FINAL REPORT

CENTRALIZING AND VIRTUALIZING SMART HOME TECHNOLOGY

PROJ354

APRIL 12 2013

ASHLEY KIERAN
JEFF PERRY
RICHARD THEN

TABLE OF CONTENTS

EXECUTIVE SUMMARY

In this report, we will discuss and outline all the necessary key points to understanding the nature of our project. Centralizing and Virtualizing Net-Zero Home Technologies refers to the over-arching objective we have completed with this project, which is essentially to upgrade, stabilize, and document IT infrastructure created by our team for a research home built by the Green Building Technologies division inside SAIT's ARIS. The home in question uses a variety of alternative forms of energy production to power itself year round. This concept is referred to as a "Net-Zero" framework. The IT infrastructure of the house is dated and malfunctioning in some regards, and we have remedied this by building a newer system that mimics the functionality of the old one, albeit in a more stable, safer, and faster environment.

Also in this document we have included the primary stakeholders/parties affected by this project, as well as the project team itself, the scope of our work (as well as what falls outside of our scope), lessons learned and recommendations for future project teams, a breakdown of our budget, and finally our conclusions.

Appended to this report, you will find a glossary of important terms, a Gantt chart detailing the workflow of our project, the User Manual we have created for future maintenance, and the Python scripts that are being used on the system (as created and deployed by the original project designer) with our commentary and analysis on them.

PROBLEM / OPPORTUNITY

The current system is no longer functioning as it once was, and if use is continued there is a chance it will out-right fail. Seeing as the house is inhabited by a full time resident and is an ongoing research project, this is a scenario that needs to be avoided. There also seems to be some unnecessary redundancy in the design that may have been created at the time of the project's original inception due to technological constraints, but the capacity to remedy this exists with the technology we have installed. In essence, the system needs to be updated while still maintaining all of its original functionality.

OBJECTIVES

- Design a system that mirrors the functionality of the current system.
- Upgrade and condense (where possible) the hardware being used by the home.
- Create a snapshot of present running systems and convert it into a virtual environment.
- Ensure the system is functional and communicating properly.
- Ensure all documentation has been completed for system maintenance, as well as have a User Guide built. Documentation of Python scripts is also required.
- Ensure system is prepared for implementation in the home itself by project's end.

BACKGROUND

Discovery 4 was built by SAIT'S very own ARIS in September, 2010. [1] In terms of the IT infrastructure that was put into place (the focus of our project), it was all designed by a single student for his own Capstone project. The system was designed in a test environment, and implemented into the house upon its completion. Since then, the house has utilized the system to keep the various components working together. Unfortunately the technology is beginning to show its age, and certain pieces of it are beginning to malfunction or aren't working properly anymore. This system has not been meeting the long term requirements that it was originally intended to. The system has also not been virtualized and thus has no means to restore information that may be lost.

The original project implementer used Python as a scripting language to design scripts to pull information being processed by the system. As a result, some knowledge of how Python functions was necessary to understanding what was being accomplished by these scripts. As none of us have any formal training with Python, we have done the best we can to teach ourselves, and as a result our documentation of the scripts should be reviewed from this perspective.

PROJECT TEAM AND PRIMARY STAKEHOLDERS

Stakeholders	**Comment**
Project Team	Ashley Kieran, Jeff Perry, Richard Then
Client	Kevin Kowal
Performing Organization	Team Wallflower
Sponsor	ARIS (via Joe Petermann)

In addition to the primary stakeholders, we also have:

Stakeholder	**Role or Influence**
Homeowner	Owner of the Discovery 4 home
Colin Chamberlain	Project Mentor
Jim Catley	Chair of School of ICT at SAIT
SAIT Polytechnic	Institution hosting the project

PROJECT SCOPE

SCOPE

Given that we had to adhere to a rigid timeline, and that we were working to upgrade a system as opposed to developing a new system entirely, we had to work within those constraints. Phases for the project were as follows:

Phase	Deliverable
1 – Planning	Finalized Plan for Development
2 – Building the System	Hardware is Functional and Communicating
3 – Setting up Software	Software Installed on System and Functioning
4 – Analyze Python Scripting	Converted to Perl or Updated Python Scripts
5 – Documentation	User Guide and Formal Report

OUT OF SCOPE

We did not build on or reworked anything when it came to the software being run by the touch screen that is used by the home owner to monitor the house's various components. We also did not do anything with the database that the sensors are pumping info into (there is a Software Development Capstone team working on this).

