Final Technical Report

DOE EERE Award Number: DE-EE0006352

Project Title:

Building Energy Management Open Source Software (BEMOSS™)

Name of Recipient

Virginia Polytechnic Institute and State University

Principal Investigator

Dr. Saifur Rahman, Professor and Director
Virginia Tech - Advanced Research Institute

August 2017

Acknowledgment:

This material is based upon work supported by the Department of Energy under Award Number DE-EE0006352.

Disclaimer:

EXECUTIVE SUMMARY

Funded by the U.S. Department of Energy in November 2013, a Building Energy Management Open Source Software (BEMOSS™) platform was engineered to improve sensing and control of equipment in small- and medium-sized commercial buildings. According to the Energy Information Administration (EIA), small- (5,000 square feet or smaller) and medium-sized (between 5,001 to 50,000 square feet) commercial buildings constitute about 95% of all commercial buildings in the U.S. These buildings typically do not have Building Automation Systems (BAS) to monitor and control building operation. While commercial BAS solutions exist, including those from Siemens, Honeywell, Johnsons Controls and many more, they are not cost effective in the context of small- and medium-sized commercial buildings, and typically work with specific controller products from the same company. BEMOSS™ targets small and medium-sized commercial buildings to address this gap.

Aiming at reducing energy consumption in commercial buildings, BEMOSS™ offers the following key features:

- **Open source, open architecture** – BEMOSS™ is an open source operating system that is built upon VOLTTRON™ – a distributed agent platform developed by Pacific Northwest National Laboratory (PNNL). BEMOSS™ was designed to have an open architecture to make it easy for hardware manufacturers to seamlessly interface their devices with BEMOSS™. Software developers can also contribute to adding additional BEMOSS™ functionalities and applications.
- **Plug & play** – BEMOSS™ was designed to automatically discover supported load controllers (including selected smart thermostats, VAV/RTUs, lighting load controllers and plug load controllers) in commercial buildings. After auto-discovery of devices in a building, BEMOSS™ can then monitor and control these devices for desired functions.
- **Interoperability** – BEMOSS™ was designed to work with load control devices from different manufacturers that operate on different communication technologies and data exchange protocols. These include both new commercially available products that operate on Ethernet, and Wi-Fi, as well as legacy devices that operate on serial communications using Modbus RTU and BACnet MS/TP protocols. BEMOSS supports the following prevalent communication technologies: Ethernet (IEEE 802.3), Serial (RS-485) and Wi-Fi (IEEE 802.11); and protocols: BACnet, Modbus, Web and OpenADR protocols.
- **Cost effectiveness** – Implementation of BEMOSS™ deemed to be cost-effective as it was built upon a robust open source platform operating on a low-cost single-board computer, such as Odroid. This feature could contribute to its rapid deployment in small- or medium-sized commercial buildings.
- **Scalability and ease of deployment** – BEMOSS™ was designed with a multi-layer architecture where multiple single-board computers hosting the BEMOSS™ platform could monitor and control a large number of load controllers in a multi-floor and high occupancy building. This makes it possible for a building engineer to deploy BEMOSS in one zone of a building, be comfortable with its operation, and later on expand the deployment to the entire building to make it more energy efficient.

- **Ability to provide local and remote monitoring** – BEMOSS™ provided both local and remote monitoring ability with role-based access control.
- **Security** – In addition to built-in security features provided by VOLTTRON™ including agent authorization & authentication, encrypted multi-layer communication and agent validation, BEMOSS™ also provided enhanced security features, including device discovery and approval process, encrypted core-to-node communication, thermostat anti-tampering and many more.
- **Support from the Advisory Committee** – BEMOSS™ system architecture was developed in consultation with the industry. BEMOSS™ advisory committee comprises of representatives from 22 organizations. Their involvement allowed us to design the BEMOSS™ operating system across the full spectrum of current state-of-the-art hardware, and their deployment practices.

BEMOSS™ solution was deployed in three demonstration sites: (1) Virginia Tech's School of Public & International Affairs in Alexandria, VA (25,000 square feet); (2) Arlington County's Equipment Bureau building in Arlington, VA (5,000 square feet); and (3) Virginia Tech's commercial office and retail building in Blacksburg, VA (40,000 square feet). At these demonstration sites, by raising air conditioning set points while maintaining occupant comfort, 10-15% energy savings in overall electricity bills was experienced; and by performing dimming control of light intensity based on a pre-defined schedule, around 33% energy savings in lighting load was achieved.

The latest BEMOSS™ source code (BEMOSS v3.5), together with its installation and developer guides, have been made available on an online repository at www.github.com/BEMOSS.

Table of Content

List of Figures

List of Tables

1.0 Summary of Tasks Proposed and Accomplishments

This section summarizes project tasks based on the statement of project objective (SOPO) and the overall accomplishments, as well as the success toward meeting the project technical requirements.

1.1 Tasks Proposed

This project was divided into three phases, where Phase 1 (October 2013-December 2014) involved the entire software platform development; Phase 2 (January 2015-December 2015) involved lab-scale testing and software enhancement; and Phase 3 (January 2016-June 2017) involved demonstrations in three buildings for functionality tests and energy savings evaluation.

Figure 1-1 summarizes all tasks of this project, which is the excerpt from the SOPO as proposed by the Virginia Tech team.

PHASE 1: BEMOSS software and software interface for plug & play hardware device integration

Task 1: BEMOSS open source software development in consultation with Industry
BEMOSS architecture will be developed, and enhancements will be made in consultation with manufacturers like GE Appliances and Danfoss who have accepted to provide advisory support.

Task 2: BEMOSS User-Interface and software tools design
User-interface will be designed with three sets of software tools, one for each load type – HVAC, plug load and lighting load. They will each have a control-system setup with manufacturer-provided default features, status display and control system auto-mapping.

Task 3: Plug & play device integration
Software tools with necessary hardware interface will be customized for HVAC, lighting and general-purpose load control. A commercially available smart thermostat will be chosen for customization and will be integrated using a communication gateway. Lighting and general-purpose load controllers will be embedded on a smart server, which will be used as the hardware controller. Appendix A in SOPO provides specifications.

PHASE 2: Lab testing and software enhancement
Task 4: Incorporate additional software features (new)
Under this task, the project team will incorporate additional software features suggested by DOE, and based on some work done by Carnegie Mellon University (CMU). This includes distributed database, plug-and-play and auto-mapping. The Virginia Tech team will work with the CMU team to get a better understanding of their algorithm and implement the required features in the BEMOSS platform.

Task 5: BEMOSS software open source access and survey
BEMOSS website (http://bemoss.org) will be developed as a portal which hosts demonstration videos, tutorials, user opinions and other useful information about BEMOSS. This website will also be used to seek feedbacks from users and promote BEMOSS software for better awareness among building owners.

Task 6: BEMOSS advanced algorithm development
Algorithms will be further enhanced to improve energy efficiency and perform DR. This will include set-point control of variable speed HVAC compressors, limiting demand restrike etc.

Task 7: BEMOSS lab scale testing
Laboratory scale testing of BEMOSS will be performed. This will include light-emitting diode (LED) lighting devices, plug load appliances and modeled HVAC system.
Deliverable 2.4: A report with results from (i) lab testing with physical lighting and plug loads and simulated HVAC performance and (ii) suggestions from stakeholders. Evaluation report, BEMOSS demonstration plan in real buildings, and enhanced BEMOSS software will also be delivered.

Task 8: Engineering design for BEMOSS integration with buildings
Engineering design of the control platform that is to be integrated with BEMOSS hardware for application in the selected three real-life buildings in year 3 will be completed.

PHASE 3: Demonstration in buildings and transition of BEMOSS software package
Task 9: Demonstration in three small and medium sized buildings
The BEMOSS platform and selected hardware controllers will be deployed in three small and medium-sized commercial buildings selected in this project. These buildings are: (1) the Equipment Bureau building operated by the Arlington County (Virginia) Government, (2) the School of Public and International Affairs academic building operated by the Virginia Tech Foundation in Alexandria, VA, and (3) the commercial office and retail building operated by the Virginia Tech Foundation in Blacksburg, VA. BEMOSS functionality test will be conducted, and its operational availability will be evaluated.

Task 10: Real-time on-site measurements and estimation of electricity savings potential
Electricity consumption of lighting loads will be recorded in the Equipment Bureau building (Arlington, VA) and that of selected HVAC loads will be recorded in the School of Public and International Affairs academic building (Alexandria, VA) and the commercial office and retail building (Blacksburg, VA). Electricity savings due to BEMOSS deployment will be estimated. These electricity consumption measurements can serve as a basis for any future measurement, verification and valuation (MV&V) projects.

Task 11: Demonstration of BEMOSS ability to perform fault detection
Under this task, BEMOSS ability to report abnormal behaviors of the HVAC units will be demonstrated in the School of Public and International Affairs academic building in Alexandria, VA.

Task 12: Transition of BEMOSS to v3.5
The current BEMOSS v2.0 will be migrated to v3.5 based on VOLTTRON 3.5. Laboratory tests will be conducted to ensure that all BEMOSS features previously available are functional after the migration.

Task 13: Delivery of BEMOSS software tools, v2.0 and v3.5
The completed BEMOSS software will be made available through an online repository and will be downloadable under BSD License, 3.0, or something similar. After the lab testing of BEMOSS 3.5, it will be deployed at the three pilot locations for field functionality tests. Software issues will be analyzed and bugs fixed to capture user inputs and seasonal variations in building operations after which BEMOSS 3.5 will be posted to GitHub and released to the public.

Task 14: Project Management and Reporting (all Phases)
Quarterly progress reports, financial reports and annual reports will be provided in accordance with the Federal Assistance Reporting Checklist following the instructions included therein. Project deliverables, including BEMOSS software and software interface for plug & play hardware device integration can be provided for peer-review/project review meetings if requested. The BEMOSS team at Virginia Tech needs one month to complete the final report after all experiments are completed and data gathered. The month of June 2017 is set aside for this activity.

Figure 1-1. BEMOSS tasks according to the SOPO

1.2 Accomplishments

Accomplishments are summarized by Task, as shown in the table below.

Table 1-1. Summary of accomplishments

Tasks	Summary of accomplishments	Details in
Task 1: BEMOSS™ open source software development in consultation with Industry	BEMOSS™ system and software architectures were developed in consultation with the industry from the onset of the project. As of June 2017, BEMOSS™ advisory committee comprises representatives from 22 organizations.	Sections 2.0 and 3.0
Task 2: BEMOSS user interface and software tool design	BEMOSS™ user interface (UI) was developed.	Section 4.0
Task 3: Plug & play device integration	BEMOSS™ was designed to automatically discover supported load controllers in buildings.	Section 5.0
Task 4: Incorporate additional software features	Additional software features include distributed database, plug & play (added security for BEMOSS™ discovery process), auto-mapping (locating devices with BEMOSS™ device identification process), as well as others, including synchronization of devices' and BEMOSS™ schedules, alarm/notification, OpenADR integration, multi-node operation, and Encrypted core-to-node communications.	Sections 5.0 and 6.0
Task 5: BEMOSS™ software open source access and survey	Information about BEMOSS™ is available online at www.bemoss.org. BEMOSS™ source code is available for download on Github, www.github.com/BEMOSS.	Section 11.0
Task 6: BEMOSS™ advanced algorithm development	These include anti-tampering, illuminance-based control and fault detection algorithms.	Section 7.0
Task 7: BEMOSS lab scale testing	Laboratory scale testing of BEMOSS™ was performed.	Section 8.0
Task 8: Engineering design for BEMOSS™ integration with buildings	The design and integration of BEMOSS™ were accomplished at the three demonstration sites.	Section 9.0
Task 9: Demonstration in three small and medium sized buildings	BEMOSS™ demonstrations were performed in three buildings: (1) the School of Public and International Affairs academic building in Alexandria, VA, (2) the Equipment Bureau building in Arlington, VA, and (3) the commercial office and retail building in Blacksburg, VA.	Section 9.0
Task 10: Real-time on-site measurements and estimation of electricity savings potential	Measurements were recorded for at least one year at each demonstration site and electricity savings potential was estimated.	Section 10.0
Task 11: Demonstration of BEMOSS™ ability to perform fault detection	Fault detection algorithms were demonstrated to detect abnormal indoor temperature patterns.	Section 7.3
Task 12: Transition of BEMOSS™ to v3.5	The latest release of BEMOSS™ was based on VOLTTRON™ v3.5.	Section 11.0
Task 13: Delivery of BEMOSS™ software tools, v2.0 and v3.5	BEMOSS™ v2.0 was released in March 2016. BEMOSS™ v3.5 was released in June 2017.	Section 11.0
Task 14: Project Management and Reporting (all Phases)	• Progress presentations were made monthly from Dec 2013 – June 2017. • Quarterly progress reports were submitted every quarter. • Continuation reports were submitted in August 2014, October 2015 and December 2016. • Final report was submitted in July 2017 (This report).	This document

1.3 Success Toward Meeting Technical Requirements

This section describes how BEMOSS™, its user interface & software tool, and its supported hardware devices meet the specified requirements identified in the SOPO.

1.3.1 Meeting Platform Requirements

a) Key platform requirements

From FOA Page 7, key platform requirements are:

- Interoperability
- Scalability
- Ease of deployment
- Open architecture
- Plug-n-play capabilities
- Ability to provide local and remote monitoring

BEMOSS™ addresses all six requirements as specified. This is summarized in Section 2.2.

b) Seamless integration with hardware components prevalent in small- to medium-sized buildings

From FOA Page 7, "the software will be designed to connect and communicate seamlessly with all the hardware components prevalent in small- to medium-sized commercial buildings, including stand-alone input devices such as the utility power meter, temperature sensors, occupancy sensors and others."

As of June 2017, BEMOSS™ was capable of integrating with selected thermostats, lighting and plug load controllers with open Application Programming Interface (API) and communicating using different communication technologies and data exchange protocols. See Section 2.5 for the list of hardware devices integrated with BEMOSS.

c) Market adoption of BEMOSS™

From FOA Page 8, "The applicant must describe how, following the end of this three-year project, the proposed open source architecture solution might be sustained as the platform matures and the building controls market adopts and deploys this solution."

Sustainability and wider adoption of BEMOSS™ had been ensured because of the following actions:

1) BEMOSS was built upon VOLTTRON™ with strong community support, contributing in developing different types of applications and providing enhanced security.

2) A Virginia start-up company, with roots in the BEMOSS™ project, applied for and received an NSF STTR grant to further develop and commercialize the BEMOSS™ software solution. The software platform under development is called BEMOSS®-Plus. Grant details follow.

- "STTR Phase I: An Agent-based Self-learning System for Efficient Building Operations and Automated Participation in Electricity Markets", sponsored by U.S. National Science Foundation (NSF), July 2016 – June 2017, PI – M. Pipattanasomporn (BEM Controls), co-PI – M. Kuzlu (Virginia Tech), Amount: $224,856.

This company, BEM Controls, LLC (www.bemcontrols.com), gave a subcontract to Virginia Tech to continue collaboration with the university. BEM Controls is now entering the lower cost building automation system market. The company has deployed the commercial solution in a public building at the Prince Georges County in Maryland and is in negotiations with other county and city-owned buildings in Virginia.

d) Diverse control needs for small- and medium-sized buildings

From FOA Page 8, "One must appreciate the diverse controls needs of small- and medium-sized commercial buildings since flexible and adaptable solutions will be required."

This requirement was addressed by making sure that several types of load controllers prevalent in small- and medium-sized commercial buildings could be interfaced with BEMOSS™.

- HVAC controllers, several thermostats, RTU and VAV were able to interface with BEMOSS™.
- Several lighting load controllers, including both stand-alone solutions and circuit-level solutions, were demonstrated to be compatible with BEMOSS™.
- Several plug load controllers, including both stand-alone solutions and circuit-level solutions, were successfully integrated with BEMOSS™.
- Data management and alarm management with email/SMS notifications were demonstrated.
- The ability to accept OpenADR signals for peak demand reduction was integrated with BEMOSS™.

e) Sharing common data/information

From FOA Page 8, "The open source architecture software solution must support data/information exchange such that devices within the building can share common data/information and enable a 'transaction' to occur against a set of requirements such as cost, comfort or energy consumption".

Since BEMOSS™ is an agent-based platform and built upon VOLTTRON™, it utilizes VOLTTRON interconnect protocol (VIP). This architecture allows data exchange among different types of devices in a building.

1.3.2 Meeting General Technical Hardware Specifications

The description below summarizes how the BEMOSS™ operating system meets general technical hardware specifications.

a) Three plug-and-play devices

From FOA Page 8, "In parallel to the software solution development, three plug and play devices must be developed to demonstrate functionality of the platform. These include: 1. Thermostat device; 2. Lighting load controller device; 3. General purpose control device."

The Virginia Tech team focused on integration of commercially available products (e.g., smart thermostats, smart plugs, VAV controllers, etc.), which could be accessed from a single-board computer running BEMOSS™. This was to ensure that they can be easily integrated with BEMOSS without any customized modifications.

b) Hardware specifications

The design and development of the BEMOSS operating system and associated components were made according to the specifications listed in the Appendix of the SOPO (and on Pages 9-10 of FOA). This is further explained in Section 2.6.

1.3.3 Meeting Requirements for User Interface and Software Tools

Overall, BEMOSS^{TM} UI allows:

1. **Control system set-up** – the BEMOSS^{TM} UI enables a user to make various inputs to all load controllers in the system, including schedules, on/off control and set points.
2. **Systems status display** – the BEMOSS^{TM} UI is capable of providing a graphical display of the status of all load controllers and sensors in the system, and allowing access to historical data.
3. **Control system point auto-mapping** – BEMOSS^{TM} is capable of communicating with and querying all supported devices. This is to detect the presence of the devices in a building; query their addresses, models; and identify device capabilities. This task is accomplished by the device discovery agent. Devices appear on the BEMOSS^{TM} dashboard page once they are discovered. The physical location of each device can be visually inspected by using BEMOSS^{TM}'s device identification feature.

In addition, BEMOSS^{TM} UI was designed to address the 'User interface and Software Tools specifications and desired features/benefits' as specified in the Appendix of SOPO (and on Pages 10-11 of FOA). This is further explained in Section 4.7.

2.0 Introduction to BEMOSS™

2.1 What is BEMOSS™?

BEMOSS™ stands for Building Energy Management Open Source Software. As an open-source open-architecture platform, BEMOSS™ was engineered to improve sensing and control of equipment in small- and medium-sized commercial buildings. BEMOSS™ offered: scalability, robustness, plug and play, open protocol, interoperability, cost-effectiveness, as well as local and remote monitoring, allowing it to work with load control devices from different manufacturers that operate on different communication technologies and protocols.

2.2 BEMOSS™ Key Features

BEMOSS™ offers the following key features:

- **Open source, open architecture** – BEMOSS™ is an open source software platform that is built upon VOLTTRON™ – a distributed agent platform developed by Pacific Northwest National Laboratory (PNNL). BEMOSS™ was designed to have an open architecture to make it easy for hardware manufacturers to seamlessly interface their devices with BEMOSS™. Software developers can also contribute to adding additional BEMOSS™ functionalities and applications.
- **Plug & play** – BEMOSS™ was designed to automatically discover supported load controllers (including smart thermostats, VAV/RTUs, lighting load controllers and plug load controllers) in buildings. Once the device is discovered, BEMOSS can then monitor and control the specific device for the desired function.
- **Interoperability** – BEMOSS™ was designed to work with load control devices form different manufacturers that operate on different communication technologies and data exchange protocols. These include both new commercially available products that operate on Ethernet and Wi-Fi, as well as legacy devices that operate on serial communications using Modbus RTU and BACnet MS/TP protocols. Currently, BEMOSS™ supports the following prevalent communication technologies: Ethernet (IEEE 802.3), Serial (RS-485) and Wi-Fi (IEEE 802.11); and protocols: BACnet, Modbus, Web and OpenADR.
- **Cost effectiveness** – Implementation of BEMOSS™ deemed to be cost-effective as it was built upon a robust open source platform that could operate on a low-cost single-board computer, such as Odroid. This feature could contribute to its rapid deployment in small- or medium-sized commercial buildings.
- **Scalability and ease of deployment** –With its multi-node architecture, BEMOSS™ provided a distributed architecture where load controllers in a multi-floor and high occupancy building could be monitored and controlled by multiple single-board computers hosting BEMOSS™. This makes it possible for a building engineer to deploy BEMOSS™ in one zone of a building, be comfortable with its operation, and later on expand the deployment to the entire building to make it more energy efficient.
- **Ability to provide local and remote monitoring** – BEMOSS™ provided both local and remote monitoring ability with role-based access control.
- **Security** – In addition to built-in security features provided by VOLTTRON™ including agent

authorization & authentication, encrypted multi-layer communication and agent validation, BEMOSS™ provided enhanced security features, including BEMOSS™ discovery approval process, encrypted core-to-node communication, thermostat anti-tampering feature and many more.

- **Support from the Advisory Committee** – BEMOSS™ was developed in consultation with an advisory committee from the beginning of the project. BEMOSS™ advisory committee comprised representatives from the following 22 organizations: Arlington County Government, Argonne National Laboratory, Automated Logic Corporation, BACnet International, Clark Energy Group LLC, Danfoss Corporations, Dominion Virginia Power (DOM), Eaton Corporation, Echelon Corporation, Electric Power Research Institute (EPRI), Emerson Electric, General Electric – GE Appliances, Honeywell International Inc., ICM Controls, Johnson Controls Inc., Lakeview Group LLC, LG Electronics, McKinstry, Rheem, United Technologies Research Center (UTRC), U.S. Department of Defense (DOD) and Wattstopper. See Figure 2-1.

  Their involvement allowed the to design BEMOSS™ across the full spectrum of current state-of-the-art hardware, and their deployment practices.

Figure 2-1. Organizations represented by BEMOSS™ advisory committee

2.3 Target Buildings and Target Load Types

Buildings consume over 40% of the total energy consumption in the U.S. According to U.S. Energy Information Administration (EIA), and a vast majority of commercial buildings are relatively small.

As shown in Figure 2-2, about half (50%) of all commercial buildings are 5,000 square feet or smaller in size. Medium-sized commercial buildings (between 5,001 to 50,000 square feet) constitute almost the other half (44.1%). These buildings (totaling about 94% of all commercial buildings in the U.S.) typically do not use Building Automation Systems (BAS) to monitor and control their buildings from a central location. This is primarily due to the unavailability of low-cost and simple BAS that can function without the presence of a building engineer on-site. BEMOSS™ was designed to target small and medium-sized commercial buildings to address this gap.

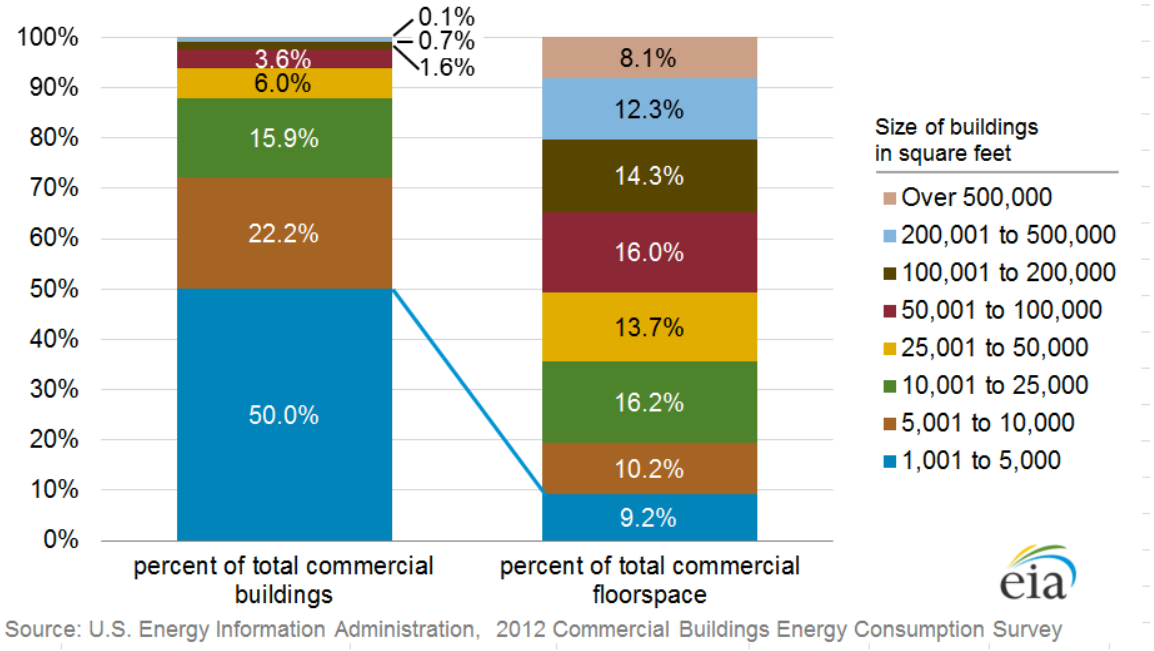


Figure 2-2. Percent of total commercial buildings and floor space by size of buildings[1]

There are three major loads in commercial buildings: HVAC, lighting and plug loads. According to the data from EIA 2012[2], electricity use by HVAC equipment, i.e., space heating, cooling and ventilation accounts for 33% of the total electricity consumption in buildings. Lighting loads constitute about 17%. Electricity use by plug loads, i.e., cooking, refrigeration, office equipment, computers, etc. accounts for 20% of total electricity use in buildings. Other loads include water heating, elevators, etc., constitute about 30% of the total electricity consumption in buildings. BEMOSS™ was designed to control three major loads in buildings: HVAC, lighting and plug loads.

[1] EIA – 2012 Commercial Building Stock [Online]: Available: https://www.eia.gov/consumption/commercial/reports/2012/buildstock/

[2] EIA - Commercial Building Energy Consumption Survey (CBECS), Table E5. Electricity Consumption (kWh) by End Use, 2012 [Online]. Available: https://www.eia.gov/consumption/commercial/data/2012/c&e/cfm/e5.php.

Figure 2-3 illustrates electricity use in buildings by load type.

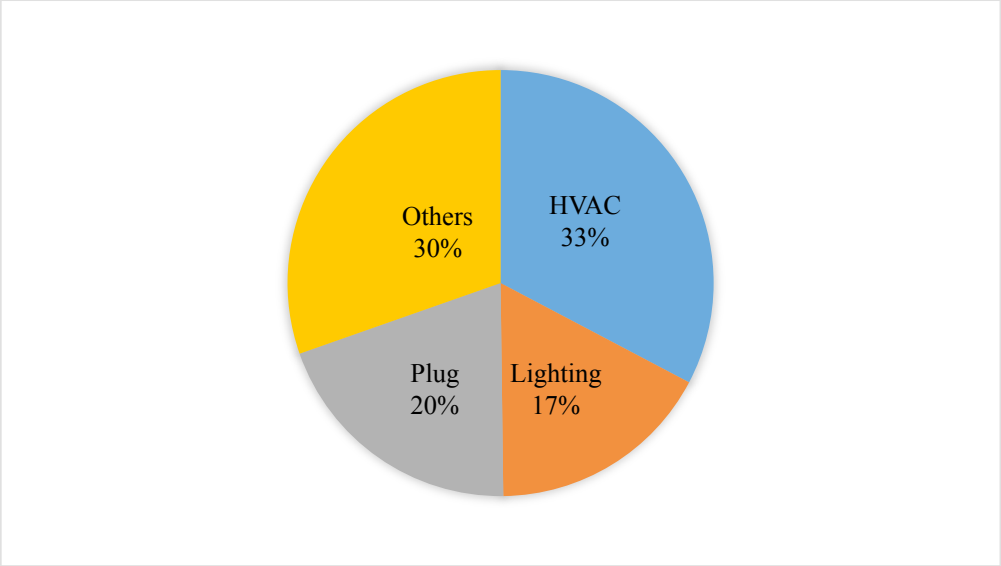


Figure 2-3. Electricity consumption by end use for all buildings (source: EIA)

2.4 List of Supported Communication Technologies and Protocols

Various technologies are available that allow communications between BEMOSS™ and selected load controllers in a building. As of June 2017, BEMOSS™ supported the following prevalent communication technologies: Ethernet (IEEE 802.3), Serial (RS-485) and Wi-Fi (IEEE 802.11), as summarized in Table 2-1.

