What's included in the HomeOS kit (and some tips on using it)

2/21/2011

HomeOS is an experimental operating system for the home which focuses on providing centralized control of connected devices in the home, useful programming abstractions for developers, and allows for the easy addition of new devices and application functionality to the home environment.

This document explains what application and driver modules are included in the kit and how to use them in your setup. It is not a programming guide; that would be programming-howto.docx. Because it might be a little out of date, your most up-to-date source of information is the source code itself ☺

Contents

Drivers

DriverAxisCamera: for IP cameras made by Axis Communications

This driver fetches the feed from Axis cameras and broadcasts the new images to all subscribed modules. It also provides operations to remotely control the camera (e.g., pan and zoom). Not all Axis cameras are capable of being remotely controlled.

The driver takes three arguments: 1) the IP address of the camera; 2) the username to access the camera; and 3) the password. Use the utility provided by Axis to discover the IP and to set/change the username and password.

We have tested this driver on models M1011 and 213 PTZ, but the driver should be compatible with other models as well.

DriverDlnaDmr: for DLNA media renderers

This driver exports the services of a DLNA-based media renderer (e.g., a DLNA-capable) TV.

It takes one argument: a substring of the (DNS) name of the DLNA device. If multiple devices match, the first one is used. The driver can be modified to search on other aspects of the device (e.g., IP address) or to do exact matches. See the function FindNetworkDmr() in DriverDlnaDmr.cs.

We have tested this driver against Windows 7 computers and a Sony TV, but it should work with other DLNA devices as well. See below on converting a Windows 7 computer into a DLNA device.

DriverDlnaDms: for DLNA media servers

This driver exports the services of a DLNA-based media server (e.g., a media storage device).

It takes one argument: a substring of the (DNS) name of the DLNA device. If multiple devices match, the first one is used. The driver can be modified to search on other aspects of the device (e.g., IP address) or to do exact matches. See the function FindNetworkDms() in DriverDlnaDms.cs.

We have tested this driver against Windows 7 computers, but it should work with other DLNA devices as well. See below on converting a Windows 7 computer into a DLNA device.

DriverImgRec: virtual device for face recognition

This driver is not yet ready for external consumption. If you need it, contact us.

DriverNotifications: virtual device for sending notifications over email and SMS

This driver offers a virtual device that other modules can use to send notifications over email or SMS. SMSs are sent using the email-to-SMS gateways provided by cellular carriers. Look up your carrier's email address at http://www.makeuseof.com/tag/email-to-sms/

It takes five arguments: 1) the SMTP server to use for sending email; 2) the SMTP username; 3) the SMTP password; 4) the email address to which notifications should be sent; and 5) the email address that corresponds to SMS.

When using this driver, make sure that your SMTP allows automated programs to send emails through the account that you using. We use smtp.live.com, which lets you do that once you prove that you are human. (Other Webmail servers also let you do that.) An informational error message is printed to the console when sending email through smtp.live.com is failing due to this error.

DriverWebCam: for Webcams

This driver fetches the feed from Webcams connected to your computer (that runs HomeOS) and broadcasts the images to all subscribed modules.

The driver takes one argument: a substring of the camera name that you want to use (in case multiple Webcams are attached). So, if your camera's name is "Microsoft LifeCam Vx-7000," you can use lifecam as the argument. To discover the list of available cameras on your computer, run the driver with empty string as an argument, and the driver will print the list on the console. Another way to discover camera names is specified in the "Testing your Webcam" section below.

Only one program can use the Webcam at a time. When you use this driver, no other program should be using the Webcam.

We have tested this driver on Microsoft LifeCam series and Axis 60380 Microcam, but it should work with other types of Webcams as well.

DriverZwaveZensys: for Z-Wave devices

This driver uses the SDK provided by Zensys (now Sigma Designs) to control Z-Wave devices. It queries the Z-Wave controller for all registered devices, and exports a port for each device. The exported Role depends on the type of the device.

It does not take any argument. See below for how to get started with using Z-Wave devices with HomeOS.