PROJECT DETAILS

When we took on this project, the IT infrastructure of the house was fairly mangled and motley. Essentially there were three PC's running that each performed a role in the system:

1) The "server" was an old IBM Thinkcentre running Windows XP. This is the heart of the IT infrastructure. Its primary purpose was to orchestrate the components of the house that required some computer interaction (such as the HBX units) [1], as well as host the database that all the sensor data gets fed into. The server needs to be run 24/7.
2) The client PC was an HP Compaq desktop model that essentially just ran the touch screen interface. The software that it runs was designed by the original project architect. It also runs Windows XP.
3) Lastly, there is a laptop that is communicating with one specific set of the sensors (the WiDAQs specifically). The laptop is part of a proprietary package from a company named SMT, and this is why it is system on top of the server. Since this piece of the system is very particular, we have left it as is for our project and focused on the above two computers.

While the system seems easy enough to grasp initially, the reality looked something like this: [1]

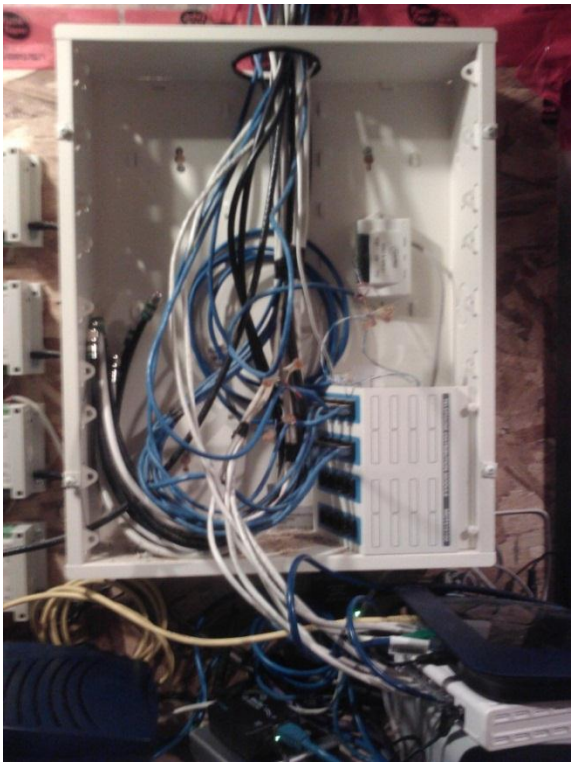

Figure 1a [1]

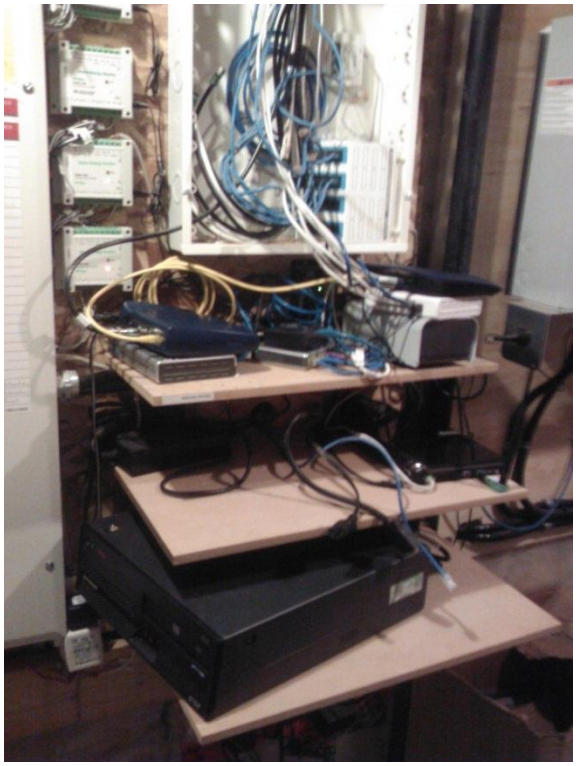

Figure 1b [1]

