Joe Thielen

GNU Phantom.Home 1.00

Table Of Contents:

Joe Thielen

GNU Phantom.Home 1.00

1. Introduction/Philosophy

This program was a spin-off of my Phantom.Security software which, in-turn, was a spin-off of an attempt at creating a Linux controlled robot. I figured, if I can read input from devices into the computer, why can't I control devices as well?!? So, I jury-wired my basement, constructed a very simple electronics circuit which can control a 12V DC relay and then built it. Now I control all the lights (3 sets of them) and even a vent fan from my Linux machine! I've looked on the Internet, but about the only thing I've found comparable to this is the X10 software, but, of course, that's not free nor open-sourced...

Previous | TOC | Next

Joe Thielen

GNU Phantom.Home 1.00

---

2. GNU Phantom.Home Installation

There are 4 steps to completing a simple GNU Phantom.Home setup. These are:

- 1. Make the Phantom.Home.Controller.
- 2. Make the Phantom.Home.Modules.
- 3. Plug everything in.
- 4. Configure the software.

Step 1. Make the Phantom.Home.Controller.

Included in this package is a JPEG image of the circuit diagram. It's relatively simple, and even those of you with limited electronics assembly should be able to make the Phantom.Home.Controller. If your an intermediate or advanced electronics hobbiest (or expert) and want to do something else with the circuit (like modify it to control something else) then go right on ahead!!! The parts for building the controller are readily available at a large Radio Shack near you (only the larger ones carry more electronics components anymore). The parts should be rather cheap. You should be able to build Phantom.Home.Controller for less than $25 (not counting the relays for the Modules).

Step 2. Make the Phantom.Home.Modules.

This part isn't necessary, as you can place the relays on the Controller itself, but I'd rather have 12VDC running around than 120VAC. The Modules are simply 12VDC relays mounted in a box. You have a 120VAC cord with a standard plug coming out of one end, and out of the other, you have either a wall jack or a jack on a cord. Somewhere on the box you have your 12VDC controlling wire running into it, too. All of these cords run into the relay. Now, you can hard-wire a relay, but it would be much easier to get the relay mounting piece, and hard-wire that. That way, if your relay burns up, then you can simply un-plug it and plug in a new one without soldering new connections. These modules can be placed right between the appliance you want to control (i.e. a desk lamp) and an electrical source (i.e. a wall jack). Of course, you can control your main lights with this system too, it just takes a little tinkering with your houses wiring. ***PLEASE EXERCISE EXTREME CAUTION WITH 120VAC WIRING! MAKE SURE THE CIRCUIT BREAKER IS 'OFF' FOR THE WIRES YOU INTEND TO WORK WITH. WORKING WITH LIVE 120VAC WIRING CAN BE HAZARDOUS TO YOUR HEALTH AND MAY EVEN BE FATAL! OK?***

Step 3. Plug everything in.

After you've constructed the Phantom.Home.Controller, and you Phantom.Home.Modules, you're almost ready to go! First, plug your Controller into the computer and a power source. Second, run the wiring from your Controller to your Modules. Third, plug your controlled devices into your Modules. Fourth, plug the Modules into an electrical source! That's it!

Step 4. Configure the software.

Now you've got all this neat stuff in place, what do you want to do with it? Make it work, of course! Updated compiling instructions are located in the Phantom.Home.README file, but it's usually done my running 'configure', then 'make', then 'make install'. This will copy the binaries and docs to the default directory of /home/Phantom/home. This can be changed by modifying the top-level configure.in (only for those familiar with this!). Then edit the Phantom.Devices & Phantom.Groups files (instructions for the last two can be found in sections 6 & 7).

Now you have everything you need for the Phantom.Homed program itself. But Phantom.Homed is simply an interface between your computer and the Phantom.Home.Controller itself. The way to manipulate the devices on the Controller is via some sort of interface to Phantom.Homed. Included in this dist. is the text-based interface: Phantom.Home.Console. A VRML (Virtual Reality Modeling Language) web based interface is also available, but can not be included in this distribution because of GNU conformance rules. There is currently no free or GPL'd VRML browser out there. So someone please write one so I can include Phantom.Home.VRML in this package! :)

Phantom.Home.Console is a text-based command interpreter. You give it commands like "turn off light1", and the device you specify as 'light1' in the Phantom.Devices file will turn off. "turn on lights" will turn off the devices you specify for the group 'lights' in Phantom.Groups.

NOTE: You can use multiple interfaces AT THE SAME TIME if you wish to. Phantom.Homed was designed to accommodate multiple programs. You can even have multiple copies of Phantom.Home.Console running at the same time as Phantom.Home.VRML!

Previous | TOC | Next

Joe Thielen

GNU Phantom.Home 1.00

3. Phantom.Homed Basics

The Phantom.Homed program can be used as a standard program, or you can put it in the background (just like a daemon). To run the program, simply run the executable (i.e. just type Phantom.Homed at the command line and press). Obviously, Phantom.Homed must be either in the current directory or in a directory in your $PATH. To run the program as a daemon, type in:

```
Phantom.Homed &
```

And the program will run in the background.

