**Aug 31**

14

**UCB OpenBAS**

Introduction to the UCB OpenBAS Prelease

UCB OpenBAS team

08

**Fall**

This document provides a overview of the UCB OpenBAS 8/21/14 Pre-release, describes some of the underlying concepts, and illustrates a typical usage case. This in not the same as installing a consumer product, as this seeks to illustrate how the current OpenBAS system is created from its Open Source code base. Much of what is presented here would be handled within typical commercial project packaging. It is not a “users manual” or “product installation guide”, but rather an introduction to the design and implementation of a powerful and flexible open source system. Our understanding is that DOE is keen to see what is “beneath the hood”.

Figure 1 shows the internal architecture of the OpenBAS system (<https://github.com/SoftwareDefinedBuildings/openbas> ) is shown It is accessed from any display device, such as a computer browser, tablet, phone, or embedded display. We discuss the OpenBAS presentation later. It is a simple, open source web application that can be extended, customized, or modified, built using the meteor framework (<https://www.meteor.com/>) over the OpenBAS api (application programming interface). The core of OpenBAS is constructed upon the open source sMAP (simple Monitoring and Actuation Protocol) infrastructure (<https://github.com/SoftwareDefinedBuildings/smap> ). Extensive documentation on sMAP is available at <http://smap.cs.berkeley.edu> which redirects to the google code site. sMAP also provides extensive pydoc documentation.

OpenBAS core comprises four kinds of RESTful web services:

* sMAP drivers: continuously running, monitored processes that connect to one of a wide variety of devices used for HVAC, Lighting, and General controllers and present it as a restful web service. Instances of sMAP drivers are described by a configuration (.ini) that specifies metadata relevant to the device and registers the data streams from these devices to the Archiver. A large collection of such drivers is available at <https://github.com/SoftwareDefinedBuildings/smap/tree/master/python/smap/drivers>.

* sMAP services: higher level, continuously running, monitored process that provider higher level OpenBAS functionality, such as discovery services (recognizing devices, loading their driver and constructing .ini files for the particular sMAP source), scheduling services (e.g., executing an integrated master schedule by publishing setpoints or commands to tstats, lighting controllers, or other devices.)
* sMAP archiver: a robust and efficient broker that stores Metadata associated with collections and streams, supports publication and subscription to streams, and implements a powerful query language over such richly tagged streams. It pushes updates to zero or more timeseries data stores.
* sMAP timeseries store: an optimized time series store for data streams that efficiently implements sMAP queries.

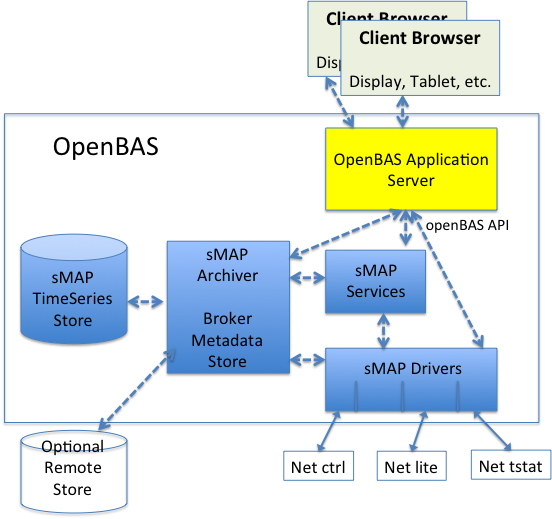


Figure 1 OpenBas RESTfull WebService Architcture

In the DOE 822 OpenBAS implementation for small to medium scale commercial buildings, all five of the components execute within the small, embedded linux box, typically on premises. However, the system is designed to scale up and down. Each of the five components can potentially run on distinct computer systems connected via the internet. For example, only the drivers might run on premises on a very small embedded device, while the rest might be on a facilities server. Or parts, such as the Timeseries store or the app server might run in the crowd.

A typical installation might be as in Figure 2 where the OpenBAS server is simply another device on the private LAN/WLAN along with the various devices it controls, such as WiFi thermostats, Ethernet thermostats, networked lighting controllers, gateways to various other networks, and so on. All of the open source code described here runs on the embedded linux box, as does the OpenBAS application.



