

# Develop a GPS-aware application for the Nokia N810, Part 1: Development environment

## Eclipse and Python are the right tools for the job

Skill Level: Intermediate

[Paul Ferrill \(paul@ferrill.net\)](mailto:paul@ferrill.net)  
CTO  
ATAC

16 Dec 2008

Learn how to configure a development environment targeted at the Nokia N810 Internet Tablet, including setting up Eclipse on a target development machine for the Python language.

The Nokia N810 Internet Tablet is a great little device with lots of functionality, all in a package that fits in your pocket. It comes with several standard applications and links to the Maemo Web site, from which you can download a whole host of other applications. (See the links in [Resources](#) below.)

Unfortunately, you can't always find an application with the specific functionality you need. If your search turns up empty, you're basically left with two options: either get someone to build it for you, or do it yourself. The developerWorks article "[Developing for the Nokia N810](#)" addressed the issue of building an alarm application to take advantage of the N810's alarm API. The article focused on using an existing API and the standard SDK.

This series of articles on developing a GPS-aware application shows you how to build a global positioning system (GPS)-aware application using the N810 and its built-in GPS receiver.

## Development options

Choosing the right development option directly affects the overall programming experience and ultimately your productivity. One big part of choosing development tools is the programming language to be used. By default, the Maemo development environment supports the C language, although you can add support for both C++ and Python. The Maemo site maintains a how-to document that provides instructions for adding the appropriate meta-packages for the additional languages (see [Resources](#)).

Efficient application development requires a good programming environment with code editing, compiling, and debugging tools. Another key requirement for embedded applications is the ability to run either in an emulated environment or right on the target device. Making it easier to cycle through the edit, compile, and test cycle helps foster programmer productivity.

The primary method for developing applications for the Nokia Internet Tablet in the past was to use the SDK and Scratchbox (see [Resources](#)). This is still a viable option and provides an emulation-based approach in a self-contained environment. The only real downside is the amount of effort required to deploy the solution on the device for testing. It's not really a big deal, but it does require several steps to get it working properly.

Key factors in choosing a programming language are the number of support libraries and the quantity and quality of sample code to help you get a jump start on your project. Choosing a programming language with which to develop your application comes down to comfort level. Many programmers are fluent in multiple languages but tend to have one or two they're most comfortable with.

For this project, I settled on Python for a number of reasons. The language is familiar and offers a wealth of libraries to handle most situations. Many existing Python applications already written for the Maemo environment give additional insight and instruction to aid the learning process.

## Eclipse and PluThon

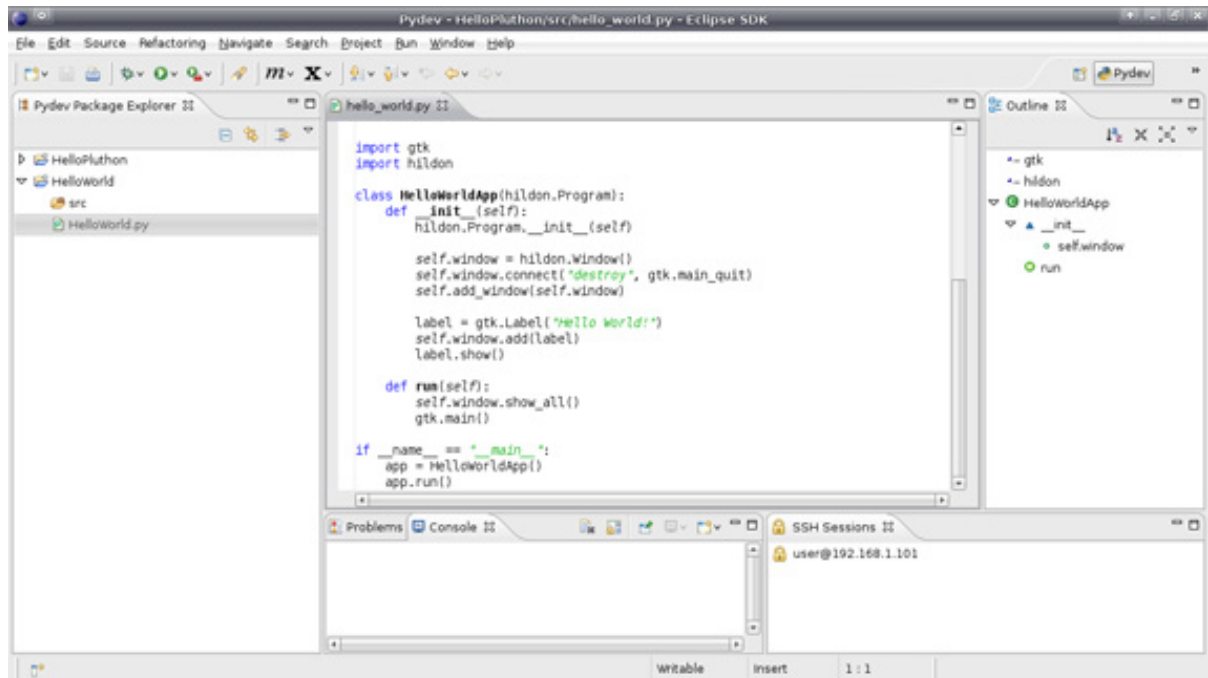
The Eclipse editor provides a solid foundation for developing applications in a multitude of languages. It's freely available and easy to install on any platform. Using Eclipse to develop in Python requires the Pydev add-on (see [Resources](#)). After you've installed Eclipse, you can use the update manager feature to include additional features.

