ORNL/TM-2015/331

Embedded Volttron Specification - Benchmarking Small Footprint Compute Devices for Volttron

Jibonananda Sanyal
David Fugate
Ken Woodworth
James Nutaro
Teja Kuruganti

17 August 2015

Approved for public release:
distribution is unlimited.

OAK RIDGE NATIONAL LABORATORY
MANAGED BY UT-BATTELLE FOR THE US DEPARTMENT OF ENERGY

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

ORNL/TM-2015/331

Energy and Transportation Science Division

Embedded Volttron Specification - Benchmarking Small Footprint Compute Devices for Volttron

Jibonananda Sanyal, David Fugate, Ken Woodworth,
James Nutaro and Teja Kuruganti

Date Published: 17 August 2015

Prepared by
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, TN 37831-6283
managed by
UT-BATTELLE, LLC
for the
US DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725

CONTENTS

_Toc424292701
Abstract.......... 4
1. Introduction.......... 5
1.1 Embedded Platforms.......... 5
1.2 Hardware specifications.......... 6
1.2.1 Intel Next Unit of Computing (NUC).......... 6
1.2.2 Raspberry Pi 2.......... 7
1.2.3 BeagleBone Black.......... 8
2. Installation.......... 10
2.1 OS Installation.......... 10
2.1.1 Intel Next Unit of Computing (NUC).......... 10
1.2.4 Raspberry Pi 2.......... 10
1.2.5 BeagleBone Black.......... 11
2.2 Installing Volttron dependencies.......... 12
2.3 Installing Volttron.......... 13
2.4 Platform Notes.......... 13
3. Applications on the platform.......... 14
3.1 Experimental setup.......... 14
3.2 Scenario I.......... 16
3.2.1 Agent Architecture.......... 16
3.2.2 On-board processing.......... 18
3.3 Scenario II.......... 19
3.3.1 Experimental Setup.......... 19
3.3.2 Results.......... 19
3.4 Limitations experienced.......... 24
4. Future Work.......... 25
5. References.......... 26

ABSTRACT

An embedded system is a small footprint computing unit that typically serves a specific purpose closely associated with measurements and control of hardware devices. These computing units are designed for reasonable durability and operations in a wide range of operating conditions. Some embedded systems support real-time operations and can demonstrate high levels of reliability. Many have failsafe mechanisms built to handle graceful shutdown of the device in exception conditions. The available memory, processing power, and network connectivity of these devices are limited due to the nature of their specific-purpose design and intended application. Industry practice is to carefully design the software for the available hardware capability to suit desired deployment needs.

Volttron is an open source agent development and deployment platform designed to enable researchers to interact with devices and appliances without having to write drivers themselves. Hosting Volttron on small footprint embeddable devices enables its demonstration for embedded use. This report details the steps required and the experience in setting up and running Volttron applications on three small low cost computing units: the Intel Next Unit of Computing (NUC), the Raspberry Pi 2, and the BeagleBone Black. In addition, the report also details preliminary investigation of the execution performance of Volttron on these devices.

1. INTRODUCTION

The Volttron software platform was developed at Pacific Northwest National Laboratory (PNNL) to address the need for enabling energy transactions on the electrical distribution grid between centralized and distributed energy generation, distributed energy storage, and distributed loads. The Volttron software platform is based on the Zero MQ publisher subscriber message bus, and supports agents that perform various tasks with local or remote data. Previous work [1, 2, 3, 4] that was supported by the Building Technologies Office at the U.S. Department of Energy demonstrated the application of Volttron to interact with Modbus devices, WIFI devices, the external cloud, internet resources, and local algorithms.

The previous work [1, 2, 3, 4] utilized platforms such as fan less box personal computers, etc. to host the local Volttron applications at the test and deployments sites. Significant interest in extending Volttron to edge devices and more deeply embedded devices has led to interest in the investigation of deploying the Volttron platform on lower level embedded single board computers. This interest includes applications on equipment such as thermostats, sensors, and HVAC equipment controls to provide a higher level of distrusted functionality for building transactional capability.

There has been some prior work in setting up Volttron on small size and low-cost computing devices. PNNL and Virginia Tech have explored the deployment of Volttron on a BeagleBone platform [5]. This document describes Oak Ridge National Laboratory’s evaluation and benchmarking of three common single board computing platforms, the BeagleBone Black, the Raspberry Pi, and the Intel Next Unit of Computing, for potential Volttron applications.

1.1 EMBEDDED PLATFORMS

An embedded system is a small footprint computing unit that typically serves a specific purpose closely associated with measurements and control of hardware devices. These units are designed for reasonable durability and operations in a wide range of operating conditions. Some embedded systems support real-time operations and can demonstrate high levels of reliability. Many have failsafe mechanisms built to handle graceful shutdown of the device in exception conditions. The available memory, processing power, and network connectivity of these devices are limited due to the nature of their specific-purpose design and intended application. Industry practice is to carefully design the software for the available hardware capability to suit desired deployment needs.

The Building Technologies Office at the U.S. Department of Energy is funding a multi-laboratory projects that will deliver a ‘Transactional Network’ that supports energy, operational, and financial transactions initially between roof top units (RTUs), between RTUs and the electric power grid using applications, or 'agents' that reside in the equipment either on local building controllers or in the Cloud.

The purpose of these projects is to demonstrate and propagate an open source, open architecture platform that enables a variety of site/equipment specific applications to be applied in a cost effective and scalable way. This will lower the cost of entry for both existing and new service providers as the data transport or information exchange typically required for operational and energy related products and services will be ubiquitous and interoperable.

Volttron is an open source agent development and deployment platform designed to enable researchers to interact with devices and appliances without having to write drivers themselves. Hosting Volttron on small footprint embeddable devices enables its demonstration for embedded use.

This report details the steps required and the experience in setting up and running Volttron applications on three small footprint devices: the Intel Next Unit of Computing (NUC), the Raspberry Pi 2, and the BeagleBone Black. In addition, the report also details preliminary investigation of the execution performance of Volttron on these devices.

1.2 HARDWARE SPECIFICATIONS

The three devices in this study have a range of possible hardware configurations. Different vendors supply different combinations of add-on hardware. Below are the specifications of the hardware used in this study.

1.2.1 Intel Next Unit of Computing (NUC)

The Intel NUC product is sold as a barebones system. It does not come with memory cards or a hard drive. These have to be purchased additionally and must be assembled by the user. The cost of a NUC with an Intel i3 processor, with 4 GB RAM and a hard drive was $496.00. Its dimensions are 4.6 x 4.4 x 1.4 inches and weighed 2.6 pounds. The baseboard DC power requirements are 19V, 65 Watts. Figure 1 shows an assembled and labeled NUC with a SD card adaptor placed side by side for a sense of scale, as well as a partially assembled NUC.