Network Cabling

On top of rebuilding the system, we were also asked to build a back-up mechanism into it to prevent a critical failure from occurring. There was pretty much nothing in the way of a hardware failsafe, so if at any point any of the computers failed, the system would begin to malfunction. Seeing as the server needs to be running all the time, it is fairly apparent there is a potential problem here. This is where the virtualization aspect of our project came into play. Virtual machines (which from here on out will be abbreviated to VMs) made the most sense to us due to the flexibility they offered. The concept of "snapshots" provided a way for us to capture and preserve the state of our systems, and we could access and revert back to these snapshots at any point in time if the system began to malfunction due to something we did to it. The same concept could be applied if somehow the system suffered a catastrophic failure, thus we have a way of

backing up the software side of things. Hardware back-up is provided via our uninterruptable power supply (UPS). [6] The original system did not employ a UPS, and we felt it was a flaw that could lead to some problems.

Using VMs also presented us with a way to centralize the entire setup in a better way than what was already implemented. By creating a VM out of both the server and touch screen PC [2], we were able to run both from a single hypervisor. Hence, we no longer needed to have a computer for both the server and the touch screen machine; they could be run from the same machine. This brings us to the centralization aspect of our project.

We decided to use an HP DL385 G7 server [4] as our host machine because it was the server model we had been working with in our lab classes, and thus we were very comfortable with how it worked. The server was a vast improvement hardware-wise over any of the computers in the house, and it also supported serial port functionality, which was a necessary component our client had stressed. We put five 300 GB hard drives into the server, and put them in a RAID 5 configuration for drive failure protection. The available memory this provided was a lot more than what the system needed to function, but we wanted to leave room for expansion down the road.

Our client had also asked us to leave the operating system on the host machine as a Windows based one, because it was what he was most familiar with. We decided to go with Windows Server 2012 for two reasons: 1) it contained Hyper-V 3.0 [8], which we had planned to use as our hypervisor, and 2) it had the latest in security technology, something we also wanted to put some focus on. Since Server 2012 looks and feels a lot like Windows 8 (an operating system we were not overly familiar with), we wanted to make it feel a little more familiar. We used a third party add-on called Classic Shell [9] that allows for some nice modification of the user interface, and namely we used this to create a “Start” button (the button in the bottom left-hand corner of Windows user interface). Windows Server 2012 does not contain this by default, and this is the same for Windows 8. This may seem like a fairly trivial thing, but it was something that we felt was a staple of familiarity and functionality.

Once we had the spine of the system built, we went through the usual steps from there, such as stress testing and installing the server into our rack. We conducted our stress testing using a program called Prime95 with a custom test to utilize the majority of our RAM. [10] From there, the next logical step was to capture hard drives of the computers running in the house and virtualize them, but we chose to do several test runs of the procedure before going out to the house to capture the actual drives. To do this, we used a Sysinternals tool called “disk2vhd”, which converted physical hard drives into virtual ones. [2]

Around this time, we began work on a couple other aspects of the project. We were required to build a website (which can be found on the DVD attached to this report), so work was commenced on that. We were also asked to take a look at the Python scripts that were developed for the original project [1], as some of them were generating errors on a regular basis on the system. Unfortunately, none of us had any real knowledge of how Python worked, so we had to do a little research on our own to get familiar with it. We had expressed to our client that, if we thought it was possible, we would convert these scripts over to a language we were more comfortable with: Perl. However at the time, we did not have access to the code, and severely underestimated how much of it there was. We quickly settled on going through it all and documenting it to the best of our abilities, because the original author had only partially done this. The results of this procedure will be discussed in the next session, and the documentation (along with the scripts) is appended to this report for your convenience.