There are two command-line arguments to Phantom.Homed, -l and -v. Argument -v is for 'verbose'. When invoked, it will send all status information to stdout (usually the screen). Usually you don't want to invoke the -v argument when the program is running in the background, for it makes for a messy session! Argument -l is for 'logging'. As opposed to the -v option, -l will send all status information to a file named Phantom.Log in the home directory

So, to run Phantom.Homed as a daemon in the background using logging, you'd do:

```
Phantom.Homed -l &
```

And off it goes. When you run the program, it will create a file called Phantom.PID. This file contains the Process ID (PID) of the Phantom.Homed program currently running. So to stop the Phantom.Homed program, you'd type:

```
kill -9 PID
```

Where PID is the number from the PID file. A nicer way to shut down the daemon is to feed it the number '9' via it's API. To do this you must be in the homedir. Execute:

```
echo 9 > Phantom.Home.Out
```

And the program will shut down in one or two seconds. It will automatically clean up after itself using this 'nicer' method. It will delete the PID file as well as write an END line to the log file (if applicable).

Previous | **TOC** | **Next**

4. Phantom.Homed Advanced

Phantom.Homed uses a file-based API (Application Program Interface, or something :)). This means that any number of programs can use Phantom.Homed. You can even create a custom interface for Phantom.Homed!!! The API is file- based, meaning that you simply create a file with the command (or request) for the Phantom.Homed program. The name of the file is Phantom.Home.Out. It must be created in the homedir. The content of the file is a single character.

Character	Meaning
0	Tells Phantom.Homed to return the current status of the devices in their binary form (0=off, 1=on). i.e. 01011010 It puts this in the file Phantom.Home.In.
1-8	Switch the current state of device x (1 through 8).
9	Stop execution & end program.
h	Switch all devices in group 1 on.
i	Switch all devices in group 2 on.
j	Switch all devices in group 3 on.
k	Switch all devices in group 4 on.
l	Switch all devices in group 1 off.
m	Switch all devices in group 2 off.
n	Switch all devices in group 3 off.
o	Switch all devices in group 4 off.

Phantom.Homed only reads the first character of Phantom.Homed and ignores any character not indicated in the above list. So if you precede your command with a space (i.e. ' 1') it will not work. However, if you put a space behind it (i.e. '1 '), it will work, as long as the character is the first in the file. Phantom.Homed deletes Phantom.Home.Out as soon as it's finished reading it, so the same command will not be executed several times in a row. If your program makes use of the '0' (status) command, make sure it deletes the Phantom.Home.In file as soon as it's done reading it in case another program needs to use it.

API NOTE: As seen by the above, 1-8 just switch the state of the device, they don't specify a specific on or off state. So it would be wise to precede your commands with a status request first. To illustrate let's assume device one is a table lamp, and it's currently in the 'off' (0) state. If you send a '1' command via the API with the intention of turning the lamp off, you're going to end up turning it on. So it would be wise to first send a '0' request via the API to establish the status of the lamp. After reading Phantom.Home.In (returned from Phantom.Homed as a result of your '0' request), you can then verify that the lamp is already off. This reasoning behind this is that more than one program may be using Phantom.Homed at the same time, so even if your program is keeping track of device status from it's own commands, it may be wrong if another program is using Phantom.Homed too!!!

Previous | **TOC** | **Next**

Joe Thielen

GNU Phantom.Home 1.00

5. Phantom.Home.Console Basics

When you first start the GNU Phantom.Home software it displays the GPL (GNU Public License) along with messages stating that it's read the Phantom.Devices and Phantom.Groups files and the current version. It will then bring you to a prompt. This prompt is where all of the human-computer interaction takes place. Using this prompt, you give GNU Phantom.Home commands like 'Turn off X', where 'X' is on your devices named in either the Phantom.Devices or Phantom. Groups files. X could be a simple desk lamp (i.e. light or light1), or it could be your jacuzzi! For a complete list of commands accepted by Phantom.Home, see the Phantom.README file. Each new version will usually add new commands, so I'll leave that in the Phantom.README instead of in this document.

In Phantom.Home you have Devices as well as Groups of Devices. For instance, let's say you have three lights (as I do). You have a single overhead light and two table lamps. You have individual control over these items by typing in commands like 'Turn on light1' or 'Turn off light3'. But you want to be able to control all three with one command, also. No problem! Just define a group containing these devices in your Phantom.Groups file. For the sake of this discussion, let's call the group 'lights'. So, you can now type in a command like 'turn off lights', and viola! All three of the devices are switched off (even if only one or two of them is on). You can then say 'turn on lights', and all three lights come on. Ahh... the bliss of laziness.

To exit the program, just use the 'exit' or 'quit' command. You'll get a 'Thank you for using Phantom.Home.Console version xxxx' message when you leave.