Table 2-1. Communication technologies supported by BEMOSS™

Technology	Standard/ Protocol	Max. Theoretical Data Rate	Coverage Range
Wired Communication Technologies			
Ethernet	IEEE 802.3	10 Mbps- 1 Gbps	up to 100 m
Serial	RS-485	100 kbps – 35 Mbps	up to 1,200 m
Wireless Communication Technologies			
WiFi	802.11x	2-600 Mbps	up to 100 m

Devices communicating using the same communication technology may utilize different data exchange protocols. For a building energy management system, there are many protocols that are popular or becoming popular. As of June 2017, BEMOSS™ supported the following protocols: BACnet, Modbus, Web and OpenADR, as summarized in Table 2-2.

Table 2-2. Data exchange protocols supported by BEMOSS™

Data exchange protocol	Application	Allow communications over:		
		Ethernet	Serial	Wi-Fi
1. BACnet (IP)	Building automation	X		X
BACnet (MS/TP)			X	
2. Modbus (RTU)	Legacy device communications		X	
Modbus (TCP)		X		X
3. Web (e.g., XML, JSON, RSS/Atom)	Numerous applications	X		X
4. OpenADR	Demand response	X		X

2.5 List of Supported Hardware Devices

List of devices supported by BEMOSS™ as of June 2017 is summarized in Table 2-3.

Table 2-3. List of devices supported by BEMOSS™

Device Model	Vendor	Protocol
HVAC load controllers		
CT30 w/ WiFi USNAP module	RadioThermostat	WiFi
CT50 w/ WiFi USNAP module	RadioThermostat	WiFi
CT80 w/ WiFi USNAP module	RadioThermostat	WiFi
PL-VC1000	Prolon	Modbus RTU
PL-VC2000	Prolon	Modbus RTU
M1000	Prolon	Modbus RTU
M2000	Prolon	Modbus RTU
Lighting load controllers		
WeMo light switch	Belkin	WiFi
Philips Hue	Philips	WiFi/Ethernet
LMRC-212 0-10V dimming room controller	Wattstopper	BACnet
Plug load controllers		
WeMo smart plug	Belkin	WiFi
WeMo insight switch	Belkin	WiFi
LMPL-201 digital plug load controller	Wattstopper	BACnet
Sensor		
LMLS-400	Wattstopper	BACnet

2.6 Capabilities of Hardware Devices per FOA Requirements

2.6.1 HVAC Controllers

The following table summarizes thermostat capabilities designed to address the 'thermostat device specifications' as specified in the FOA on Page 9. As can be seen in the table, not all selected thermostats can meet the specified requirements. However, BEMOSS™ was designed to allow these thermostats to meet the requirements specified in the FOA.

Table 2-4. Capability of selected thermostats based on specifications as specified in the FOA, and BEMOSS™ capabilities to supplement the thermostat requirements

Thermostat device specifications	CT30/CT50	CT80	PL-VC1000/2000	M1000/2000	Handled by BEMOSS™
Configured for wired or wireless communication	X	X	X	X	
Use of a new or existing industry-recognized communication protocol	X	X	X	X	
Use of a local display with keypad, touchscreen or laptop computer for programming purposes	X	X	X	X	
Capable of accepting set point changes, schedule changes and time synchronization commands from a gateway/master controller, if connected to them	X	X	X	X	X
Capable of supporting web services	X	X			X
Capable of handling holiday scheduling with the ability to make changes locally	X	X			X
Capable of handling occupancy scheduling, including capabilities for: occupied and unoccupied set points; occupancy sensors; occupancy override; and programmable alarm					X
Capable of receiving signals from external sources and act on them, for example, demand response, price, reliability, ancillary service signals from utilities or ISOs		X			X
Multiple zone temperature sensor inputs (average, high or low; wired and wireless)					X
Economizer mode with the ability to make changes locally at the thermostat	X	X			
Spare digital and analog inputs (two each) for monitoring/diagnostics			X	X	
Outdoor air temperature inputs (wired, wireless or web-connected)			X	X	X
Alarm parameters for temperatures, equipment runtime hours or performance issues		X			X
Capable of sending email or text upon alarm activation via a communication service					X
Three levels of security via password authorization					X

2.6.2 Lighting Load Controllers

The following table summarizes lighting controller capabilities designed to address the 'lighting controller specifications' as specified in the FOA on Pages 9-10. As can be seen in the table, not all selected lighting controllers can meet the specified requirements. However, BEMOSS™ was designed to allow these lighting load controllers to meet the requirements specified in the FOA.

Table 2-5. Capability of selected lighting controllers based on specifications as specified by the FOA, and BEMOSS™ capabilities to supplement the lighting controller requirements

Lighting controller specifications	WeMo light switch	Philips Hue	Wattstopper LMRC-212	Handled by BEMOSS™
Configured for wired or wireless communication	X	X	X	
Use of a new or existing industry-recognized communication protocol	X	X	X	
Use of a local display with keypad, touchscreen or laptop computer for programming purposes	X	X	X	X
Capable of accepting set point changes, schedule changes and time synchronization commands from a gateway/master controller, if connected to them	X		X	X
Capable of supporting web services	X	X		X
Capable of handling holiday scheduling with the ability to make changes locally			X	X
Capable of handling occupancy scheduling, including capabilities for: occupied and unoccupied set points; occupancy sensors; occupancy override; and programmable alarm			X	X
Daylighting control for perimeter spaces with window/natural daylighting with the ability to make changes locally at the controller			X	X
Exterior lighting control via astrological clock and photocell with the ability to make changes locally at the controller			X	X
Capable of receiving signals from external sources and act on them, for example, demand response, price, reliability, ancillary service signals from utilities or ISOs				X
Alarm parameters for equipment runtime hours or equipment performance issues				X
Capable of sending email or text upon alarm activation via a communication service				X
Three levels of security via password authorization				X

2.6.3 Plug Load Controllers

The following table summarizes plug load controller capabilities designed to address the 'general purpose controller specifications' as specified in the FOA on Page 10. As can be seen in the table, not all selected plug load controllers can meet the specified requirements. However, BEMOSS™ was designed to allow these plug load controllers to meet the requirements specified in the FOA.

Table 2-6. Capability of selected plug load controllers based on specifications in the FOA, and BEMOSS™ capabilities to supplement the plug load controller requirements

Plug load controller specifications	WeMo smart plug	Wattstopper LMPL-201	Handled by BEMOSS™
Configured for wired or wireless communication	X	X	
Use of a new or existing industry-recognized communication protocol	X	X	
Use of a local display with keypad, touchscreen or laptop computer for programming purposes	X	X	X
Capable of supporting web services	X		X
Capable of handling holiday scheduling with the ability to make changes locally	X	X	X
Capable of handling occupancy scheduling, including capabilities for: occupied and unoccupied set points; occupancy sensors; occupancy override; and programmable alarm		X	X
Capable of receiving signals from external sources and act on them, for example, demand response, price, reliability, ancillary service signals from utilities or ISOs			X
Three levels of security via password authorization			X

3.0 BEMOSS™ Architectures

3.1 System Architecture

A simple BEMOSS™ architecture is illustrated in Figure 3-1 for a small commercial building with a few load controllers of each type. In this architecture, only one single-board computer (e.g., Odroid) with BEMOSS™ installed is sufficient to enable monitoring and control features of all load controllers in the building. This embedded system can communicate with different types of load controllers, i.e., thermostats, lighting load controllers and plug load controllers via wireless signals (e.g., Wi-Fi). Local and remote monitoring and control via a smart phone or a tablet are also enabled.

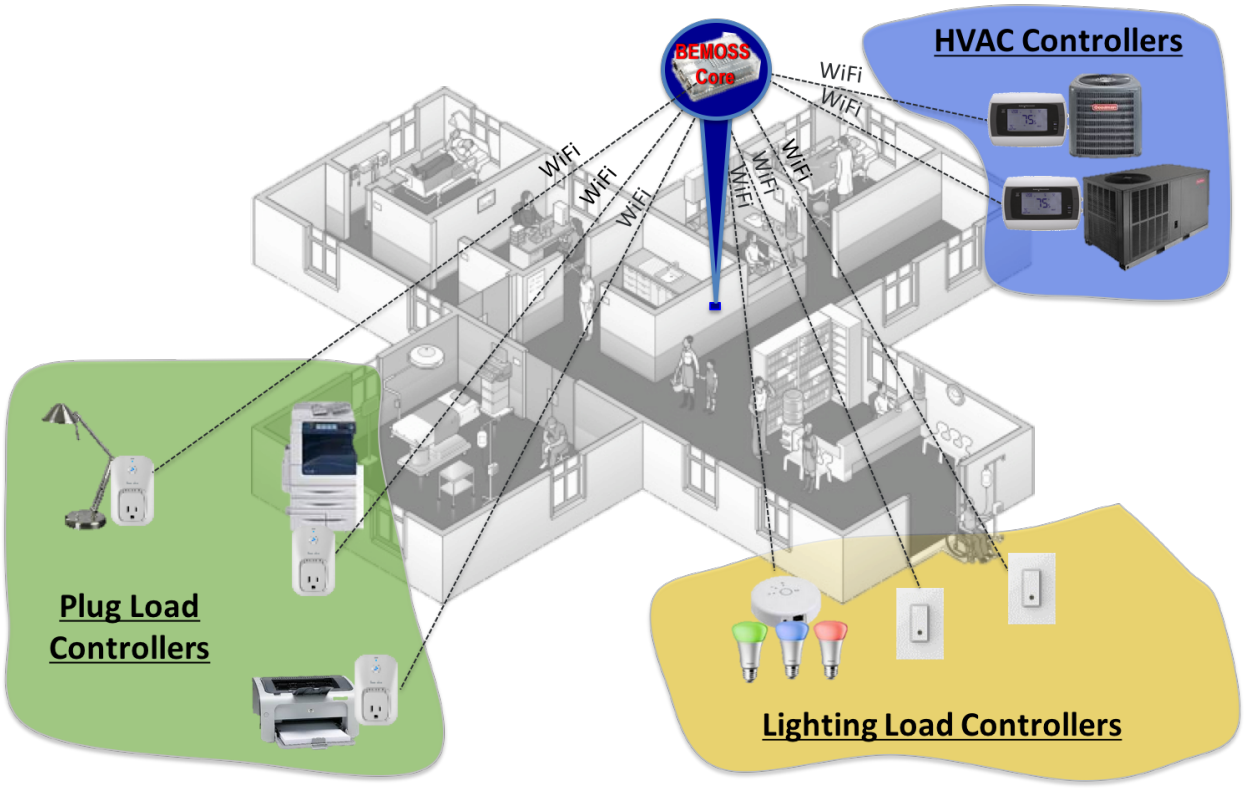


Figure 3-1. BEMOSS™ system architecture in small buildings with a few load controllers

For buildings with multiple floors and larger number of devices, BEMOSS™ can be set up to deploy its multi-node architecture, as shown in Figure 3-2. In this architecture, several BEMOSS™ nodes[3] communicate with the BEMOSS™ core[4]. Each node is responsible for monitoring and control of a zone in which a selected set of load controllers reside (in Figure 3-2, a zone is one floor), while the core is responsible for supervising operation of the overall system and allowing local and remote access for monitoring and control of all devices in buildings. This multi-node architecture enables BEMOSS™ to be scalable and makes the overall building operation more reliable. That is, it allows the migration of device agents from a failed node to the core. In the case where a BEMOSS™ node fails, devices responsible by that node will be moved to the core, enabling continuous operation of all devices.

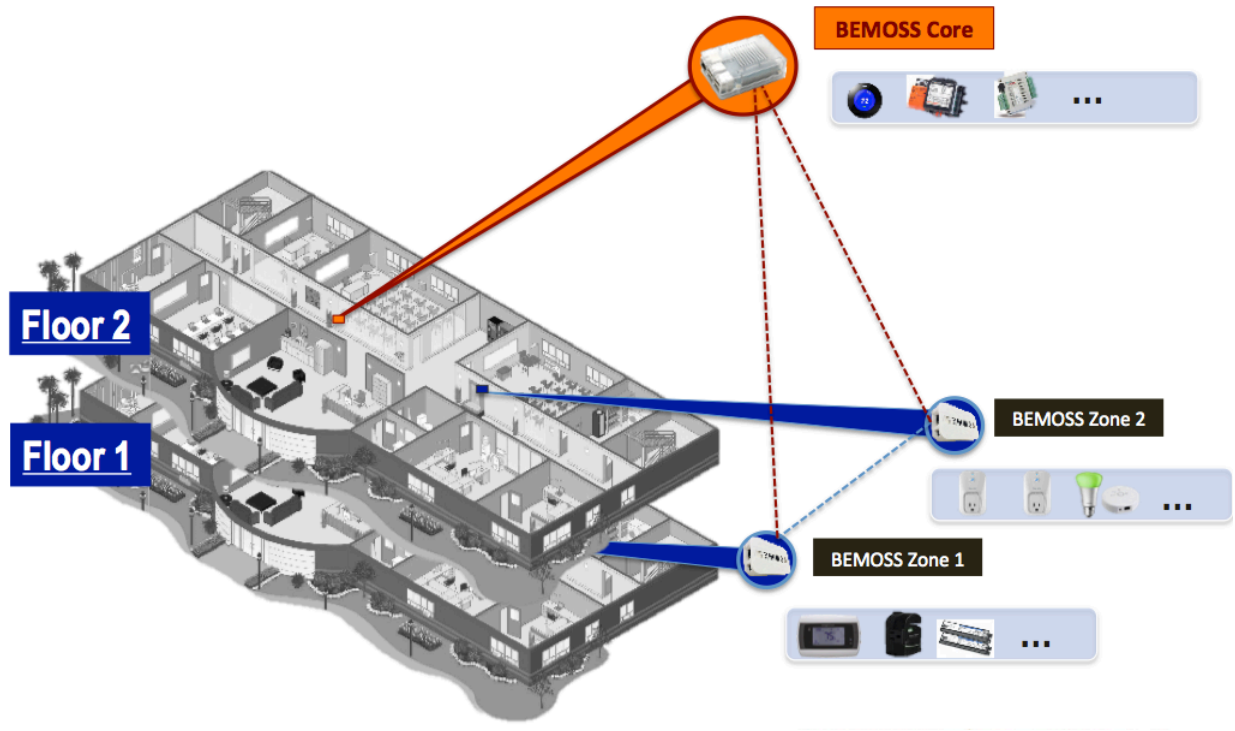


Figure 3-2. BEMOSS™ system architecture in larger buildings

3.2 Software Architecture

BEMOSS is a robust open source operating system for building energy management that is built entirely on open source software tools. The entire BEMOSS™ system comprises four layers: User Interface (UI) layer, Application and Data Management Layer, Operating System and Framework layer, and the Connectivity Layer.

[3] **BEMOSS™ node** is BEMOSS™ with limited functionalities. It supports connections with hardware devices, hosts selected agents, can run limited applications/algorithms and supports minimal database. It does not support user interface.
[4] **BEMOSS™ core** is a full version of BEMOSS™ that supports user interface.

Figure 3-3 depicts the overall BEMOSS™ software architecture.

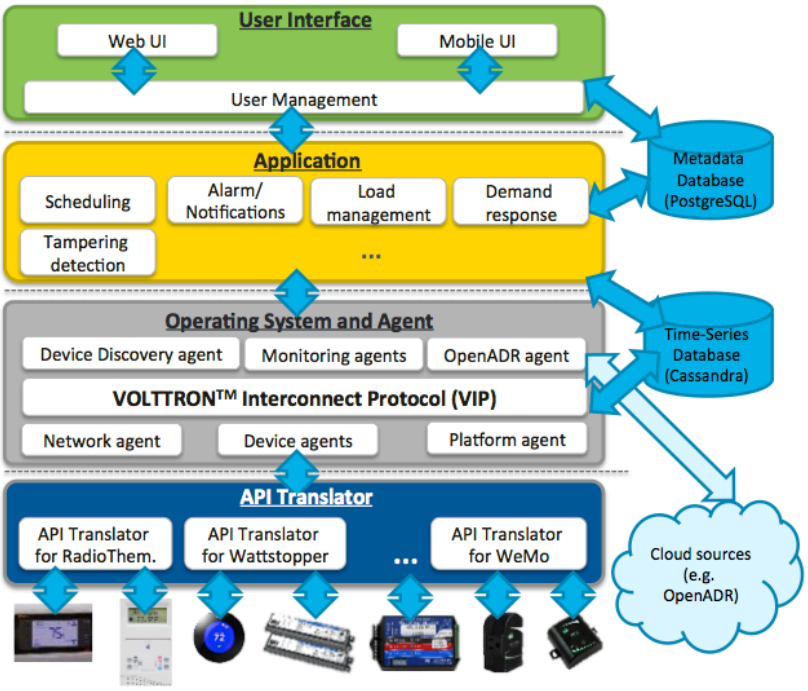


Figure 3-3. BEMOSS™ software architecture

Each layer is explained in more details below:

Layer 1: User Interface (UI) - The BEMOSS™ UI layer has two components: user management and web UI. Regarding user management in BEMOSS™, role-based access control is implemented to allow different levels of access to different individuals. For example, a building operator has full authority to adjust set points and schedules of loads in buildings, while tenants have limited access to view current status and historical load data, or control selected loads in specific zones. In BEMOSS™, this role-based access control is achieved using access control lists. With respect to BEMOSS™ web UI, it is a dashboard type interface with visuals and graphs to show current settings and status of devices. Authenticated users can control these devices through the interface.

Layer 2: Application and Data Management - This layer embeds algorithms to allow monitoring and control of hardware devices interfaced with BEMOSS™. Examples of possible applications include demand response, price-based management, planning and scheduling, behavior pattern analysis, load management, as well as alarm/notifications. BEMOSS™ databases are PostgreSQL and Apache Cassandra. While PostgreSQL stores metadata, Apache Cassandra stores time-series data due to its ability to access and deliver data in near real-time, as well as its distributed data storage architecture, which provides redundancy.

Layer 3: Operating System and Framework - In this layer, VOLTTRON™, a distributed agent platform developed by Pacific Northwest National Laboratory (PNNL), was chosen as the software platform for BEMOSS™. Several agents were developed to support BEMOSS™, including device discovery agent, device agents, demand response (DR) agent, and other system agents. Each agent is explained in more details below:

- **Device discovery agent** – is responsible for detecting the presence of devices in a building, querying their model numbers, identifying their APIs and launching a device agent to monitor/control the discovered device. With this approach, there is no need to manually identify each device beforehand using an approach, like bar codes or QR codes.
- **Device agents** – includes device agents for thermostat, lighting and plug load. These agents are instantiated to monitor, communicate and control hardware devices after being discovered by the device discovery agent. Once a device agent is initiated, it is assigned particularly to one hardware device.
- **DR agent** – performs priority-based load control when a DR signal is received from an electric utility or a DR aggregator.
- **Other system agents** –Other agents such as the network agent, approval helper agent and platform monitor agent, are responsible for the functioning of the BEMOSS™ system by facilitating the packaging, installation, starting, stopping, monitoring and managing the agent execution.

All agents communicate over the VOLTTRON™ VIP. The entire BEMOSS™ system is also designed to allow email/SMS notifications through its alarm/notification app.

Layer 4: Connectivity Layer - This layer takes care of the communication between the Operating System and Framework layer and all physical hardware devices. To allow BEMOSS™ to communicate with hardware devices that use different communication technologies, data exchange protocols and have device functionalities (different device APIs), the BEMOSS™ team created several API interfaces. Each API interface allows BEMOSS™ agents to communicate with a group of devices based on their unique APIs. Basically, API interfaces provide a translation service for BEMOSS™ agents so that agents can get readings and send control commands to devices (without knowing their APIs) using simple function calls: getDeviceStatus and setDeviceStatus.

4.0 BEMOSS™ User Interface

BEMOSS™ User Interface (UI) layer consists of two components: the UI (the web interface) and user management. The web application and user management layer resides in the central server along with the BEMOSS software.

4.1 Overview of Open Source Tools Used for BEMOSS™ UI

BEMOSS™ UI has an open architecture and can be expanded to add more pages and functionalities in the future. The web interface uses a collection of open source projects and is developed primarily in Python for the server side, and HTML, CSS, JavaScript/ jQuery for the client side (scripting). Open-source tools used to develop BEMOSS™ UI include:

- *Django web framework*: Django is an open source python web framework and follows the model-view-controller architectural pattern. The web interface is built using this framework. The powerful Object-Relational Mapping (ORM) features, template tags and the flexibility to add new components to the framework that is loosely coupled with the original framework allow developers to widen the web platform.
- *ZeroMQ:* is the BEMOSS™ message bus, which is used to send and receive messages and support communications between agents and UI. ZeroMQ is used because of its high performance asynchronous messaging library, and its ease of integration with VOLTTRON™, which is another open source tool being used in the Operating System and Framework Layer.
- *Twitter Bootstrap:* BEMOSS™ web interface uses Bootstrap as its front-end framework. Bootstrap uses HTML and CSS-based design templates for typography, forms and buttons and other interface elements along with optional JavaScript extensions. Bootstrap is chosen because of the responsive design it offers and the ease of design. The responsive design makes the web interface adaptable to a web browser of any dimension, smartphone, tablet or laptop or PC.
- *FontAwesome:* In addition to the CSS and HTML elements offered by Bootstrap, BEMOSS™ leverages the iconic font and CSS framework, FontAwesome, for its scalable vector icons for instant and quick customization. It is a supplement to Bootstrap framework and the icons are scalable.
- *jQuery and jQueryUI:* BEMOSS™ leverages the rich script collection from jQuery to display interactive and dynamic data on the web interface.
- *Python:* Python is the base language for the entire BEMOSS™ application. Python is a choice as VOLTTRON™ platform is built using Python.
- Other web interface open source tools used include *weather-icons* and *gauge.js*.

4.2 BEMOSS™ UI Pages

Dashboard is the BEMOSS™ main UI page. This is shown in Figure 4-1. It displays device icons and their current status organized by load types (i.e., HVAC, lighting and plug load controllers). This is also accessible from the side navigation bar.

Figure 4-1. BEMOSS™ dashboard page

Each of the widgets in the Zone dashboard has the following features:

- Current Status (ONLINE/OFFLINE; current temperature and set point for thermostats)
- A visual image of the device (depending on device type)
- Identify button (available to the administrator)
- View/Edit Information (available to the administrator)

In addition to the dashboard, BEMOSS^{TM} UI also has the following pages listed on the side navigation bar:

- ***HOME:*** The home page shows number of devices being monitored and controlled by BEMOSS^{TM} by device type.
- ***DISCOVER NEW DEVICES:*** This page allows the administrator to initiate the discovery process of new devices in a building. After the discovery process is completed, the number of discovered devices will appear on the Discover/Manage tab on the left navigation bar.
- ***DISCOVER/MANAGE:*** This page allows the administrator to approve the devices that belong to his/her building and delete any unknown devices from the system.
- ***NETWORK STATUS:*** On clicking, it expands to show 'Node Status' and 'Device Status' sub-nodes. These nodes show the status information (online/offline) of the nodes and devices approved in the system.
- ***ALARMS & NOTIFICATIONS:*** The Alarm & Notification tab is used to set alert events and check for notifications. The building operator can set an alarm to send a notification when a BEMOSS^{TM} node is offline or any BEMOSS^{TM} devices are offline. Notification channels are of three types: (i) BEMOSS^{TM} notification, which is the in-application notification; (ii) Email notification and (iii) SMS notification.
- ***MANAGE USERS:*** This page has a list of all active users and users with pending registration requests. The administrator can approve or deny a registration request, view the roles of registered users, and can assign or modified their roles.
- ***MISC SETTINGS:*** The Misc Settings tab available on the side bar is a quick way to navigate to the system settings, where general settings including managing zip code of the building location, holidays.
- ***APPLICATIONS:*** The Applications tab allows a user to initiate new illuminance-based control and fault detection features.

4.3 Device Monitor and Control Capability

When clicking any of the device icon in the dashboard page, the user is redirected to the respective device page. In this discussion, a thermostat is used as an example. The thermostat page shows detailed information about the thermostat referred to, including current temperature, set point, mode, and fan mode. The page also shows the outdoor temperature in the area, thereby making it comfortable to decide on a temperature set point. The user with a 'building admin' role can change the settings of the thermostat using this interface.

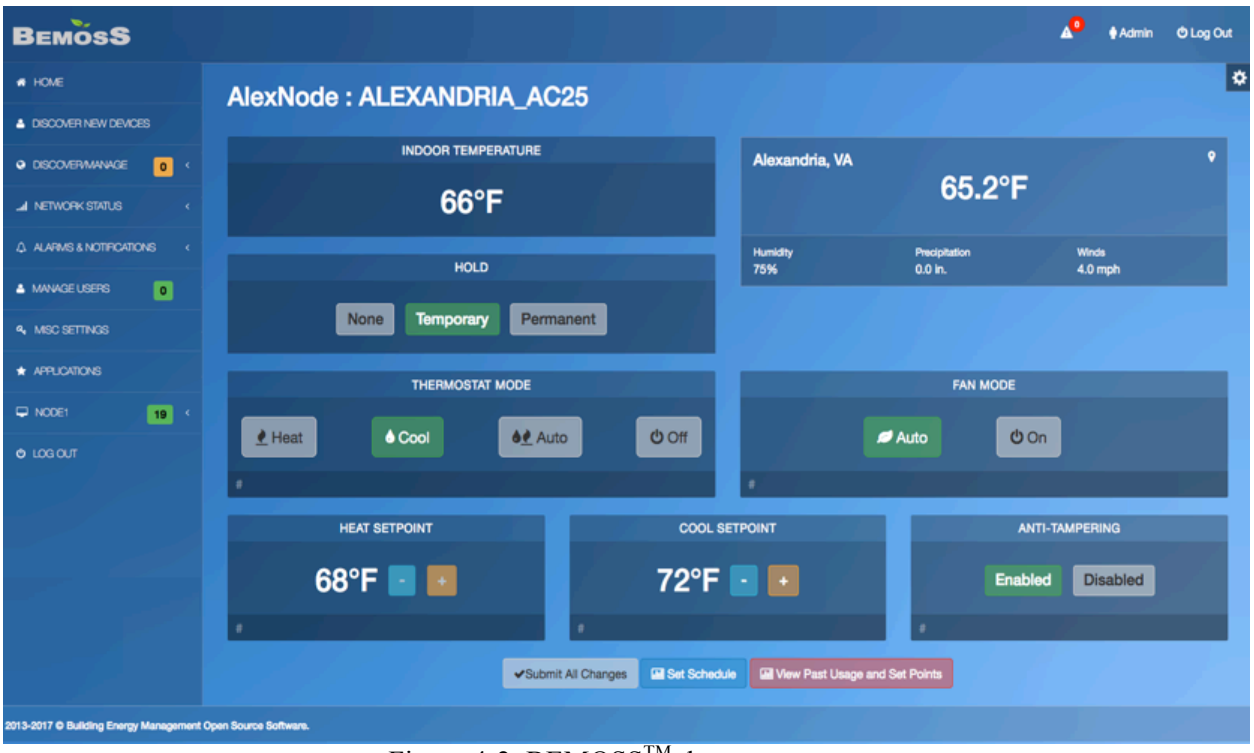


Figure 4-2. BEMOSS™ thermostat page

The ‘Set Schedule’ button, on click, redirects the user to the schedule page (see Figure 4-3) where the user can set weekday, weekend and holiday schedules.