Once we were able to get into Discovery 4, we managed to convert both the server and client PCs into virtual hard drives (or vhd for short). Next was the process of setting up our hypervisor, which in our case was Server 2012’s Hyper-V 3.0 [8]. Upon initially booting up the VMs inside Hyper-V, we noticed that they were running very slowly even with the Hyper-V integrated tools installed, much to our confusion. We soon discovered that Hyper-V didn’t like to play nice with either Windows XP or AMD processors, which is what our server was using. Some tweaking of the system needed to

occur, and we opted to do this through a driver provided by AMD called AMDV Opt. [12] This substantially improved performance in the virtual machines.

At this point in time, we hit what was arguably the biggest problem point of our project. We went to test serial port functionality in the VMs using our UPS (which conveniently communicated through serial), but Hyper-V had no option to let us use a serial port inside a virtual machine. After some confusion and research, we realized much to our horror that every version of Hyper-V 2.0 and up did not support physical serial ports being used by virtual machines. This was an incredibly big problem for us as a lot of the components in the house that needed to be hooked up to our system upon implementation communicated via serial ports. Seeing as the system was now entirely virtualized, we wouldn't be able to have any communication between the house's hardware and our virtual machines.

There appeared to be a way to get the serial ports talking to the VMs by using "named pipes", but this was a very foreign concept to us and we were unable to grasp what needed to be done in any kind of meaningful way. We tried a few other approaches, such as utilizing third party software to send the serial port data over TCP/IP or UDP and doing that all over an isolated network, but our numerous attempts proved fruitless. [13][14] We were now considering completely switching hypervisors, which would render a lot of our previous work obsolete. On top of that, our choices of available hypervisors were narrowed due to the necessity of the host system remaining Windows based (ie. We could not employ a Linux type setup or a bare-metal hypervisor; it had to be something that would run on top of Windows Server 2012).

Ultimately, the use of VMware Workstation proved to be our best bet. The primary problem with switching to that hypervisor was that our virtual hard disk files that we captured from the house were not directly supported by VMware Workstation, and we did not have enough time left to set up a time to go back to the house and re-extract the systems in a different way. We ended up using a converter called "WinImage" [17] to change our .vhd files to a "Fixed Sized Virtual Hard Disk" and save them as .vmdk formats, which would then be interpreted correctly by VMware Workstation. There was another option called "Dynamically Expanding Virtual Hard Disk" which seemed like the better candidate, but every time we tried to load an image that was classified as this, it would fail.

Finally, we had to establish how the networking aspect of the project was going to work. Since we could not really manipulate how the network was set up, and we had to essentially hot-swap our system into the network come implementation time, we had to keep things pretty much as they were and work our system into it. For the majority of the project we had assumed that the networking aspect was done through static IP addresses, and we knew this would be one of the last steps, so we never put much thought into it. We found ourselves very stumped when we discovered there was no static IP set on either the server or the touch screen PC. How would info be passed from the server to the touch screen PC if neither knew where to find the other at all times? The answer to our problem was in the router settings, as well as the network map we were provided. It took us a little while to connect the dots, but we determined that the MAC addresses were being used to reserve IP addresses through the router's DHCP propagation. We had some suspicions that the software, DynDNS, might have had something to do with the network, as the software was being run on the server. However we later figured out that this piece was actually responsible for the VNC access, which our client was using to remotely access the server. Now that we are using Server 2012, our client can RDP into the host machine (and thus the VMs), so VNC access will no longer be necessary.

BUDGET

Our proposed budget at the project's inception:

Budget Item	Estimated Cost
HP ProLiant DL385 G7 Server	$3,398.99
8GB DDR3-1333MHz	$109.99
Client PC	$789.99
CPS1500AVR UPS	$379.95
Netgear ProSafe 24-Port Switch	$392.85
Supermicro Server Rack Cabinet	$865.24
Miscellaneous cables, adapters, taxes and shipping costs	$562.99
Software (Operating Systems, Licensing, etc.)	$1000
TOTAL PROJECT BUDGET	**$7500**

We were also asked to calculate a theoretical budget for labor based on how many cumulative hours would be worked on the project, and what that would cost if we were charging approximately $100 per hour.

