture Monitor']  [IT Temperature Monitor: GCCa01Model:E4 FirmwareVersion:3.4Manufacturer:Sen satronicsRelease Date:July 7, 2004 Website: http: //www.sensatronics.comSerial Number: EA1 G5C0T143Unit name: GCCa01140x525', ':H-A4-T', '180x525', ':H-A4-M', '140x475', ':C-A4-T', '180x475', ':C-A4- M', '79.1 &amp;deg;F84.5 &amp;deg;F65.1 &amp;deg;F65.3 &amp;deg;F Sensa tronicsEnvironmental SensingModel E4IT Temperature Monitor']</pre>



### **Merging Data and Image**

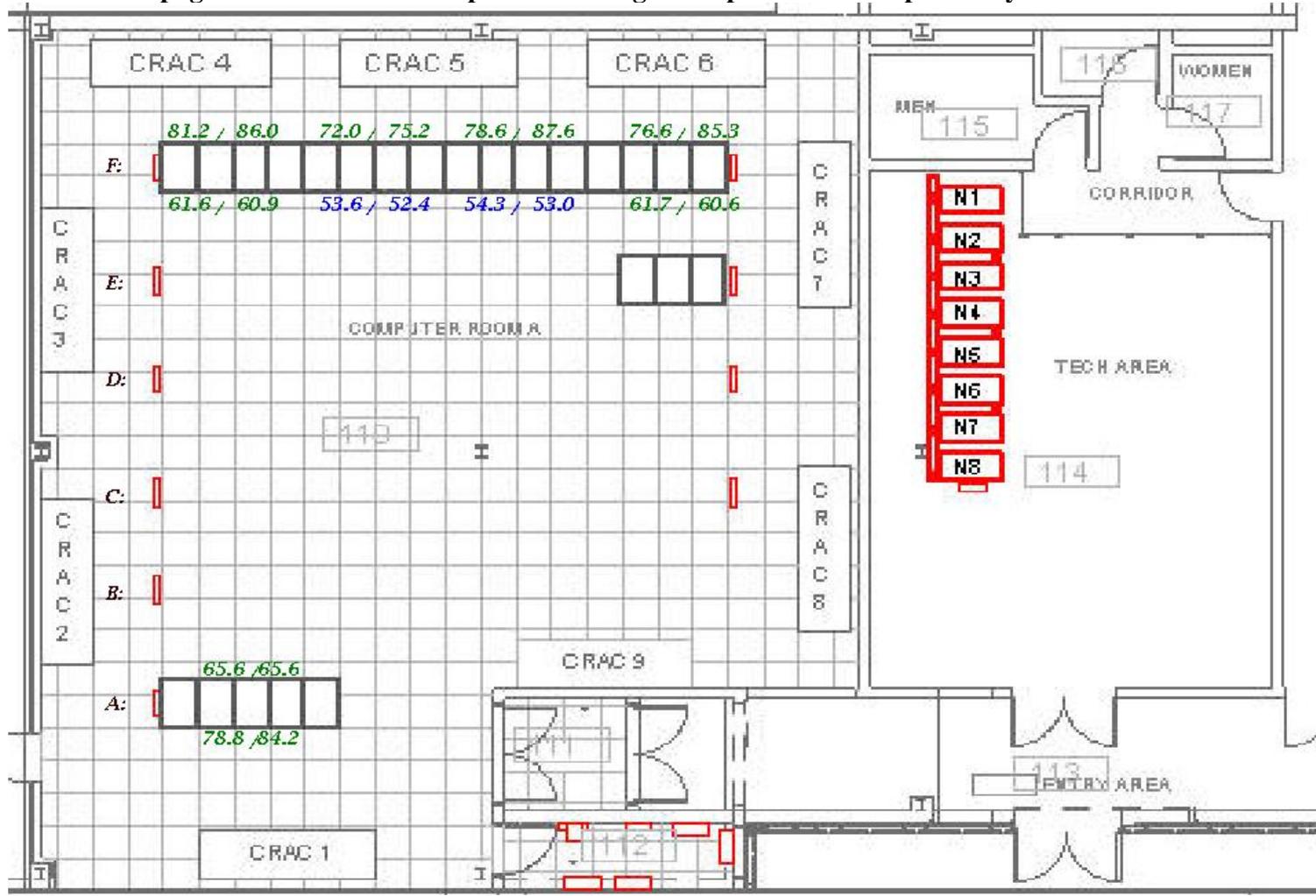
The next step in the program involved using the Image module to open a file containing the floor map of the entire Grid Computing Center building. By assigning it to another variable, I was able to produce a copy of the GCC image as an object that could be modified within the confines of my program. Through a “guess and check” process, I determined the coordinates that specified the rectangular area of GCC that is being monitored. By cropping the image around these corners using Image module routines, I obtained an object that could be resized to perfectly fit in the target PC screen display. My final product had the appearance of the original floormap, but restricted to the data center portion of GCC.