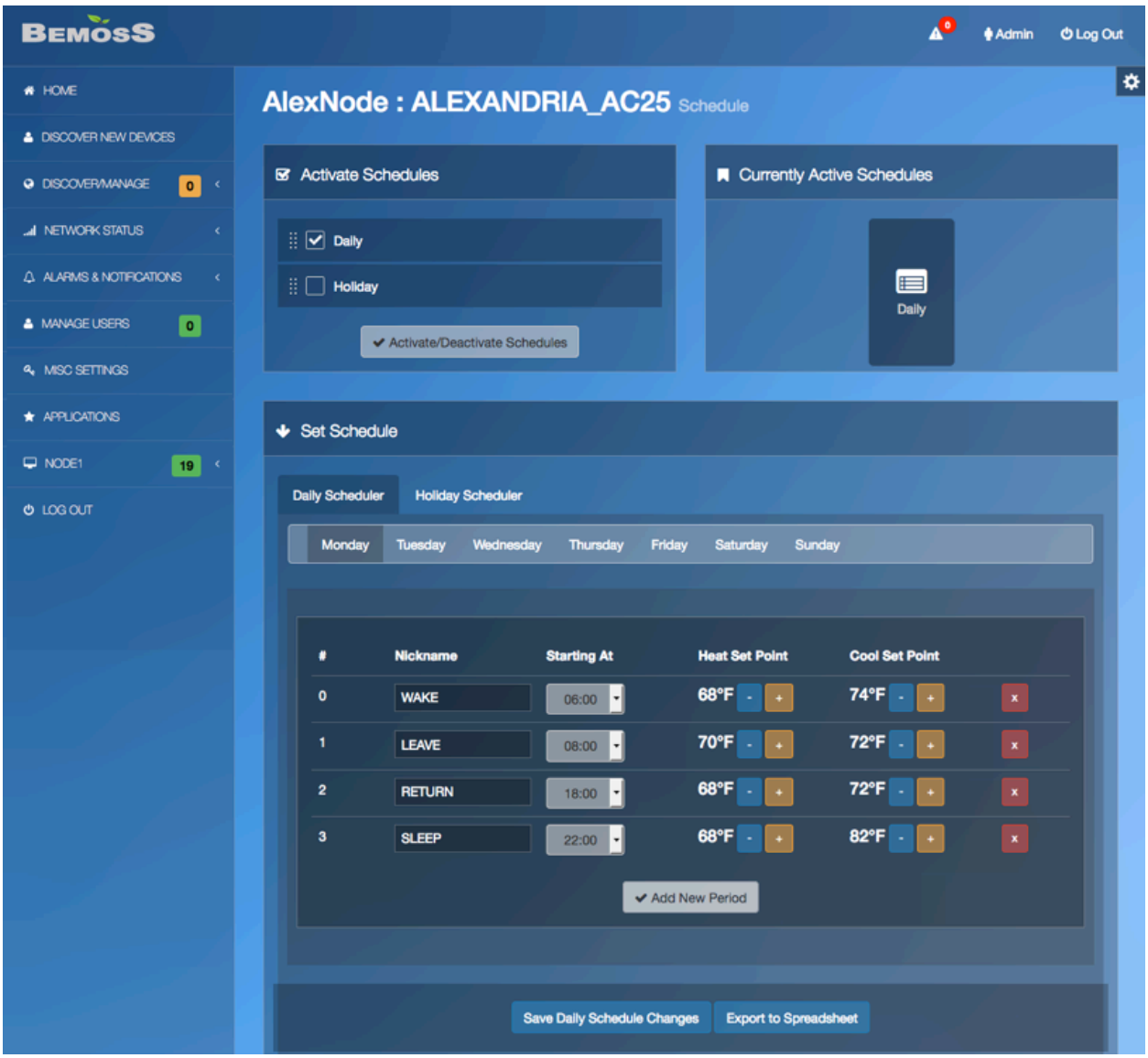


Figure 4-3. Thermostat schedule page

The 'View Past Usage and Set Points' button on the thermostat page, on click, redirects the user to access the historical data (see Figure 4-4) where the user can retrieve historical data of the thermostat, i.e., indoor temperature, as well as heat and cool set points.

Figure 4-4. Thermostat historical data page

4.4 User Management

On login, a user is redirected to a page based on his/her specific role. The user's role determines his/her access areas in the application. BEMOSS™ provides three user roles:

- **Administrator:** The administrator has the highest privileges and complete access to the system and can monitor and control the system without any restrictions. The administrator approves any new user registrations, and assigns zone managers to particular zones. The administrator authorizes new devices. The administrator can schedule devices, setup alarms, and notifications for any registered events in the system. The administrator can also generate reports for further analysis.
- **Zone Manager:** The administrator can assign a user to manage a particular zone in a building. The zone manager can then control devices within this zone. The zone manager effectively acts as an administrator to the zone. However, the zone manager does not have control over device name modification. This restriction preserves system naming conventions. The zone manager role can be assigned and revoked as per the needs of the building engineer.
- **Tenant:** Tenant is a general user in a building. A tenant can monitor devices but cannot control them. A tenant can request for any change in system or device settings, if needed. This is allowed using a 'Request Change' button in device pages. The zone manager or administrator can service this request by either allowing it or rejecting it based on system conditions.

4.5 Alarms & Notifications

The 'alarms and notifications' feature is included in BEMOSS^{TM} 3.5. With this feature, a building operator can register a specific event on the BEMOSS^{TM} UI and chooses a method to receive an alert (email/SMS). See Figure 4-5.

Modes of notifications: BEMOSS^{TM} allows three modes of notifications:

- *Email* – a building operator receives an email.
- *SMS* – a building operator receives an SMS.
- *BEMOSS^{TM} notification* – a building operator can see an alert on BEMOSS^{TM} UI. This is a default option, and all alerts/notifications will be shown on BEMOSS^{TM} UI.

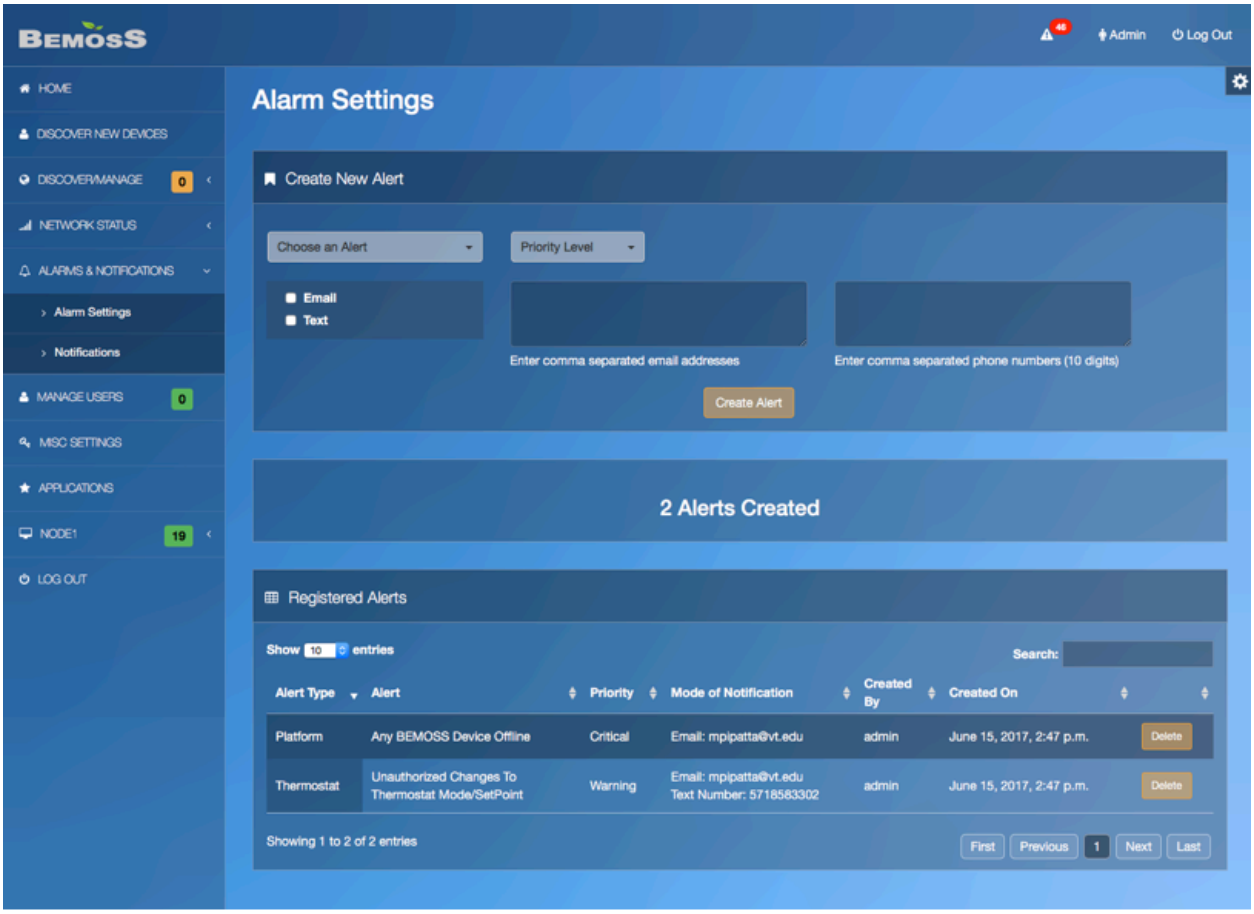


Figure 4-5. Alarm/notification page

Types of alerts: For BEMOSS^{TM} 3.5, four types of alarms are enabled:

- *Unauthorized changes to thermostat mode/set point* – a notification will be sent by email or SMS to a building operator if a thermostat has been tampered when the override mode is not allowed. The notification includes the information about the tampered thermostat (e.g., ID), original set point/mode, tampered set point/mode, and time of tampering.
- *AC system failure* – please refer to the fault detection section (Section 7.3) for more detail.

- *Any BEMOSS™ node offline* – a notification will be sent to a building operator when any BEMOSS™ node offline. The notification includes information about the offline node, such as its ID.
- *Any BEMOSS™ device offline* – a notification will be sent to a building operator when any BEMOSS™ device offline. The notification includes information about the offline device, such as its ID.

Alert priority levels: a building operator can choose the priority level associated with the alert created. The alert priority level selected will be included in the subject line of the email or SMS sent. Three alert options are:

- Warning
- Important
- Critical

4.6 Report Generation

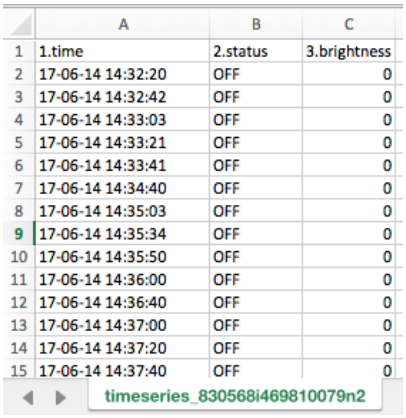
Report generation involves collecting information from a variety of sources, consolidating into appropriate tabs and sheets, and generating a report document at the click of a button.

In BEMOSS™ v3.5, reports can be generated for historical data. Generated reports are stored in a spreadsheet format. From the visualizations page, reports about time-series data collected for different data points for a device can be downloaded. This can also be filtered by date and time.

Figure 4-6 shows the device historical data page. On clicking the 'Export All to Spreadsheet' button, a file open dialog opens with a default file name asking for user instructions.

Figure 4-6. Export to spreadsheet option on the device historical data page

Figure 4-7 shows an example of device historical data file that is exported using the report printing application.



	A	B	C
1	1.time	2.status	3.brightness
2	17-06-14 14:32:20	OFF	0
3	17-06-14 14:32:42	OFF	0
4	17-06-14 14:33:03	OFF	0
5	17-06-14 14:33:21	OFF	0
6	17-06-14 14:33:41	OFF	0
7	17-06-14 14:34:40	OFF	0
8	17-06-14 14:35:03	OFF	0
9	17-06-14 14:35:34	OFF	0
10	17-06-14 14:35:50	OFF	0
11	17-06-14 14:36:00	OFF	0
12	17-06-14 14:36:40	OFF	0
13	17-06-14 14:37:00	OFF	0
14	17-06-14 14:37:20	OFF	0
15	17-06-14 14:37:40	OFF	0

timeseries_830568i469810079n2

Figure 4-7. Exported data shown as a spreadsheet

4.7 Summary of BEMOSS™ UI Features

BEMOSS™ UI was designed to address the 'User interface and Software Tools specifications and desired features/benefits' as stated in the FOA, as follows:

Table 4-1. Design of BEMOSS™ UI to meet UI specifications as specified by FOA

FOA specifications for UI	**How BEMOSS™ UI addresses the specifications**
Designed for open source standards	BEMOSS™ UI has been designed based on open source standards, using open source software, like Django, jQuery, ZeroMQ, Java Script and Twitter Bootstrap. See the discussion in Section 4.1.
Uses the language of the web (HTTP) to communicate over the Internet or intranet without special software or plug-ins	BEMOSS™ UI uses HTTP to enable communications over the Internet. See additional discussion in Section 4.1.
Extensive use of menus, toolbars, and icons to allow intuitive navigation and fast access to important information	BEMOSS™ uses sidebar menu and icons to allow intuitive navigation and fast access. In addition, the BEMOSS™ dashboard page provides a user an easy access to all discovered devices in the building. See the discussion in Section 4.2.
Adjusting of set points and other control properties	BEMOSS™ UI allows adjustment of thermostat set points and ON/OFF status of each type of load controllers. This is discussed in Section 4.3.
Display and control of field equipment	All load controllers can be controlled from its respective UI pages. An example of a thermostat page is discussed in Section 4.3.
Setting and changing of schedules	A schedule page is available to control each load type: thermostat, lighting and plug load controllers. A user is able to: • Schedule the change of thermostat set points • Schedule the ON/OFF(DIM) status of lighting loads • Schedule the ON/OFF status of plug loads.

	See an example of thermostat schedule page in Section 4.3.
Graphical trending of all important building conditions, including energy and comfort	BEMOSS™ UI shows graphical trending of selected parameters based on historical data stored in BEMOSS™ database. This is discussed in Section 4.3.
Intuitive, comprehensive building operation with dynamic, interactive graphical access	BEMOSS™ UI allows selected data to be shown in an interactive graphical format. A user can also customize the display period and data resolution. See Section 4.3.
Display of point status and history information	BEMOSS™ displays point status and history of each load controller in its respective UI page. See Section 4.3.
Multi-level passwords and Secure Sockets Layer (SSL) with 128-bit encryption for security	BEMOSS™ UI uses role-based access control, and SSL with at least 128-bit encryption. See Section 4.4. Also, please see a detailed description of BEMOSS™ security in Section 6.5.
Programming of alarms	BEMOSS™ UI allows a user to receive customized alarm/notification via email and/or text message. See the discussion in Section 4.5.
Advanced alarm management capabilities including email, pagers, network printers, etc.	Advanced alarm management capabilities, including email, SMS, have been delivered with BEMOSS™ v3.5. See the discussion in Section 4.5.
Acknowledgement of alarms on a priority basis	Alarm priority can be set as discussed in Section 4.5.
Presentation of data to a user in a clear and concise format	BEMOSS™ uses icons with few texts to allow its user-friendly feature. Data are presented in graphical format with an option to download as a CSV file. See Section 4.6.
Ability to initiate printing of reports	BEMOSS™ provides the ability to initiate printing of reports as discussed in Section 4.6.
High-resolution color graphics system status display that can be tailored to the requirements of each individual facility and designed to accommodate novice and experienced operators	BEMOSS™ UI allows a user to customize its look and layout using color setting widget. This is the gear icon in Figure 4-1.
Viewing, archiving and retrieval of event logs	BEMOSS™ provides the ability to view, archive, and retrieve of event logs in the historical data pages of each device. A user can also retrieve notification history in the 'Alarm/notification' page, as well as data on online/offline status of each device in the 'Network status' page.
Ability to monitor data communications channels	BEMOSS™ provides the ability to monitor data communications channels on the 'Network Status' page.

5.0 BEMOSS™ Plug & Play Feature

BEMOSS™ has an ability to discover supported devices in a building at the beginning of a deployment, and once any new devices are deployed. This is accomplished through the BEMOSS™ discovery process. Once discovered, discovered devices will need approval by the administrator before being added to the building control system. The device identification feature enables the building administrator to visually inspect the location of a particular device. These features are discussed below.

5.1 BEMOSS™ Discovery Process

Device discovery is carried out by the device discovery agent. As its name suggests, its primary purpose is to discover supported devices in the building. The discovery process for any device is divided into the following steps:

1. Detect the presence of a device in the building
2. Query the device to find out its model information
3. Look up the device in the list of supported devices and check if the corresponding API translator is available

The figure below illustrates the BEMOSS™ discovery page.

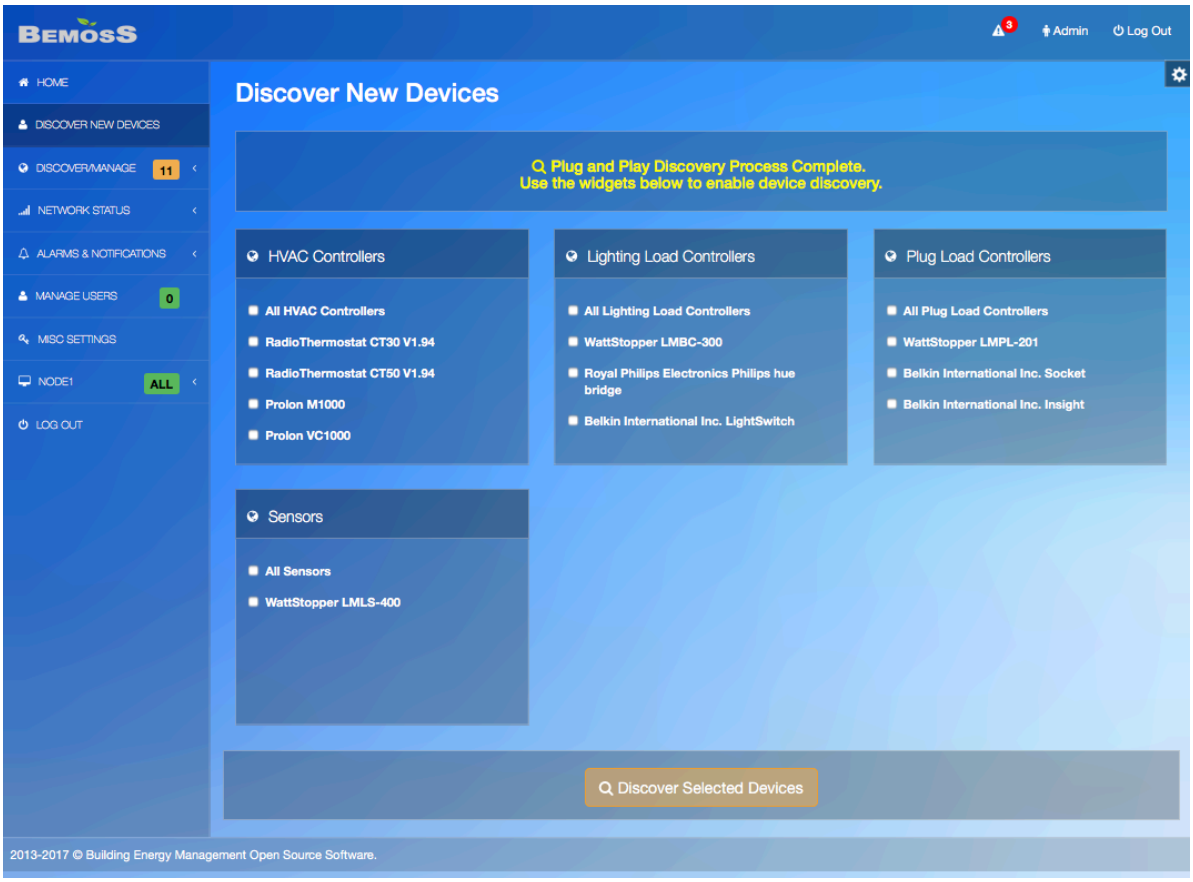


Figure 5-1. BEMOSS™ discovery page

If no new device is discovered in a preset number of consecutive cycles, the discovery process will stop to save resources. This is to eliminate unnecessary discovery broadcast messages. At any time, a manual discovery can be triggered from BEMOSS™ UI, under the 'Discover New Devices' tab. This allows a building administrator to add new devices later. As shown in Figure 5-1, manual discovery can be triggered for selected device types, as the administrator will usually know the type of the newly added device in the building.

5.2 Device Approval Process

As an added security for BEMOSS™ plug & play device integration, all devices can be controlled only after they are approved by the building administrator. Devices approval is located at the 'Discovery/Manage' tab on the left navigation bar. The building administrator can approve the devices that belong to his/her building and leave the unknown devices in the 'Pending' or 'Non-BEMOSS' status. This adds another layer of security where malicious devices are disregarded from the BEMOSS™ system.

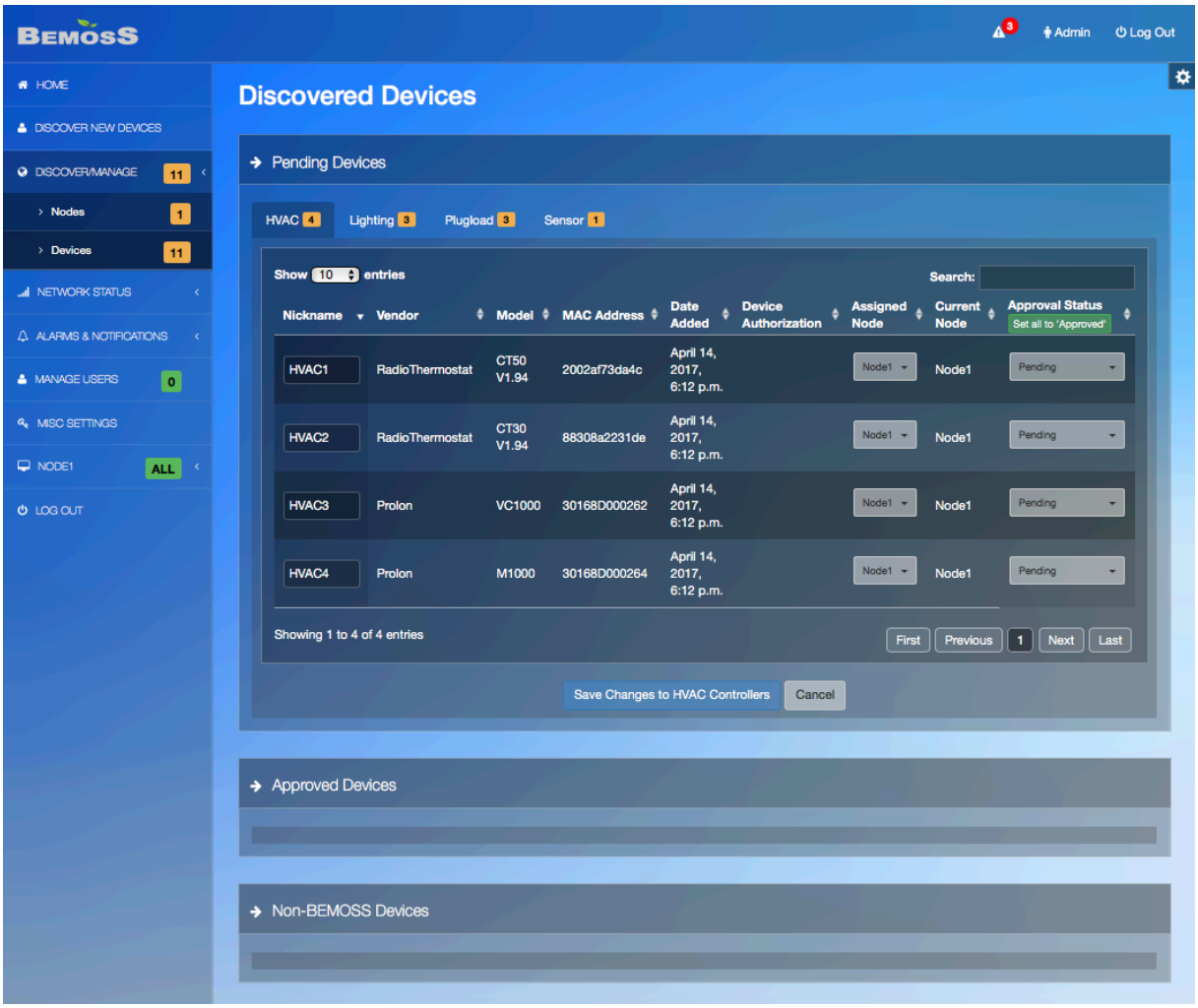


Figure 5-2. Device approval

5.3 Device Identification

BEMOSS™ also allows the building administrator to identify a particular device by sending an 'identification' signal to the device. This will make a selected thermostat to blink or its backlight turned on, or switch ON/OFF lights/plug loads. This step helps identify the physical location of devices, which will help the building administrator to assign a reasonable nickname for the device and change its zone (e.g., conference room thermostat, break room light switch, etc.). The 'Identify Device' button appears on the BEMOSS™ dashboard in a respective device widget, as shown below.

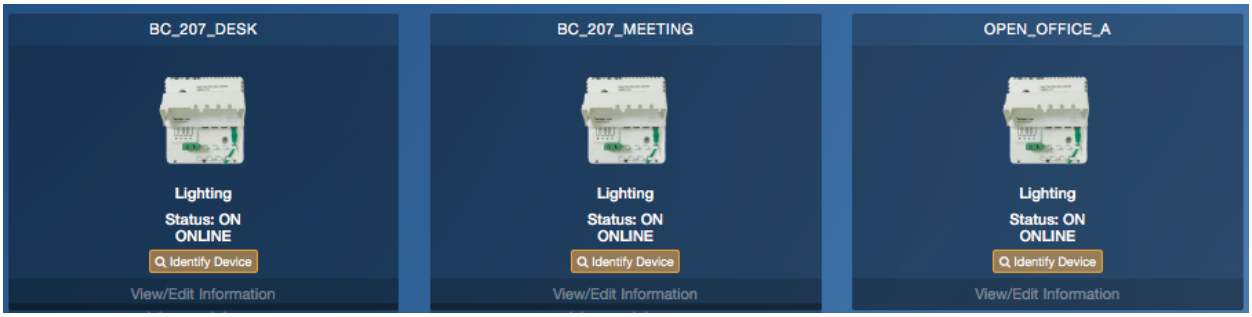


Figure 5-3. 'Identify Device' button

6.0 Additional BEMOSS™ Features

This section discusses additional BEMOSS™ features, including distributed database, thermostat schedule synchronization, OpenADR integration, multi-node operation and encrypted core-to-node communication.

6.1 Distributed Database

BEMOSS™ has two types of databases: one to store time-series data and the other is to store metadata. Cassandra was selected to host BEMOSS™ time-series database and PostgreSQL was selected to store device metadata. Cassandra is an open-source NoSQL distributed database system suited for high frequency and high quantity data. It has been used by Netflix, eBay, Apple, Comcast, etc. for large active data management. It is popular because of its familiar SQL like query language (called Cassandra Query Language (CQL)), linear scalability, high performance and fault tolerance. Cassandra is chosen for BEMOSS™ implementation mainly because it is capable of distributing data across multiple BEMOSS™ nodes. That is, it enables storing data in a distributed fashion and allows replication of data to multiple nodes. Additionally, a data-filtering algorithm was also incorporated into BEMOSS™, where data are collected only when there is a change or every 10 minutes even if there is no change. This approach proves to reduce data storage requirement overtime (by approximately 95% for a thermostat). Data display method is also revised accordingly to be able to plot varied time-step data.

6.2 Thermostat Schedule Synchronization

In BEMOSS™ 3.5, thermostat schedules are synchronized among the thermostat, the thermostat app and BEMOSS™. This is to avoid any confusion to a user/building operator when dealing with thermostat schedules either from the device itself, the thermostat app or BEMOSS™ UI, as schedules displayed on any of these locations will be the same. See Figure 6-1.