The Z-Wave devices that it recognizes thus far include dimmers and sensors, but it is easy to extend to other devices. Let us know if there are particular devices that you are interested in.

Applications

AppCamera: displays camera output

This application subscribes to image feeds from all cameras in the home, and displays them on its UI.

It comes with three types of UIs. The first is WPF-based local UI that pops up on the computer where HomeOS is run. The second is a Silverlight-based UI intended for use within a browser. It can be accessed from http://<ip-address>:<port>/uihtml?<friendly-name-of-app>. The port is configured in Globals.cs (InfoServiceAddress); it current value is 51430. The third is a Windows Phone application.

AppDoorNotifier: sends notifications when door/window sensors are triggered

This is an example of a security application. It monitors a switch (e.g., a doorbell) and sensor (e.g., a door sensor) and sends notifications when the switch is toggled, the sensor indicates open, and the time of the day is between a configured time window. When email notifications are used, the emails contain images from all cameras in the home. It uses DriverNotifications to send the notifications.

The application takes five arguments: 1) the friendly name of the switch to monitor; 2) the friendly name of the sensor to monitor; 3) how to send notifications (email, sms, or both); 4) the start time of the window in which notifications should be sent; and 5) the end time of the window.

The application includes the same three types UIs as AppCamera.

AppSwitchMultiLevel: for controlling a dimmer

This is a simple application to read or control the level of a dimmer.

It takes as its only argument the friendly name of the dimmer.

It comes with two types of UIs: a local UI and a browser-based UI.

AppBenchmarker: for benchmarking system performance

This application is for our performance testing purposes.

AppBrowser: for browsing devices in the house -- Obsolete

This application is now obsolete

AppDlna: for playing any media on any renderer

This application demonstrates the use of DLNA drivers. It compiles the list of media available on all DMS (server) ports registered with HomeOS, and then lets users pick any registered DMR on which to play that media.

AppDlnaSwitching: for playing media where presence is detected

This application implements the classic smart home scenario in which music is played where presence is detected. It uses dimmers as the proxy for presence. When lights are turned brighter than a certain level in a room, music starts playing in that room. If later lights are brightened in another room, music starts

playing there as well from the same point in the track. The music stops when the brightness level reduces.

AppDoorUnlock: turns on lights based on audio and camera input

When an audio command is received from a remote device, it takes an image from the camera. If the image matches a configured user and the audio matches the user's password, lights are turned on (as a proxy to a lock being unlocked). The code in this sample is out of date.

AppDummy: a "hello world" application to play with HomeOS basics

This module exports a port with Role *dummy* and two operations *echo* and *echosub*. If it finds another *dummy* port registered, it invokes that port's *echo* operation periodically. It also subscribes to the other port's *echosub* operation and sends periodic notifications to modules that subscribe to its own *echosub* operation.

AppImgRec: a sample application that use DriverImgRec

This is a sample application that shows how to use DriverImgRec.

AppSimpleDlna: a sample application for media rendering

This application plays media from the specified media server to the specified media rendered. It takes as arguments substrings of the server and renderer names.

AppWinMoAudio: receives wav files from a remote device

This application is now obsolete.