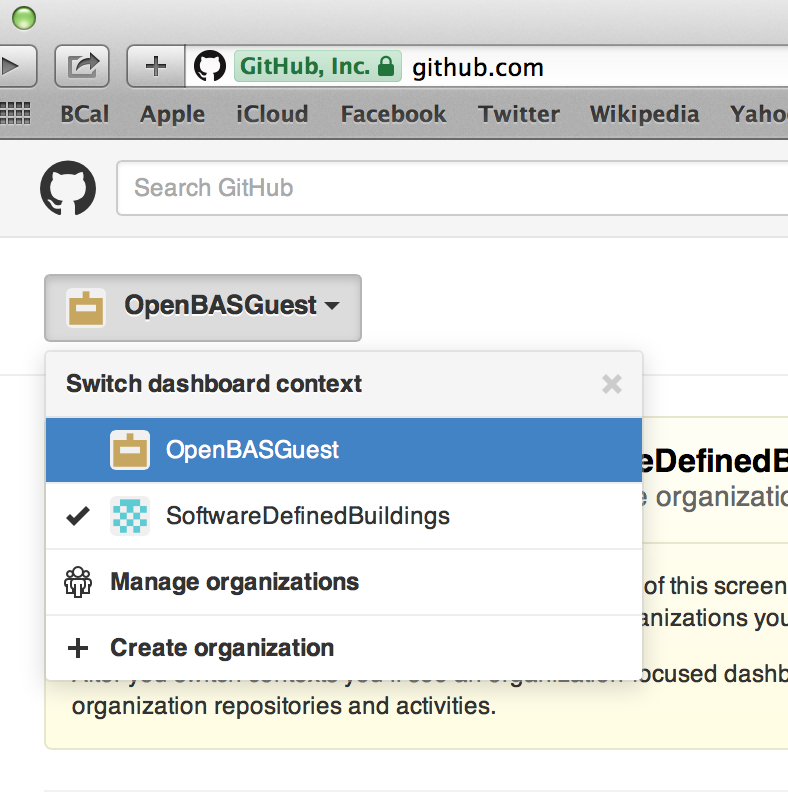
Figure 2 OpenBAS running on premises within an existing private LAN/WLAN

Another common deployment strategy would be to utilize the OpenBAS server as rooting the network of HVAC, lighting and general controller units within the building, optionally connected to other networks, as Shown in Figure 3. OpenBAS can operate with a variety of network organizations.



Figure 3 OpenBAS forming a private Building Area Network

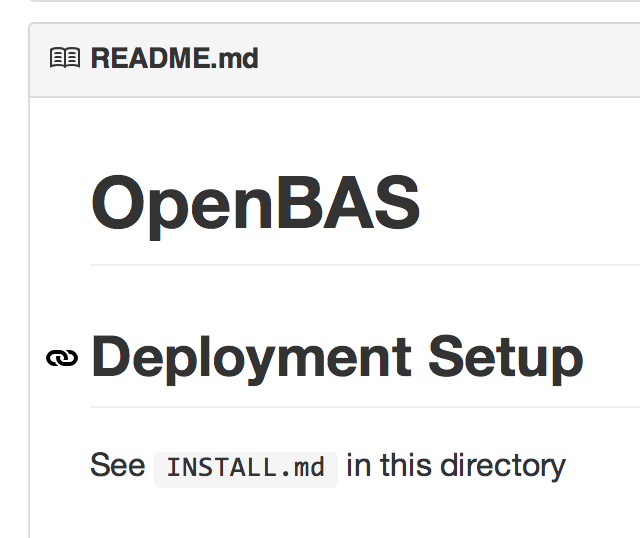
To access the UCB OpenBAS repository, use a browser to access <http://github.com>. Sign in as OpenBASGuest using the password openbas822project. In the repository selection on the left, select SoftwareDefinedBuildings.



click View SoftwareDefinedBuildings. You will see the complete repository for sMAP as well as OpenBAS.