The final, and arguably toughest phase of my project, was to take the string values obtained via the regular expressions applied to the HTML source code, and merge them with the newly cropped image of the GCC floor map that showed only the data center. A “for” loop was used to process each probe’s data, extract the name and temperature values, and then display each at the correct location on the GCC data center map according to its coordinates. In addition, I added an “if-then-else” statement to the “for” loop so that any temperature higher than 90 degrees would appear on the map in red to warn of a high temperature level, temperatures lower than 60 degrees appear on the map in blue, while all other temperature in between appear in green.

### **Benefits of the Project**

With the completion of my project and the impending installation of the entire set of 2800 computers at GCC, the Fermilab CD Operations Group will be able to monitor the temperature throughout the data center. Not only can they determine the temperature of an area, but also the area of the building from which that temperature is originating. It will be much easier to determine the functioning or malfunctioning of both computers and the air ventilation systems. In addition, this determination can be done remotely without the need to visit the data center. The program also is adaptable, allowing for the easy addition or removal of sensors from the program and, thus, from the map display, as illustrated by the E16’s installation, which will required only minor modifications to the program itself. As more thermocouples and E-microcomputers are needed and ordered, each microcomputer will be assigned a different IP address, making the monitoring of all of the temperatures at once difficult to accomplish. My program will improve upon this by retrieving each IP address and displaying the information collectively, so as to be easier to interpret. The program will have great value for the entire time that the GCC data center is in operation.

The web page of the GCC floor map with its merged temperatures as reported by the E4 and E16 units.



## **Conclusion**

I came to Fermilab with some knowledge of the programming procedures and syntax of the C++ language. I was not aware of many of the computer languages that I have encountered here. Now I have a fair capability of programming in the Perl language. Although I am not as skilled as I would like, I have enough background and practice to accomplish many tasks of programming in this language. This knowledge will prove most helpful as I continue my study to become a Computer Engineer. Also, my summer project has also taught me a great deal about the Python language. I have written simple programs. The programs that I constructed in accordance with my project are straightforward, yet quite effective, and took quite a bit of preparation, investigation, and frustration to complete.

I have learned volumes about the value of hard work and deep thought, and the ever-important ability to depend on and learn from others. I can write functions, use methods, save files, manipulate pictures and images, and create internet-accessible html source code within a Python program.

More importantly, I've learned about the Fermilab facility. I was previously unaware of its existence. Through the weekly lectures given by Fermilab scientists, I know the uncertainties of dark matter and dark energy-concepts that were completely foreign to me. I know that quarks are still not the smallest particles possible, and the amount of work required to view these miniscule particles for even a few seconds is tremendous. I learned that some cancers can be treated with neutrons, and that the process is ongoing to improve this procedure. I also learned that scientists are normal people, too, and that they play sports, lift weights, have picnics, and possess a sense of humor.

### **Degree of Project Success**

I felt that my project was immensely successful. I am overjoyed to have accomplished the objectives set forth by my supervisor. Not only that, but it was able to effectively handle every temperature probe that my supervisor desired to install. As the project progressed, I was also able to modify the program to meet other requirements that my supervisor requested.

I believe that because of its successful execution, the program will be of great benefit to many others who are interested in monitoring the temperatures of GCC can make more use of my program, also. My supervisor gave me specific reasons for the project that he had me do. I believe that I have answered all of his reasons with the execution of my program.

## **Acknowledgements**

I owe the majority and most prominent of gratitude to God for allowing me the opportunity to have such an experience so early in my college education. It is a strict blessing to have heard of this wonderful program, and to have been able to actually participate in such an enriching endeavor.