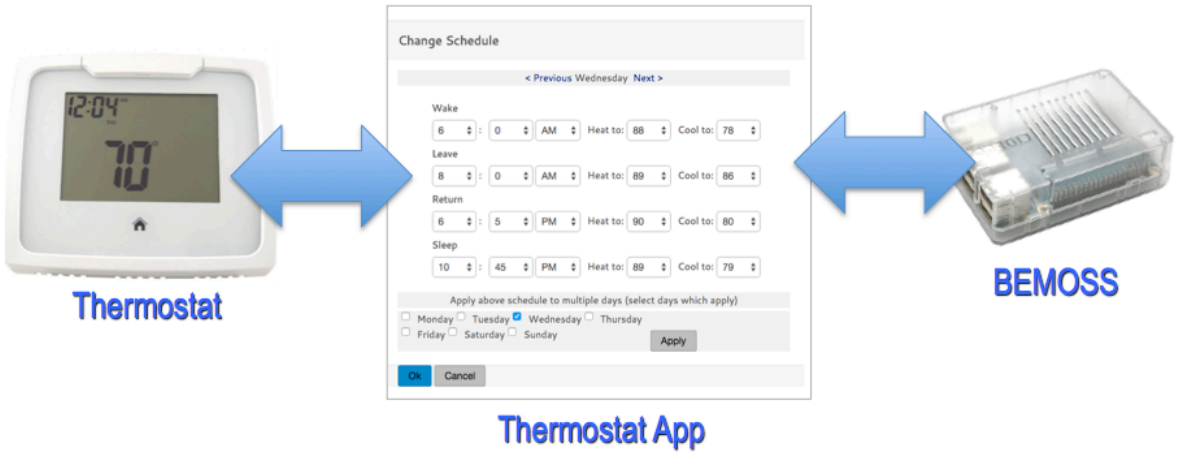


Figure 6-1. Synchronizing thermostat schedules

The synchronization procedure: The synchronization procedure works as follows:

1) When a new thermostat is discovered and added to BEMOSS^{TM}, its schedule will be automatically imported to BEMOSS^{TM}. This implies that the BEMOSS^{TM} UI will show the same thermostat schedule that is available on the thermostat. The schedule in the thermostat app provided by the manufacturer will be the same as the schedule on the device by default.

2) When a user/building operator changes the thermostat schedule in BEMOSS^{TM} UI, this schedule will be pushed to both the thermostat and the app.

Given that the 'ANTI-TAMPERING' in BEMOSS^{TM} thermostat page is set to 'Disabled':

3) When a user/building operator changes the thermostat schedule at the device, this schedule will be copied to BEMOSS^{TM}. Note that: the thermostat app will be automatically updated to reflect the new schedule by default.

4) When a user/building operator changes the thermostat schedule in the thermostat app, this schedule will be copied to BEMOSS^{TM}. The thermostat app will automatically update the device schedule by default.

However, if the 'ANTI-TAMPERING' in BEMOSS^{TM} thermostat page is set to 'Enabled', when a user/building operator changes the thermostat schedule at the device or the thermostat app, this is considered as a **tampering event** in BEMOSS^{TM}. BEMOSS^{TM} will not allow this change and will revert the schedule back to its original schedule.

6.3 OpenADR Integration

This is another feature in BEMOSS^{TM} 3.5. OpenADR was integrated with BEMOSS to enable the ability to receive an OpenADR signal from a local utility/a DR aggregator, in other words, an OpenADR virtual top node (VTN) server. This is accomplished by the use of the existing OpenADR agent in VOLTTRON^{TM} with some modifications. Once BEMOSS^{TM} receives an OpenADR signal, BEMOSS^{TM} can implement device control decisions based on the signal. This is accomplished by collaboration of the OpenADR agent – who receives the signal – and the demand response (DR) agent – who implements the control decision, as shown in Figure 6-2.

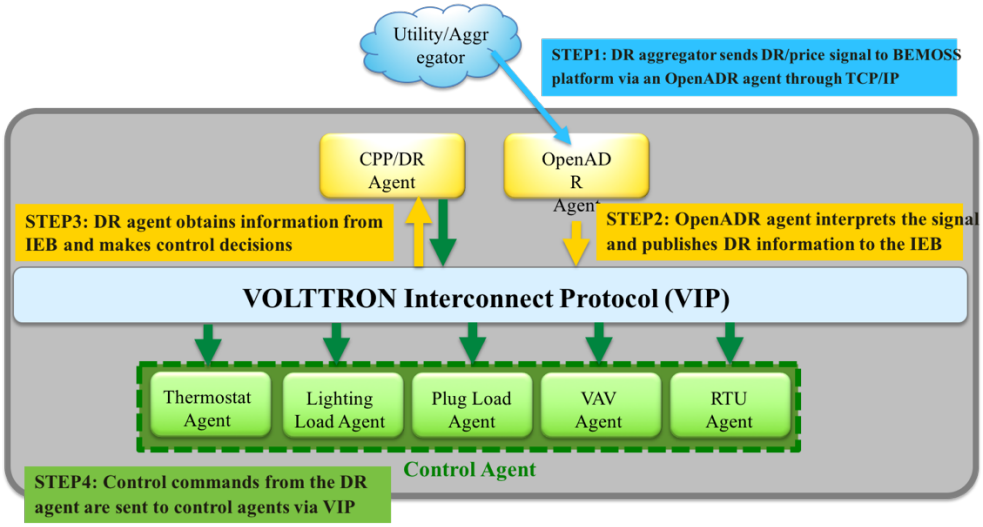


Figure 6-2. Method of enabling OpenADR signal and DR algorithms in BEMOSS^{TM}

This process can be explained as follows:

Step 1: An OpenADR agent in BEMOSS™ receives the signal from the OpenADR VTN. This signal can be, for example, price signals or peak demand reduction signals.

Step 2: Then, the OpenADR agent interprets the signal, and publishes the information to the IEB (Information Exchange Bus) where relevant agents can pick up.

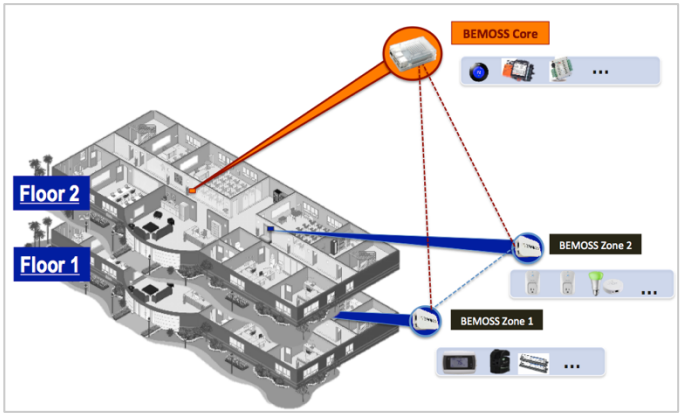
Step 3: The DR agent uses the DR information to make control decisions based on its built-in DR algorithm. Then, it publishes the set of commands for relevant agents to implement through the IEB.

Step 4: The control commands are picked up by respective agents (i.e., thermostat agents, lighting load agents, plug load agents, VAV agents and RTU agents) to take necessary actions.

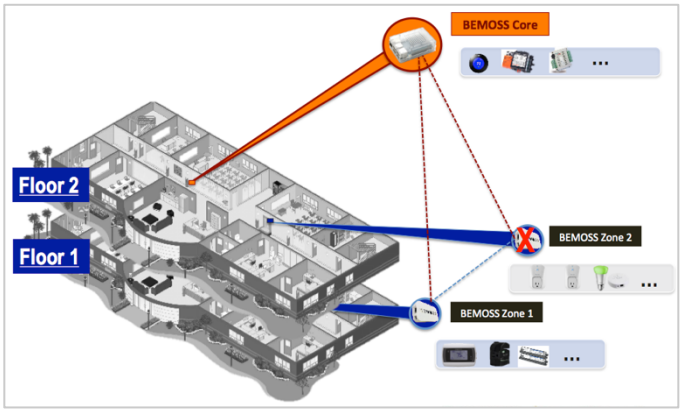
6.4 Multi-node Operation

BEMOSS™ 3.5 was designed to include the multi-node architecture to provide robustness for the deployment. That is, if any BEMOSS™ node fails, devices supervised by that BEMOSS™ node can still be monitored and controlled. This is accomplished by moving associated control agents to the BEMOSS™ core or its neighboring BEMOSS™ node. BEMOSS™ multi-node/multi-layer architecture also enables distributed control, which each BEMOSS™ node can control devices in its zone based on its objective function.

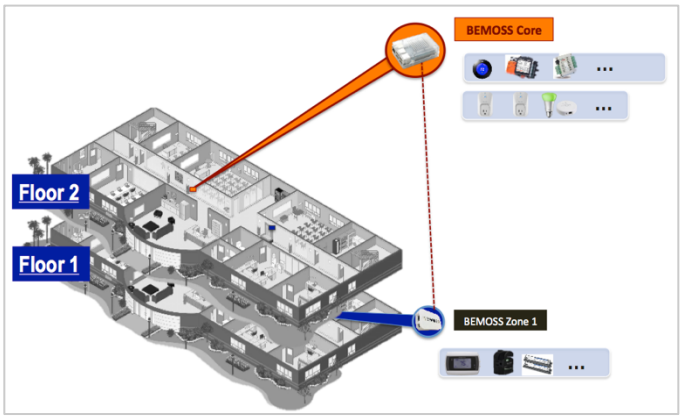
The concept of BEMOSS™ multi-node architecture is illustrated in the series of figures below:



(a) A two-story building with a BEMOSS™ core and two BEMOSS™ nodes. Each node is responsible for monitoring and control of devices on each floor.



(b) Suppose the BEMOSS™ node#2 fails.



(c) The BEMOSS™ core can pick up devices from the failed node and provide continuous building operation.

Figure 6-3. BEMOSS™ multi-node architecture and its operation

6.5 BEMOSS™ Security

Based on the BEMOSS™ software architecture presented in Section 3.2, BEMOSS™ security concerns exist in the following areas:

- BEMOSS™ UI layer
- BEMOSS™ database
- BEMOSS™ information exchange security
- BEMOSS™ operating system and framework layer
- BEMOSS™ connectivity layer
- Connection to the public Internet (i.e., connection to local utility and cloud network)

Security concerns in each area are addressed below, together with associated BEMOSS™ security measures.

6.5.1 BEMOSS™ UI layer: Security Concerns and Measures

The web platform in general is vulnerable to a lot of malicious hacking activities on the Internet. The software that makes up the UI has to be a secure framework to make sure no security leaks occur. Some of the major vulnerabilities in the BEMOSS™ UI layer are related to client-side security. Client-side refers to operations performed by the client, which can be a computer application such as a web browser that runs on a user's local computer, and connects to a server. Possible threats under this category, and associated security measures, are summarized below.

Security concern 1: Cross-site scripting (XSS). XSS allows an attacker to embed malicious JavaScript, VBScript, ActiveX, HTML, or Flash into a vulnerable dynamic page to fool the user, executing the script on his machine in order to gather data. The use of XSS might compromise private information, manipulate or steal cookies, create requests that can be mistaken for those of a valid user, or execute malicious code on the end-user systems. The data is usually formatted as a hyperlink containing malicious contents and which is distributed over any possible means on the Internet. It allows the attacker to inject client side scripts into the browser of the victim. This is usually achieved by storing the malicious scripts in the database where it will be retrieved and displayed to other users, or by getting users to click a link, which will cause the attacker's Java Script code to be executed by the browser. XSS attacks can originate

from any untrusted source of data (e.g., cookies, web services), whenever the data is not validated before being included in a page.

> ***BEMOSS™ security measure.*** *BEMOSS™ security ensures that BEMOSS™ web pages do not have any variable or text that is not validated. To avoid this problem, BEMOSS™ makes sure that each page is run through an escape filter, which converts the potentially harmful HTML characters into un-harmful ones. BEMOSS™ uses automatic escaping to achieve this safety. This security threat has a risk level of less to moderately critical.*

Security concern 2: Cross-site request forgery (CSRF). CSRF is a type of attack that occurs when a malicious website contains a link, a form button or some script that is intended to perform some action on BEMOSS™, using the credentials of a logged in user who visits the malicious site in their browser. Another attack in the same area is to trick a user's browser into logging into a site with someone else's credentials.

> ***BEMOSS™ security measure.*** *BEMOSS™ adds a CSRF cookie that is set to a random value (a session independent nonce value), which other sites do not have access to. A hidden form field is used with the name 'csrfmiddlewaretoken' containing the cookie value and it is present in all outgoing POST forms. For all incoming requests a CSRF cookie must be present, and the csrfmiddlewaretoken field must be present and correct. Otherwise a 403 error is logged. This way the BEMOSS™ application is protected against cross site request forgery attacks.*

Security concern 3: Errors in web applications not handling properly. When errors in the web application are not handled properly, for example, an error page in an ill-designed web page may give out unnecessary information, this may allow a malicious user to obtain implementation-level information from the server. This might seem harmless initially, but it can lead to the attacker gaining access to the server and may become the root to higher security issues like denial of service attacks.

> ***BEMOSS™ Security measure.*** *BEMOSS™ handles all errors with utmost care and makes sure the error message is generic and does not allow for indirect guessing attacks.*

Security concern 4: Denial of service (DOS) attack. A DOS attack in the BEMOSS™ context is an attempt to make the BEMOSS™ platform unavailable to its intended users. One common attack method involves saturating the target machine with external communication requests such that it cannot respond to legitimate request. This is one of the easiest and the most powerful attacks when considering web applications. But since applications like BEMOSS™ are restricted to certain users and limited to the building, the scope for this attack is minimum.

> ***BEMOSS™ security measure.*** *BEMOSS™ ensures that a minimum protection against DOS attacks is provided by use of a stateless cookie. When the client intends to communicate, verifying the legitimacy of the connection can be done by the server sending a hash of a secret key (known only to the server) and the IP address of the client to the client as a cookie. This cookie is different from the browser cookie. The client is expected to reply with the cookie value, which the server can verify. A server overloading attacker will not want to reply to a cookie. The server need not wait for the client to respond either, because the cookie is stateless. If the client responds, the connection is legitimate and the server can continue to send message to authenticate, establish and continue communication.*

Security concern 5: SQL injection attack. This attack enables a malicious user to execute arbitrary SQL

code on a database. This can result in database records being accessed by attackers, resulting in addition of malicious records, or data leakage. This attack usually occurs because of the web application not properly stripping user input of unnecessary special characters.

> ***BEMOSS™ security measures.*** *BEMOSS™ security ensures that the user input is validated. Special validation has been done to ensure that the data input for SQL queries are valid, and ensure that the application will not process SQL commands from the user directly. BEMOSS™ also implements logical security at the database level by using roles and permissions at the database layer.*

Additional BEMOSS™ security measures include:

(1) BEMOSS™ uses encrypted communications. So malicious users will not be able to sniff the authentication credentials or any other information transferred between client and server. Using encrypted communications provides three major security guarantees: server authentication, data confidentiality, and data integrity.

(2) BEMOSS™ uses session security. By purging session data at the end of a session or for expired sessions, BEMOSS™ ensures that the system is secure to any hackers trying to use the session keys for replay attacks. BEMOSS™ stores the session ID that maps to session data stored in the backend. That way, it is safe unlike when storing session keys in cookies. It is also ensured that developers escape out session data if they display it in a template. Although it is nearly impossible to detect if someone's session ID is hacked, BEMOSS™ ensures that brute-force attacks don't happen by storing session IDs as hashes.

6.5.2 BEMOSS™ Database: Security Concerns and Measures

BEMOSS™ database is of two types: (1) Relational Database Management System (RDBMS) database; and (2) Time-series database. BEMOSS™ uses RDBMS to store the metadata, e.g., user profile, preference setting and device information; and uses Cassandra to store time-series data, e.g., room temperature, power consumption of devices and device status.

Security concern: Most database security concerns are related to SQL injection.

The following measures are implemented in the BEMOSS™ architecture to secure BEMOSS™ database:

Security measures common for BEMOSS™ RDBMS and time-series databases:

- **Authentication:** BEMOSS™ ensures database security by using authentication mechanisms. Only registered users can access BEMOSS™, and registration is done semi-manually using a secure process. Thus authentication ensures that only certain people have access to the database.
- **Authorization:** Users in a building access BEMOSS™, based on their assigned roles. Users are separated into groups and authorization is restricted based on the user role in the building. A tenant is given access to view all the pages, and corresponding read access to the database. A building owner/manager is given access to read and modify values in the database. Both the tenant and the building owner/manager may be restricted from modifying archived data. However, the building owner is the ultimate authority and may be allowed to modify the database by activating special permissions if needed.

Additional security measures specific to BEMOSS™ RDBMS databases:

- **Database access security:** BEMOSS™ uses stored procedure calls wherever possible. This limits data access to tables via defined roles in the database. A stored procedure is like a function

call and acts as an interface to the underlying data structure, so that the data is shielded. Alternatively, Object Relational Mapping may also be used.

Additional security measures specific to BEMOSS™ time-series database:

Communications with Cassandra happens from agents and from the UI. With every connection request from either agents or the UI, authentication is performed at the database end. Since all data to be sent to the database are available in the BEMOSS™ message bus, the message bus will need to be secured. This is discussed in the Information exchange security in Section 6.5.4.

6.5.3 BEMOSS™ Operating System and Framework Layer: Security Concerns and Measures

The operating system and framework is an essential layer of the BEMOSS™ as it links user interface layer, application and data management layer, and connectivity layer together. In this layer, VOLTTRON™ is used as an agent platform to provide BEMOSS™ functionalities.

Security concerns: Possible security risks in this layer include:

- An attacker may execute a malicious agent in order to sniff data from agents' communication messages over VOLTTRON™ VIP. The data can include important information such as user profile, user preference, or occupancy leading to the user privacy concerns.
- An attacker may execute an unverified agent to control or tamper with hardware devices without authorization by a user or a corresponding agent. This can result in shorter life of hardware devices, hardware devices malfunction or permanent damaged hardware devices.
- An attacker can bring in a tampered device that is supported by BEMOSS, and once the device is discovered and added into BEMOSS™, the attacker can use that device to eavesdrop to the communication protocol between BEMOSS™ and devices, which may cause vulnerability towards communication security and data privacy.
- An attacker can bring in a malicious BEM that discovers and controls the devices in the building. An attacker can also bring in a malicious BEMOSS™ node and eavesdrop to the communication between BEMOSS core and nodes.

Security measures implemented by BEMOSS:

1) VOLTTRON™ built-in Security

BEMOSS relies partly on VOLTTRON™ in-built features to address agent's confidentiality, integrity, and availability requirements. VOLTTRON™ Interconnect Protocol (VIP) has been implemented to provide a secure communication protocol between agents, controllers, services, and the supervisory platform. For secure communications among agents, VIP extends the ZeroMQ Authentication Protocol ZAP by implementing authentication in the auth module (*volttron/platform/auth.py*). As per this, any agent can connect to the message bus only if it is authenticated to the VOLTTRON™ platform by proving its identity. VIP authorization defines a platform owner with the ability to limit the capabilities of authenticated agents. Hence, an agent should be authorized along with necessary capabilities to make a transaction with peer agents.

When a BEMOSS™ platform is launched, it adds a new auth entry into volttron auth file with BEMOSS™ specific user_id and capabilities. Hence, any agent launched by BEMOSS™ has the required capabilities. Any communication BEMOSS™ agent looks for capability of the communicating agent and

accepts message from its peer agents only if it has the necessary capability. With this agent authorization feature non-BEMOSS launched agents can never intrude with BEMOSS™ agents.

2) BEMOSS™ Discovery Approval Process

BEMOSS™ discovery agent and UI have been designed to incorporate approval process for new device discovery to enhance security of the system. In this approval process, new devices are added to BEMOSS™ after approval from the user. It is to be noted that this process is applied only once during the lifetime of the device in the system, unless the device is lost from the database due to a software glitch or something similar.

The way this has been implemented is: instead of immediately launching an agent for a newly discovered device and adding it to BEMOSS™, in the modified system, the discovery agent keeps the status of a device pending after discovering it. It only adds an entry to the BEMOSS™ database with the discovery info of the newly discovered devices with its status as 'pending', without launching any agent for the device. The UI picks up entries in the database with 'pending' status and shows it in UI as newly discovered devices with approve/decline button beside it. This allows the user to check the device and then approve it based on a manual authentication protocol. Whether the user approves the device or marks it as a non-BEMOSS device (NBD), the UI makes the corresponding change to the status of the device in the database and also sends a message to the discovery agent notifying it about the status change. When discovery agent receives such notification from UI with the 'approval' status, it adds it to BEMOSS™ device data tables and launches agent for the device, thereby adding it to BEMOSS™. If the device is marked as NBD, discovery agent ignores the device and makes no further communication with the device.

This approval process provides the following security benefits:

1. It enhances BEMOSS™ security by adding an additional layer of authentication for devices before adding to BEMOSS™. This reduces the possibility of a security-compromised device to be added to BEMOSS™, as it has to go through the user's approval first.
2. It prevents the BEMOSS™ from controlling other people's devices, which are not part of a user's building management. For example, BEMOSS™ may discover devices on a floor that is not part of the user's system. This approval process ensures that BEMOSS™ does not control third party device on the same premises.

3) Encrypted Core-to-node Communication

This feature employs a public-private key combination for core-to-node communication. A schematic of the core-to-node communication is shown in Figure. 6-4.

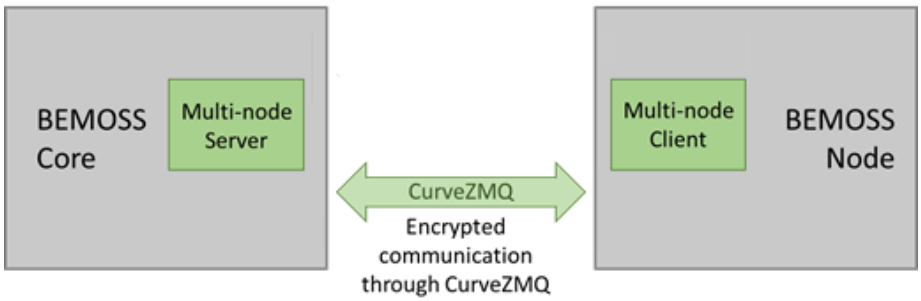


Figure 6-4. Secured BEMOSS™ Core-to-node communication.

The key points of this security feature are:

- The BEMOSS™ core and nodes have each other configuration (IP address and public addresses) pre-configured during setup.
- Once every node and core are configured, they have each other's public key in their configuration having private key safe within itself.
- Once a secured communication has been established, all further message exchanges are encrypted through CurveZMQ.
- As messages between the core and nodes are encrypted and can only be decrypted by the correct key pair, it enhances the security of BEMOSS™ by ensuring that a tampered and unauthorized BEMOSS™ node cannot take part in communication within a BEMOSS™ network.

6.5.4 BEMOSS™ Information Exchange Security

BEMOSS™ information exchange refers to information exchanges between any two BEMOSS™ agents (i.e., agent-agent interaction) and between an agent and any components in the BEMOSS™ software architecture, including database (e.g., agent-database, agent-UI, agent-devices). This can be either within the same machine, or between machines. A machine in this case refers to a PC, a tablet accessing BEMOSS™ UI, or an embedded system hosting BEMOSS™. BEMOSS™ uses ZeroMQ for all messaging and communications. ZeroMQ provides many of the building blocks to implement encrypted and authenticated communications over a shared socket. Router pattern socket type is one such pattern specifically designed to implement peer to peer communication. Hence, BEMOSS™ bases all its local within PC agent to agent communication on ZMQ's router pattern to provide a single connection point along secure message passing instead of each agent having a capability to make ipc calls to one another.

For information exchange within the same machine (e.g., within a BEMOSS™ core or within a BEMOSS™ node), the router instance binds to a ROUTER socket and peers (agents) connect using a DEALER. Agents communicate along the ROUTER socket adhering to VIP (volttron interconnect protocol). VIP is routing protocol which takes care of Message-level authentication. By defining a standard message-based protocol between the router and peer agents, non-standard messages are disregarded by the agent thereby allowing only valid agent to initiate communication with other agent. Additionally, VIP defines agent authorization which can limit the extent of access of valid agent with other.

For information exchange between machines (e.g., between a BEMOSS™ core and a BEMOSS™ node or between a BEMOSS™ core and a BEMOSS™ database), BEMOSS™ uses a security wrapper framework "CurveZMQ" around ZeroMQ for secure messaging. CurveZMQ aims to protect the message queue from eavesdropping, fraudulent data, altered data, replay attacks, amplification attacks, man-in-the-middle attacks, key theft attacks, identity attacks, and certain denial of service attacks.

6.5.5 BEMOSS™ Connectivity Layer: Security Concerns and Measures

As agents communicate with different hardware devices using the connectivity layer and different communication mediums, security aspects of this layer are mainly dependent on the security of the message exchanges between the device and the agent.

Security concerns: The most probable ways of attack to this layer are:

- An attacker may eavesdrop to the messages between the agent and the device. If they are not encrypted, the attacker may gain information about device status and control commands, which

may give away private information such as: occupancy, user preferences etc., which may be used for malicious purposes. It also creates privacy concerns.

- An attacker can alter information in the data messages, thereby causing the agents to receive corrupted device data and make undesirable decisions.
- An attacker can send spurious control messages to devices, causing nuances. This can be done from external entities or from malicious code inside agent hosts.

Security measures: Encryption and message validation should be implemented to ensure data privacy and prevent data corruption. Authentication could prevent devices from reacting to fake control messages, but not all hardware devices support encryption/authentication procedure. Following is a description of how BEMOSS implements specific security features in different protocols.

- **BACnet:** Addendum G to the BACnet standard specifies requirements for security of BACnet IP or MS/TP network. If this standard is followed, authentication and encryption are ensured to prevent eavesdropping or malicious control commands. The only issue is a lot of vendors do not include these security features in their products due to complexity and computational requirement at the device end. For BACnet MS/TP devices, securing the MS/TP messages with additional encryption/decryption may impact performance. However, it is also difficult to attack MS/TP network, because the attacker has to gain access to actual physical RS-485 wires to launch any type of attack. Therefore, an alternate to securing MS/TP network through encryption is to secure the MS/TP hardware devices and wires physically through careful conduit design and making hardware devices accessible to authorized personnel only. For BACnet IP devices, key-based encryptions should be implemented. BACnet IP devices without any security feature should be connected through a separate LAN to isolate it from the main building LAN, which is more accessible to an attacker.
- **Modbus:** Modbus protocol by itself does not provide any security measures. Some vendors develop security features overlaid on top of the actual Modbus protocol to ensure security. But most devices do not offer any security by their own. In such cases, similar to BACnet, the Modbus RS-485 network and devices could be secured using physical security measures. For Modbus TCP, additional security could be overlaid on top of the actual message exchange protocol.
- **Wi-Fi:** A good number of Wi-Fi devices currently in the market use HTTP get and post messages for communication. This makes them very vulnerable, as an attacker can easily eavesdrop and generate spurious control messages. One solution to this is to upgrade device firmware by vendors to use HTTPS instead of HTTP. Another solution is to use separate wireless network for these devices. This minimizes possibility of threats, but does not eliminate it completely.

For devices without proper security in message exchanges, BEMOSS™ implemented additional checking algorithms on the agent side to ensure that the device is not being controlled by unauthorized messages. These algorithms can be described as follows:

1. Periodic checking of status of a device and comparing it with expected status from the UI or algorithm can reveal if a device status has been undesirably changed. The agent can then send corrective control messages to the device and also issue warning messages to the upper layers of probable threats to the device.
2. Sniffing of outgoing control messages from the agent hosting system and comparing it with actual control commands in the message bus for control agents can reveal if any malicious code/agent is sending unauthorized control messages to the device. This can identify attacks to the agent host

and issue warnings. In extreme cases, the communication interface of the agent host can be blocked to prevent any more sending of malicious control messages to devices.