The NUC used in this study is the Intel® NUC Kit D34010WYK and had the following system configuration:

Processor	4th generation Intel® Core™ i3-4010U processor (soldered down) with active fan heatsink
Memory	Two SO-DIMM slots supporting up to 16 GB of 1600/1333 MHz 1.35V DDR3L memory (Note: 1.5V DDR3 memory is not supported)
Display	One mini DisplayPort 1.2 with audio support One mini HDMI port 1.4a with audio support
Audio	Intel® High Definition Audio (Intel® HD Audio)[1] subsystem in the following configuration: • 8-channel (7.1) digital audio via HDMI 1.4a output and via one DisplayPort 1.2 connector • Headphone/microphone jack on the front panel
LAN support	Intel® Gigabit Ethernet Controller
Peripheral interfaces	Two USB 3.0 connectors (front panel) Two USB 3.0 connectors (back panel) Two USB 2.0 ports (internal headers) One SATA port (internal header) Consumer infrared sensor on the front panel
Expansion capabilities	One full length mini PCI Express slot with mSATA support One half-length mini PCI Express slot

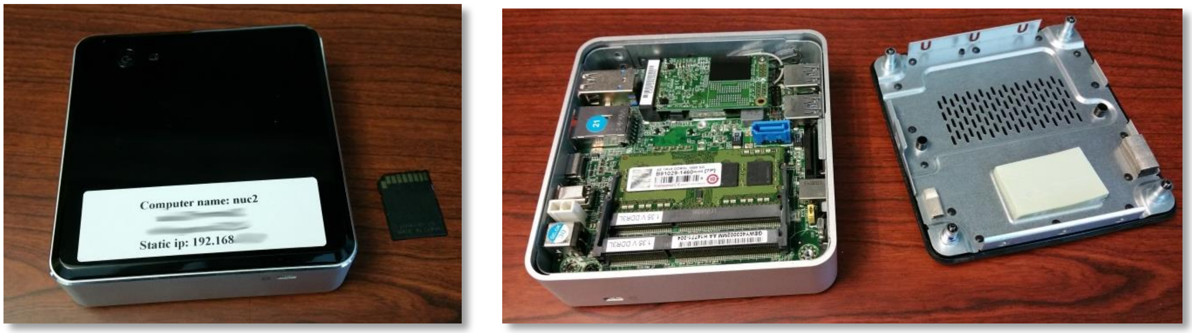


Figure 1: Left: An assembled and labeled NUC with a SD card adaptor for a sense of scale, and Right: A partially assembled NUC.

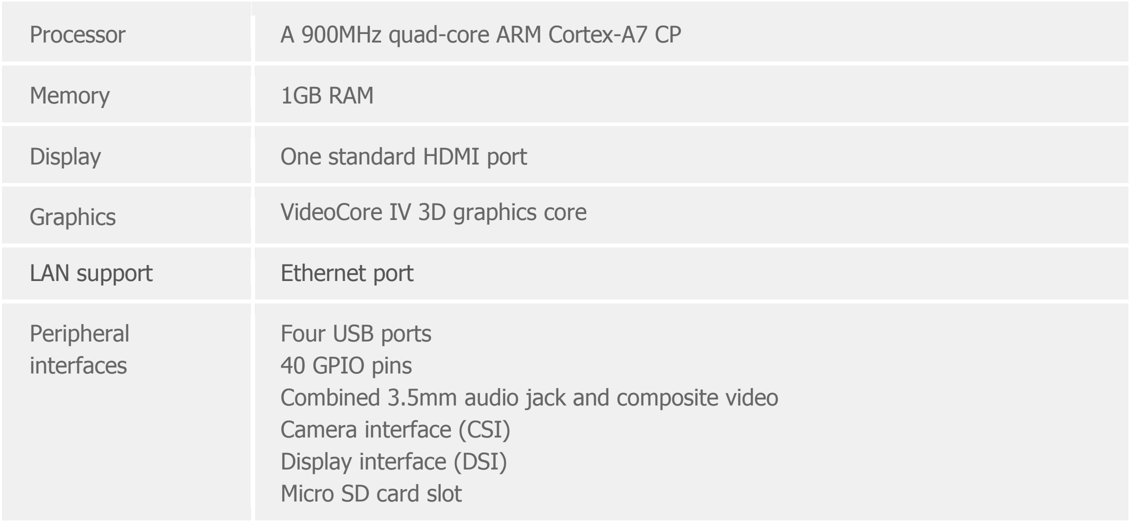
1.2.2 Raspberry Pi 2

The Raspberry Pi 2 Model B is a newer and more powerful system in comparison to the original Raspberry Pi. The devices are available from different vendors in different packages. The one used in this study is a packaged kit from Canakit which included a Raspberry Pi 2, two heat sinks, a wireless Ethernet adapter, a transparent plastic enclosure, and an HDMI cable for connection to a monitor. Note that this is a standard Type A HDMI connector. The user must assemble these components.

The Canakit Raspberry Pi 2 kit retailed for $84.50 and had product dimensions of 3.37 x 2.13 x 0.67 inches. The weight of just the board was 1.59 ounces. The power requirements depend on the number of USB units connected and can be between 700 -1000mA for the Model B although the maximum is 1 Amp. A powered external USB hub is recommended when multiple USB devices are to be connected to the device to keep from drawing too much power. The keyboard, mouse, USB WiFi, and display can add to the power draw from the device. Figure 2 shows a Canakit Raspberry Pi 2 fully assembled in its transparent enclosure.

Figure 2: A Canakit Raspberry Pi 2 fully assembled in its transparent enclosure. A credit card and an SD card adapter are placed nearby for a sense of scale.

The hardware specifications are below:



Processor	A 900MHz quad-core ARM Cortex-A7 CP
Memory	1GB RAM
Display	One standard HDMI port
Graphics	VideoCore IV 3D graphics core
LAN support	Ethernet port
Peripheral interfaces	Four USB ports 40 GPIO pins Combined 3.5mm audio jack and composite video Camera interface (CSI) Display interface (DSI) Micro SD card slot

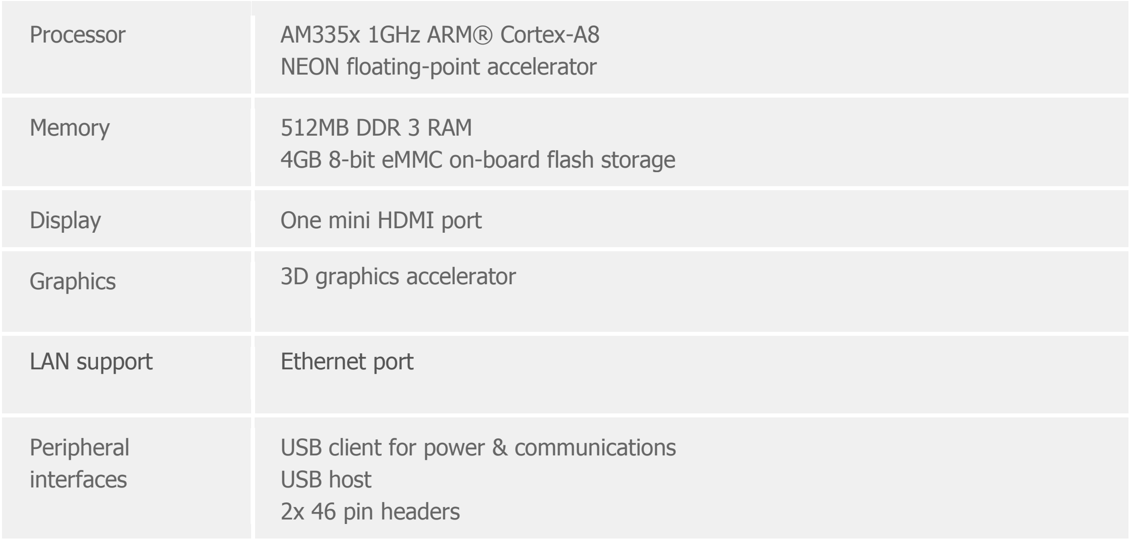
1.2.3 BeagleBone Black

The BeagleBone Black is another small footprint credit card sized device. Like the Raspberry Pi 2, the BeagleBone Black is also available from multiple vendors. The one used in this study was a kit from MakerShed which included the barebones BeagleBone Black and several additional wire connectors for assembling by the user. Figure 3 illustrates a BeagleBone Black with a credit card and an SD card adapter placed nearby for a sense of scale.

Figure 3: A BeagleBone Black with a credit card and an SD card adapter placed nearby for a sense of scale. The BeagleBone Black kit did not have an enclosure.

The kit retails for $119.99 on MakerShed and has product dimensions of 3.5 x 2.15 x 0.187 inches. The weight of just the board was 1.4 ounces. The power requirements are similar to the Raspberry Pi 2 and dependent on the connected USB devices, however, a 2 Amp power supply is recommended.