Eclipse is fully supported for Maemo development through an integration layer described on the Maemo site. Support for the Maemo version 4.0.x Chinook and version 4.1.x Diablo SDK releases and both the N800 and N810 devices is provided. New features added since the Diablo release include a PC Connectivity tool, the ESBox plug-in for launching the Scratchbox emulator, and the PluThon plug-in for

on-device execution.

Along with the PC Connectivity tool, PluThon represents a tremendous improvement, and the combination makes it possible to link to the tablet over an SSH connection, download applications and tools, and run your application directly on the device. Getting everything configured correctly does require a bit of effort, but it's definitely worth it. Figure 1 shows the Hildon Hello World application loaded and ready to run on the device.

**Figure 1. Eclipse and PluThon**



## Setting up the development environment

One of the quickest ways to get up and running with a fully configured Maemo development environment is to download the Maemo SDK VMware Appliance from the Maemo site. And all you need after that is the VMware Player (see [Resources](#) for links to both). Launching the appliance presents a KDE-based Linux® environment with everything you need to develop for both the Nokia N800 and N810 devices.

For this project, it's important to test the application directly on the device, because the GPS functionality is required. You can perform this testing with the PluThon add-on. All installation occurs from within the Eclipse environment. For detailed instructions, consult the PluThon installation page (see [Resources](#)). You will have to follow all the instructions from within the VMware Appliance environment, as the current version doesn't have it installed by default.