This software release builds on a number of open source projects. We have provided simple install scripts to build the entire OpenBAS from scratch. Of course, in any product utilizing this technology, the consumer would do none of this. It would be packaged into the product. It you wanted to download these open source repositories you would simply clone them using git. Developers will be familiar with that process. We wont go into it here.

Instead, we have provided a complete install procedure for an OpenBAS deployment. It shows at the top of the README.



To see the full install process, click on INSTALL.md.

[https://github.com/SoftwareDefinedBuildings/openbas/blob/master/INSTALL.md](https://github.com/SoftwareDefinedBuildings/openbas/blob/master/INSTALL.md" \t "_blank)

The OpenBAS engine is built on the open source linux distribution Ubuntu. Instructions for installing version 14.04 are at <https://github.com/SoftwareDefinedBuildings/openbas/blob/master/INSTALL.md#installing-ubuntu-1404>

Starting from a clean Ubuntu system, the entire openBAS engine can be installed by simply:

curl http://install.openbas.cal-sdb.org/ | sudo bash –

The installation instructions give an additional step or two to configure this once installed.

This install script installs all of the other open source tools that are utilized by OpenBAS. You can inspect the script and see all of what it does to download all the necessary packages and then set up the installation. Again, this would not be something seen by a customer of an OpenBAS product, it is essentially what the manufacturer would use to build such a product on whatever cost-effective embedded device they were shipping it on.

With a clean ready to deploy OpenBAS in hand, <https://github.com/SoftwareDefinedBuildings/openbas/blob/master/INSTALL.md#getting-started>

Briefly outlines how to setup the actual deployment. So that you will not need to get ahold of actual hardware so try all this out, we have set up couple of VirtualBox virtual machines at various stages of the process that can be used to perform a simulated deployment – it have no real devices and no real space, but drivers for virtual devices provide all the connections. This will be presented to you under separate cover.

So that you can get a sense of what a deployment is like we have provided access to two of them.