To summarize, the security measures taken in connectivity layer and hardware devices are:

1. Using key-based encryption methods whenever possible, to ensure encryption, authentication and message integrity.
2. Securing physical communication medium, by securing network wires, using tamper resistant technologies for hardwire devices or in case of wireless network, by creating separate networks less accessible from outside.
3. Using checking algorithms in agents to ensure devices are being controlled by authorized control commands only.

6.5.6 Connection to the Public Internet (i.e., connection to local utility and cloud network)

BEMOSS™ connection to a local utility exists to allow BEMOSS™ to receive electricity price signals and demand response (DR) signals via a standard protocol, like OpenADR. OpenADR is a communication data model designed to interact with DR signals by automating DR actions performed by BEMOSS. A generic architecture of OpenADR is shown in Figure 6-5.

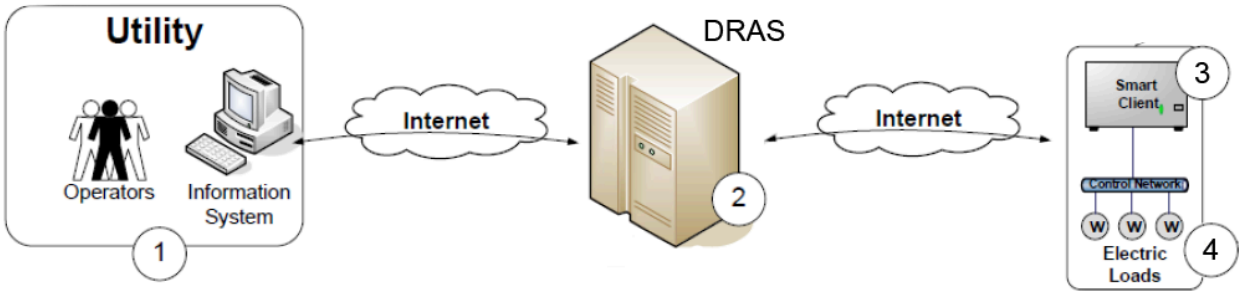


Figure 6-5. Generic OpenADR interface architecture

Information flow in the OpenADR architecture can be discussed in the following steps:

1. The utility or ISO defines DR events and price signals sent to Demand Response Automation Server (DRAS).
2. DR event and price signals are published on a DRAS.
3. A DRAS client, in this case BEMOSS™, requests event information from the DRAS every specific time interval, e.g., one minute.
4. BEMOSS determines load management actions based on events and price signals, and carries out load shedding/shifting strategies.

Security concerns: Security concerns related to connection to a local utility, which is based on Internet communications, includes both the utility side (server side) and the BEMOSS™ side (client side). Main security concerns are:

- An attacker intercepts information sent between the server side and the BEMOSS™ client side to gain knowledge of DR events, pricing information and customer information. This can lead to loss of confidentiality, e.g., the exposure of customer data, unauthorized modification of information, manipulation of information and malicious attacks.
- An attacker may manipulate BEMOSS™ to communicate with a fake DRAS. This may make BEMOSS to receive false price/DR signals, which may result in turning on (or off) selected loads in the building. This can cause excessive loads to the grid, and grid instability, as well as financial impacts on customers. An attacker may also issue false time synchronization, causing events to occur sooner or later than the original schedule. In addition, it may make the utility server not be able to record the actual response of BEMOSS™ to the DR events received.
- An attacker may disable BEMOSS™ from receiving incoming DR signals. This can be done using denial of service attacks when an attacker floods the communication channels between the server and the client (BEMOSS™) with no-DR related Internet traffic.

BEMOSS security measures: Security features of OpenADR 2.0 conform to NIST Cyber Security requirements, and follow the guidelines provided by the "Security Profile for OpenADR" prepared by the UCAIug OpenADR Task Force and SG Security Joint Task Force. OpenADR 2.0 provides security services like authentication, confidentiality and integrity by implementing Public Key Infrastructure (PKI) certificates in both the utility server and the OpenADR client. Two primary public key cryptography algorithms supported by the utility server are:

- Elliptic Curve Cryptography (ECC) – 256 bits or longer keys
- Rivest, Shamir, and Adelman (RSA) – 2048 bits or longer keys

To establish a secure communication channel between the utility server and BEMOSS, BEMOSS could use a certificate from the approved list of certificates. BEMOSS™ could obtain a CA certificate(s) from a commercial vendor, i.e., an approved certificate authority (CA), and use this certificate to issue OpenADR certificates to connect to the utility server.

7.0 Advanced Algorithm Development

This section discusses advanced algorithms available in BEMOSS™ 3.5, including thermostat anti-tampering, illuminance-based control and fault detection.

7.1 Anti-Tampering

In a typical commercial building setting, a building operator might not want occupants to arbitrary change thermostat cooling/heating set points. For that reason, the anti-tampering feature was developed to detect such tampering and allow thermostats to revert the tampered set point back to its authorized value +/- some tolerance.

Enable/disable the anti-tampering feature: A building operator can enable or disable the anti-tampering feature for each individual thermostat via the BEMOSS™ UI. See Figure 7-1.

- The anti-tampering feature is enabled when the 'ANTI-TAMPERING' is set to 'Enabled'.
- The anti-tampering feature is disabled when the 'ANTI-TAMPERING' is set to 'Disabled'.

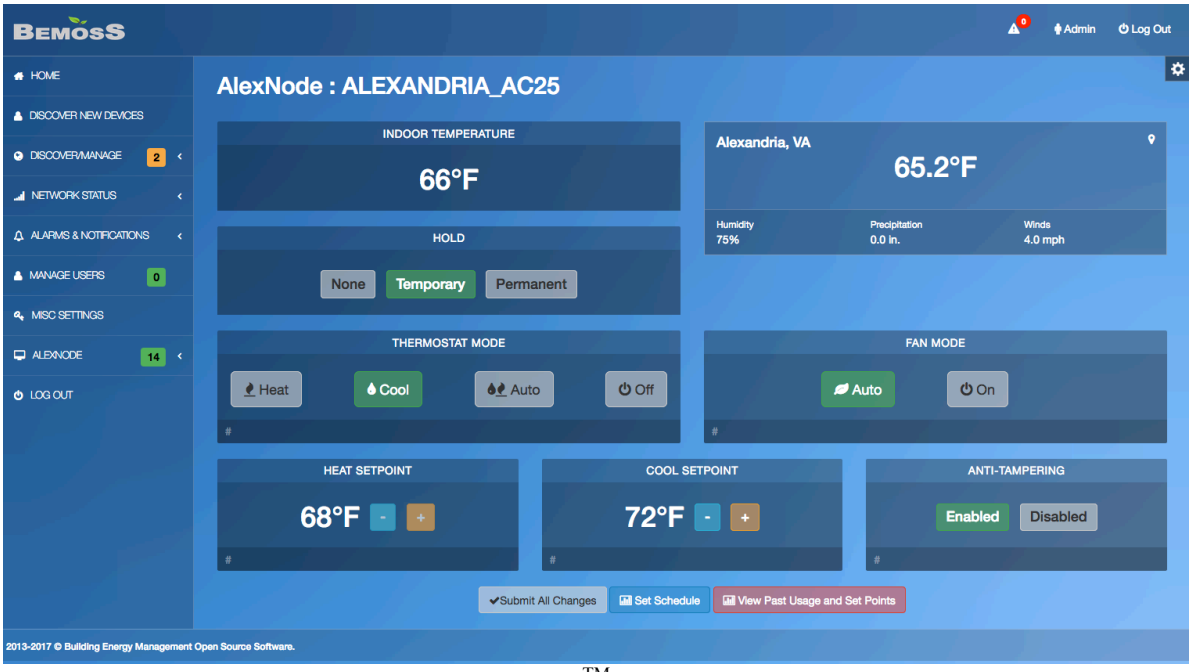


Figure 7-1. BEMOSS™ UI – thermostat page

If the building operator does not allow override (ANTI-TAMPERING is set to 'Enabled'), BEMOSS™ will periodically obtain a set point reading from the thermostat and compare it with the authenticate value, which is saved in the BEMOSS™ database. Once BEMOSS™ discovers that there exists discrepancy between the two values, BEMOSS™ will know that a thermostat has been tampered and take the action.

Set point correction mechanism: The set point correction mechanism gives occupants a certain freedom. That is, it allows the change of ± 2°F tolerance (□t) from the original set point. That is, occupants are allowed to change the temperature set point by ± 2 °F. In other words, a set point change from 75°F to any value between 73°F and 77°F is allowed. However, if occupants change the set point to other values

outside this allowable range, BEMOSS™ will automatically revert the set point it to the nearest allowable value. For example, if a set point is changed from 75°F to 69°F, BEMOSS™ will revert the temperature back to 73°F. See Figure 7-2.

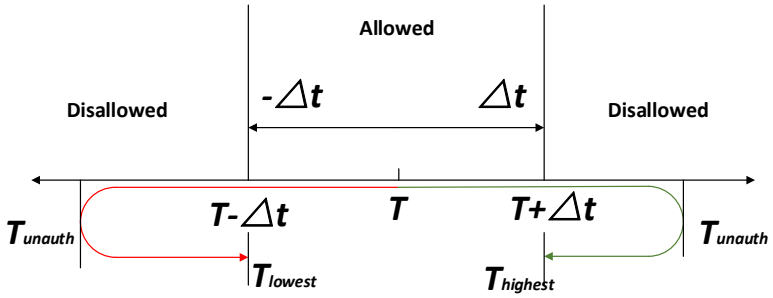


Figure 7-2. Set point correction mechanism

7.2 Illuminance-based Lighting Control

In many commercial buildings, indoor illuminance is influenced by both outdoor ambient light as well as indoor artificial light. In order to save energy as well as providing a constant illuminance to a workspace, an illuminance-based lighting control application was developed on BEMOSS™. According to inputs from ambient light sensor(s), BEMOSS™ intelligently dims the artificial light level to maintain a preset illuminance level.

The user interface for this application is shown in Figure 7-3:

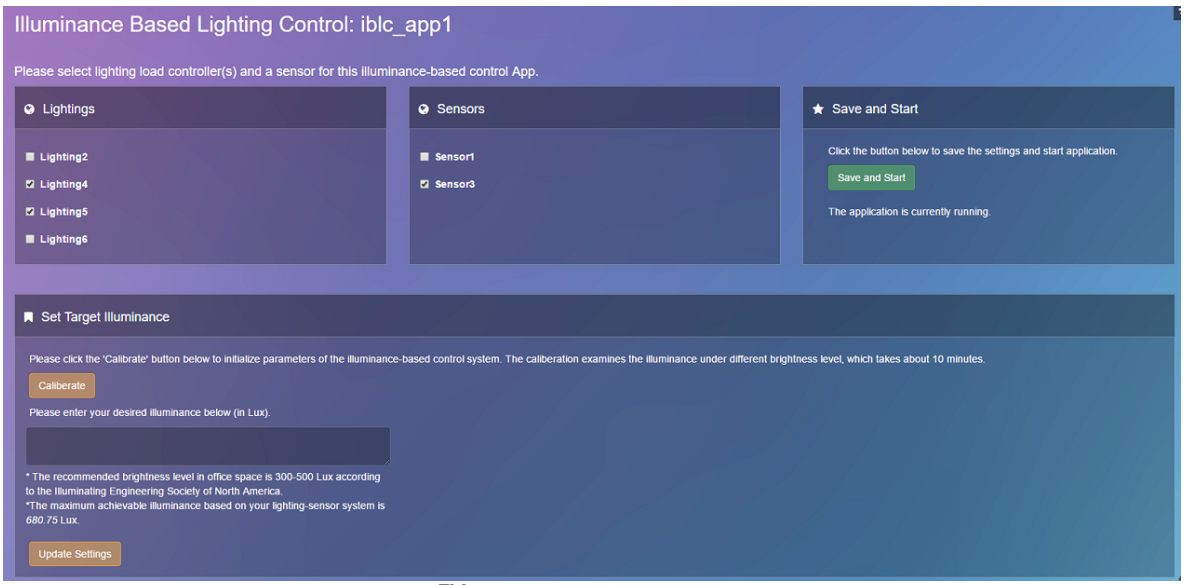


Figure 7-3. BEMOSS™ UI – Illuminance-based lighting control page

To implement this control, a user can configure the application using the UI by performing the following steps:

1. Select light(s) to be controlled and sensor(s) to get illuminance readings from. If multiple lights are selected, they will be controlled together; if multiple sensors are selected, the average reading from all sensors is used for the control.
2. For the intelligent control, a calibration process is needed at the start of the application to determine the relationship between the brightness level and sensor readings (Lux).
3. A user needs to enter the target illuminance level (in Lux)

After these three steps are performed, BEMOSS™ will start the close loop lighting control application.

Below are two algorithms for Steps 2 and 3 in detail:

1. Calibration

When the 'Calibrate' button is clicked, BEMOSS™ will conduct a series of test to determine the brightness-illuminance relationship specific to a given environment. Lights will be set at different brightness levels (i.e., 10%, 40%, 70% and 100%) and illuminance readings from the sensor(s) will be recorded.
Figure 7-4 shows an experiment conducted in the lab environment at Virginia Tech, which demonstrates the linear relationship between the brightness level set and illuminance readings. These four data points (as shown in the black squares in Figure 7-4) are used for determining the slope of the linear regression line representing the relationship between light brightness and sensor reading.

Figure 7-4. Relationship between brightness percentage and illuminance (control sensitivity)

One additional usage for the calibration is to get the maximum illuminance given 100% artificial light. This number is provided on the user interface, as shown in Figure 7-3, to be the upper-limit of the target illuminance.

2. Illuminance-based lighting control

The algorithm for the illuminance-based lighting control is illustrated in Figure 7-5.

```
for each monitoring period:
    Get illuminance reading from sensor(s)
```

```
if |target illuminance – current illuminance| > 30:
    Calculate needed brightness:
        needed brightness = (target illuminance – current illuminance) / sensitivity
    Get current brightness from lighting agent
    Calculate new brightness setting:
        new brightness = current brightness + needed brightness
    if new brightness > 100%:
        Set the brightness of light(s) to be 100%
    else if new brightness < 0%:
        Set the brightness of light(s) to be 0%
    else:
        Set the brightness of light(s) to be new brightness
else:
    do nothing since illuminance is with tolerable band
```

Figure 7-5. Pseudocode for illuminance-based lighting control algorithm

Notes:

1. A tolerance of ±30 Lux is used: as long as the current illuminance level is within this band of target illuminance, no action is needed.
2. The relationship between brightness and illuminance level is used for the control. If the light is changed or the sensor is relocated, a new calibration is necessary.

7.3 Fault Detection

Sometimes the HVAC system in a building does not operate as desired. This could be because of some fault in the thermostat, the heat/cool air/water delivery system or the heating/cooling equipment. A basic fault detection app has been added to BEMOSS that can detect when the heating/cooling operation deviates from its regular or normal behavior.

Figure 7-6. Fault detection UI

The user has options for watching out for three kinds of abnormal temperature behaviors.

7.3.1 Fault Detection 1: Abnormal Temperature behavior

Detecting abnormal temperature behaviors can be discussed in two different operating modes, i.e., cooling or heating.

a) Thermostat in Cooling Mode

When the thermostat is in the cooling mode of operation, the temperature is expected to be within the deadband of the cool setpoint in short amount of time. If the temperature is higher than the cool setpoint for a long period (in this case, its time integration is larger than 9 degree hours), that means there is some problem with the cooling mechanism, and an abnormal cooling alert is created.

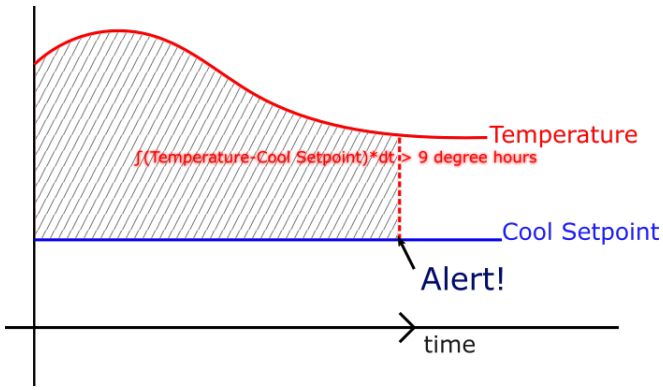


Figure 7-7. Illustration of time integration of the difference between indoor temperature and cool setpoint

b) Thermostat in Heating Mode

This is similar to cooling mode of operation. If the temperature is below the heat setpoint for a long time (the time integration of the difference is larger than 9 degree hours), then that means the heating system has failed to heat the room and a heating fault alert is generated.

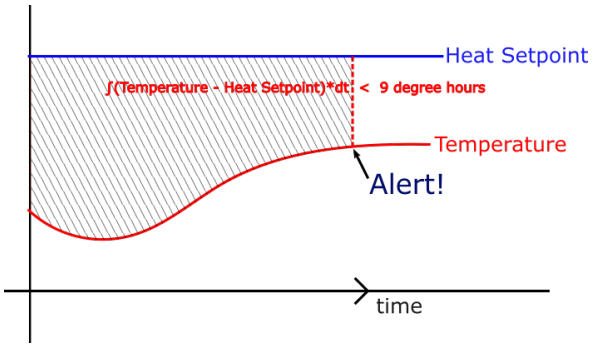


Figure 7-8. Illustration of time integration of the difference between indoor temperature and heat setpoint

7.3.2 Fault Detection 2: Temperature Goes below a Preset Level

Regardless of HVAC operation mode (cooling or heating), if the indoor temperature goes below a preset value (options are 55°F, 60°F and 65°F), the user is alerted.

7.3.3 Fault Detection 3: Temperature Goes above a Preset Level

Regardless of HVAC operation mode (cooling or heating), if the indoor temperature goes above a preset value (options are 80°F, 85°F and 90°F), the user is alerted.

8.0 Lab-scale Testing

This section describes BEMOSS™ laboratory set up at the Virginia Tech – Advanced Research Institute (ARI), including the list of hardware available, how the communication network is set up and the list of test procedures, test results and problem found.

8.1 Hardware Devices in BEMOSS™ Laboratory

As of October 2015, BEMOSS™ lab has the following hardware devices.

Table 8-1. List of devices used in the lab

No.	Device Model	Vendor	Protocol
BEMOSS™ Host			
1	BEMOSS™	Odroid	N/A
Thermostat			
2	CT30 w/ WiFi USNAP module	RadioThermostat	WiFi
3	CT50 w/ WiFi USNAP module	RadioThermostat	WiFi
Lighting controller			
4	WeMo light switch	Belkin	WiFi
5	Philips Hue	Philips	WiFi/Ethernet
6	LMRC-212	Wattstopper	BACnet MS/TP
Plug load controller			
7	WeMo smart plug	Belkin	WiFi
8	WeMo insight switch	Belkin	WiFi
9	LMPL-201	Wattstopper	BACnet MS/TP
RTU & VAV			
10	M1000/2000 RTU - Rooftop Controller	Prolon	Modbus RTU
11	PL-VC1000/2000 – VAV Controller	Prolon	Modbus RTU
Sensor			
12	LMLS-400	Wattstopper	Modbus RTU
Communication gateway			
13	The BAS Router	Contemporary Controls	BACnet MS/TP - BACnet/IP
14	LMBC-300 Digital Network Bridge	Wattstopper	Wattstopper DLM – BACnet MS/TP

BEMOSS™ host: BEMOSS™ host is a single board computer on which BEMOSS™ is installed.

HVAC load controllers: The lab set up has CT30 (#2), CT50 (#3), as well as VAV and RTU controllers (#10-11).

Lighting load controllers: Lighting load controllers are WeMo light switch (#4), Philips Hue (#5) and Wattstopper lighting controller LMRC-212 (#6).

Plug load controllers: These include WeMo smart plug (#7), WeMo insight switch (#8), and Wattstopper plug load controller (#9).

Senosr: One ambient light sensor (#12) is included to evaluate the illuminance-based control algorithm.

Communication gateways: These include: the BACnet router (#13), to convert BACnet MS/TP to BACnet/IP and the Wattstopper digital network bridge (#14), to convert Wattstopper DLM protocol to BACnet MS/TP.

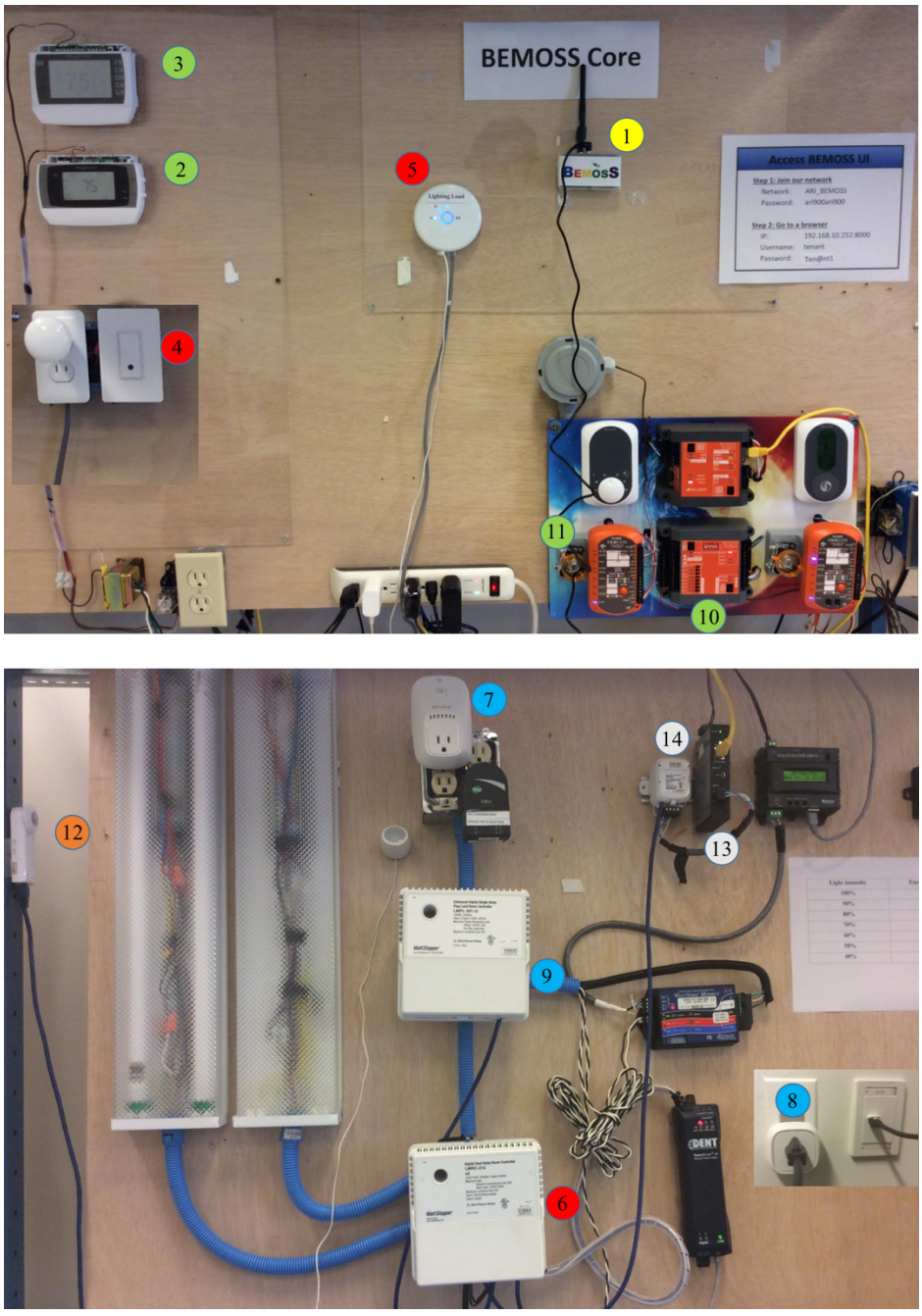


Figure 8-1. BEMOSS™ lab set up

Single board computers: Various single board computers (#24-34) have been acquired for testing the performance of BEMOSS™ software, as shown below.

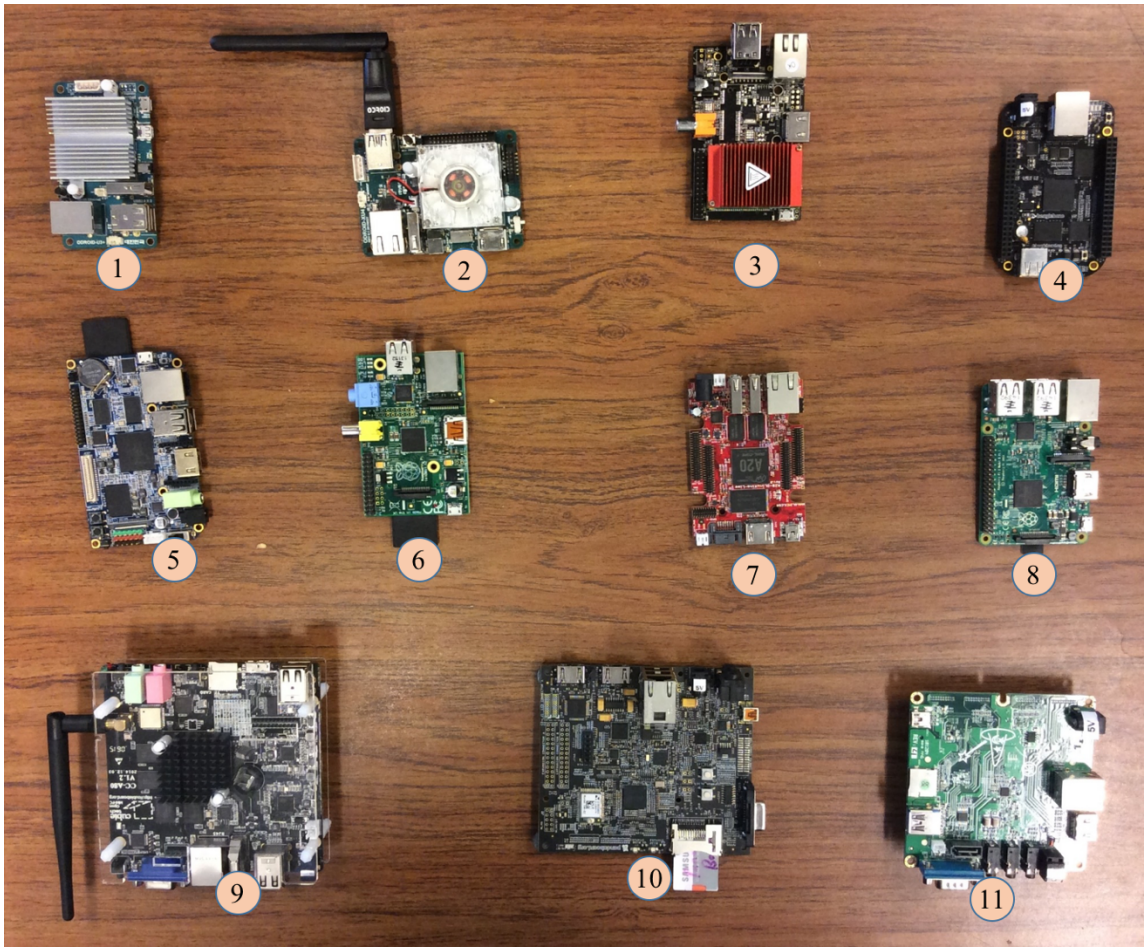


Figure 8-2. Single board computers used in the BEMOSS lab

	Single Board Computer	Vendor
1	ODROID-U3	Hardkernel Co., Ltd.
2	ODROID-XU4	Hardkernel Co., Ltd.
3	HummingBoard	SolidRun
4	BeagleBone Black	BeagleBone Team
5	NanoPC-T1	FriendlyARM
6	Raspberry Pi 1 model B revision 2	Raspberry Pi Team
7	Olinuxino A20	Olimex Ltd
8	Raspberry Pi 2	Raspberry Pi Team
9	CubieBoard A80	CubieTech
10	Panda Board	Panda Board Team
11	WandBoard Quad	WandBoard Team

8.2 Communication Set up in the BEMOSS™ Laboratory

In order to allow the BEMOSS™ operating system to interface with Wi-Fi, Ethernet and Serial (RS-485) devices, certain physical communication features are implemented. These include:

(1) Allowing BEMOSS™ to communicate with **Wi-Fi** devices, a Wireless Local Area Network (WLAN) can be setup using existing wireless infrastructure in a building or by using wireless routers to setup a new wireless network. Both Wi-Fi hardware devices and BEMOSS must be configured to join the same network.