The hardware specifications are as follows:



Processor	AM335x 1GHz ARM® Cortex-A8 NEON floating-point accelerator
Memory	512MB DDR 3 RAM 4GB 8-bit eMMC on-board flash storage
Display	One mini HDMI port
Graphics	3D graphics accelerator
LAN support	Ethernet port
Peripheral interfaces	USB client for power & communications USB host 2x 46 pin headers

2. INSTALLATION

This section describes the step-by-step instructions for installing the operating system, Volttron's required dependencies, and the Volttron software platform on each of the three small footprint embeddable hardware platforms.

2.1 OS INSTALLATION

This section details the steps required to install the operating system on the three devices.

2.1.1 Intel Next Unit of Computing (NUC)

Ubuntu 14.04 (Trusty) was installed on the NUC. Following were the steps taken.

Step 1: On a desktop or a laptop computer, download the latest Ubuntu 14.04.2 LTS release ISO file from http://www.ubuntu.com/download/desktop

Step 2: Burn the ISO file onto a DVD using optical disk image writing software.

Step 3: After assembling the NUC, connect a keyboard, mouse, and a display to the device.

Step 4: Connect a USB optical media drive to the NUC.

Step 5: Power on the device and press F2 to enter the Intel Visual BIOS interface.

Step 4: Under Boot Order, ensure the Optical drive has priority. If not, click 'Advanced' to make required changes.

Step 5: Save changes and exit the Visual BIOS interface.

Step 6: Insert the Ubuntu DVD into the optical drive and boot from the media.

Step 6: Follow on-screen instructions to install Ubuntu on the device.

1.2.4 Raspberry Pi 2

Installation of Ubuntu 14.04 (Trusty) was done in the following steps:

Step 1: On a desktop or laptop computer running Linux, download the Ubuntu 14.04 image for the Raspberry Pi 2 from https://wiki.ubuntu.com/ARM/RaspberryPi. Note that this is a community maintained image and is not supported by Ubuntu.

Step 2: Download the zipped image file from http://www.finnie.org/software/raspberrypi/2015-04-06-ubuntu-trusty.zip

Step 3: Extract the contents into a directory.

Step 4: Use the bmap-tool to write the image onto a micro-SD card. You may use an SD card adapter to plug in the micro-SD card into your laptop or desktop. Execute the following command where /dev/sdX is the SD card device.

```
$ sudo bmaptool copy --bmap ubuntu-trusty.bmap ubuntu-trusty.img
/dev/sdX
```

Step 4: Optionally, you may wish to resize the partitions on the SD card. Instructions for partioning can be found at https://wiki.ubuntu.com/ARM/RaspberryPi

Step 5: Insert the SD card into the raspberry Pi 2's SD card slot. Connect a keyboard, monitor, and connect a power supply to boot up into Ubuntu. The Raspberry Pi 2 has a standard sized HDMI port for connecting a monitor.

Note: An attempt was made to install the unofficial Ubuntu Mate for the Raspberry Pi 2 from https://ubuntu-mate.org/raspberry-pi/. While the software installed correctly, the screen resolution was set to 1920x1080 and does not adjust to a monitor supporting a different resolution. It was also slow in responding to keyboard and mouse events. This is a known problem and a class 6 or a class 10 high-throughput micro-SD card are recommended. A class 4 micro SD card was used in the experiments here.

1.2.5 BeagleBone Black

Installation of Ubuntu 14.04 on the BeagleBone Black was performed in the following steps:

Step 1: Download a complete image from https://rcn-ee.com/rootfs/2015-05-08/ubuntu-14.04.2-console-armhf-2015-05-08.tar.xz

Step 2: Extract the image into a directory.

Step 3: For this experiment, a Windows flasher was used which can be downloaded from https://wiki.ubuntu.com/Win32DiskImager (Note that Linux install steps can be found at http://elinux.org/BeagleBoardUbuntu#eMMC:_BeagleBone_Black)

Step 4: Use the Windows flasher software to write the image onto a micro-SD card. You may use an SD card adapter to plug in the micro-SD card into your laptop or desktop. The inserted SD card will show under the 'Device' drop down on the interface as illustrated below.

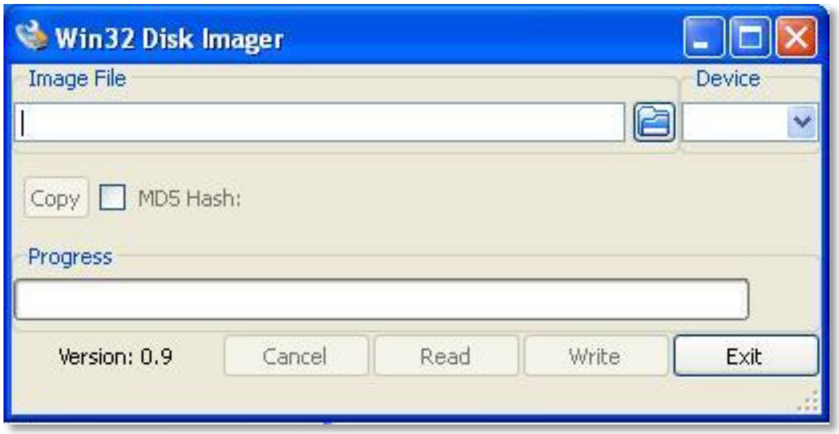


Figure 4: A Windows based flasher program to write boot images to an SD card or flash drive.

Step 5: Insert the SD card into the SD card slot on the BeagleBone Black.

Step 6: Connect a monitor and a keyboard to the board and while holding down the power button down, connect the 5v power supply to the board to power the device. Then the device should now begin to flash the image on the MMC of the BeagleBone Black.

Step 7: Once the flashing is complete, remove the SD card and reboot the device to boot into the freshly installed OS.

2.2 INSTALLING VOLTTRON DEPENDENCIES

Step 1: A software update that applies the latest Ubuntu updates is highly recommended after successfully booting up for all three small form-factor devices. Connect to a wired or wireless network to access the internet. The Intel NUC comes with a wireless adapter and connecting to a network is the easiest. The Raspberry Pi 2 and the BeagleBone Black have an Ethernet port which makes wired Ethernet connections very easy. The Canakit setup also came with a WIFI adapter. Drivers are required for the adapter and at the time of this writing, the adapter was not used for connecting wirelessly.

Open a terminal window and execute:

```
sudo apt-get update
```

Step 2: The software will not install correctly if the system date and time are not correctly set. Install the ntpdate package if it is already not installed and update the date and time.

```
sudo apt-get install ntpdate

sudo ntpdate –s time.nist.gov
```

Note that it was observed that the previous command step appeared to exit gracefully without changing the time. Please use the date command to force set the date and time if the command fails.

Step 3: Install the required software dependencies for Volttron. The main packages to be installed are:

- `git`
- `build-essential`
- `python-dev`
- `openssl`
- `libssl-dev`
- `libevent-dev`

Execute the following command to install these packages:

```
sudo apt-get install build-essential python-dev openssl libssl-dev
libevent-dev git
```

This should install all your required dependencies. For details, please visit:
https://github.com/VOLTTRON/Volttron/wiki/VOLTTRON-Development-Quick-Start

2.3 INSTALLING VOLTTRON

Step 1: To install Volttron, you must first obtain a copy of the Volttron codebase. Execute the following in a terminal to obtain the source code.

```
git clone https://github.com/VOLTTRON/Volttron
```

Step 2: Change your working directory to the root directory of the Volttron code and bootstrap Volttron. In the opened terminal, execute the following commands.

```
cd Volttron
python2.7 bootstrap.py
```

The installation may take some time to finish. To test that installation worked, activate the platform in verbose mode and set a log file as below:

```
. env/bin/activate
Volttron -vv -l Volttron.log&
```

If the setup is correct, there will be no errors.

2.4 PLATFORM NOTES

Installation of Volttron on the Raspberry Pi 2 and the BeagleBone Black took significantly longer than the Intel NUC. The Raspberry Pi 2 took about 1.5 hours while the BeagleBone Black took well over 2.5 hours. Fetching, unpacking, and the installation of numpy, pandas, and pyzmq are the most time-consuming pieces.