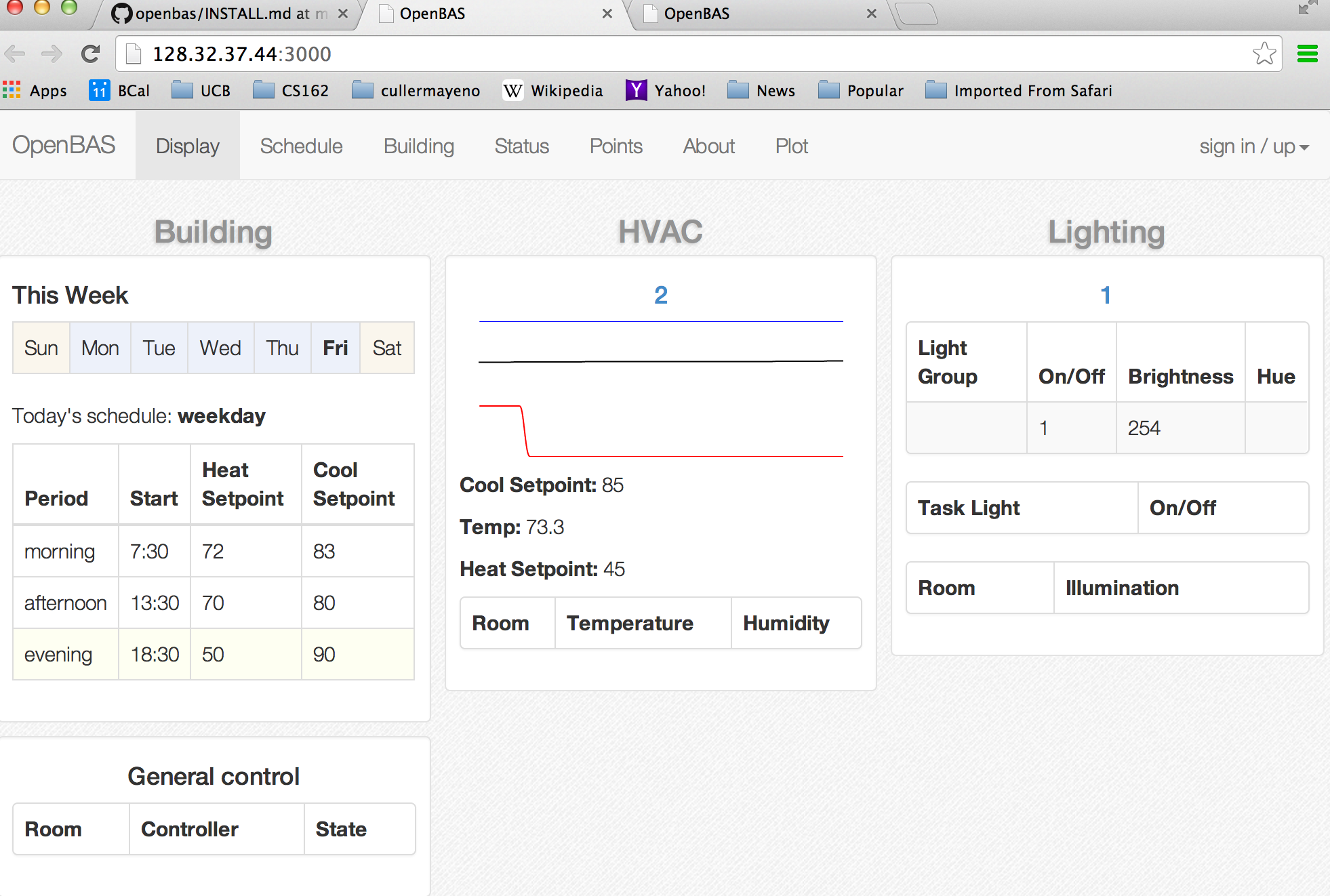
PlexiBAS: <http://128.32.37.44:3000/>

Is a lab benchtop demonstration that is a two-room building of about a cubic meter in the 410 Soda Laboratory. It will be evolving as we approach the final demo in October.

CIEE: <http://169.229.141.11:5900>

Is an at scale medium commercial demonstration in the CIEE facility.

The following shows a screenshot of the PlexiBAS benchtop OpenBAS application.



The Display application tab provide a dashboard view of integrated whole-building HVAC, Lighting, and General Controllers. The building section gives a snapshot of the current week’s schedule and whole building level monitoring. For example, if the building has a electric metering device it will appear in this panel.