I am eternally grateful to Mrs. Diane Engram, Mr. Elliot McCrory, and the rest of the SIST selection committee for allowing me the opportunity to learn first-hand at an institution with the prestige and prominence that only Fermilab carries with it.

I owe a major debt of gratitude also to Dr. David Ritchie, my summer supervisor. He helped me write a complete, functioning program in a language that I did not even know existed. Dr. Ritchie was always there on the many occasions that I had become stuck, or simply lost. He'd always point in the right direction. Dr. Ritchie went beyond the call of simply supervisor by helping me write my report. His suggestions and comments were vital to the construction of the document you now read. In an unprovoked manner, Dr. Ritchie also offered me assistance

on my presentation, helping me in the summarization process and sitting through my auditions. Neither of the latter two actions are part of any supervisor's job requirements, but Dr. Ritchie decided himself to help me advance myself and truly speak what I wanted to say, and for that, I thank him.

Thank you to Dr. Davenport, who also provided guidance and insight into the formation of my report. I am happy that he was willing to speak to the students informally, and was nice enough to discuss everyday interests and places. Thank you for all of the knowledge that you have instilled in me.

I also thank Orlando and Tim, employees in the Computing Division. Without their help, it would have been much more difficult to get my programming functioning. Thank you to everyone else in the Computing Division for the kind greetings and food gatherings.

I must also thank Ryan Shelby, the president of the Honor Society at Alabama A&M University. Not only did he introduce me to this awesome program, but he also constantly encouraged me to apply. In the end, I cannot think of a better way to have spent my summer, and, had it not been for him in particular, I would not have gained so vast an amount of knowledge. Thank you also to Eric Rivera, a fellow intern under Dr. Ritchie's supervision. For the assistance in installing the temperature monitoring equipment and simple conversation in the office, I am grateful.

Last, but certainly not least, I want to thank my fellow SIST interns. I have learned nearly as much from you as I have from the internship project itself. The new experiences and friendships that I have forged this summer will change me forever. You made everything perfect.

## **References**

Grayson, John. Python and Tkinter. Manning Publications Co., Greenwich, CT, 2000

Harms, Daryl; McDonald, Kenneth. The Quick Python Book. Manning Publications Co., Greenwich, CT, 2000

Lutz, Mark. Programming Python. 2<sup>nd</sup> Edit. O'Reilly & Associates, Inc., Sebastopol, CA, 2001

--, Ascher, David. Learning Python. O'Reilly & Associates, Inc., Sebastopol, CA, 1999

Wall, Larry; Christainsen, Tom; Schwartz, Randall. Programming Perl. 2<sup>nd</sup> Edit. O'Reilly & Associates, Inc., Sebastopol, CA, 1996