Item	Hours	Rate	Cost
Cumulative Team	265	$99.95	$26,486.75

Our budget at the project's completion:

(see next page)

Team Wallflower Budget

Item	Description	Price	Qty.	Total
2PORT PCIE SERIAL CARD DUAL PORT SERIAL RS232 CARD 16950	Serial port adaptor	$ 44.00	2	$ 88.00
Win Svr Std 2012 2 Proc - Select Plus Academic	Windows Server License	$ 233.88	1	$ 233.88
24PT GBIT STK SMART SWCH	24 port switch	$ 411.00	1	$ 411.00
1500VA UPS SMART APP AVR RM-T 2U 6OUT RJ11/45 5-15P 3YR	UPS (Uninterruptable Power Supply)	$ 316.00	1	$ 316.00
DL385 G7 6234 2P 16GB SVR-SBY	Server	$ 3,279.15	1	$ 3,279.15
RK1236BKF 12U ENCLOSED RACK 36IN W/ DOORS BLACK	Server rack	$ 542.00	1	$ 542.00
3FT BLACK SNAG-LESS CATEGORY 6 PATCH CABLE -ETL VERIFIED	Network/patch Cables	$ 3.00	20	$ 60.00
SMARTBUY 300GB SAS 10000 RPM 6G 2.5IN DP ENT HD	Hard drive	$ 245.00	5	$ 1,225.00
VMWare	VMWare Workstation 9	$ 249.00	1	$ 249.00
Cables To Go Model 27901 3ft. 18 AWG Universal Flat Panel Power Cord	Power cord	$ 3.99	2	$ 7.98
				$ 320.60 Taxes
				$ 11.04 Additional Shipping Costs
				$ 6,743.65 Grand Total
				$ 756.35 Budget Remaining

And our theoretical labor came to:

Team Wallflower Budget

Item	Hours	Rate	Cost
Cumulative Team	188.6	$ 99.95	**$18,850.57**
			$26,486.75 Budgeted For
			$7,636.18 Budget Remaining

Notes about the Budget:

- Initially, our target budget was closer to $7000 than $7500. The rationale behind the additional funding was primarily in potential licensing issues. At the project's inception, we were under the impression that the majority of the licensing costs would be covered by SAIT as our project was being conducted under their banner. However, whether this was true or not could not be determined at the time of our budget submission, and thus we decided to plan for the worse. As it turned out, the operating system license we ended up using was not covered by SAIT.
- The inclusion of a "Client PC" on the initial budget was meant to replace the touch screen interface computer in the home. The design employed by the original implementer used a standalone PC alongside the server. We ended up modifying this by virtualizing the "Client PC" and condensing it all on our host machine's drives.

DISCUSSION OF RESULTS AND ACHIEVEMENTS

At the conclusion of this project, we have:

- Assembled a system setup that meets our requirements
- Successfully virtualized all the aspects of the system that we could
- Created a backup and contingency plan for the infrastructure
- Documented and made recommendations for optimization of Python scripts
- Developed a user manual for continued project maintenance and accessibility

We felt the project had its challenging moments, but also its victories (much like any big project). Initially, our biggest concern was timing. We were unable to order our equipment until January due to funding issues, and the hardware in question did not arrive until mid-February. That effectively removed a month and a half of our project time, at least in terms of building the system. We spent most of that period playing around with our own copy of Windows Server 2012, as well as studying the scripting language Python so we could understand what was happening in the scripts.

We also initially thought the process of converting a physical drive to a virtual one would be much more difficult than it actually proved to be. We tried a Microsoft converter as well as a VMware converter [3], but ultimately the Sysinternals tool we used (disk2vhd) met our needs. [2] The process is as simple as inserting an external hard drive and pointing the tool at what drive you want to convert. We experimented with this quite a bit during this initial period so that we were prepared to capture the images when we got our hardware.