It is anticipated that with the appropriate skill level about 2.5 man-hours of time is required from unpacking an Intel NUC to installing the Volttron platform. This estimate goes up to about 4 hours for the Raspberry Pi 2 and 5 hours for the BeagleBone Black. The time required can become significant if a large number of devices are to be setup. For large scale deployments, it may be possible to create an operating system image with Volttron and agents preinstalled. This will significantly reduce the required time for setup. It was outside of the scope of this work to evaluate the effectiveness of the proposed deployment methodology, therefore, the above should be considered speculative.

3. APPLICATIONS ON THE PLATFORM

After the setup and installation of Volttron on the three computing platforms, their behavior and performance was tested in two scenarios while running Volttron applications. The first evaluation scenario employs several agents, which publish and communicate via the Volttron message bus. Its agent architecture is described next, followed by observed behavior of the applications on the three platforms.

The second was a simpler application where the Python pymobus module was used to make temperature readings at different speeds to understand hardware performance for Modbus communication and Volttron execution. This consisted of four scenarios: commutation with thermostats using pymodbus alone, via a Volttron app, via a Volttron app while publishing to SMAP, and via a Volttron app while pushing to a MySQL database. The experimental setup is described next followed by the first and second application scenarios and a summary of the limitations experienced.

3.1 EXPERIMENTAL SETUP

The experimental setup is illustrated in Figure 5. A set of five TEMCO Modbus thermostats are connected accessed via a RS-485 serial computer interface. A serial to USB converter is used to connect the thermostats to the selected computing device. The test setup also a thermostat relay status display panel with light emitting diodes to visually illustrate the state and the mode of operation of the thermostats. Figures 6, 7, and 8 illustrate the wiring diagrams in detail.

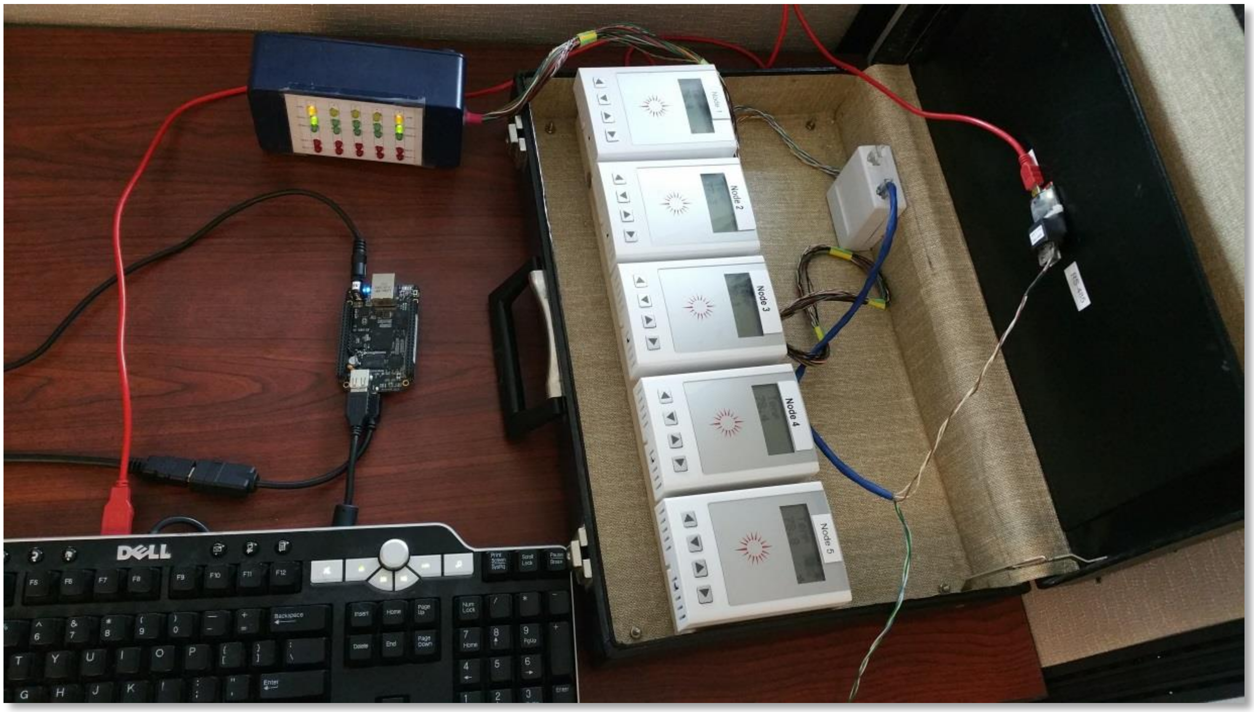
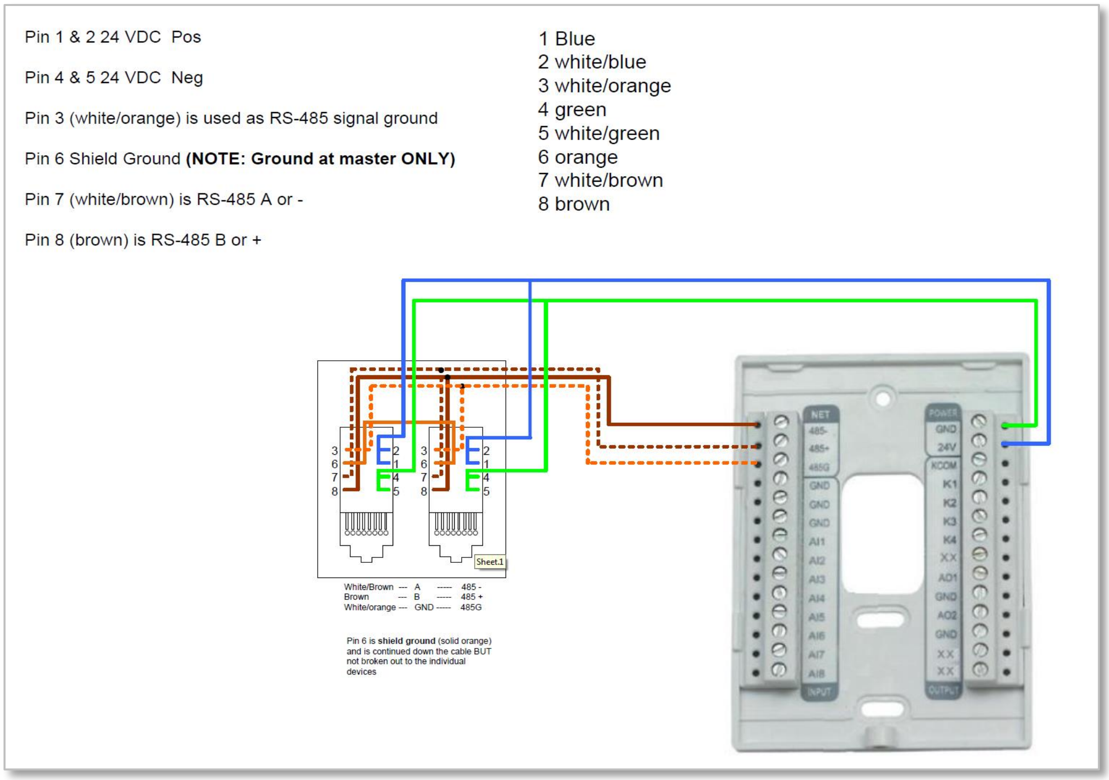


Figure 5: The experimental setup with thermostats connected over the serial to USB interface.