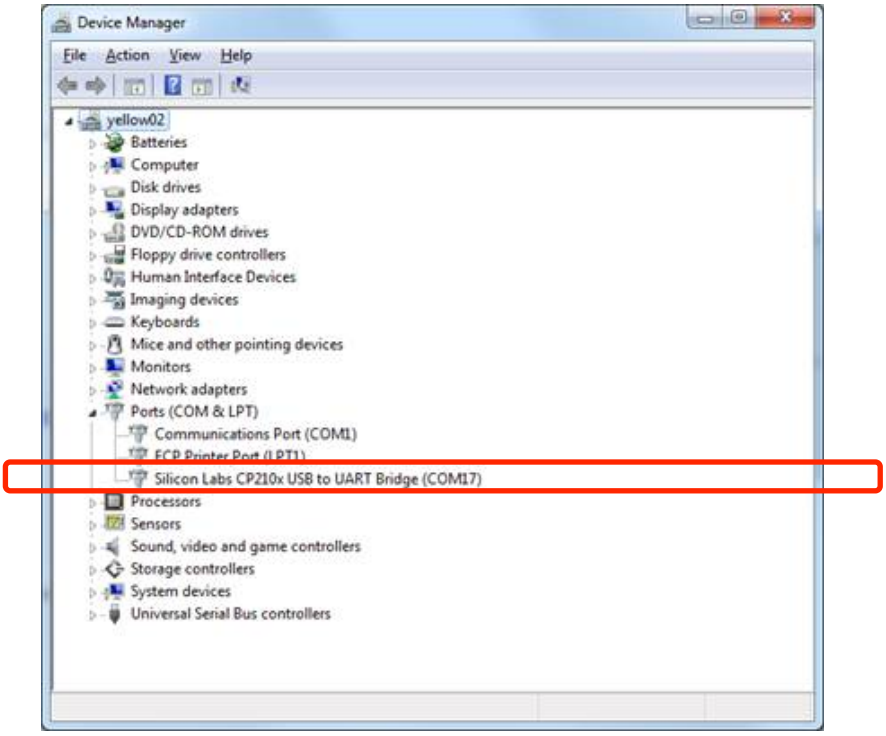
Tips and tricks

Getting started with Z-Wave

To get started with using Z-Wave in your project, in addition to the devices that you want to use, you'll need to get a USB-based Z-Wave controller. We use the one from Aeon labs (http://www.aeon-labs.com/site/products/view/2/).

To get the USB stick working:

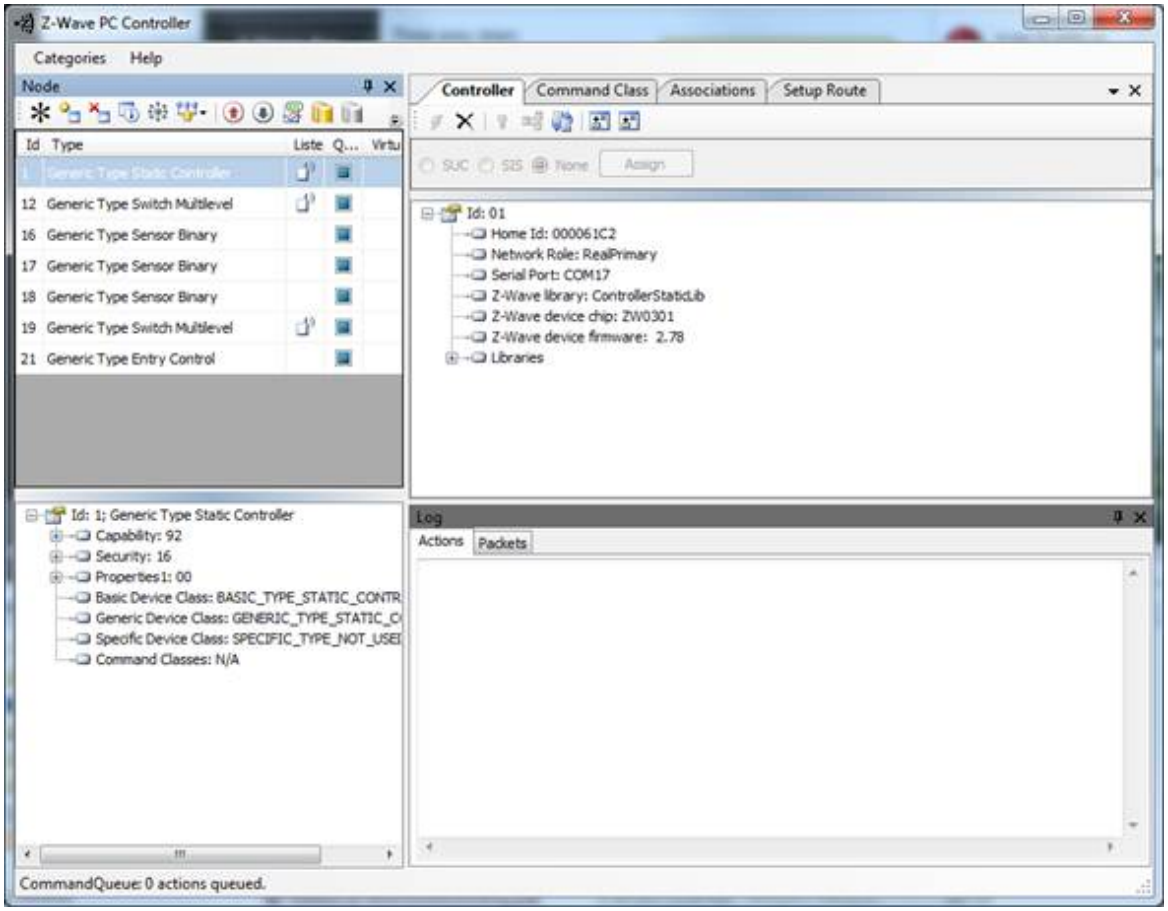
1. Insert it into your computer
2. If Windows does not automatically install the driver for the stick, install the one provided by your stick vendor.
3. Upon install, you should see an entry that suggests a "USB to UART (serial) Bridge" under Ports (COM and LPT).