Previous | **TOC** | **Next**

6. Phantom.Devices

The format of an entry in the Phantom.Devices file is this:

DEVICEx:Enabled/Disabled:Aliases

Here is an example:

DEVICE1:Enabled:light1,mainlight

This defines DEVICE number 1 (DEVICE1). It is enabled (so the software can control it), and it has two aliases: 'light1' and 'mainlight'. That means you can issue commands like 'Turn on light1' or 'Turn off mainlight', and control the same device using different aliases. Aliases should be a group of words separated by commas (,).

You can have up to eight (8) devices in the Phantom.Devices file.

Previous | **TOC** | **Next**

Joe Thielen

GNU Phantom.Home 1.00

7. Phantom.Groups

The format of an entry in the Phantom.Groups file is this:

```
GROUPx:Enabled/Disabled:Aliases:Devices
```

Here is an example:

```
GROUP1:Enabled:lights,allights:1,2,3
```

This defines GROUP number 1 (GROUP1). It is called 'lights' or 'alllignts', and contains the devices 1, 2, and 3. This means when you type in a command like 'Turn on alllights', it will turn devices 1, 2, and 3 on. Likewise, a 'turn off lights' will turn off all three devices (regardless of whether they are already off or not).

Of course, you can assign a group to a singular device also:

```
GROUP2:Enabled:tablelamp:2
```

All Aliases and Devices should be lists separated by commas.

You can have up to four (4) devices in the Phantom.Groups file.

Previous | TOC | Next

Joe Thielen

GNU Phantom.Home 1.00

8. Autolighting

As of Phantom.Home Beta0.62, autolighting is now supported! Autolighting is a way of automatically controlling your lighting via a door magnet on the door leading into a room (this can be modified for multiple doors, or for using a motion sensor).

To use the autolighting features, you must first connect one end of a normally open magnetic switch to the parallel ports ground, and the other end to pin 11. Then you have to start Phantom.Homed with the '-a' option to enable autolighting. When you're ready to leave the room & enable autolighting, type in 'im leaving' (in Phantom.Home.Console). NOTE: You still have to shut off the lights off via Phantom. The next time you enter the room, whatever device you have connected to D4 (i.e. Device 4) will turn on! To leave the light (or whatever is connected to D4) on, type in 'im here' or 'im back'. If you don't type in 'im here' or 'im back', the light will turn off in 3 minutes (180 seconds). The device and time length can be modified in the Phantom.Homed.cpp source (in the Door_Check function).

Enjoy!

Previous | **TOC** | **Next**

Joe Thielen

GNU Phantom.Home 1.00

9. Hardware/Electrical Info

The circuit diagram is relatively simple (as is intended). However, if you have a device that is more complicated than ON or OFF, then you can modify the hardware/software to your content. If you have a jacuzzi and want to control the temperature from your computer, you can go right ahead, as long as you know how :).

Please be *EXTREMELY CAREFUL* when dealing with electricity. *ESPECIALLY 120VAC* house wires. *NEVER DEAL WITH LIVE WIRES!* Flip the circuit breaker to 'OFF' before you try to deal with house lines. Please be careful!

As for the wires running between the Controller and Modules, you can manage this in one of several ways. If your sure your setup isn't going to change then you can hardwire the 12VDC line from the Controller to the Modules. However, a better approach would be to use RCA jacks and plugs. You can buy a board from Radio Shack that has 8 female jacks on one board and mount that to your Controller. Then, put a single female jack on each of your Modules. Now you can use either pre-made OR home-made RCA cables to run from your Controller to your Modules. Obviously the RCA cables should have male plugs on BOTH ends.

Previous | **TOC** | **Next**

Joe Thielen

GNU Phantom.Home 1.00

10. Thanks/Legal Stuff

I'd like to thank Riku Saikkonen <Riku.Saikkonen@hut.fi> for his mini-HOWTO on IO-Port-Programming. I used his example code almost verbatim in my early Beta versions. This is the only document I could find that smacked me in the face and said "Look idiot, it's that easy!". And it worked. Thanks!

I'd also like to thank Scott Roberts for his help with the electronics components and circuit diagram creation. He was a big help in creating the circuit. Thanks Scott!!!

I'd like to thank Rick Crelia <rcrelia@mail.RATH.PeachNet.EDU> for his input in making GNU Phantom.Home a better package! He evaluated Phantom.Home for the GNU and was very helpful in making it a GNU program. Thanks Rick!

I'd like to thank George Cragg <gcragg@compactmembrane.com> for creating the very nice circuit diagram for phantom_home_controller.jpg.

I'd like to thank Steve Cote <scote@microlegend.com> for his help in nipping the CPU resource hog issue!

Previous | **TOC** | **Next**

Joe Thielen

GNU Phantom.Home 1.00

11. Comments/Questions/Complaints/Contacting Me...

I'd love to hear anything you have to say about the software, so please drop me an e-mail at jthielen@accesstoledo.com or visit my website at www.joethielen.com!

Previous | TOC | Next