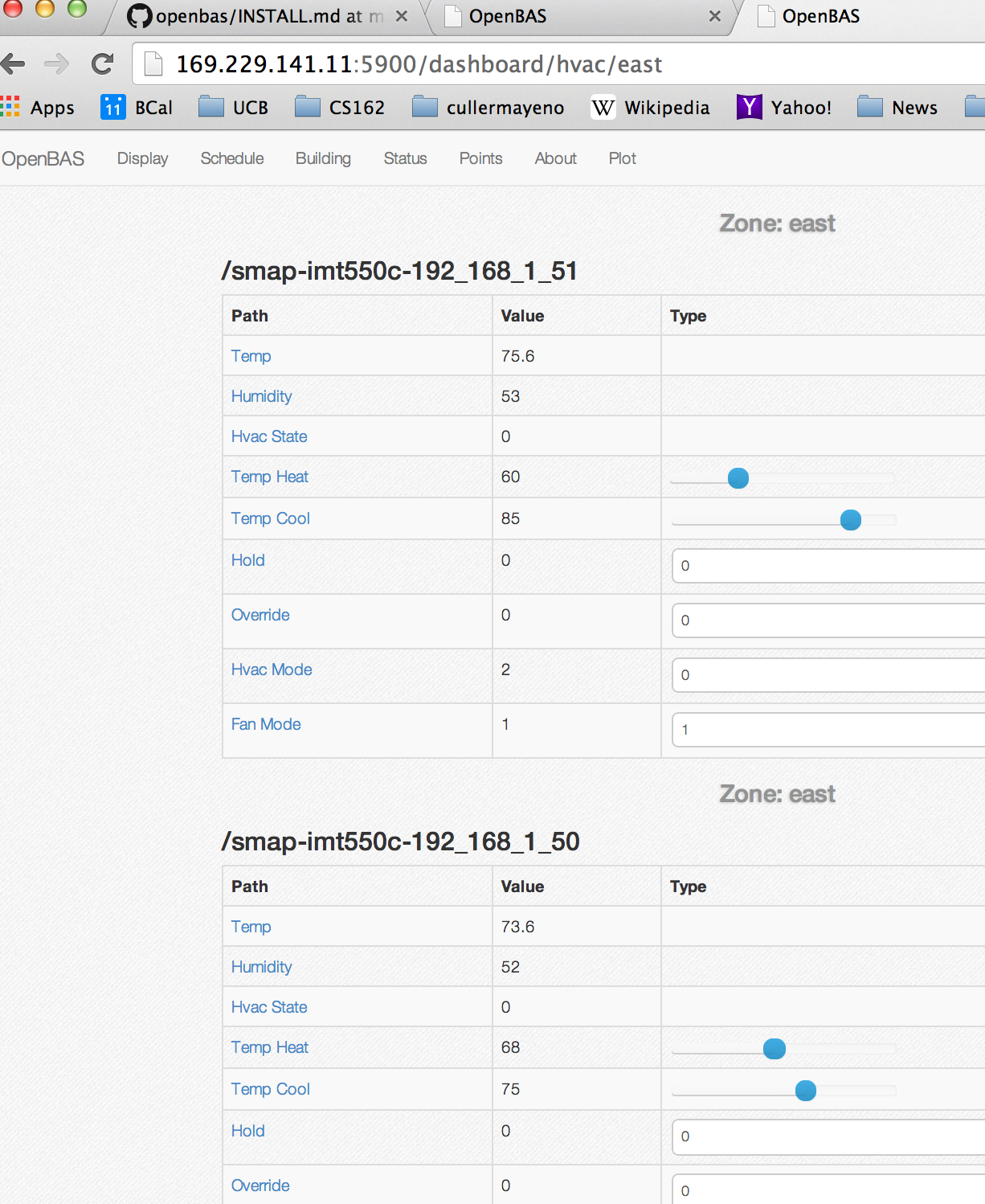
The HVAC section shows current setpoints and temperature readings for each HVAC zone in the building along with sparkline graphs to indicate the trends. If additional room-by-room sensors are deployed, these show up within the HVAC zone encompassing the room. Here we have a single zone; the CIEE deployment has multiple zones and environmental sensors in multiple rooms per zone.

The Lighting panel is a corresponding display for each of the lighting zones and various task lighting. A general control panel, which is empty in the screen shot is provided as well.