When our hardware did finally arrive, and we got everything set up, it mostly came down to figuring out how our hypervisor was going to interact with everything. As discussed earlier, the majority of our time was spent getting Hyper-V to interact with our VMs properly, which inevitably did not play out. Our solution was to use VMware Workstation in the end because it enabled us to have serial port functionality, something that was absolutely essential for our project.

Our backup plan was focused on the hardware and software aspects of the system versus the database side of things. The Python scripts indicated that database backup was already happening, and interaction with the database was not within our scope so we did not look any further into that aspect. The snapshots we have taken of our virtualized server and touch screen PC can be reverted back to at any time if the system becomes corrupt or begins to malfunction in any serious way. We have outlined how to take a snapshot in our User Manual so that a backup image can be created at any point in time. Hardware failure is prevented through our UPS. If there is a power failure, the UPS is set to run for 45 minutes before shutting down the systems, and an alert email is sent to discoveryhomeserver@gmail.com.

The Python scripts were documented to the best of our abilities, and the documentation and scripts themselves are appended to this report. Inside the documentation we have provided a recommendation for cleaning up the scripts, though this mostly deals with removing the scripts that we think are not doing anything anymore. We have created an image of the finished server prior to any script removal to be reverted back to if our recommendation proves to cause more problems than fix them.

Lastly, a user manual has been created that is intended for future maintenance of the project. The manual contains steps on how to do everything we did to get the server up to where it is now, as well as the login information of various different pieces of the system, and the steps necessary to recover from a hardware or software failure.

LESSONS LEARNED

- Hyper-V 2.0 and up does not support physical serial port access for virtual machines. It also does not support USB devices in virtual machines. Though this problem did not really affect us, it is something to keep in mind.
- You can mount a .vhd as a hard drive in your host machine, much like you would mount a physical drive
- VMware Workstation 9.0 does not boot .vhd files, nor does Oracle's VirtualBox [11]
- To boot .vhd files using VMware Workstation, you must convert it to a "Fixed Sized Virtual Hard Disk" and save it as .vmdk formats. "Dynamically Expanding Virtual Hard Disk" does not appear to work, at least not for our purposes.
- To install the Standard version of Windows Server 2012, we had to first install the Data Centre version and install Standard on top of that as our disc image was not a bootable one. This caused issues with activation as the MSDNAA activation and the official OEM server authenticate in two different manners. We had to uninstall all product keys and reinstall with the Standard 2012 product key to get it to work.
- You can reboot a machine in RDP by opening up command prompt and typing "shutdown /r"
- To repartition the disc size of a .vmdk file, we first had to extend the volume with VMware Workstation, and then install EaseUS Partition Master Home Edition to extend the XP file system drive. [16] XP will not allow you to extend the file system drive with integrated tools installed; you have to use third party software.
- Networking inside the Discovery 4 Net-Zero home is done through the integrated DHCP reservations (via MAC addresses) as opposed to setting static IP addresses within the operating system.

RECOMMENDATIONS

- If your system is using serial ports in a virtual environment, do not use Windows Hyper-V. There are work-arounds that exist, but they are incredibly complex.
- If you plan to update or redesign a system, having communication with the person or team that designed the original system will prove greatly beneficial. Thorough documentation can, to some extent, remedy this as well. In other words, have a full understanding of the system before you begin to work on it.
- Have a full understanding of how you are being funded.
- Document and time stamp everything you do for accountability.

If we were to redo this project, the biggest thing we would stress would be thoroughly researching the system before setting out to work on it. In our case, the documentation we were given to work with felt really scattered, and it made having a cohesive understanding of the system something difficult to obtain initially. There were a lot of little surprises we kept discovering as we went through the work, and ultimately had we had a better understanding of how the system was working we could have avoided some unnecessary steps. Another point we would really stress is extensively looking into the hypervisor you plan on using. Something that may seem like an obvious inclusion or feature may not actually be part of the package, and it is much better to discover these things prior to doing a bunch of work with the system and then having to undo it, much like what we had to do.