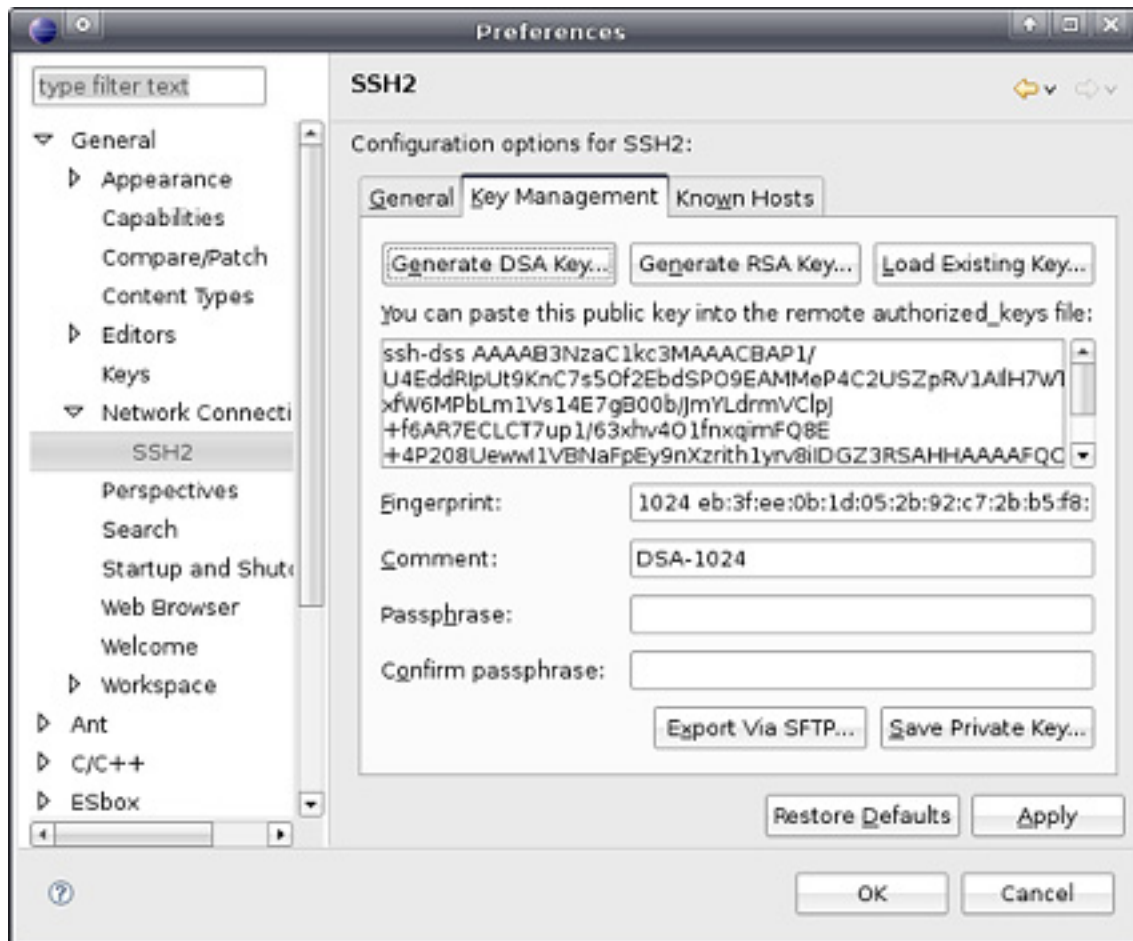
Connecting to the tablet requires a few prerequisites and a bit of configuration before it will work. First, you must install the three packages mentioned on the PluThon installation page. PluThon requires an SSH server running on the tablet to complete the connection. Installing an SSH server on the tablet requires a single command at a terminal prompt:

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

Establishing the SSH link between your development machine and the tablet requires public and private keys to eliminate the need for a password when connecting. The PC Connectivity page (see [Resources](#)) provides the details for creating keys from the command line. This process also requires that you copy the keys to specific locations on both the host and target machines. Fortunately, creating the keys from within Eclipse is much easier.

To create keys within Eclipse, click **Window > Preferences**, as shown in Figure 2. Then, click **General > Network Connections > SSH2** to display a window with three tabs. From the middle tab—**Key Management**—you can create both your public and private keys.

### Figure 2. Generating keys the easy way



At this point, click **Generate DSA Key**. Optionally, you can enter a passphrase for increased security, but it isn't required. Next, click **Save Private Key** to save the key locally. You must save the key in your home folder in the `.ssh` directory.

**Note:** I couldn't save the key from the Eclipse menu directly and had to copy the files from the command line after the fact.

The public key must be stored on the target machine to complete the public-private key combination. To store the key, click **Export Via SFTP** in the same SSH2 key window. You must know the IP address of your tablet and have the SSH server up and running. The other requirement is that you need to connect as *user* when prompted by the target site:

```
user@192.168.1.111
```

With that accomplished, repeat the process for the RSA key. When both sets of keys have been created and copied into the right directories, you should be good to go.

## Testing

With setup and configuration complete, you can test your installation to make sure everything has been configured correctly. The easiest way to do that is to create and run a simple Hello World application. To do so, launch Eclipse on your development machine (or from within the Maemo SDK VMware Appliance). Next, click **File > New > Project** to start the New Project Wizard. Then, click **PluThon > PluThon project**. Type a project name (`Hello World`), then click **Next**.

Next, create an SSH session for your project to use for communication with the target device. Doing so requires a host name or IP address and a user name. You can also choose to use a custom port or just stick with the default. Finally, click **Finish** to complete the process. You should now have a blank project ready for your code.

If you want a quick way to test your setup with an application that actually displays something on the tablet, you can use the Hildon Hello World template. Using this template requires essentially the same steps listed above, with the only change coming right before you click **Finish**. If you click **Next** after configuring the SSH session, you'll be presented with the option of creating your project based on an existing template. Selecting the Hildon Hello World template, then clicking **Next** gives you one last page with the option of linking multiple projects. You can safely ignore this and click **Finish**.