Pin 1 & 2 24 VDC Pos

Pin 4 & 5 24 VDC Neg

Pin 3 (white/orange) is used as RS-485 signal ground

Pin 6 Shield Ground **(NOTE: Ground at master ONLY)**

Pin 7 (white/brown) is RS-485 A or -

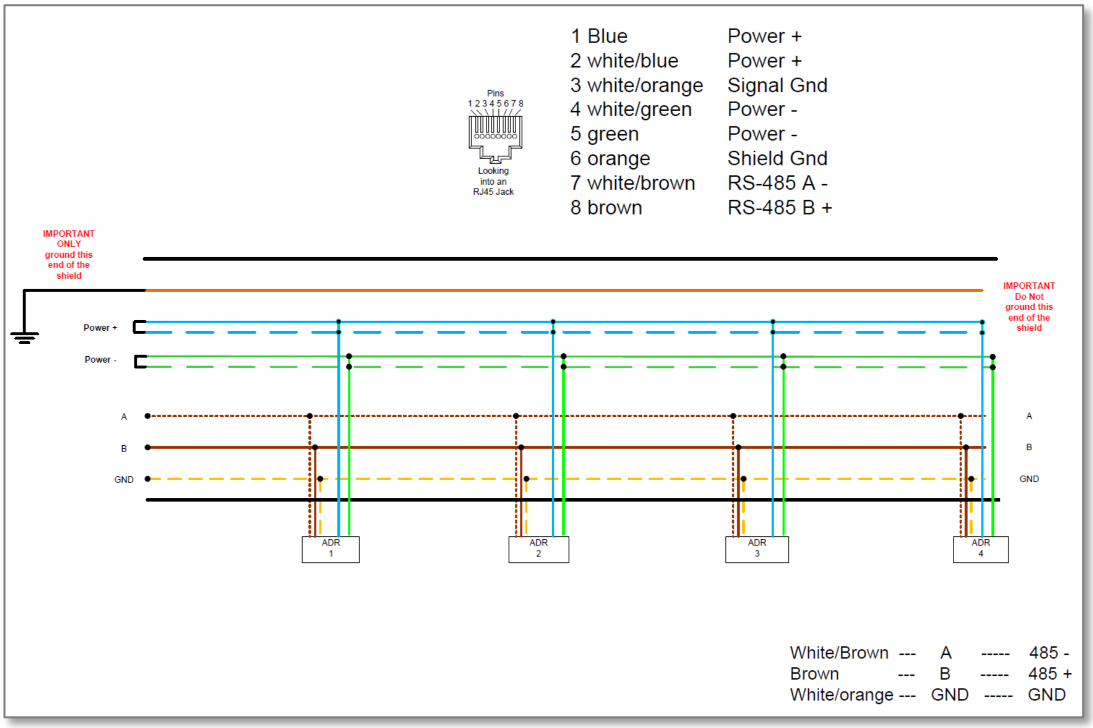
Pin 8 (brown) is RS-485 B or +

1 Blue
2 white/blue
3 white/orange
4 green
5 white/green
6 orange
7 white/brown
8 brown

White/Brown --- A ----- 485 -
Brown --- B ----- 485 +
White/orange --- GND ----- 485G

Pin 6 is **shield ground** (solid orange) and is continued down the cable BUT not broken out to the individual devices

Figure 6: Wiring diagram of the thermostats and relays.



1 Blue	Power +
2 white/blue	Power +
3 white/orange	Signal Gnd
4 white/green	Power -
5 green	Power -
6 orange	Shield Gnd
7 white/brown	RS-485 A -
8 brown	RS-485 B +

IMPORTANT ONLY ground this end of the shield

IMPORTANT Do Not ground this end of the shield

White/Brown --- A ----- 485 -
Brown --- B ----- 485 +
White/orange --- GND ----- GND

Figure 7: Wiring diagram of the thermostats and the Modbus interface.

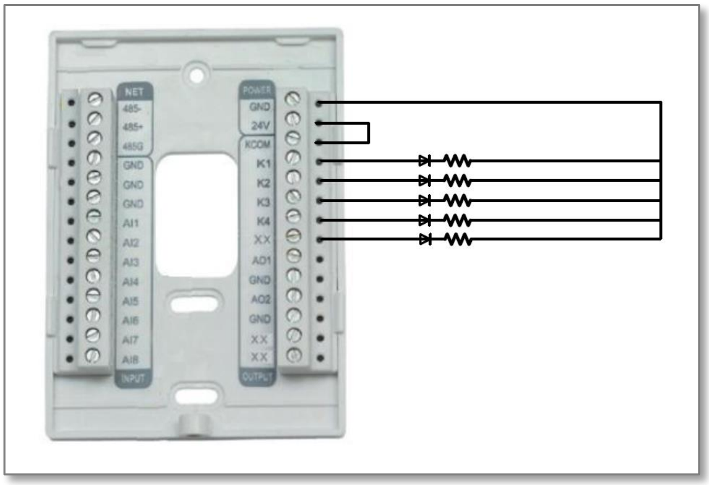


Figure 8: Display panel light emitting diode connections.

3.2 SCENARIO I

3.2.1 Agent Architecture

The following figure (Fig 9 and 10?) illustrates the overall architecture of the Volttron agents in the first application.

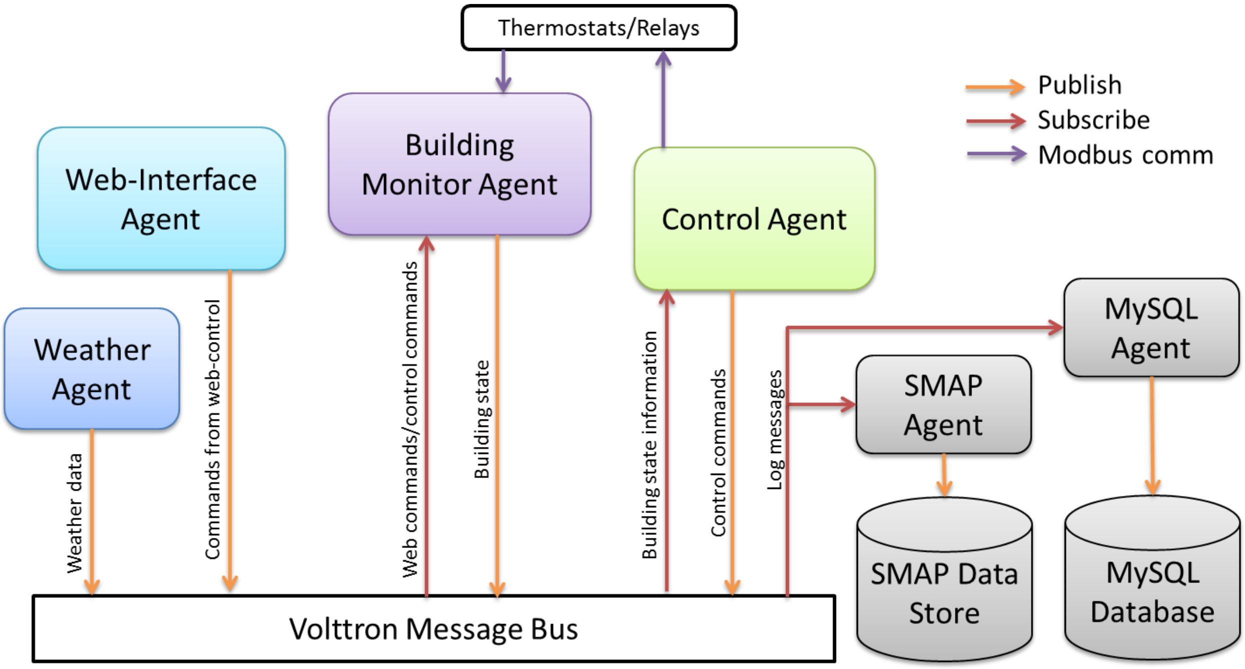
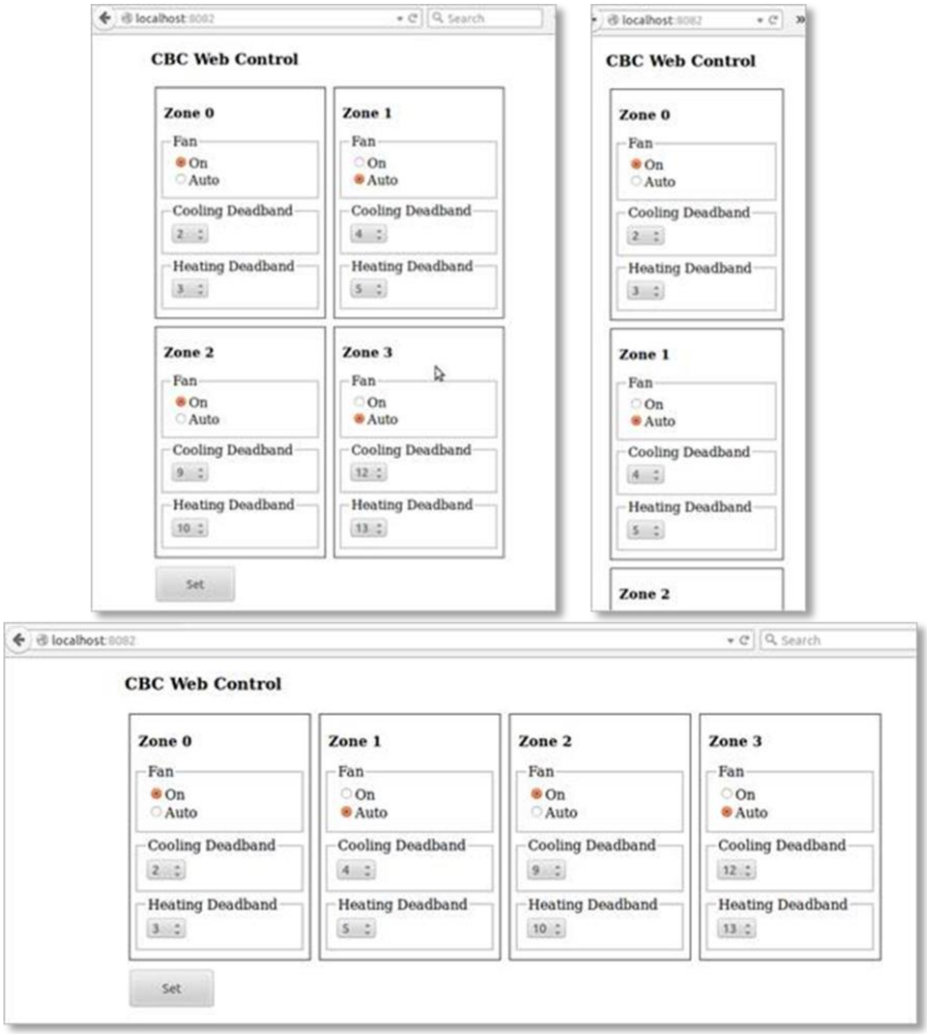