CONCLUSION

Overall, we hope our project can become an integral part of the Discovery 4 infrastructure. We feel like this upgrade was a necessary step forward for the house, and we enjoyed having a part to play with this incredible building. As a group, we feel working on a project that was environmentally mindful has been very refreshing, and the sustainable values that the home embraces are very much aligned with our own. We have learned a lot from this project, both in the realm of IT but also in the realm of project management. We hope to take the knowledge we have learned over the past few months and use it for our work down the road, as we believe sustainable-minded technology is the key to the future.

BIBLIOGRAPHY

[1] Sait, *Discovery 4 Home Documentation,* Calgary: Sait, 2009.

[2] M. Russinovich and B. Cogswell, "Windows Sysinternals," Microsoft, 14 October 2010. [Online]. Available: http://technet.microsoft.com/en-ca/sysinternals/ee656415.aspx. [Accessed 10 April 2013].

[3] VMware , "Download VMware vCenter Converter Standalone 5.0," VMware , 25 October 2012. [Online]. Available: https://my.vmware.com/web/vmware/info/slug/infrastructure_operations_management/vmware_vcenter_converter_standalo [Accessed 10 April 2013].

[4] "HP ProLiant DL385 G7 Server - Models," HP, 2013. [Online]. Available: http://h10010.www1.hp.com/wwpc/us/en/sm/WF25a/1 15351-3328412-241644-241475-4132832.html?dnr=1. [Accessed 10 April 2013].

[5] Netgear , "Netgear ProSafe 24 Port GigabitStackable Smart Switch GS724TS," Netgear , 2013 . [Online]. Available: http://www.netgear.com/business/products/switches/stackable-smart-switches/gs724ts.aspx. [Accessed 10 April 2013].

[6] cyberpowersystems, "PowerPanel Business Edition," Cyber Power Systems, 2013. [Online]. Available: http://www.cyberpowersystems.com/products/management-software/ppbe.html?selectedTabId=resources&imageI=#tab-box. [Accessed 10 April 2013].

[7] cyberpowersystems, "CPS1500AVR," Cyber Power Systems, 2013. [Online]. Available: http://www.cyberpowersystems.com/products/ups-systems/smart-app-ups/rackmount-lcd/CPS1500AVR.html. [Accessed 10 Ap 2013].

[8] Microsoft , "Microsoft Hyper-V Server 2012," Microsoft , 2013. [Online]. Available: http://www.microsoft.com/en-us/server-cloud/hyper-v-server/default.aspx. [Accessed 10 April 2013].

[9] Classic Shell, "Classic Shell," Classic Shell, April 2013. [Online]. Available: http://www.classicshell.net/. [Accessed 10 April 2013].

[10] Mersenne Prime Search, "Prime95 (64bit) - 25.11," extremeoverclocking, 26 March 2010. [Online]. Available: http://files.extremeoverclocking.com/file.php?f=205. [Accessed 10 April 2013].

[11] virtualbox, "virtualbox," oracle, 15 March 2013. [Online]. Available: https://www.virtualbox.org/. [Accessed 10 April 2013].

[12] AMD, "Low I / O performance and / or increased CPU utilization in a virtualized Windows® XP 32-Bit System," AMD, 7 January 20 [Online]. Available: http://support.amd.com/us/kbarticles/Pages/WinXPVirtualizationHotfix.aspx. [Accessed 10 April 2013].

[13] eltima, "Serial Over Ethernet," Eltima, 2013. [Online]. Available: http://www.eltima.com/products/serial-over-ethernet/. [Acces April 2013].

[14] ABC-Deploy, "Automated Windows Client Management," ABC-Deploy, 2009. [Online]. Available: http://abc-deploy.com/. [Acces April 2013].

[15] goodjobsucking, "Hyper-V Serial Ports and Windows 2008," Good Job Sucking., 2 February 2010. [Online]. Available:

http://www.goodjobsucking.com/?p=214. [Accessed 10 April 2013].

[16] CNET staff , "Easeus Partition Master Home Edition," CNET, 29 August 2012. [Online]. Available: http://download.cnet.com/Ease
Partition-Master-Home-Edition/3000-2248_4-10863346.html. [Accessed 10 April 2013].