(2) Allowing BEMOSS™ to communicate with **Ethernet** devices, all Ethernet devices are connected via Ethernet cables and switches. Then, Ethernet devices must be configured such that they are in the same subnet with BEMOSS. Routers can be used to route messages between devices in different subnets.

(3) Allowing BEMOSS™ to communicate with **Serial (RS-485)** devices, these devices are connected using physical wires. These devices can be interfaced with BEMOSS - which supports Ethernet - via a BACnet router (that converts BACnet MS/TP to BACnet/IP) or a Modbus gateway (that converts Modbus RTU to Modbus TCP).

- BACnet MS/TP devices communicate with BEMOSS through a BACnet router, providing route communications between BACnet/IP and BACnet/MSTP networks. This allows BACnet/IP devices to communicate with other BACnet devices on a BACnet/MSTP network. A BACnet router connects to the local area network via the Ethernet interface. This device requires no mapping because all data on the one side (for example, the BACnet/IP side) is transparently passed to the other side (BACnet MS/TP) and vice versa.

- Modbus RTU devices communicate with BEMOSS through a Modbus gateway, allowing serial Modbus RTU devices to communicate and interoperate with Modbus TCP-based gateways. The Modbus TCP gateway connects to the local area network via the Ethernet interface. This gateway converts the Modbus TCP/IP protocol to/from Modbus RTU protocols.

8.3 BEMOSS™ Functionality Tests and Issues Found

The BEMOSS™ software platform has been tested in a laboratory environment during its development to evaluate its functionalities. Specifically, the following tests were conducted:

Table 8-2. BEMOSS™ functionality test

Test/Description	Status
Test#1: Auto discovery and device approval This test was conducted to make sure that all BEMOSS supported devices can be discovered by BEMOSS, and the approval process works correctly.	OK – all supported devices were discovered
Test#2: BEMOSS™ UI test This test was conducted to ensure all UI pages are displayed correctly. Any errors found have been fixed. These include the dashboard, thermostat pages, lighting load controller pages, plug load controller pages, schedule pages, historical data pages, data export features and many others.	OK – all UI pages functioned as expected
Test#3: User registration/user settings/log-in and log-out procedure	OK – User registration

This test was conducted to ensure that BEMOSS allows user registration and their roles (i.e., admin, tenant) permit users to have access to their respective pages and features. User settings and logging in/out features were also tested.	and role-based access control worked as expected.
Test#4: Alarms and notifications Several alerts were created to ensure that BEMOSS™ could detect registered alert conditions and sent notifications via BEMOSS™ UI and email.	OK – email notifications and SMS were received
Test#5: Scheduler test Schedules were added to change settings of thermostats, status of lighting and plug load controllers at specific times. This is to test whether the scheduler app functions correctly.	OK – devices follow the pre-programmed schedule
Test#6: Device monitoring and control via a local network This test was conducted to ensure that all BEMOSS™ supported devices can be controlled within a building network.	OK – devices can be monitored and controlled via a local network
Test#7: Thermostat tampering Manual changes of thermostat set points were conducted while BEMOSS™ is set to disabled override. This is to ensure that the anti-tampering feature prevents such unauthorized changes and send notifies a building owner of unauthorized actions.	OK – thermostat tampering detection worked as designed
Test#8: Multi-node test A laboratory set up with one single-board computer (BEMOSS™ core) and another one (BEMOSS™ node) is used for this test. Each computer is responsible for monitoring and control of a group of devices in one building. Power was unplugged from the BEMOSS™ node to simulate a node failure. Then it was observed that all monitoring and control functions of the failed node were picked up by the core and all functions continued without any interruption.	OK – BEMOSS core and node worked as designed

Several bugs were found during the tests, and all bugs were fixed. Examples of bugs found include:

- Anti-tampering did not work.
- Schedule did not implement.
- Some buttons on the UI did not communicate.
- Tenant accounts could access data download and edit device nicknames.
- Tenant accounts saw a misplaced copyright bar.
- Some UI pages showed wrong copyright years, e.g., 2014 or 2015.
- Alerts came but showed wrong time in the message.
- Alerts came twice for the same event when running in a multi-node environment.

One major issue the team found as functionality and robustness tests on BEMOSS™ 3.5 were performed was: after running the system for several days at a time, the system fails due to 'too many open files' issue originating from VOLTTRON 3.5 agents**. The team fixed this problem, and all other bugs before the BEMOSS™ 3.5 release in June 2017.

** The 'too many open files' issue was suspected to come from a temporary VOLTTRON™-agent that was created every time a user opened a webpage in BEMOSS™. This temporary VOLTTRON-agent was used to subscribe to messages on the VIP message bus, so that when a device updated its state (e.g., when the temperature reading of a thermostat changes), it could listen to such a message and sent an update to the webpage in real time. When a webpage was closed, the VOLTTRON-agent was removed, but internal sockets and other files opened by the agent remained lingered in the system, resulting in 'too many open files' eventually.

The solution implemented was to replace this temporary VOLTTRON™-agent with a temporary ZMQ socket to allow relaying any messages in the message bus to the webpage. Since this temporary ZMQ socket is automatically removed when the webpage closes, there are no connections or files lingered open. Another change that helped reduce the number of open files was to use a single agent to handle the connection to time series Cassandra database instead of each device agents creating their own connection with the Cassandra database. This also reduces the number of open files in the system.

8.4 Performance On Different Hardware Hosts

BEMOSS™ performance tests were conducted on different hardware hosts, including Virtual Box with 2GM RAM, Virtual Box with 4GB RAM, Linux PC, Rugged PC and Odroid XU4.

8.4.1 Hardware Host Specifications

The specification of each hardware host is summarized in Table 8-3.

Table 8-3. Hardware host specifications

Device	Processor	Processor Frequency	Storage/ Memory	Device Type	Operations System
Lenovo Yoga Virtual Box (2GB RAM)	Intel Core i5-5200U CPU	2.20GHZ	217.5GB/ 2GB RAM	All in one PC	Ubuntu 16.04 64-bit LTS
Lenovo Yoga Virtual Box (4GB RAM)	Intel Core i5-5200U CPU	2.20GHZ	217.5GB 4GB RAM	All in one PC	Ubuntu 16.04 64-bit LTS
Linux PC	AMD A8-5600K APU with Radeon™ HD Graphics x4	3.6GHz x4 (4 processors)	976.6GB 7.2GB RAM	All in one PC	Ubuntu 16.04 LTS 64-bit
Rugged PC	Intel Celeron CPU N3160	1.60GHz x4 (4 processors)	117.8GB 7.5GB RAM	Industrial computer	Ubuntu 16.04 LTS 64-bit
Odroid XU4	ARMv7 Processor rev 3 (v7l)x8	1.4MHz x4 (4 Processors) 2MHz x4 (4 Processors)	64GB 2GB RAM	Single board computer	Ubuntu Mate 1.12.1 32-bit

8.4.2 BEMOSS™ Installation on Each Hardware Host

Before running the test, Ubuntu was installed on each hardware host. This was followed by BEMOSS™ installation, following the installation guide available on Github, URL: https://github.com/bemoss/BEMOSS3.5/wiki/BEMOSS-Installation-Guide.

8.4.3 Test Procedure

On each hardware host, CPU and RAM usage was monitored based on the following scenarios:

- Linux-only (no BEMOSS™)
- BEMOSS™-only (no devices)

- BEMOSS™ with 10 devices to monitor & control
- BEMOSS™ with 20 devices to monitor & control
- BEMOSS™ with 30 devices to monitor & control
- BEMOSS™ with 50 devices to monitor & control
- BEMOSS™ with 75 devices to monitor & control

8.4.4 Performance Test Results

Test results are summarized in Tables 8-4 and 8-5, for CPU and RAM usage, respectively:

Table 8-4. CPU usage on different hardware hosts and loading levels

Hardware host	CPU Usage						
	Linux-only (no BEMOSS)	BEMOSS (no device)	10 devices	20 devices	30 devices	50 devices	75 devices
Odroid XU4	4.4%	12.5%	14.7%	16.2%	15.6%	-	-
Virtual box (2 GB)	22%	30%	35%	40%	43%	50%	60%
Virtual box (4GB)	22%	25%	30%	35%	38%	46%	52%
Linux PC	4%	4.25%	5.5%	8.5%	11%	14%	18%
Rugged PC	11%	11.3%	14%	12.5%	12.5%	12.5%	14.25%

Table 8-5. RAM usage on different hardware hosts and loading levels

Hardware host	CPU Usage						
	Linux-only (no BEMOSS)	BEMOSS (no device)	10 devices	20 devices	30 devices	50 devices	75 devices
Odroid XU4 (2GB RAM)	366MB	1.3GB	1.5GB	1.7GB	1.9GB	-	-
Virtual box (2 GB RAM)	806MB	1.2GB	1.6GB of RAM and 600MB of swap memory	1.6GB RAM and 950MB of swap memory	1.8GB RAM and 1021MB swap memory	1.8GB RAM and 1.5GB in swap memory	1.8GB RAM and 2.1 GB in swap memory
Virtual box (4GB RAM)	664.6MB	2.1GB	2.3GB	2.6GB	2.9GB	3.4GB	3.7GB
Linux PC (7.2GB RAM)	933.5MB	2GB	2.3GB	2.9GB	3.2GB	3.8GB	4.8GB
Rugged PC (7.5GB RAM)	1.9GB	2.6GB	3.1GB	3.4GB	3.6GB	4.2GB	5.4GB

Performance test indicates that:

- Odroid XU4 can support monitoring & control of up to 35 devices.

- Virtual box with 2GB RAM can support monitoring & control of up to 10 devices before it goes into swap memory. However, using swap memory, 75 devices can still be monitored and control using this Virtual box.
- Virtual box with 4GB RAM can support monitoring & control of 85 devices before going into swap memory.
- Linux PC with 7.2GB RAM can support monitoring & control of up to 195 devices. After this, the PC slows down significantly.
- Rugged PC with 7.5GB RAM can support monitoring & control of up to 170 devices.

9.0 BEMOSS™ Demonstration in Three Buildings

This section summarizes BEMOSS™ demonstration in three buildings:

Figure 9-1. List of demonstration sites

#	Building	Address	Control
1	Virginia Tech's Architecture Building	1021 Prince St., Alexandria, VA, 22314	HVAC, plugs
2	Arlington County's Equipment Bureau	2701 S Taylor St., Arlington, VA 22206	Lighting
3	Virginia Tech's Building in Blacksburg, VA	460 Turner St. NW, Blacksburg, VA 24060	HVAC

9.1 VT Architecture Building in Alexandria, VA

9.1.1 Site Description

The first BEMOSS™ demonstration site is the Virginia Tech building located in Alexandria, VA (1021 Prince St.) hosting the School of Public and International Affairs (SPIA) and Center for Public Administration and Policy (CPAP). See Figure 9-2. This building has an approximate area of 25,000 square feet, and consumes about 14-25 MWh per month. Its peak load occurs in summer and the peak load is about 61 kW. Monthly electricity bill is about $2,000 per month.

Figure 9-2. Virginia Tech building at 1021 Prince St., Alexandria, VA
(source: http://ncr.vt.edu/locations/Alexandria.html)

9.1.2 Floor Plan and Electrical Loads

This building has three floors. The first floor mainly hosts faculty offices, reception areas and workshop space for students. The second floor hosts a number of offices. The third floor has several classrooms and extra offices. Figure 9-3 shows the floor plan and marks the locations of thermostats.

Figure 9-3. Floor plan of the Virginia Tech building in Alexandria, VA (▲ = location of thermostats)

HVAC and plug load controls are of focus in this building.

HVAC Loads: There are the total of twelve (12) thermostats in the building, of which one (1) thermostat is located on the first floor, five (5) thermostats are located on the second floor and six (6) thermostats are located on the third floor, as shown in Figure 9-3. Sample pictures of non-communicating thermostats existed in the building before BEMOSS™ deployment are shown in Figure 9-4.

Figure 9-4. Sample pictures of existing thermostats before BEMOSS™ deployment

Eleven (11) thermostats out of twelve are associated with a residential-type split HVAC system with a compressor on the roof and a respective air handler. Figure 9-5(a) illustrates a number of compressor units located on the roof the building. The compressors#1-5 ask marked in the figure are the units that serve the second floor. One thermostat on the third floor is used to control a packaged rooftop unit (RTU) shown in Figure 9-5(b). The associated air handling units of compressors #1-5 are shown in Figure 9-6.

(a) Residential-type split HVAC systems (b) A packaged rooftop unit (RTU)

Figure 9-5. Rooftop compressors

Figure 9-6. Air Handlers serving the second floor

Plug Loads: About 30% of the total number of receptacles in the buildings have loads connected to them, mostly consisting of desktop computers and monitors. On the second floor there are approximately 90 receptacles. Sample plug loads of the building are illustrated in Figure 9-7.

Figure 9-7. Sample plug loads in the building

9.1.3 BEMOSS™ Integration with Building

BEMOSS™ was deployed in this building to monitor and control HVAC load for the entire second floor and the classroom on the third floor; as well as selected plug loads (i.e., Xerox machine) on the third floor.

Two BEMOSS™ nodes were deployed. One serves as the core, and the other serves as the node. A total of six thermostats were replaced with communicating thermostats; and three smart plugs were used. Figure 9-8 illustrates BEMOSS™ deployment in this building.

With respect to the communication network, use was made of the existing Wi-Fi network of the building. Hence, all devices as well as the BEMOSS™ core/node were configured to use the existing building network for their operation. Furthermore, a dedicated IP address was assigned to each BEMOSS™ core and BEMOSS™ node, allowing remote monitoring and control of devices.

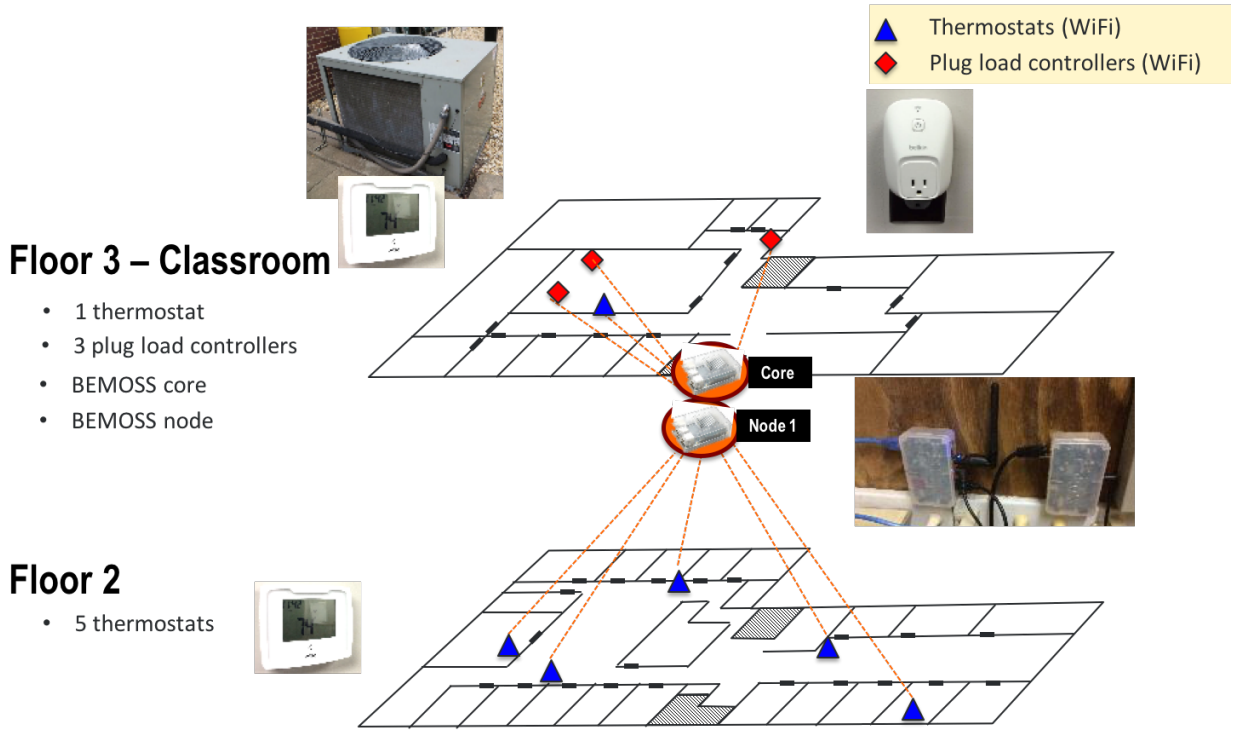


Figure 9-8. BEMOSS™ deployment in Alexandria building

9.2 Equipment Bureau Building in Arlington, VA

9.2.1 Site Description

The Equipment Bureau building (Figure 9-9) is a full-service vehicle maintenance and repair facility for Arlington County. It includes an extensive compressed air system, hydraulic lifts, and operates 18+ hours per day, which makes it an energy-intensive building. Total size of the building is about 45,000 sq ft, of which about 40,000 sq ft is the first floor facility where vehicle maintenance and repairs take place. The office space of about 5,000 sq ft is located on the second floor of the building.

Figure 9-9. Equipment Bureau, Arlington, VA (Source: http://environment.arlingtonva.us/energy/county-operations/building-energy-report-cards/specialty-facilities/equipment-division/)

9.2.2 Floor Plan and Electrical Loads

A photograph of the building floor plan is shown in Figure 9-10. The office space on the second floor is the square shaped space in the middle of the image with two tapered corners.

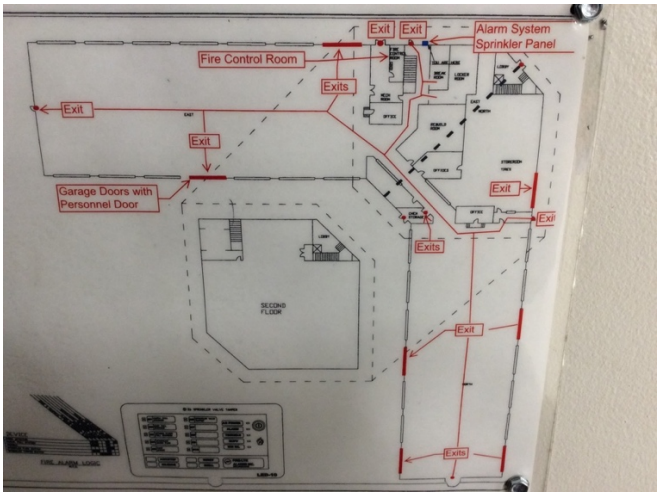


Figure 9-10. Building floor plan – Equipment Bureau building in Arlington, VA

Lighting loads in the office space on the second floor consist of Cree ZR-series troffers with dimmable LED fixtures. Dimming is achieved using the 0-10V dimming wires present at every fixture. In total, there are about 60 light fixtures in the office space. See Figure 9-11 and 9-12.

Figure 9-11. Photographs showing lighting loads in the building

Figure 9-12. Building drawing showing location of light fixtures

9.2.3 BEMOSS™ Integration with Building

The objective is to control six lighting circuits in this 5,000 square feet space. The following devices were deployed:

1) Two BEMOSS™ nodes
2) Three Wattstopper LMRC-212 lighting load controllers
3) One BACnet gateway
4) One BACnet converter

Two BEMOSS™ nodes (one serves as the core; the other serves as the node), together with the BACnet converter were installed in a NEMA box, which was secured in an electrical room in the building. One BACnet gateway was used for daisy-chain connection with three Wattstopper LMRC-212 lighting load controllers, each of which controls two lighting circuits.

- One Wattstopper controls light fixtures in open office areas (A&B) shown in green (Figure 9-13)
- One Wattstopper controls light fixtures in conference rooms (A&B) shown in blue (Figure 9-13)
- One Wattstopper controls light fixtures in Chief offices (A&B) shown in red (Figure 9-13)

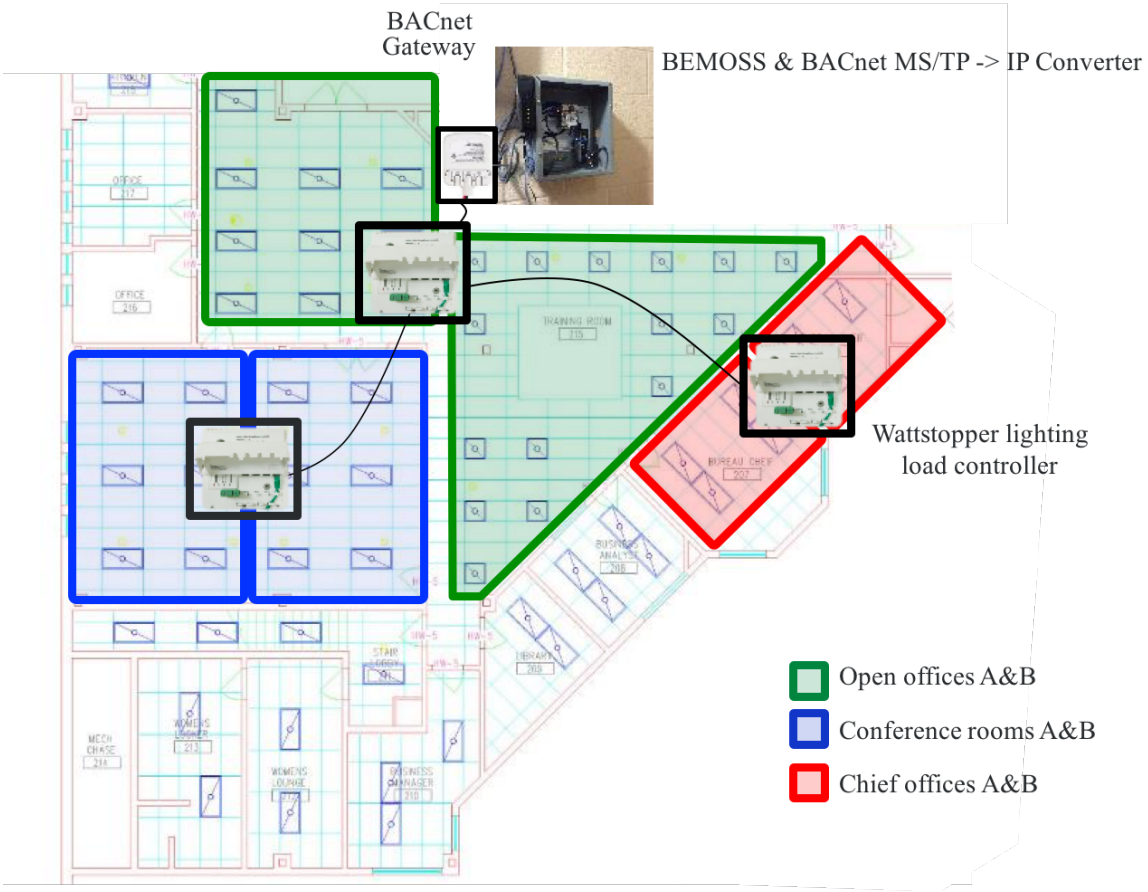


Figure 9-13. BEMOSS™ deployment in Equipment Bureau building

For the communication network, each BEMOSS™ node was assigned a dedicated static IP address by the County's IT administrator to allow monitoring and control from a remote location.

9.3 Virginia Tech Building in Blacksburg, VA

9.3.1 Site Description

This building is located near the main campus of Virginia Tech in Blacksburg, VA. See Figure 9-14. It is primarily used for retail and commercial offices leased to several individual tenants. It has a total floor area of about 41,301 sq ft on two floors, and consumes about 46-65 MWh per month. Its peak load occurs in summer and the peak load is about 160 kW.

Figure 9-14. VT building in Blacksburg, VA

9.3.2 Floor Plan and Electrical Loads

The floor plan of this building on the second floor is illustrated in Figure 9-15. This floor has seven thermostats, represented by the letter "T" in the figure, and one temperature sensor, represented by the letter "S". Each thermostat controls one rooftop unit, which is located on the roof shown in Figure 9-16.

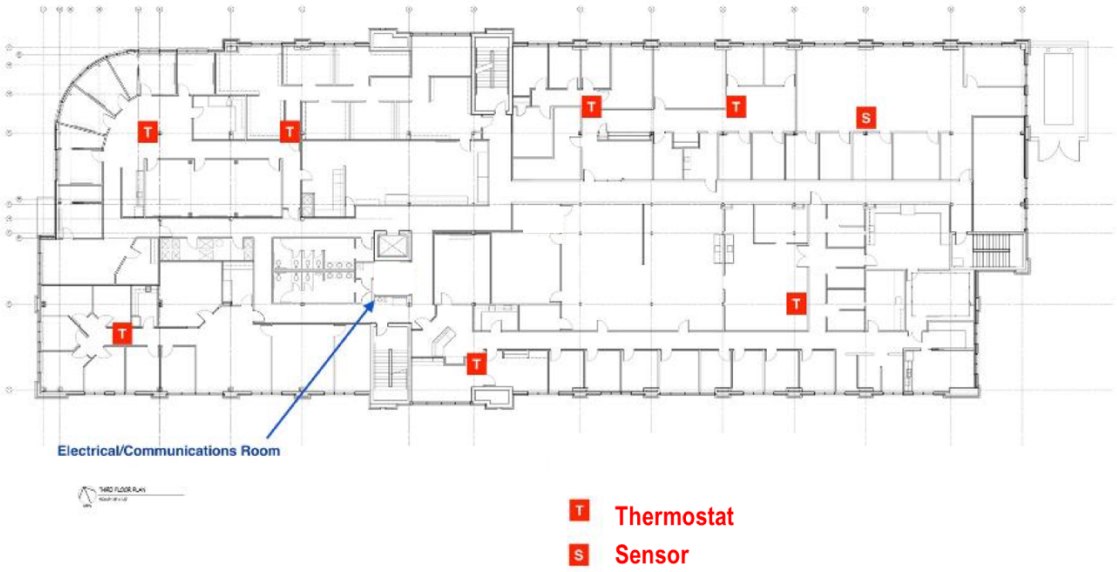


Figure 9-15. Floor plan of the building in Blacksburg, VA

Figure 9-16. Rooftop units on the roof of the Blacksburg building

Sample pictures of non-communicating thermostats existed in the building before BEMOSS™ deployment are shown in Figure 9-17.

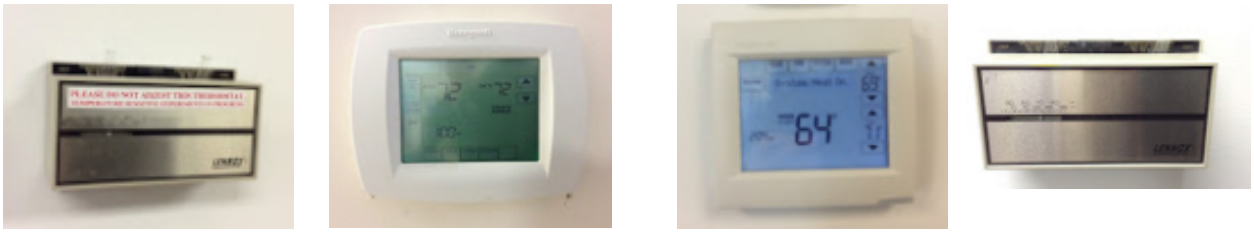


Figure 9-17. Sample pictures of existing thermostats before BEMOSS™ deployment

9.3.3 BEMOSS™ Integration with Building

BEMOSS™ was deployed in this building to monitor and control HVAC load for the entire second floor.