<http://www.fnal.gov/pub/about/whatis/history.html>

<http://history.fnal.gov/discover.html>

## Appendix

A copy of my program, in its entirety:

```
"""
This program reads the temperatures from a web server
and merges them with a map of the computing room in which the
temperature sensors are located.
"""
"""
Import Modules
"""
from Tkinter import *
import os, sys
import Image
import urllib
import re
import ImageDraw
import ImageFont

"""
I.      Here we define both Global Data and the internet address of the web
        servers with the temperatures.
"""

aTempURL = "http:// w.x.y.z /"
bTempURL = "http:// w.x.y.z /"

Note: the actual IP addresses have been removed per local policy requirements.

"""
When the other E16's and thermocouples are ordered and installed, these will
likely be the catalyst
for defining the additional IP addresses.
cTempURL = ""
dTempURL = ""
eTempURL = ""
"""

URLS = []
URLS.append(aTempURL)
URLS.append(bTempURL)

"""
In a similar manner, the additional E16's IP addresses can be added to the URLS
list with these commands:
URLS.append(cTempURL)
URLS.append(dTempURL)
URLS.append(eTempURL)
"""

"""
II.     Here is where we define the Main module's code.  This is also where
        the definitions of helper classes and functions are placed.
"""

def main():

    """
    III.  Read in the floor map.
          1.  Crop the map.
          2.  Resize the map to fit the window.
    """
```

```

pic = Image.open("GCCmap.jpg")
floor = pic.crop((375,300,850,650))
view = floor.resize((957,647))

for itemURL in range(len(URLS)):

    """
    IV.      Use the urllib module's capabilities to read the index page
            provided by the web server.
    """

    fp = urllib.urlopen(URLS[itemURL])
    page = ""
    NumberOfReads = 0
    n = 0
    while 1:
        s = fp.read(8192)
        if not s:
            break
        NumberOfReads = NumberOfReads + 1
        page = page + s
        n = n + len(s)
    fp.close()

    """
    V.      Process using regular expressions the information coming back
            from the web server to extract the
            text strings of the names and values of the temperatures. The
            format of the name will be
            hhhxvfv: [H|C] - [A-F] \d\d? - [T|M] where hhh is the horizontal
            coordinate from 0 to 900 and vvv is
            the vertical coordinate from 0 to 700. [H|C] designates
            whether the probe on the hot or cold aisle.
            [A-F] is the computer aisle in GCC (could be Aisle A, Aisle C,
            or Aisle F). \d\d? is the number of
            the rack in each aisle (i.e. D7 is the seventh rack in Aisle
            D, while F16 is the sixteenth rack in
            Aisle F). [T|M] tells whether a thermocouple is located at Top
            or Middle of a rack (i.e. E3-T means that the thermocouple is
            at the top of the third rack in Aisle E.
    """
    """
    VI.     Remove all HTML from the web page text returned
            1. Extract the temperatures from the web page.
            2. Extract the coordinates of each probe from the web page.
            3. Extract the names of the probes from the web page.
    """

    page = re.sub('<([a-z] | [A-Z])+>|</([a-z] | [A-Z])+>', '', page)
    page = re.sub('<[^>+>', '', page)

    temps = re.split('(\d\d.\d) &deg;F',page)
    coors = re.split('(\d\d\d\d\d)',page)
    names = re.split('(: [H|C] - [A-F] \d\d? - [T|M])', page)
    print "page is:\n", page, "\n"
    print "temps are:\n", temps, "\n"
    print "coors are:\n", coors, "\n"
    print "names are:\n", names, "\n"

    """
    VII.    Merge the temperature value with the map at the correct spot
            on the map according to the
            coordinates in the temperature name.
            1. Make an object named draw that we can use to draw on the

```

```

        image object named view
    2. Make a font info object by loading the font file
       lubBI10.pil.
    """

    draw = ImageDraw.Draw(view)
    ifo=ImageFont.load("lubBI10.pil")

    """
    The temperatures that are displayed on top are recorded from the
    probes that are on top.
    """

    draw.text((70,110), 'F:', fill=50, font=ifo)
    draw.text((70,195), 'E:', fill=50, font=ifo)
    draw.text((70,265), 'D:', fill=50, font=ifo)
    draw.text((70,350), 'C:', fill=50, font=ifo)
    draw.text((70,420), 'B:', fill=50, font=ifo)
    draw.text((70,500), 'A:', fill=50, font=ifo)

##
##     for item in range(len(temps)/2):
##         if float(temps[2*item+1]) > float(90.0):
##             textcolor = "red"
##         elif float(temps[2*item+1]) > float(60.0):
##             textcolor = "green"
##         elif float(temps[2*item+1]) < float(60.0):
##             textcolor = "blue"
##
##         probenames = names[2*item+1]
##         topormiddle = re.split('.*(T|M$).*', probenames)
##
##         xcoor = re.split('(\d\d\d)x\d\d\d', coors[2*item+1])
##         ycoor = re.split(' \d\d\d(\d\d\d)', coors[2*item+1])
##         toptemp = temps[2*item+1]+'/'
##         if topormiddle[1] == 'T':
##             draw.text([int(xcoor[1]), int(ycoor[1])], toptemp,
##                       fill=textcolor, font=ifo)
##
##         else:
##             draw.text([int(xcoor[1]), int(ycoor[1])], temps[2*item+1],
##                       fill = textcolor, font=ifo)

    """
    VIII. Write the image out as a jpg file so that it can be viewed by a
    web browser.
    """

##     view.show()
##     view.save("GCCupdate.jpg")
    return None

#-----#
#-----#
    """
    IX. Initiates execution of program if the program definition is named
        'main'
    """
    if name == '__main__':
        main()

    else:
        pass

    """
    X. THAT'S ALL, FOLKS
    """

```

'''