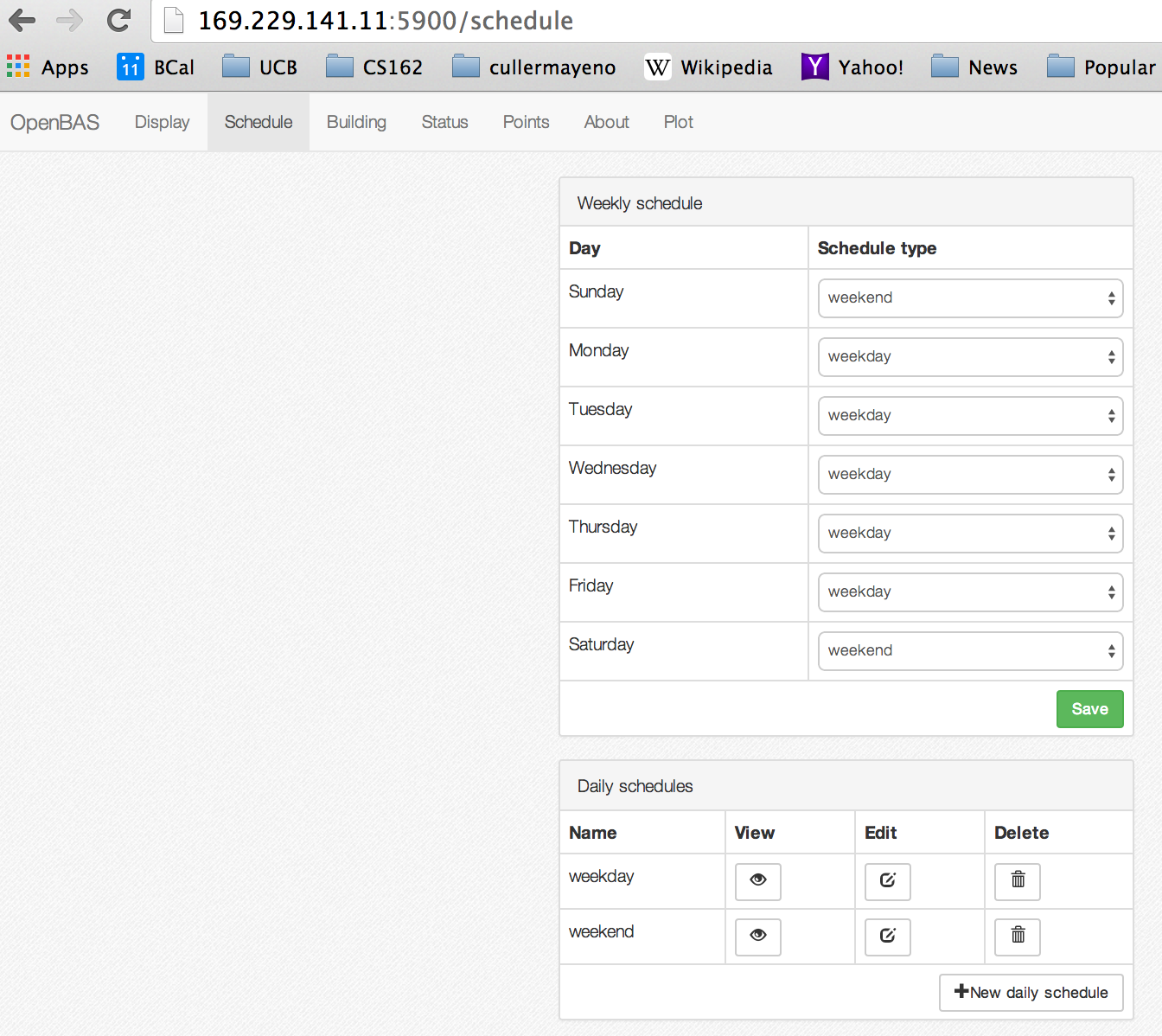
All of what appears in the display page is automatically generated from the basic Building profile and the discovery process.

Clicking on a zone drills down into the zone. For example in the CIEE deployment clicking on the East zone (<http://169.229.141.11:5900/dashboard/hvac/east>) provide a direct interface to each of the thermostats in that zone, as shown below. These are also automatically constructed from the discovery process, upon recognizing the thermostat and loading its driver. It shows the value of each of the monitoring points on the device and provides sliders and pull-down menus the control points, i.e., actuators.