Figure 9: Agent architecture of the Volttron applications tested.

A Building Monitor Agent observes thermostats in a building and publishes observed values. It is also subscribed to commands from the Control Agent as well as the Web-Interface. When a control command is published by either of these agents, the Building Monitor Agent takes action upon the request. The communication with the thermostats is over the Modbus protocol.

The Control Agent houses the control logic and is subscribed to messages from the Building monitor as well as the Weather Agent. It periodically publishes control messages to alter the state of the building. A command is issued every 5 seconds.

The Web-Interface Agent provides a convenient remote web-based interface to interact with the devices. Figure 6 illustrates a screenshot of the web-interface agent which supports a fluid layout to adapt to different screen sizes such as laptops, tablets, and smart phones. The purpose of this agent is to facilitate occasional control of the devices by a human. The frequency of use of the web-interface is dependent on the human user and is anticipated to be used sparingly. As such, only a handful of web requests are issued to the platform on a given day.



CBC Web Control

Zone 0
Fan
● On
○ Auto
Cooling Deadband
2
Heating Deadband
3

Zone 1
Fan
○ On
● Auto
Cooling Deadband
4
Heating Deadband
5

Zone 2
Fan
● On
○ Auto
Cooling Deadband
9
Heating Deadband
10

Zone 3
Fan
○ On
● Auto
Cooling Deadband
12
Heating Deadband
13

CBC Web Control

Zone 0
Fan
● On
○ Auto
Cooling Deadband
2
Heating Deadband
3

Zone 1
Fan
○ On
● Auto
Cooling Deadband
4
Heating Deadband
5

Zone 2

CBC Web Control

Zone 0
Fan
● On
○ Auto
Cooling Deadband
2
Heating Deadband
3

Zone 1
Fan
○ On
● Auto
Cooling Deadband
4
Heating Deadband
5

Zone 2
Fan
● On
○ Auto
Cooling Deadband
9
Heating Deadband
10

Zone 3
Fan
○ On
● Auto
Cooling Deadband
12
Heating Deadband
13

Figure 10: The web-based control interface for local human-machine-interface. The three screenshots illustrate the fluid design of the control interface to support small touch-screen devices such as smartphones.

The Weather Agent fetches the current weather conditions from WUnderground every 4 minutes and publishes it on the Volttron message bus. This is the standard Weather app that is part of the Volttron codebase.

All agents publish pertinent log messages. The SMAP Agent is subscribed to these log messages and its purpose is to get the data out of the compute platform and log these messages in a remote data store. The SMAP agent pushes the weather data out of the platform every four minutes when the weather data is updated.

(Note that although the MySQL agent is illustrated in the architecture and described here, it was a later addition for the Scenario II test cases only and therefore not part of Scenario I test results.)

The MySQL Agent is also subscribed to relevant log messages and its purpose is to log the data values to a MySQL database. At the time of this experiment, the MySQL database was hosted locally on the same compute device. Its behavior was designed to be similar to that of the SMAP agent, which creates an HTTP request per log message and receives a success or failure response. The MySQL agent creates a database connection when initializing and creates a new cursor to submit an INSERT query per log message. Internally, the agent uses the Python MySQLdb module for MySQL communication.

3.2.2 On-board processing

During installation, these boards had a monitor and keyboard connected to them. However, to minimize display and keyboard interrupt loads on these small devices while testing Volttron applications, a secure shell terminal was used to connect to the devices. Note that the sshd daemon was already installed in the BeagleBone Black OS image; however, it has to be installed on the Raspberry Pi 2. The following command was used:

```
sudo apt-get install openssh-server
```

As a control, an experimental session with a connected keyboard and monitor was done; however, there was no major change in the CPU load of the small computing units. The following table illustrates system, CPU, and memory footprint while running the Volttron applications described above.

	BeagleBone Black	**Raspberry Pi 2**	**Intel NUC***
CPU load	~2.5 - 4.5%	~2.5 – 4.5%	<2%
Memory use	331 MB of 490 MB	336 MB of 923 MB	2.2 GB of 3.8 GB
* The Intel NUC had the Ubuntu Desktop (GUI) installed.			

The numbers reported were similar across top, iostat, and mpstat Linux utilities. The command top computes an average use since the last screen update. The other two commands, iostat and mpstat also compute average processor utilization. It was rare to see processor utilization above 10% while Volttron applications were executing. Only times processor utilization was in the 10 – 15% range was during communication with the Weather Underground server. It is anticipated that applications that require heavy communication with external data stores will drive up the CPU utilization. The above test results indicate that this configuration of Volttron agents as a use case performs successfully on all three computing devices. Scenario II described next illustrates the effect of significantly higher communication calls on CPU utilization.

3.3 SCENARIO II

3.3.1 Experimental Setup

While the first experiment represents a benchmark for a possible use case scenario, it does not provide adequate information on the capabilities, operating margin, or potential scalability of these computing devices while running Volttron applications. A second, more controlled experiment was conducted to run a more thorough experiment as well as to derive preliminary benchmarks of performance while pushing the hardware to a certain degree.

a. A simple Modbus read operation from the five thermostats (as illustrated in the setup in Figure 5) was tested in four execution scenarios:

b. Without Volttron, simple repeated reads from the thermostats

c. As an agent using the Volttron platform

d. As an agent using the Volttron platform and publishing values on the Volttron message bus while a smap agent pushed the data to a remote smap server.

e. As an agent using the Volttron platform and publishing values on the Volttron message bus while a MySQL agent pushed the data to a local MySQL database.