Now, you should have a project with a single Python source file named `hello_world.py` that consists of 40 lines of code, including comments, to display the text *Hello World* in the middle of the window. You can change the text if you really want to, but for this test, it isn't necessary. This is a good time to point out the usefulness of Python and the way it uses libraries to add functionality to your program. Two lines in the hello world program—`import gtk` and `import hildon`—bring in everything you need to display the text on the screen.

The last step is actually to deploy the application to the target device. Click **Run > Run As**, and then click **PluThon Application** to use the SSH connection to copy everything over to the host and run it. If everything works, you should see a white screen with the words *Hello World* (or whatever you changed it to) in the middle of the screen. Eclipse provides a Console window to display all status messages and any output that would have gone to the console if you were running the application natively from the Python command line.

## Wrapping up

With all these steps completed and the connection fully tested, you're now ready to start on serious development. In Part 2 of this article series, I'll take a look at several

different Python libraries offering both UI options and utility functions to help with the GPS access. Choosing the proper UI model is key to making the application work on the small screen.

Other potential library considerations include database access to store position information as received from the GPS, mathematical libraries for making distance calculations, and communication libraries should the need arise to transmit the current position over a network connection. You could even use freely available libraries to connect to public services like Twitter, but I leave that to you.

Choosing Python and Eclipse as the core technologies for this development makes it easy to create unit tests to add built-in testing to the end product. The Pydev Eclipse module comes with several code snippets, including one for rapidly creating a unit test framework around which to build your code. If you follow the standard Python coding conventions using Docstrings, you will automatically have a solid foundation for documentation. It's also much easier to maintain your documentation if it's embedded directly in the code to help keep everything in sync.

Deciding on a good UI library requires some prototyping and testing with potential candidate offerings. While the standard Hildon libraries provide all the widgets and buttons you'd expect to find in a typical desktop application, they aren't necessarily the best fit for a finger-friendly interface. You can scale the different objects to meet your needs, although that wouldn't be recommended for a complex interface. Other potential candidates include the Enlightenment Foundation Library (EFL), used by the Canola media player application. This solution looks nice but takes a good bit of work on your part to make it work well. It should be fun giving these different options a good workout!

# Resources

## Learn

- Learn more about the [PluThon Project](#).
- Visit the official [Maemo SDK site](#).
- Read how to [add the appropriate meta-packages for additional languages](#) on the Maemo site.
- Read how to [install PluThon on the host and target machines](#) on the PluThon installation page.
- Learn more about [IDE integration](#) with Maemo.
- Efforts are underway to assist in [developing GPS applications for the Nokia N810 device](#).
- Get started with [Pydev](#), and learn how to use it with Eclipse.
- Learn about [Scratchbox](#), including how to set it up.
- Check out "[Developing for the Nokia N810](#)" (developerWorks, August 2008), which shows you how to build an alarm application using the N810's alarm API and the standard SDK.
- In the [developerWorks Linux zone](#), find more resources for Linux developers (including developers who are [new to Linux](#)), and scan our [most popular articles and tutorials](#).
- See all [Linux tips](#) and [Linux tutorials](#) on developerWorks.
- Stay current with [developerWorks technical events and Webcasts](#).

## Get products and technologies

- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- With [IBM trial software](#), available for download directly from developerWorks, build your next development project on Linux.
- Learn about and install the [Pydev add-in](#) from the Pydev Web site.
- Get your hands on the [Maemo SDK VMware Appliance](#).
- Get started with the [PC Connectivity](#) meta-package.
- The VMware Player program is available directly from the [VMware site](#).

## Discuss

- Get involved in the [developerWorks community](#) through blogs, forums,

podcasts, and spaces.

## About the author

### Paul Ferrill

Paul Ferrill has been writing in the computer trade press for more than 20 years. He got his start writing networking reviews for *PC Magazine* on products like LANtastic and early versions of Novell Netware. Paul holds both BSEE and MSEE degrees and has written software for more computer platforms and architectures than he can remember.

## Trademarks

IBM, the IBM logo, ibm.com, DB2, developerWorks, Lotus, Rational, Tivoli, and WebSphere are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. See the current list of [IBM trademarks](#).

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.