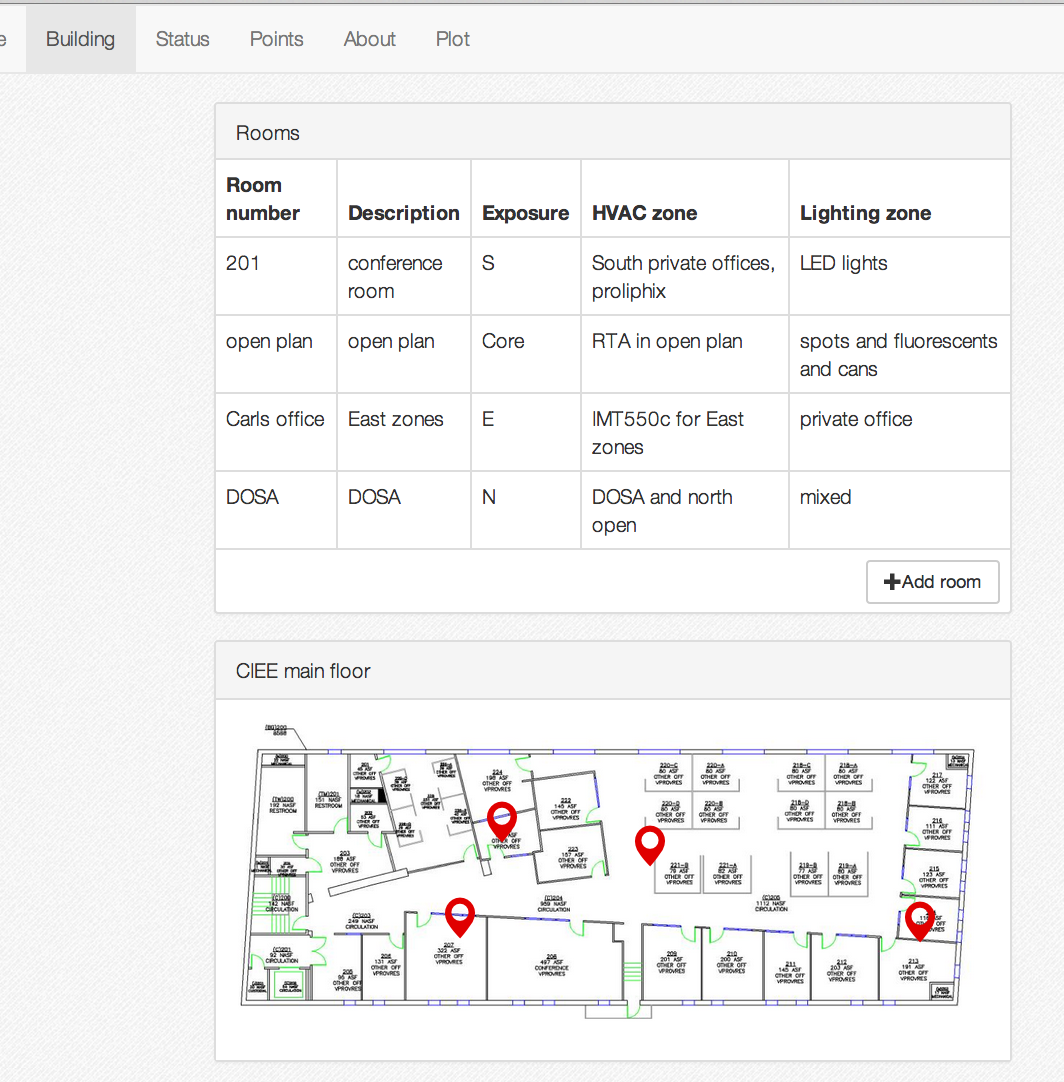
Clicking on any of these points drills down further to permit trending, or rather analysis and visualization of the historical data associated with the point.



The schedule tab allows building operators to construct as many daily schedules as they choose to employ and to assign schedules to the days of the week. For example, CIEE is initially using two periods in each day and two schedules, weekend and weekday.