Two BEMOSS™ nodes were deployed. One serves as the core, and the other serves as the node. A total of seven thermostats were replaced with communicating thermostats; and the existing sensor was replaced with another communicating thermostat.

With respect to the communication network, the University's IT Department created a secured network specifically for BEMOSS™ deployment. All thermostats as well as BEMOSS™ core/node were configured to connect to this secured network provided. A dedicated IP static address was assigned to each BEMOSS™ core and BEMOSS™ node, allowing remote monitoring and control of devices.

10.0 Functionality Test, Availability and Evaluation of Electricity Savings

10.1 Functionality Test

Functionality tests were performed at the demonstration sites. Since the Alexandria building has both HVAC and plug load controls, tests were performed to make sure that thermostats/plug load controllers can be controlled, scheduled can be set and historical data can be accessed and downloaded. For the Equipment Bureau building, BEMOSS™ ability to control lighting loads were tested. For the Blacksburg building, BEMOSS™ ability to control HVAC loads was evaluated.

Tests performed and results are summarized in the table below.

Table 10-1. BEMOSS™ functionality test results

	School Of Public And International Affairs Academic Building (Alexandria)	**Equipment Bureau Building (Arlington)**	**Virginia Tech Building (Blacksburg)**
HVAC control			
- change thermostat set point	OK	-	OK
- change thermostat mode	OK	-	OK
- change fan mode	OK	-	OK
- set schedule	OK	-	OK
- perform anti-tampering	OK		OK
- access historical data	OK	-	OK
- download historical data	OK	-	OK
Lighting control			
- switch lights ON/OFF	-	OK	-
- dim the lights	-	OK	-
- set schedule	-	OK	-
- access historical data	-	OK	-
- download historical data	-	OK	-
Plug load control			
- switch plug load ON/OFF	OK	-	-
- set schedule	OK	-	-
- access historical data	OK	-	-
- download historical data	OK	-	-

Test results indicate that BEMOSS™ can perform flawless functions to change set point, set schedule and allow historical data access/download for all three load controllers (HVAC, lighting and plug load) at the demonstration sites.

10.2 Availability

BEMOSS™ downtime was recorded at three demonstration sites to evaluate its operational availability. BEMOSS™ operational availability is defined as: $\left\{1 - \left(\frac{Total\_downtime}{Total\_testPeriod}\right)\right\} \times 100\%$. The table below summarizes BEMOSS™ availability (%) and downtime (in total minutes each month and number of events) at three demonstration sites.

Table 10-2. BEMOSS™ availability and downtime from June 2016 to June 2017

	School Of Public And International Affairs Academic Building (Alexandria)	Equipment Bureau Building (Arlington)	Virginia Tech Building (Blacksburg)
June 2016	99.97% (13 mins, 4 events)	100% (0 min, 0 event)	99.98% (11 mins, 4 events)
July 2016	99.95% (24 mins, 5 events)	100% (0 min, 0 event)	99.93% (32 mins, 8 events)
August 2016	99.97% (13 mins, 2 events)	100% (0 min, 0 event)	99.94% (28 mins, 7 events)
September 2016	99.96% (17 mins, 6 events)	100% (0 min, 0 event)	99.97% (12 mins, 2 events)
October 2016	99.99% (2 mins, 2 events)	100% (0 min, 0 event)	99.99% (5 mins, 1 events)
November 2016	100% (0 min, 0 event)	100% (0 min, 0 event)	99.95% (19 mins, 2 events)
December 2016	99.99% (3 mins, 3 events)	100% (0 min, 0 event)	99.98% (10 mins, 1 event)
January 2017	100% (0 min, 0 event)	100% (0 min, 0 event)	N/A
February 2017	99.99% (4 mins, 3 events)	100% (0 min, 0 event)	N/A
March 2017	99.99% (4 mins, 3 events)	99.98% (9 mins, 1 event)	N/A
April 2017	99.99% (6 mins, 1 event)	99.98% (7 mins, 1 event)	N/A
May 2017	100% (0 min, 0 event)	100% (0 min, 0 event)	N/A
June 2017	100% (0 min, 0 event)	100% (0 min, 0 event)	N/A

V2.1 on Odroid

V3.5 on Odroid

V3.5 on a cloud server

V2.1 on Odroid

It can be seen that since the beginning of BEMOSS™ deployment in three buildings, BEMOSS™ provides better than 99.9% availability. This is attributed to the fact that BEMOSS™ has a watchdog mechanism to detect any agent crashes. Once an agent crash is detected, the system will automatically restart the agent.

Also, it can be seen that BEMOSS™ downtime improved over time from July 2016 to January 2017 in the Alexandria building due to code improvement and deployment of BEMOSS™ v3.5. Additional bugs found during February-March-April 2017 were fixed, and 100% availability were observed in May and June 2017.

For the Equipment Bureau, BEMOSS™ operated reliably since its installation in mid-2016; however, one failure event was noticed in March and April 2017. The reason of the crash was the large volume of accumulated data during the past several months. Since the crash, the team backed up the data every month and deleted old data periodically to prevent accumulating large volume of data in the database. It is also recommended to restart the database periodically to allow Cassandra to initiate its own clean-up capability.

10.3 Electricity Measurements and Estimated Savings

This section discusses electricity measurements and estimated electricity savings at three demonstration sites.

10.3.1 Alexandria Building: Electricity Measurements

In this building, BEMOSS™ is used to monitor and control five RTUs serving the cooling/heating needs for office spaces on the 2nd floor. The daily energy consumption (kWh) of the 2nd floor compressor load at the academic building, Alexandria, VA, together with the daily average outdoor temperature (°F), from July 2016 to June 2017 are provided in Figures 10-1, 10-2, 10-3 and 10-4.

It can be seen that:

- Maximum daily HVAC energy consumption was around 110kWh in the months of July and August 2016 when the average daily outdoor temperature was about 90°F.
- Daily HVAC energy consumption decreased as outdoor temperature decreases, and the lowest daily energy consumption of 1.7kWh appeared in December 2016 and January 2017 when the building was heated through a closed-loop hot water system fueled by natural gas.

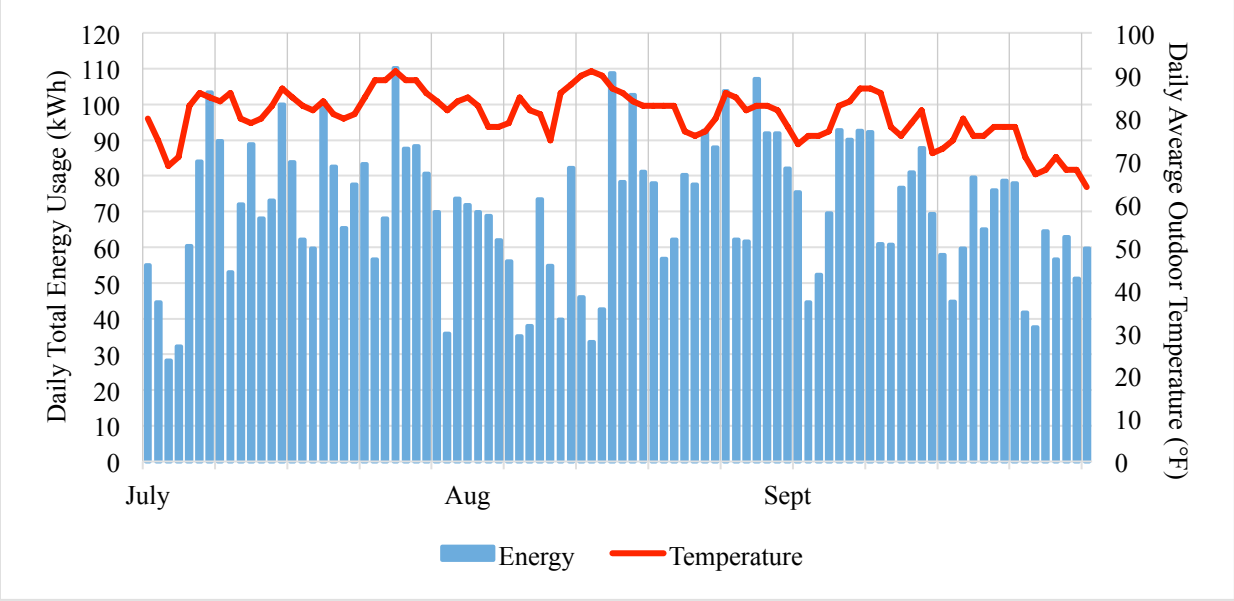


Figure 10-1. Daily energy consumption of the 2nd floor compressor load at the Alexandria building and daily average outdoor temperature from July to September 2016.

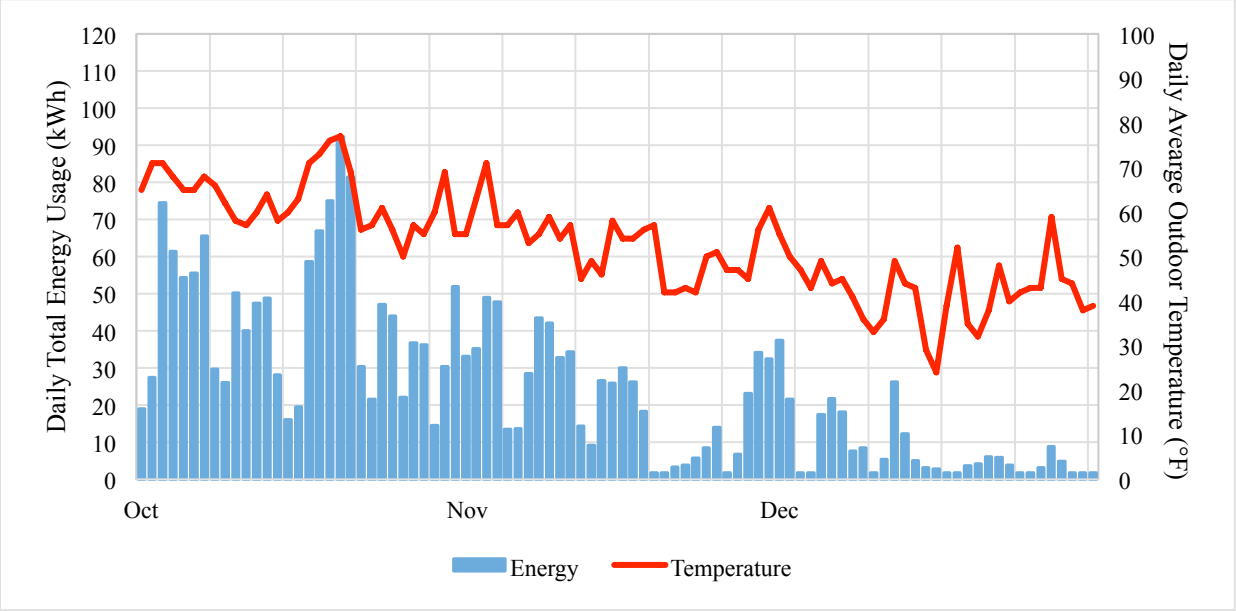


Figure 10-2. Daily energy consumption of the 2nd floor compressor load at the Alexandria building and daily average outdoor temperature from October to December 2016.

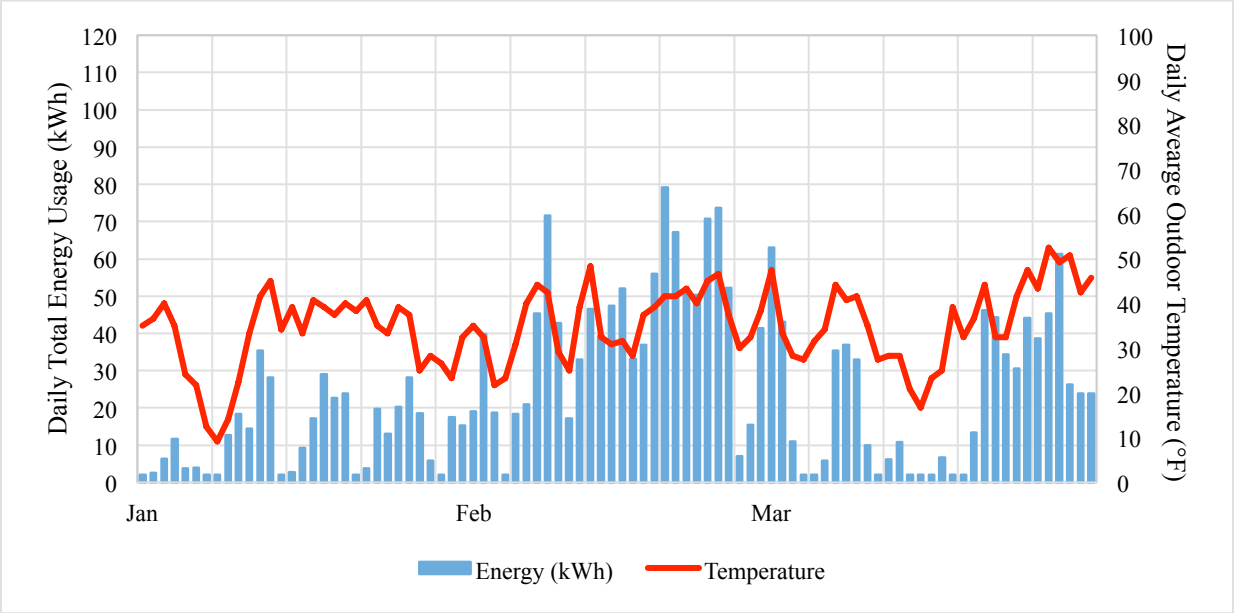


Figure 10-3. Daily energy consumption of the 2nd floor compressor load at the Alexandria building and daily average outdoor temperature from January-March 2017.

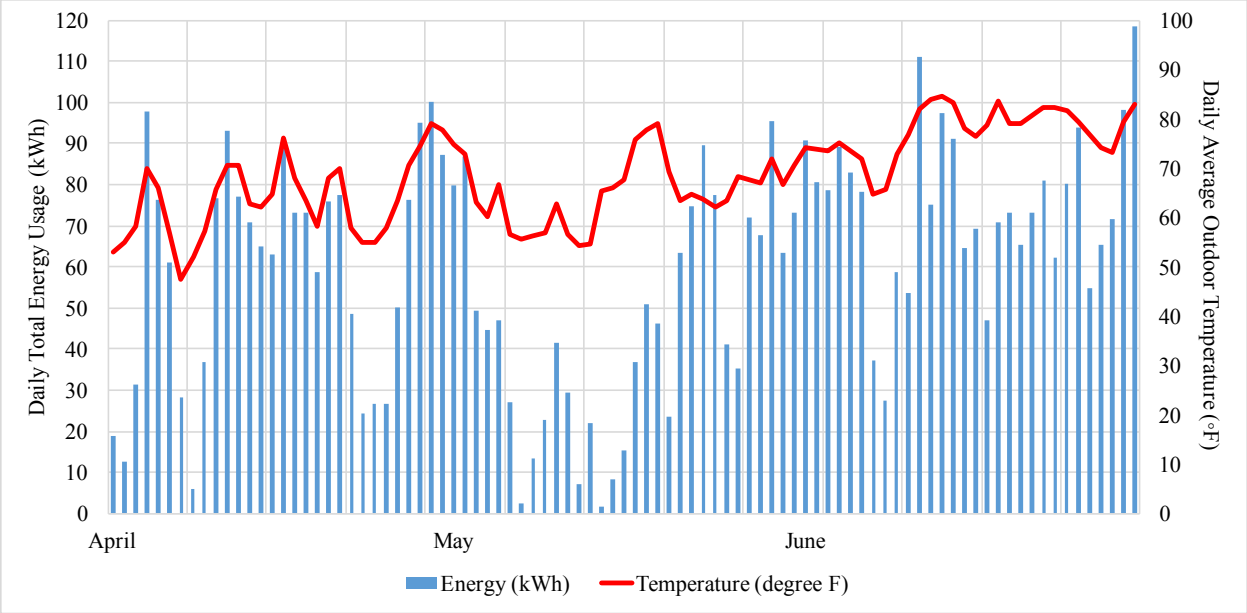


Figure 10-4. Daily energy consumption of the 2nd floor compressor load at the Alexandria building and daily average outdoor temperature from April-June 2017.

The table below summarizes monthly energy consumption (kWh) for five RTUs on the second floor.

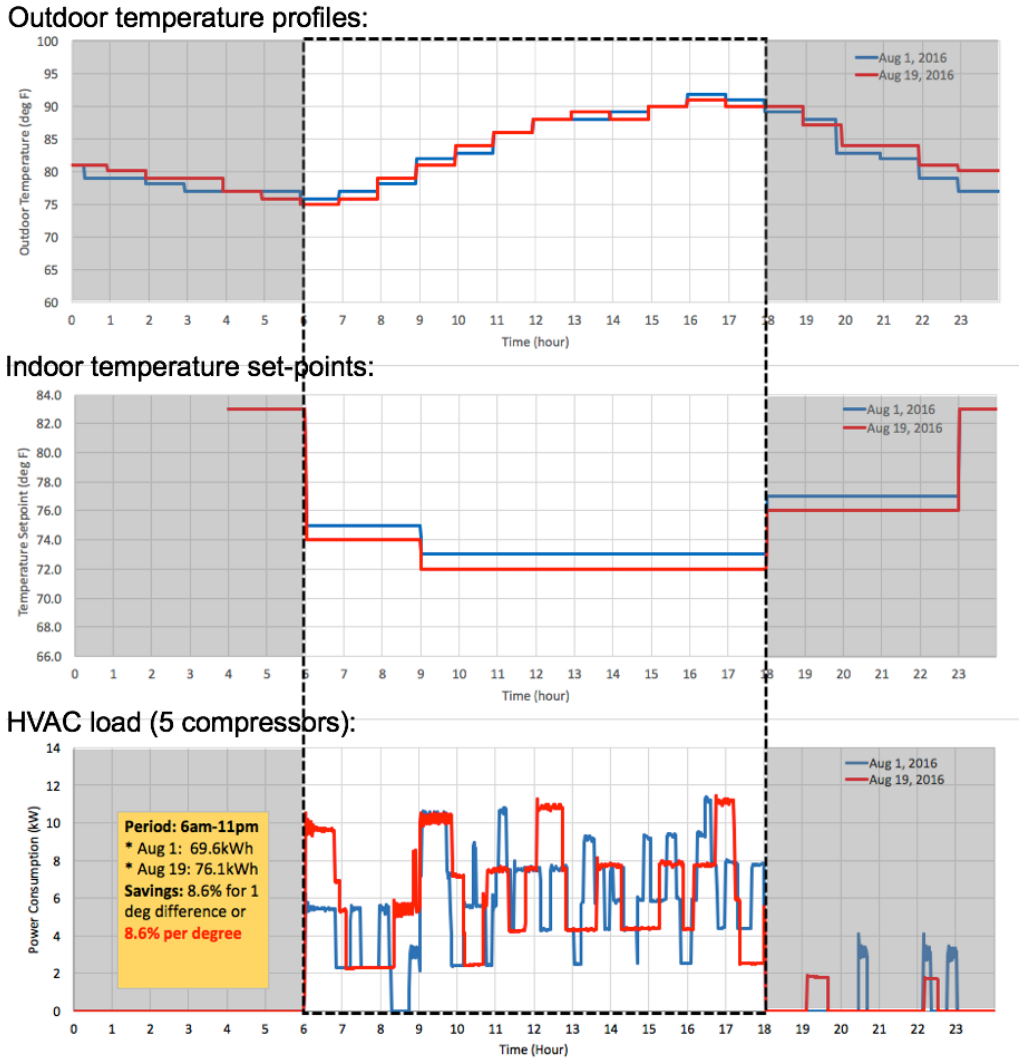
Table 10-3. Monthly energy consumption (kWh) for five RTUs on the second floor of the Alexandria building from July 2016 to June 2017

Month	July 2016	Aug 2016	Sep 2016	Oct 2016	Nov 2016	Dec 2016	Jan 2017	Feb 2017	Mar 2017	Apr 2017	May 2017	June 2017
kWh	2,238	2,198	2,041	1,373	661	245	330	965	592	1,798	1,410	2,240

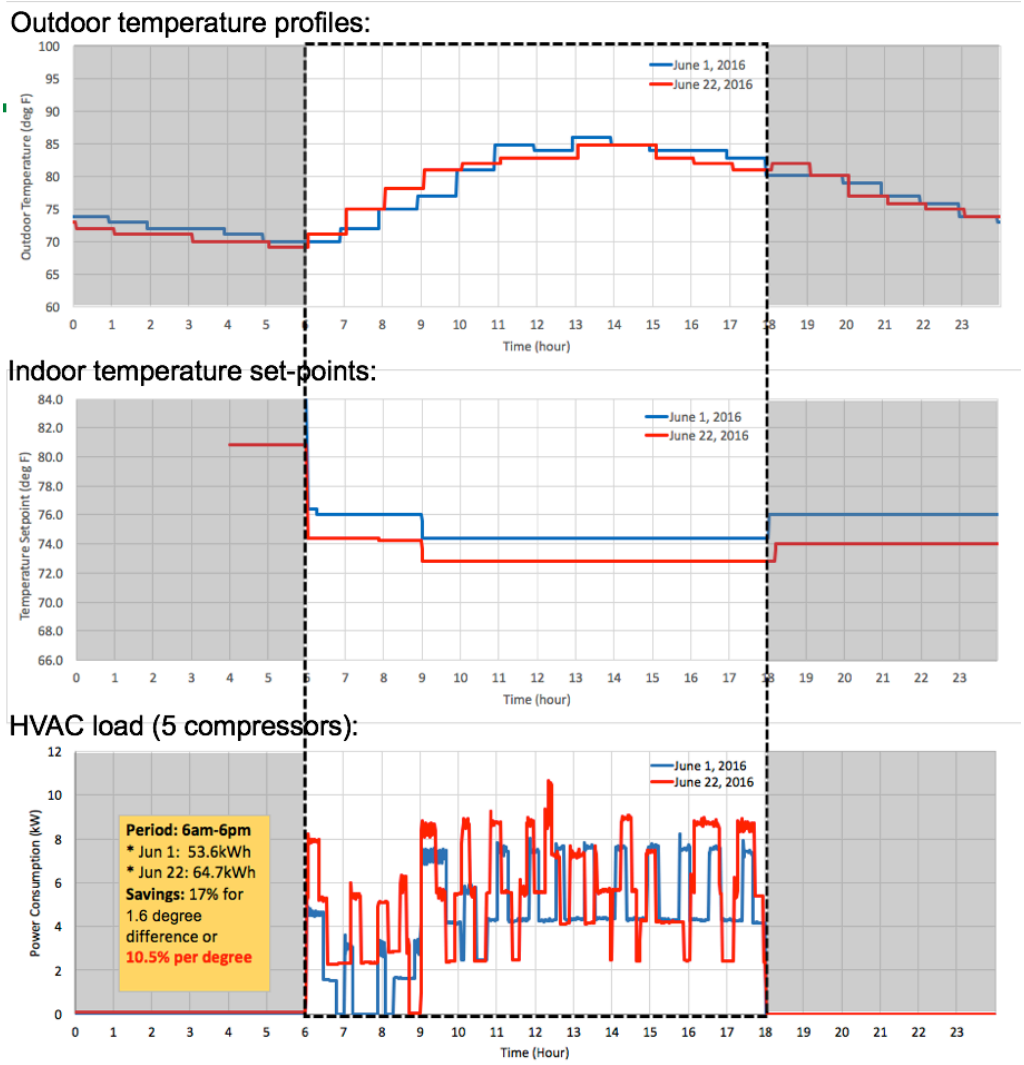
10.3.2 Alexandria Building: Estimated Savings

Experiments were conducted to investigate the impact of adjusting cooling set point on the total energy consumption of five compressors at the Alexandria building. Figure 10-5 shows the comparison of the following parameters: outdoor temperature, temperature set points and power consumption of five compressors at the Alexandria Building, of two days with similar outdoor temperature profiles, i.e., August 1 vs August 19, 2016 and June 1 vs June 22, 2016. August 1&19 had the peak outdoor temperature of about 92 deg F, while June 1&22 had the peak outdoor temperature of about 85 deg F.

For each set (two days with similar outdoor temperature profiles), daily temperature set point of Day 1 was manipulated such that it was higher than that of Day 2 by one degree or more. The total daily HVAC kWh consumption was then compared between Day 1 and Day 2 to determine the amount of HVAC energy savings (kWh) achievable per degree of temperature set point increase. Note that: the energy consumption (kWh) was the sum of all five compressors serving the 2nd floor of the Alexandria building.



(a) August 1 vs August 19, 2016



(b) June 1 vs June 22, 2016

Figure 10-5. Outdoor temperature, temperature set points and power consumption of five compressors at the Alexandria Building.

The table below summarizes the results:

Table 10-4. Energy savings analysis

	Date	Lowest/Highest outdoor temperature	Difference in temperature set points	Energy savings
Set (a)	August 1, 2016 August 19, 2016	76°F/92°F 75°F/91°F	1°F	8.6%
Set (b)	June 1, 2016 June 22, 2016	70°F/86°F 69°F/85°F	1.6°F	17% (or 10.5% per degree)

It can be seen that HVAC energy savings by raising the temperature set point varied depending on a case-by-case basis. The variation can be attributed to the difference in outdoor temperature profiles, indoor temperature set points, as well as the variation in occupancy and internal loads. Set (a) indicates energy savings per degree of 8.6% with the peak outdoor temperature of 92°F, while Set (b) indicates energy savings per degree of 10.5% with the peak outdoor temperature of 86°F.

10.3.3 Equipment Bureau Building: Electricity Measurements and Estimated Savings

At the Equipment Bureau building, six circuits of LED lights were monitored and controlled by BEMOSS™. These lights were dimmed based on the pre-set schedule. Scheduled dimming is from 6:00am to 9:00pm at the following brightness levels:

- Open office area A: 50%
- Open office area B: 45%
- Conference room A: 50%
- Conference room B: 45%
- Chief office area A: 30%
- Chief office area B: 30%

Table 10-5 shows the total energy consumption and estimated energy savings at the Equipment Bureau building during from June 2016 to May 2017. The data shows energy savings of around 34% due to dimming control.

Table 10-5. Energy savings due to dimming control at the Equipment Bureau building (June 2016 – June 2017)

Month	**Total Measured Energy Consumption (kWh)**	**Total Calculated Energy Consumption without Dimming (kWh)**	**Energy Savings by Dimming (%)**
June 2016	260.08	391.03	33.49%
July 2016	92.48	137.85	32.91%
August 2016	-	-	-
September 2016	263.51	397.29	33.67%
October 2016	264.37	399.90	33.89%
November 2016	278.13	423.78	34.37%
December 2016	280.76	426.40	34.16%
January 2017	243.38	365.40	33.39%
February 2017	229.32	357.53	35.86%
March 2017	264.26	414.34	36.22%
April 2017	210.20	323.25	34.97%
May 2017	248.51	388.20	35.98%
June 2017	222.70	349.37	36.25%
		Average savings	**34.50%**

10.3.4 Blacksburg Building: Electricity Measurements

In this building, BEMOSS™ is used to monitor and control seven RTUs serving the cooling need for office spaces on the second floor of the building.

The daily energy consumption (kWh) of the compressor load at the commercial office and retail building, Blacksburg, VA, from July to December 2016 is provided in Figures 10-6 and 10-7, together with the daily average outdoor temperature (°F).