The scan rate and the associated time-out for communicating with the Modbus thermostat devices were varied from 0.01 seconds to 0.5 seconds or 100 scans per second to 2 scans per second. Each scan consisted of a Modbus read of one of the thermostat devices to obtain the temperature measurement value. In other words, 100 scans per second is equivalent to reading from one thermostat 100 times per second or 10 thermostats 10 times each per second. In the test execution, the communication sequenced between all five thermostats evenly to achieve a realistic test of communicating with multiple devices. For each test condition of scan rate, each thermostat was read 50 times to obtain a reasonably statistical sample for benchmarking.

3.3.2 Results

Tables 1, 2, and 3 summarize the results and the observed behavior of reading data off thermostats at the different scan rates. The tool mpstat was executed to summarize system load every 5 seconds. The system utilization reported in the tables is the maximum load observed in any 5 second interval. The number reported is percentage of time the system is not idle (100% – idle%). This value constitutes sum of all user level activity, priority activity, system level, and interrupt service activity across all cores. The number serves as a benchmark for the maximum observed load in any 5 second period and does not imply continuous load over the execution of the scenario. This data is summarized in a plot in Figure 11.

The incidence of no data being returned increases with a low timeout value/faster can rate. Occasionally, more severe exceptions would be thrown in the pymodbus function calls. It should be noted that for the scenarios with SMAP and MySQL, only valid data points were published onto the Volttron message bus for archiving. It should also be noted that the Intel NUC was running the Ubuntu desktop graphical user interface.

Table 1: Performance on the BeagleBone Black.

	PyModbus only			PyModbus on Volttron			PyModbus on Volttron with SMAP archiving			PyModbus on Volttron with MySQL archiving		
	Number of errors			Number of errors			Number of errors			Number of errors		
Timeout value (s)	No value read	Exceptions	Max % CPU use	No value read	Exceptions	Max % CPU use	No value read	Exceptions	Max % CPU use	No value read	Exceptions	Max % CPU use
0.01	239	16	22.27	243	7	89.62	241	9	76.54	246	4	58.39
0.02	232	23	8.76	215	26	38.05	195	22	72.15	227	11	45.85
0.03	220	11	8.07	124	5	34.03	123	6	69.65	116	1	85.42
0.04	107	0	6.41	112	0	38.67	89	0	57.00	93	0	88.38
0.05	59	0	6.62	142	0	31.25	130	0	57.67	145	0	67.01
0.10	26	0	5.30	9	0	23.33	7	0	41.50	9	0	61.86
0.15	0	0	3.98	0	0	19.46	0	0	37.78	0	0	56.72
0.20	0	0	5.04	0	0	17.99	0	0	43.48	0	0	62.89
0.25	0	0	3.76	0	0	14.71	0	0	42.62	0	0	51.77
0.30	0	0	10.85	0	0	18.28	0	0	43.22	0	0	42.35
0.35	0	0	7.05	0	0	11.92	0	0	41.74	0	0	53.15
0.40	0	0	12.21	0	0	20.25	0	0	47.93	0	0	47.81
0.45	0	0	4.63	0	0	10.27	0	0	37.24	0	0	46.04
0.50	0	0	6.14	0	0	11.09	0	0	35.34	0	0	45.82

Table 2: Performance on the Raspberry Pi 2.

	PyModbus only			PyModbus on Volttron			PyModbus on Volttron with SMAP archiving			PyModbus on Volttron with MySQL archiving		
	Number of errors			Number of errors			Number of errors			Number of errors		
Timeout value (s)	No value read	Exceptions	Max % CPU use	No value read	Exceptions	Max % CPU use	No value read	Exceptions	Max % CPU use	No value read	Exceptions	Max % CPU use
0.01	255	0	2.08	245	5	29.33	242	8	21.59	249	1	18.85
0.02	237	18	0.96	224	26	8.83	221	29	7.24	224	26	8.53
0.03	242	12	0.66	159	8	6.31	187	6	12.97	183	7	15.00
0.04	86	1	0.51	92	0	5.33	3	246	14.50	85	0	21.57
0.05	28	0	0.56	76	0	4.43	77	0	8.93	83	0	24.32
0.10	10	0	0.41	30	0	4.42	37	0	11.68	46	0	14.81
0.15	0	0	0.36	0	0	1.98	0	0	11.18	0	0	9.84
0.20	0	0	0.30	0	0	3.10	0	0	4.08	0	0	11.38
0.25	0	0	0.30	0	0	3.36	0	0	9.01	0	0	12.01
0.30	0	0	0.30	0	0	1.27	0	0	13.66	0	0	6.38
0.35	0	0	0.30	0	0	1.17	0	0	6.97	0	0	8.30
0.40	0	0	0.30	0	0	1.02	0	0	4.99	0	0	9.93
0.45	0	0	0.30	0	0	2.50	0	0	2.84	0	0	9.16
0.50	0	0	0.30	0	0	2.60	0	0	4.53	0	0	11.39

Table 3: Performance on the Intel NUC.

	PyModbus only			PyModbus on Volttron			PyModbus on Volttron with SMAP archiving			PyModbus on Volttron with SMAP archiving		
	Number of errors			Number of errors			Number of errors			Number of errors		
Timeout value (s)	No value read	Exceptions	Max % CPU use	No value read	Exceptions	Max % CPU use	No value read	Exceptions	Max % CPU use	No value read	Exceptions	Max % CPU use
0.01	247	3	8.78	248	2	14.81	248	2	7.42	248	2	6.87
0.02	236	14	7.99	236	14	10.78	237	13	4.66	229	21	6.27
0.03	241	8	6.87	236	9	10.60	244	6	5.87	239	7	3.66
0.04	80	2	5.82	104	0	9.03	105	2	10.89	89	1	5.21
0.05	28	0	6.22	41	0	8.63	29	0	12.55	42	0	5.61
0.10	13	0	5.07	1	0	8.22	8	0	6.17	12	0	3.21
0.15	0	0	4.66	0	0	3.56	0	0	3.91	0	0	3.50
0.20	0	0	4.42	0	0	3.86	0	0	2.60	0	0	3.10
0.25	0	0	4.87	0	0	3.21	0	0	3.91	0	0	2.86
0.30	0	0	4.01	0	0	7.97	0	0	3.06	0	0	8.61
0.35	0	0	3.91	0	0	3.61	0	0	2.91	0	0	3.00
0.40	0	0	10.49	0	0	3.61	0	0	4.31	0	0	7.31
0.45	0	0	9.62	0	0	3.26	0	0	3.46	0	0	2.61
0.50	0	0	7.21	0	0	3.41	0	0	4.16	0	0	2.51