[17] WinImage, "Download WinImage," WinImage, 2013. [Online]. Available: http://www.winimage.com/download.htm. [Acces
April 2013].

TERMINOLOGY

AMD – Stands for "Advanced Micro Devices." AMD develops computer processors.

ARIS – "Applied Research and Innovation Services" abbreviated. It is the research branch of SAIT Polytechnic, and is the organization our project has been completed for.

Bare-Metal Hypervisor – Refers to a hypervisor that can be run without an operating system. For more information, see "Hypervisor."

D4 – Short for "Discovery 4", which is the designation given to the home our project is centered around.

DHCP – "Dynamic Host Configuration Protocol." Used to connect hosts over a network using IP addresses.

GB – Short for Gigabyte, a unit of data measurement.

Hypervisor – Acts as a monitor and controller for virtual machines. Can be considered the master in a master/slave model. It coordinates multiple virtual machines and their resource utilization, among other things. For more information, see "Virtual Machine."

IBM – "International Business Machines." IBM is primarily known as a computer manufacturer.

ICT – Short for "Information and Communications Technology." It is one of many schools at SAIT Polytechnic, and is the school we belong to.

IT – "Information Technology" abbreviated. Refers to the use of computers to achieve a goal, meet a need, or provide a service.

Linux – Another type of operating system besides Windows. It is known for its open-source development and deployment.

MAC – Stands for "Media Access Control (Address)." It is a physical address indicator on network devices.

MSDNAA – "Microsoft Developer Network (MSDN) Academic Alliance" abbreviated. A venue for academic institutions to access Microsoft products for educational purposes.

Net-Zero – Refers to the concept of equal parts power production and consumption, in terms of the Discovery 4 house. In the summer months, due the abundant amount of sunlight, the house produces more power than it consumes, but the converse is true for the winter months; it uses more than it produces. The end result levels itself out, which is where the term net-zero comes from.

OEM – Stands for "Original Equipment Manufacturer."

PC – "Personal Computer." Can refer to a wide variety of computers.

Perl – An interpreted scripting language. It is a language our team has formal training using.

Python – Another common scripting language. This is the language the original student developer used to create custom scripts to scrape data into Discovery 4's database. Our team does not have formal training on this language.

RAID – "Redundant Array of Individual Disks." There are various different kinds of RAID setups with the intention of providing availability, backup, and information storage.

RAM – "Random Access Memory." A form of data storage on computers, mostly used for short term storage.

RDP – "Remote Desktop Protocol." This is a mechanism used to connect and control a computer from a remote location, using this protocol.

SAIT – "Southern Alberta Institution of Technology." A polytechnic institution based out of Calgary, Alberta, Canada. It is the institution we belong to and are conducting our project for.

Sysinternals – A suite of computer tools used to measure and gauge various aspects that can be found on a computer. The suite is available for free online, and is widely used.

TCP/IP – "Transmission Control Protocol" and "Internet Protocol." The two of them are a set of common protocols used to communicate via a network.

UDP – "User Datagram Protocol." Similar in principle to TCP/IP, though executed very differently.

UPS – "Uninterruptable Power Supply." A form of backup power for computer hardware that takes over if the main source of power to the equipment fails. Its primary purpose is to avoid power failure from crashing a system ungracefully.

VHD – "Virtual Hard Disk." Essentially a hard disk that is used by virtual machines, much like a physical hard drive would be used for a physical machine.

VM – "Virtual Machine." A virtual instance of a computer system running in another machine (called the host). The virtual machine can achieve functionality by emulating computer hardware, or by having physical hardware from the host machine be passed through to it.

VMDK – "Virtual Machine Disk." A file format used by VMware for virtual machines. Similar in concept to a virtual hard disk, but different due to its proprietary attachment to VMware.

VNC – "Virtual Network Computing." Similar to Remote Desktop Protocol, but different in its execution. At face value, the connection experience is slower than RDP. This was the connection protocol used to connect to the home's server remotely prior to our project.