Before you can use your devices, you need to pair them with the stick. The easiest way to do that is using the controller that is bundled with the Z-Wave SDK. It is also included in the HomeOS distribution at homeos\Drivers\DriverZwaveZensys\libs\ZWaveController.exe. The usage guide for this utility is homeos\docs\INS10240-7 - PC based Controller User Guide.pdf

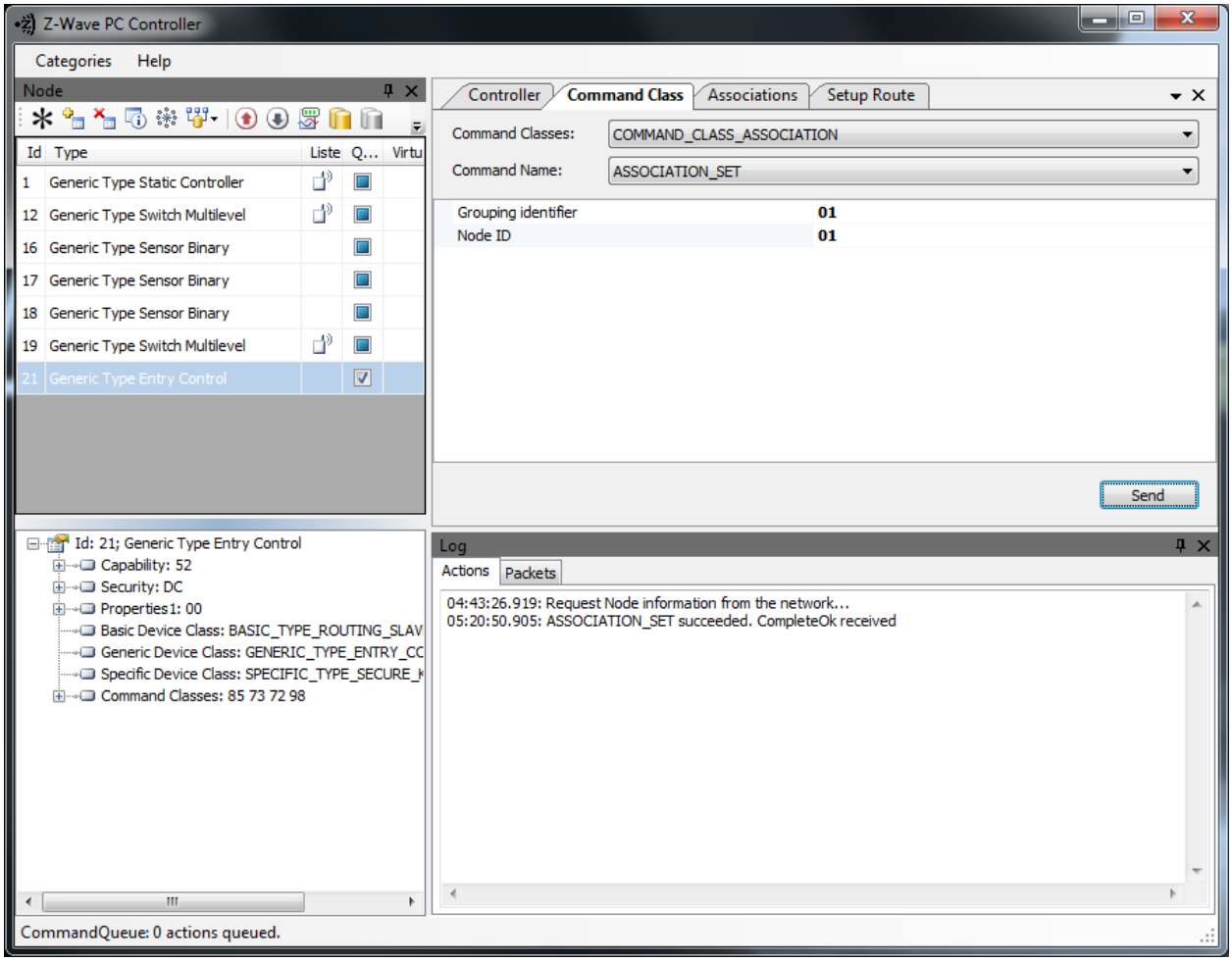
To pair your device:

1. Start ZWaveController
2. Categories (menu at the top) -> Settings
3. Select the COM port that corresponds to your stick, and then press OK
4. You should now see an entry for Generic Type Static Controller in the top left pane (where there are seven entries in the image below.)
5. Press the “Add” button on the controller (second from left under “Node”), and then push the pairing button on your device (Section 5.1.1.1 of the controller guide).
6. After successful pairing, you should see an entry for your device in the top left pane.



Some devices send notifications to the controller when certain events happen (e.g., sensor is triggered). To subscribe to these notifications, you should associate your USB controller to the device. To create an association:

1. In the top left pane, click on the device you want to associate
2. Check the Queue Overrided box for the device
3. Press the Node Info button (fourth from left button under Node), which will populate the set of Command Classes supported by the device (bottom left pane)
4. Click on the Command Class tab in the top right pane
5. Pick COMMAND_CLASS_ASSOCIATION from the Command Classes drop down menu
6. Pick ASSOCIATION_SET from the Command Name
7. Enter 1 for Grouping identifier and 1 for Node ID (which is the ID of the USB controller, not the device)
8. Click Send. The controller utility should look something like the following when this is done.
9. A test of working association is that events should be printed in the bottom right pane when they occur.



Converting a Windows 7 computer to a DLNA device

You can configure Windows 7 computers to act as DLNA devices (that is, you don't necessarily need special devices such as TVs or DVRs to build DLNA-based applications).

1. Start Windows Media Player
2. Open the menu under Stream
3. Turn on media streaming
4. Check "Allow remote control of my Player" to become a DLNA renderer
5. Check "Automatically allow devices to play my media" to become a DLNA server
6. Leave Window Media Player running when you are using the computer as a DLNA device.

You can test if the conversion worked using another DLNA device (which could also be a Windows 7 computer). The Play To feature of Windows 7 (http://windows.microsoft.com/en-US/windows7/using-the-play-to-feature-to-stream-media) uses DLNA. If that works for you, things are properly configured. Being able to play from Device A to Device B implies that A is a DLNA server and B is a renderer.

Testing your Webcam

You can test your Webcams using the binaries in the directory `homeos\Drivers\DriverWebCam\libs`.

1. Run Demo.exe.
    a. If you are on a 64-bit machine, Demo.exe may not run as the dlls that it uses are 32-bit only. If you run into this problem, use CorFlags.exe (`C:\Program Files\Microsoft SDKs\Windows\v7.0A\bin` or find it online) to set the 32bit flag:

```
> CorFlags.exe Demo.exe /32Bit+
```

2. The list of available cameras should now appear in the drop down box next to the Stop button
3. Select the camera you want to use and press Start
4. If you see the video, your Webcam is working

Note that only one program can use the Webcam at a time. When you do this, no other program should be using the Webcam.