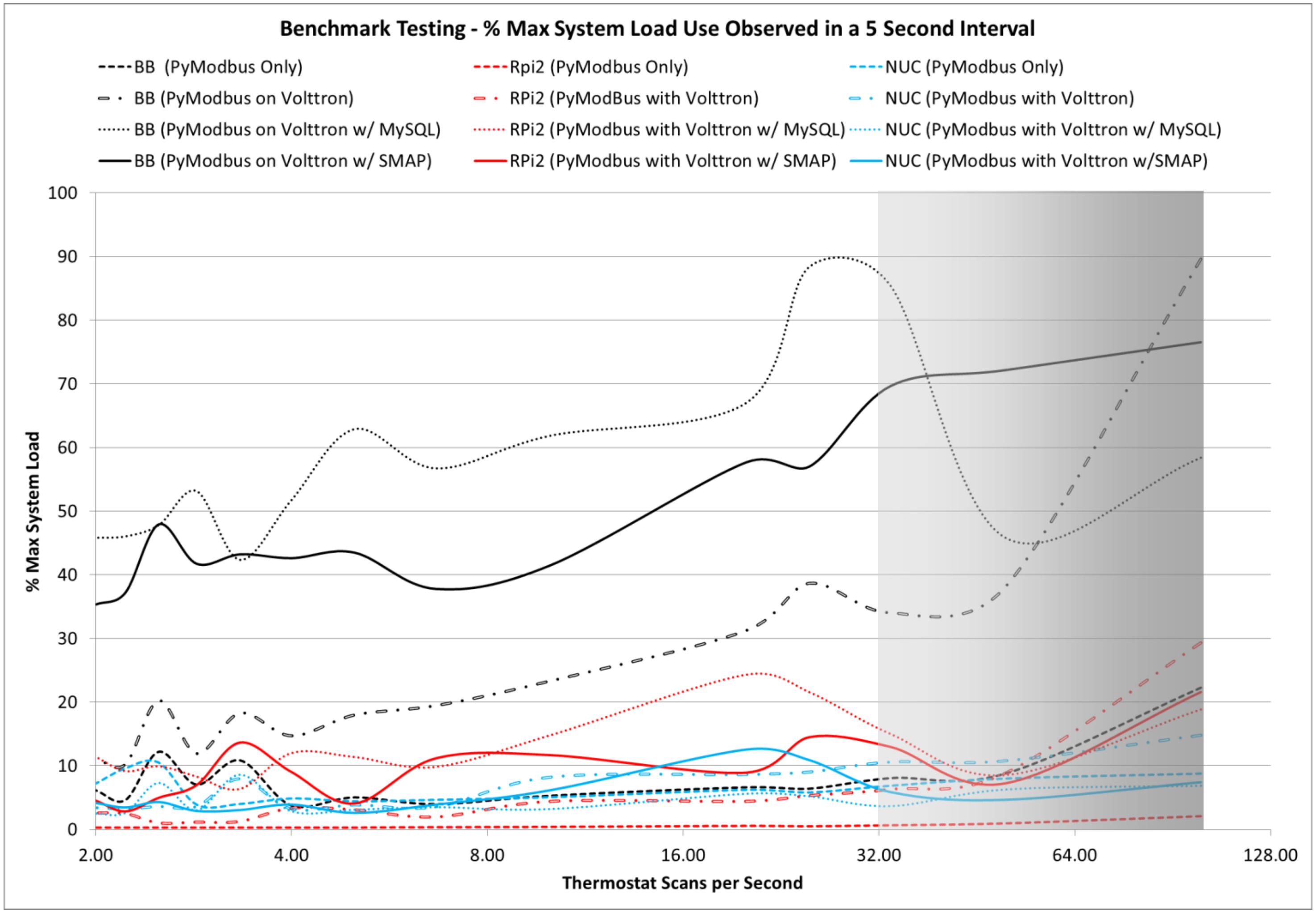


Figure 11: Summary Plot of Percent Maximum System Use in Five Second Intervals for all three Computing Devices. The BeagleBone Black performance is illustrated using black lines, the Raspberry Pi 2 in red, and Intel NUC in blue. The grey region should be interpreted with caution since the device communication exhibited repeated increasing read and communication errors.

3.4 LIMITATIONS EXPERIENCED

Volttron was successfully installed on all three computing devices. There were not many technical limitations experienced in the installation and running of Volttron on these devices. Installation time was significantly longer on the BeagleBone Black than the other two platforms.

The overall execution of applications on the Volttron platform was successful for all platforms and the overall system memory use did not exceed 400 MBs. Testing of the first scenario, a baseline Volltron configuration, demonstrated that processor utilization for all three devices was rarely above 4%, except during installation when it ran at ~99%, and during communication with the Weather Underground server. Testing of the second scenario, performing Modbus thermostat scanning with different scan rates and software configurations, demonstrated that all platforms showed higher processor utilization with faster scan rates. The maximum demonstrated processor utilization rates were ~89% for the Beagle Board, ~29% for the Raspberry Pi 2, and ~14% for the Intel NUC at 100 thermostat scans per second (see section 3.3.1 for the experimental setup).

It should be noted that the Volttron performed in a stable and predictable manner even with the highest thermostat scan rates for all platforms. The data indicates that unsuccessful Modbus communication is observed at the higher scan rates due to the nature of the serial Modbus network performance which is expected. The processor on the BeagleBone Black does not include a heat sink. During the installation of Volttron and during the execution of Volttron, the circuit board was observed to be warm. The behavior of the Raspberry Pi 2 and the NUC was not observable because of their protected enclosures.

It should also be noted that the mpstat routine differed in the manner in which it reported the timestamps on the Raspberry Pi 2 and the Intel NUC or the BeagleBone Black. It used a 24 hour time format while the other devices used a 12 hour time format. The mpstat documentation indicates that the S_TIME_FORMAT environment variable affects the timestamp format; however, the environment variable was not defined on the devices. Additional investigation was not done to identify the cause of the difference in reporting styles. While speculative, there may be some differences in implementation of the routine for the different computing architectures and its investigation is considered future work.

As expected, the Intel NUC demonstrated better performance compared to the other two computer platforms tested. It outperformed the BeagleBone Black and the Raspberry Pi 2 for both the Volttron installation time as well as perceived Volttron application execution performance.

The automatic network update feature for date and time was an issue on the BeagleBone Black as well as the Raspberry Pi 2. On a cold start of the devices, the BeagleBone Black’s clock would resume from the timestamp of when it was shut down, and the Raspberry Pi 2 would reset its clock back to 1 January 1970. It was necessary to force set the date and time on the BeagleBone Black every time the device was powered back on. The NUC had no issues in this regard.

4. FUTURE WORK

The experiments described in this report represent a straightforward application scenario for benchmarking and document preliminary findings on executing Volttron applications on small and low cost single board computing platforms. The results provide useful information, and observed limitations and assumptions warrant additional investigation. The following is a notional list of several next steps:

- Benchmarking of distributed Volttron communication and decision algorithms. Multiple small computing platforms will be running Volttron (connected to thermostats) and communicating with each other to arrive at a collective control decision.

- Additional IO and network benchmarking of the computation platforms.

- Ensuring benchmarking tools/utilities such as mpstat are similar in implementation across platforms.

- Testing on other single board computing devices such as the Arduino, Intel Edison, or others in the evaluation.

- The only Linux operating system version that was included in this testing was Ubuntu. It will be useful to understand the requirements and ease of deployment of Volttron on other operating systems.

- Deployment of each of these small footprint computational devices in real work settings.

5. REFERENCES

[1] Haack, J.N., Akyol, B.A., Katipamula, S., Lutes, R.G., Volttron Lite: Integration Platform for the Transactional Network, PNNL-22935, October 2013.

[2] Katipamula, S., Lutes, R.G., Ngo, H., Underhill, R.M., Transactional Network Platform: Applications, PNNL-22941, October 2013.

[3] Nutaro, J., Starke, M., Kuruganti, T., Fugate, D., An Inexpensive Retrofit Technology for Reducing Peak Power Demand in Small and Medium Commercial Buildings, *22nd International Compressor Engineering Conference at Purdue, 15th International Refrigeration and Air Conditioning Conference at Purdue, 3rd International High Performance Buildings Conference at Purdue*, July 2014.

[4] James Nutaro, David Fugate, Teja Kuruganti, Brian Fricke. Refrigerated Display Case Defrosting using Inferential Ice Sensing, *The 24th IIR International Congress of Refrigeration,* August 16 – 22, 2015 Yokohama, Japan.

[5] W. Khamphanchai, A. Saha, K. Rathinavel, M. Kuzlu, M. Pipattanasomporn, Saifur Rahman, B. Akyol, and J. Haack, Conceptual Architecture of Building Energy Management Open Source Software (BEMOSS), *IEEE PES ISGT-Europe Conference*, Istanbul, Turkey, October 12 – 14, 2015.