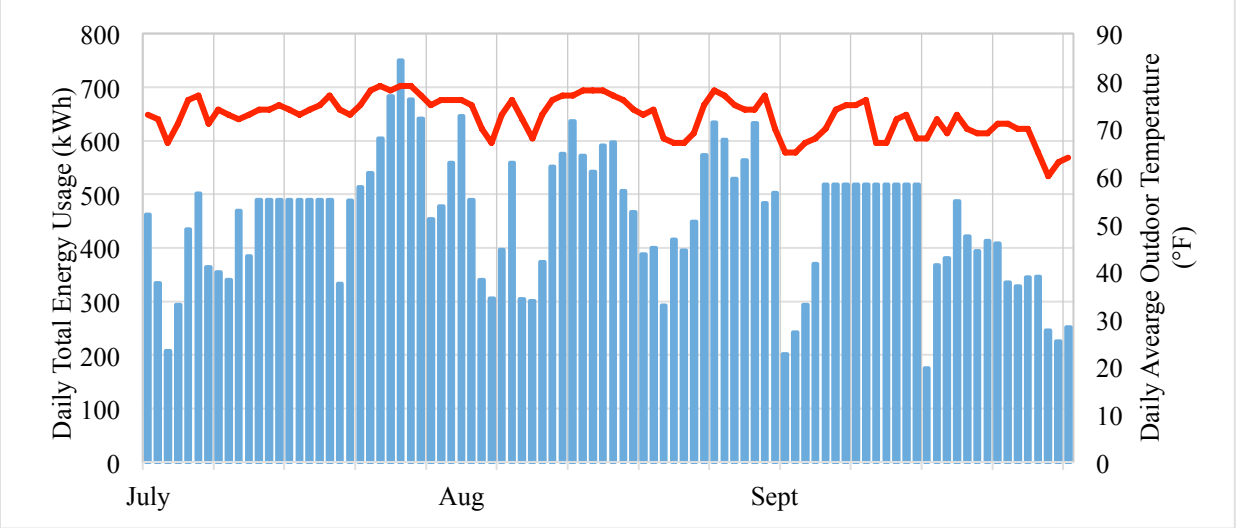


Figure 10-6. Daily energy consumption of the compressor load at the Blacksburg building and daily average outdoor temperature from July to September 2016.

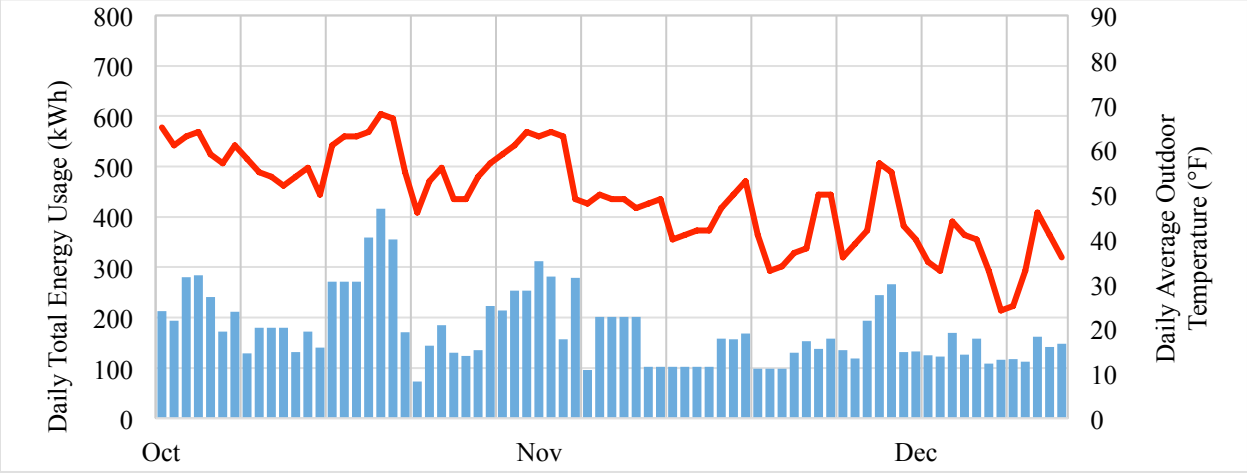


Figure 10-7. Daily energy consumption of the compressor load at the Blacksburg building and daily average outdoor temperature from October to December 2016.

It can be seen that:

- The maximum daily energy consumption was around 750 kWh in July 2016, when the daily average outdoor temperature was also high.
- The minimum daily energy consumption was around 100 kWh in November-December 2016 in winter since gas heating was used and thus only fans were consuming electricity.

The table below summarizes monthly energy consumption (kWh) for seven RTUs on the second floor of this building.

Table 10-6. Monthly energy consumption (kWh) for seven RTUs on the second floor of the Blacksburg building from July 2016 to December 2016

Month	July 2016	Aug 2016	Sep 2016	Oct 2016	Nov 2016	Dec 2016
kWh	14,779	15,113	11,717	6,553	4,867	N/A

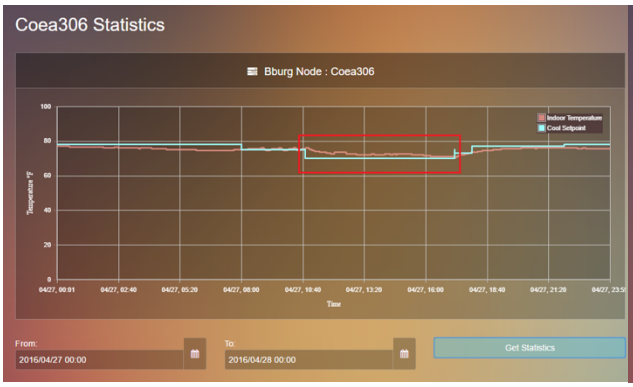
10.3.5 Blacksburg Building: Estimated Savings

Experiments were conducted to investigate the impact of the anti-tampering feature[5] on energy savings potential. This case was demonstrated using two days with similar outdoor temperature profiles.

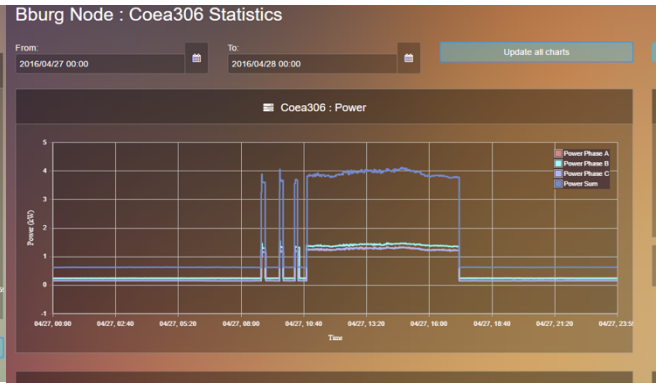
On Day 1, the anti-tampering feature was OFF, i.e., adjusting thermostat setpoint was allowed. At around 10AM, the thermostat setpoint was tampered from 75°F to 70°F and remained at 70°F until 5PM. See Figure 10-8(a). The corresponding RTU power consumption is shown in Figure 10-8(b) -- the daily energy consumption of 37.5kWh.

On Day 2, the anti-tampering feature was ON, i.e., adjusting themostat setpoint was not allowed, and any change in set point was reverted back to the original setpoint. Figure 10-8(c) indicates that the temperature setpoint remained at 70°F despite two tampering attempts. Figure 10-8(d) illustrates the corresponding RTU power consumption -- the daily energy consumption of 29.2kWh.

This implies that the BEMOSS™ anti-tampering feature results in **daily energy savings of 8.3kWh or about 22% ((37.5kWh-29.2kWh)/37.5kWh).**



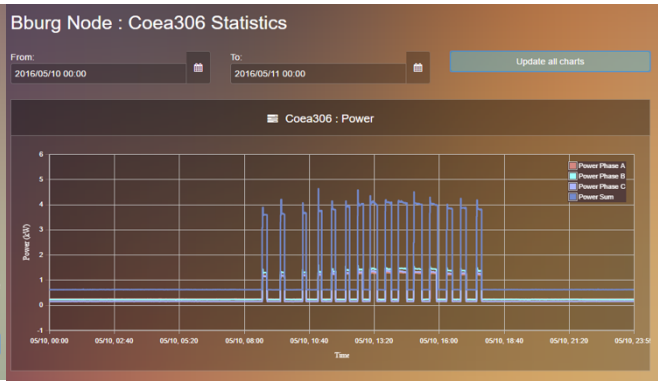
(a) Indoor temperature setpoint on Day 1



(b) RTU power consumption on Day 1

[5] Anti-tampering feature is the ability to detect a change in thermostat temperature set point (i.e., increase or decrease the set points) at the thermostat and revert it to the original set point automatically.

(c) Indoor temperature setpoint on Day 2



(d) RTU power consumption on Day 2

Figure 10-8. Indoor temperature setpoint and RTU power consumption on Day 1 (left) and Day 2 (right)

10.4 BACnet IP Gateway Issue in Blacksburg Building

This section summarizes the BACnet IP gateway issue in Blacksburg building, and discusses how the team investigated the issue and conducted the lab experiment.

10.4.1 Issue Found

Six WattNode power meters and one BACnet IP gateway (for converting BACnet IP to BACnet MS/TP) were installed and started operation in the Blacksburg building since May 2016. Power meters had been in operation without any problem for the first several months.

Starting from September 2016, the BACnet IP gateway began experiencing repeated offline events. This happened following the occurrence of a campus network issue on August 30, 2016, which made all BEMOSS™ devices disconnected from the network.

After a couple of offline and reset events, the team thought that it was the hardware issue and decided to replace the existing BACnet IP gateway with a new unit on November 11, 2016. However, the new gateway went offline again on November 18, 2016.

Following several offline events (again), a troubleshooting request was made to Virginia Tech – Communications Network Services (CNS) on November 21, 2016 to find out if there was the network related problem. A repair confirmation was provided by CNS the next day -- in which the port where the BACnet IP gateway and BEMOSS™ connected to the Ethernet was fixed. The gateway worked fine for two weeks and again went offline again.

10.4.2 Lab Experiment

By ruling out the hardware failure (by replacing the gateway in November 2016), the team suspected that the possible cause of frequent offline events could be related to software settings used between the gateway and power meters, i.e., inappropriate data collection frequency of BEMOSS™, and communication baud rate (i.e., data transmission rate between the BACnet IP gateway and power meters) In the Blacksburg deployment, BEMOSS™ requests the data from each power meter every 20 seconds collecting 30 measured objects shown in Table 2. The baud rate at the BACnet IP gateway and power meters was set at 9600 bits per second.

Table 10-7. Data measurement objects collected by WattNode Devices

Type	***Data***
Energy Objects	Energy_Sum, Energy_Pos_Sum, Energy_Pos_Sum_NR, Energy_Phase_A_Net, Energy_Phase_B_Net, Energy_Phase_C_Net,
Reactive and Apparent Energy Objects	Energy_Reactive_Sum, Energy_Reactive_Phase_A, Energy_Reactive_Phase_B, Energy_Reactive_Phase_C,
Power Objects	Power_Sum, Power_Phase_A, Power_Phase_B, Power_Phase_C
Reactive and Apparent Power Objects	Power_Reactive_Sum, Power_Reactive_Phase_A, Power_Reactive_Phase_B, Power_Reactive_Phase_C,
Voltage Objects	Voltage_Average_LN, Voltage_Phase_A, Voltage_Phase_B, Voltage_Phase_C,
Power Factor Objects	Power_Factor_Average, Power_Factor_Phase_A, Power_Factor_Phase_B, Power_Factor_Phase_C
Current Objects	Current_Phase_A, Current_Phase_B, Current_Phase_C
Frequency Object	Power Line Frequency

Our hypothesis was that – the data collection frequency may be set too frequent and/or baud rate setting may be inappropriate, making the BACnet IP gateway overloaded, thus forcing the gateway to go offline.

Experiments were then performed in the lab to investigate the impact of BEMOSS™ data request intervals and baud rate on the operation of the BACnet IP gateway. To conduct this experiment, a test bed was set up in a lab environment. The test bed consisted of two identical WattNode power meters (WNC-3Y-208-BN) and one BACnet IP gateway. The WattNode power meters and the BACnet gateway were RS-485 daisy chained with 22 AWG twisted pair shielded wire to the main data line. It was approximately 40 feet from the BACnet gateway to the last WattNode power meter. This set up is illustrated in Figure 10-9.

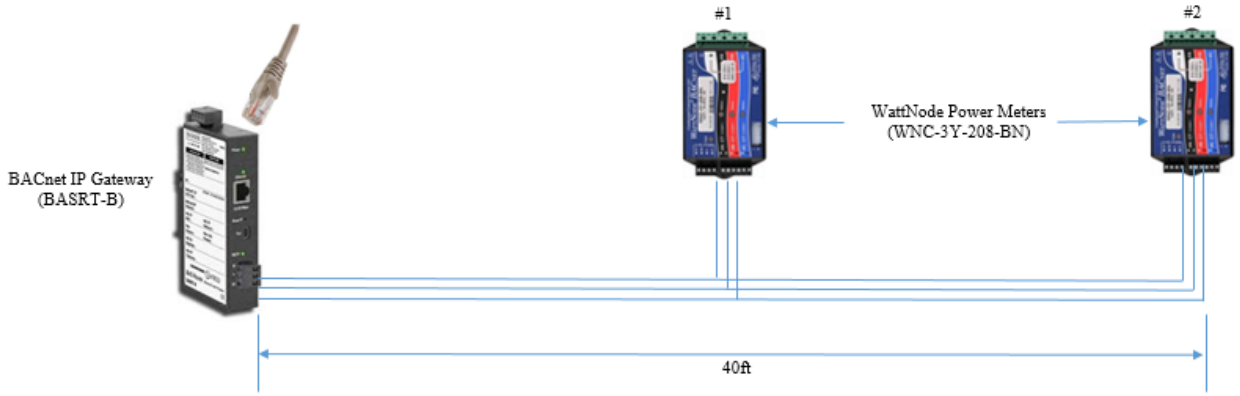


Figure 10-9. Lab set-up consisting of two identical WattNode power meters and one BACnet IP gateway.

BEMOSS™ was configured to interrogate data from WattNode power meters at four different data collection intervals, i.e., 20, 30, 60, 120 seconds, to archive the measurement objects listed in Table 10-7. Data request intervals were changed by updating the settings file in BEMOSS™.

Two different baud rates, i.e., 9,600 and 38,400 bits per second, were used. Baud rates were changed by modifying the setting page of the BACnet IP gateway; and changing the position of dip switch at each power meter.

Table 10-8 summarizes lab experiments conducted. Each case study was conducted for the duration of (at least) three days. The entire experiment was performed over one-month period.

Table 10-8. Selected case studies at different baud rates and data request intervals
(✓ = no failure event)

		Data Request Intervals			
		20 seconds	30 seconds	60 seconds	120 seconds
Baud Rate (bps)	9600	✓	✓	✓	✓
	38400	✓	✓	✓	✓

As shown in Table 10-8, there was no failure event of the BACnet IP gateway, nor communication errors between BEMOSS™ and the gateway and between the gateway and the power meters, regardless of the baud rate and data request intervals used.

10.4.3 Conclusion

The team concluded that the offline event of BACnet IP gateway in the Blacksburg building is case-specific and is due to stringent restrictions of building network architecture and/or settings -- because of the following reasons:

- The lab experiment showed no failure event with different BEMOSS™ data request intervals and baud rates.
- It was not hardware issue as the problematic BAS gateway was replaced with the new one, but the problem still persisted.
- The same BACnet IP gateway being used at the other BEMOSS™ demonstration site (i.e., Arlington County's Equipment Bureau) operates reliably (at 9600 baud rate and 60-second data collection intervals) without any offline event.
- The BACnet IP gateway event happened following the occurrence of a campus network issue on August 30, 2016

Recommendation: Even though the BACnet IP gateway did not go offline during the experiment when high frequency data collection (i.e., 20 seconds) and high baud rate (i.e., 38,400 bps) were used, it is suggested that BEMOSS queries the data from the BACnet IP gateway at 1-minute intervals, and set the baud rate of the BACnet IP gateway at 9,600 bps to avoid data congestion, data dropping, and the risk of the router going offline. For power meters, data collection at 1-minute intervals is reasonable to avoid database overload during long-term building operation. Additionally, it is recommended to assign a static IP address to a BACnet IP gateway. This will allow uninterrupted communications between BEMOSS and the gateway for continuous monitoring.

11.0 BEMOSS™ Software Access

The latest version of BEMOSS™ source code (version 3.5) has been made publically available on Github, URL: www.github.com/BEMOSS, together with its Wiki that provides description about BEMOSS overview, features, installation guides as well as developer resources.

The screen capture of BEMOSS™ 3.5 page on Github is shown in Figure 11-1.

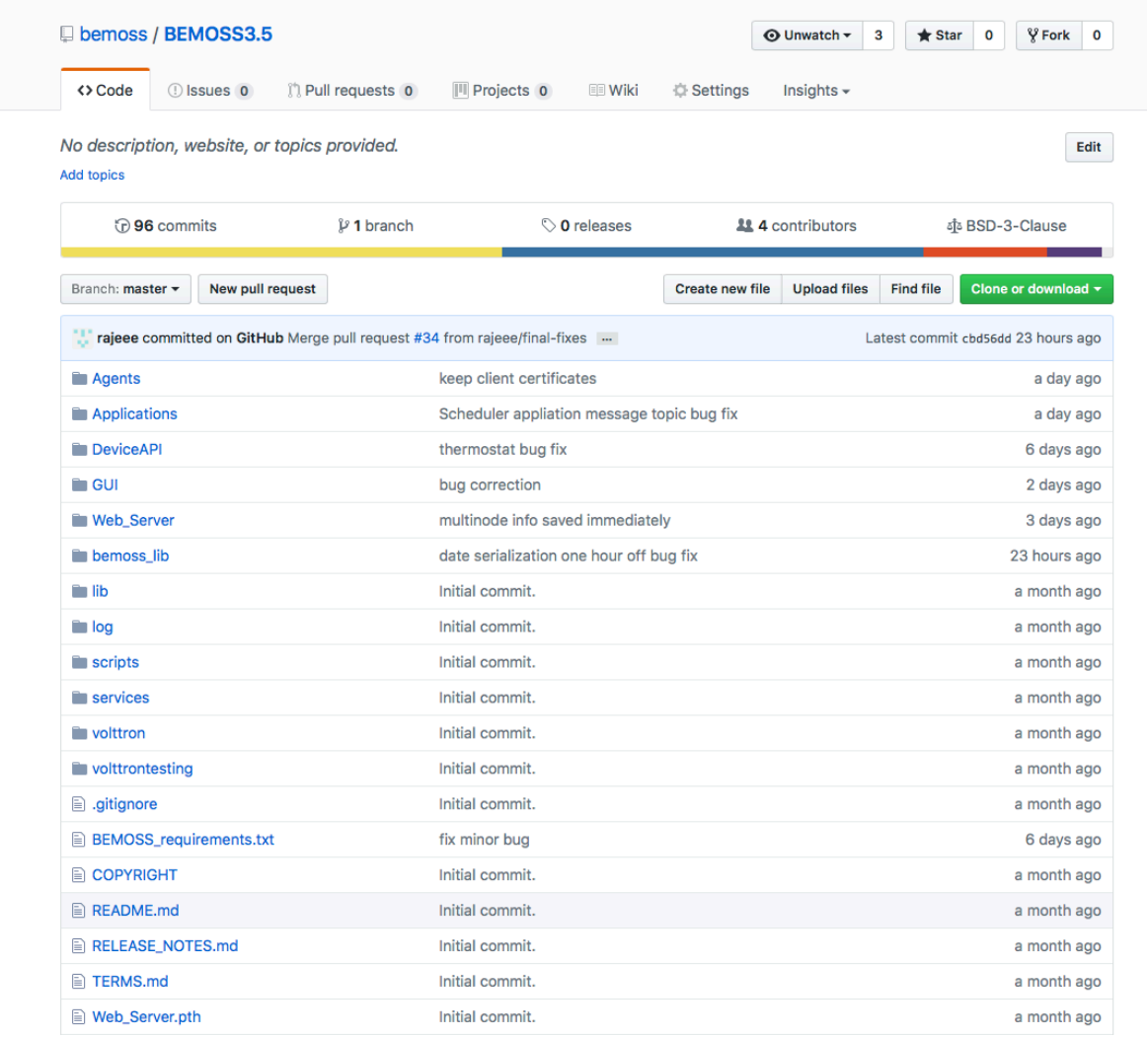


Figure 11-1. Screen capture of BEMOSS™ 3.5 page on Github

The screen capture of BEMOSS wiki page on Github is shown in Figure 11-1.

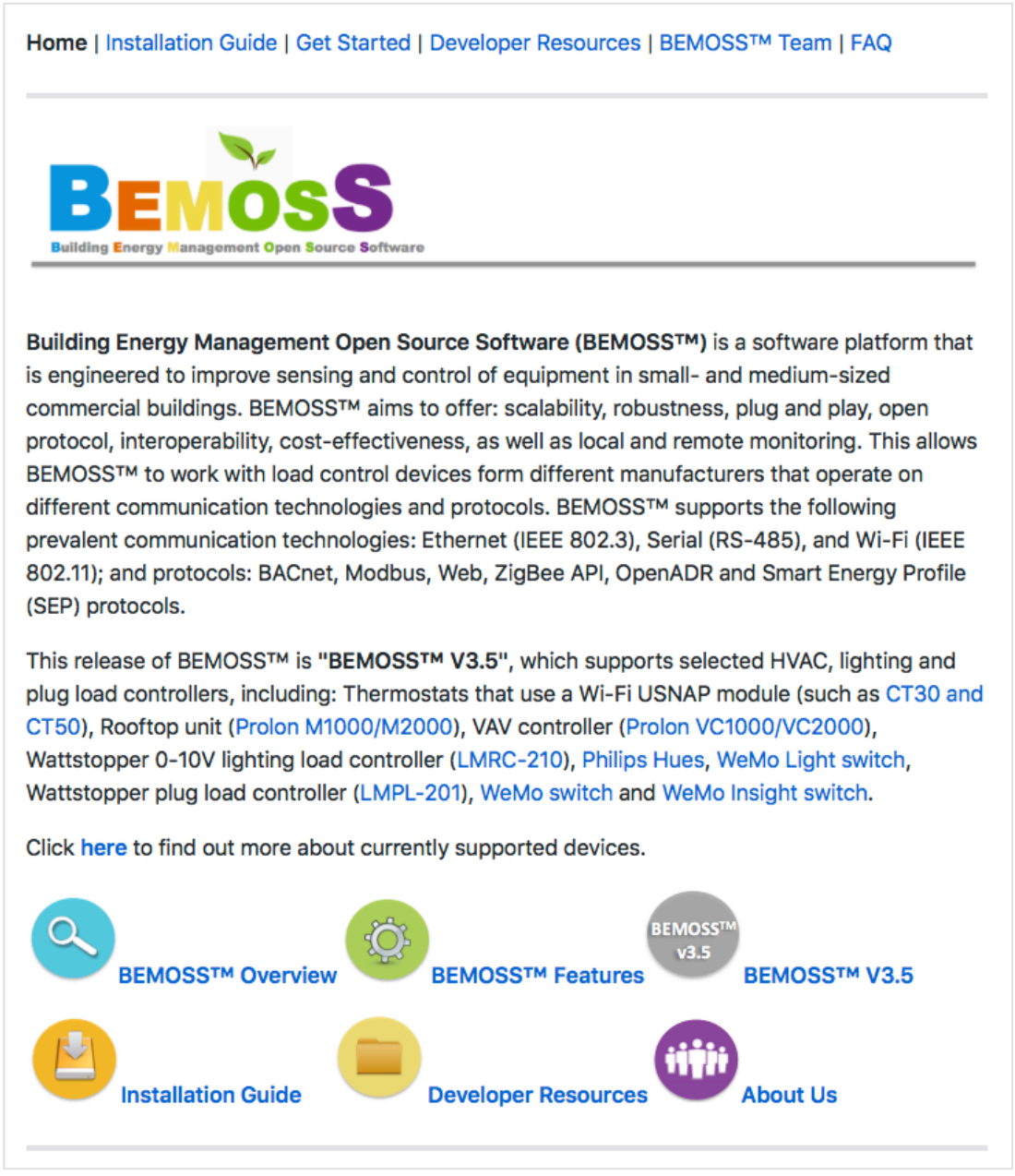


Figure 11-2. Screen capture of BEMOSS™ Wiki page on Github

BEMOSS official website can be found at www.bemoss.org. See Figure 11-3.

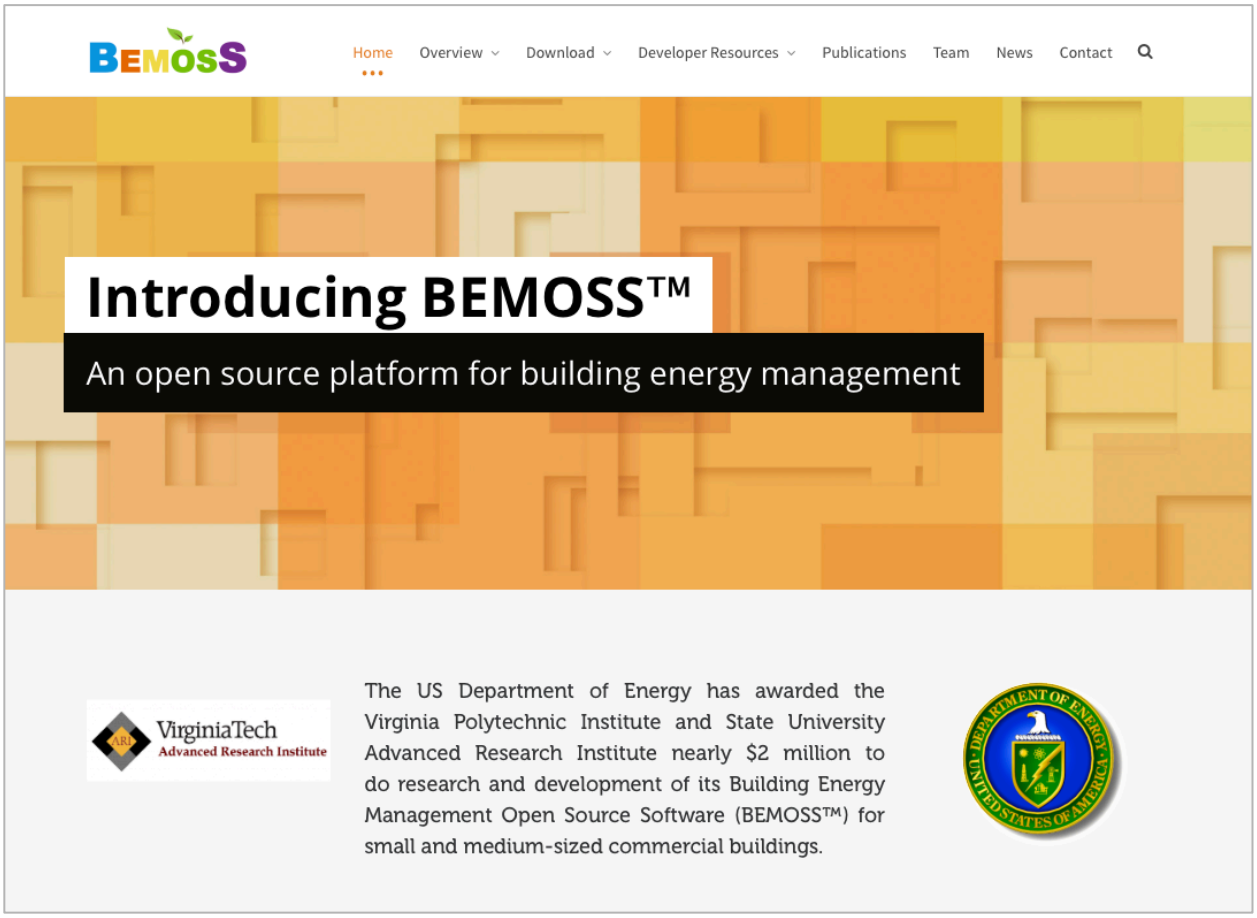


Figure 11-3. Screen capture of BEMOSS™ official website (www.bemoss.org)

In addition, BEMOSS discussion forum is also available at:
https://groups.google.com/forum/#!forum/bemoss.