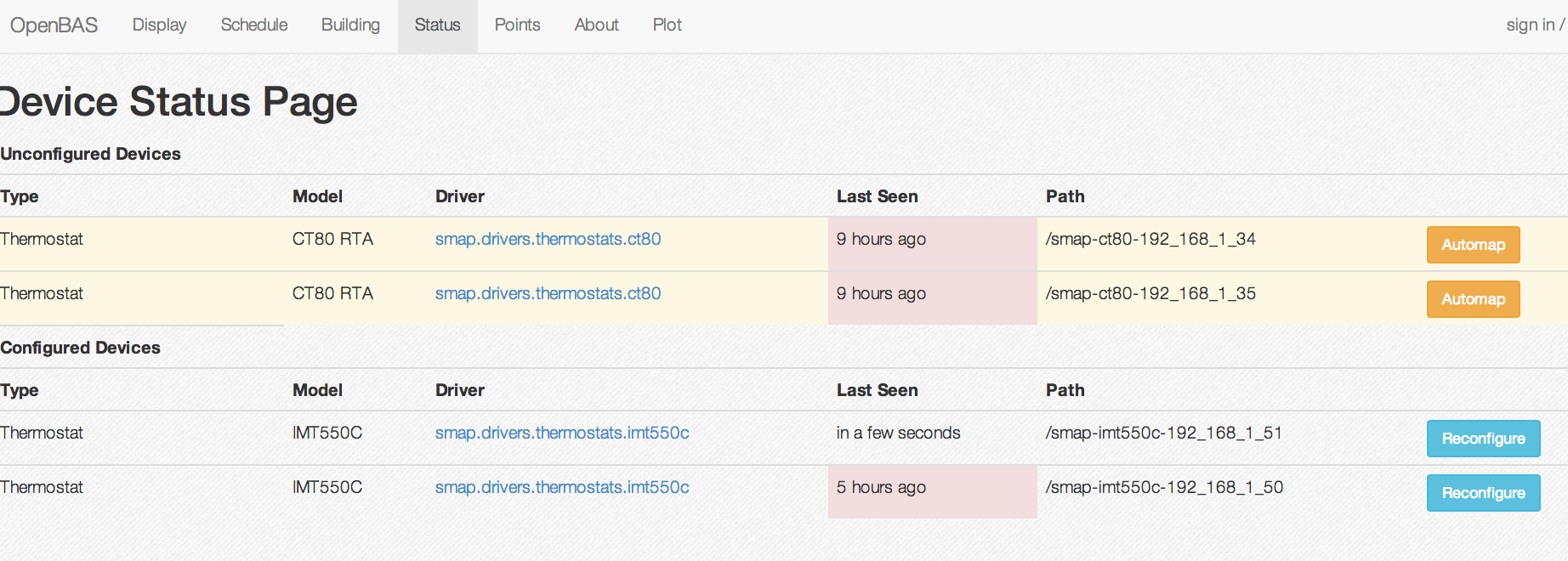
The building tab provides a way for the building operator to describe the facility in their own terms – describing rooms and associating them with HVAC and Lighting Zones. The discovery process allows devices to be associated with the zones they control or to which they provide additional monitoring inputs. An optional floor plan can be included to support spatial visualization, which will continue to be developed as OpenBAS is developed further.



The Status page reports on the status of each device in the OpenBAS deployment. This page provide the facilities inventory view, rather than the environmental comfort or spatial views presented above. For each device, it specifies the type, the model, and a link to actual code of the driver that is loaded for that device. And it provides various status information, including how long it has been since the device last reported (here showing a period during debugging where things were intentionally disconnected).

The status page is intrinsically related to the discovery and automaping process. When a new device is plugged into the openBAS building LAN, it will be noticed by the OpenBAS discovery service. The discovery service runs an identification process to determine the particular type and model of device and to load a driver for it. It creates an initial configuration for it and makes it another OpenBAS sMAP web service.

The discovery process propagates all the way up to the status page, where the device appears in the list as ready for automaping.



Clicking on the automap button provide a dialog box that allows the device to be associated with its particular role in the building profile. For example, for a themostat it recognizes that it should be associated with an HVAC zone and presents the HVAC zones for simple selection. Additional information, such as the room, can be provided.

The reconfigure button is used if previously mapped devices are in some way reconfigured.

In further OpenBAS developments analytics can be incorporated to verify aspects of the device to zone relationship.

The building profile, discovery, and automapping processes build the Metadata representation of the entire deployment. This Metadata is described in

<https://github.com/SoftwareDefinedBuildings/openbas/blob/master/METADATA.md>

The Metadata is represented as sMAP .ini files within the particular installation. All of the application is automatically created from this Metadata. So a thermostat is discovered it is associated with a zone. A new HVAC zone panel is automatically created on the display space. A drill-down page is constructed automatically for the particular interface for that device. It becomes part of the status and points pages. All of this becomes accessible through the OpenBAS API allowing third party software to utilize it.

Other parts of the application provide access to data.

The points tab provides a simple reporting of all points in all devices in the deployment.

The plot tab is used to explore the data that streams into OpenBAS for all of the devices.

The entire OpenBAS application, which is written in meteor node.js server-side and client-side javascript environment. If many clients were accessing the same OpenBAS installation, there would all be consistent – i.e., they would see each others actions simultaneously, as well as any new changes in the actual building facility.

The application provides a straightforward security model allowing users to have accounts with various privileges. Typically, displays in the physical space might want to provide visualization of OpenBAS setting and data, but only allow changes or actuation under a login with the appropriate rights and credentials.