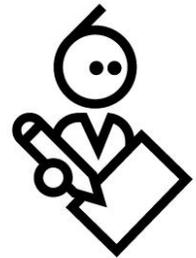




Nabduino user guide



Content

Content	2
Getting started	3
Updating the Nabduino firmware.....	3
Modifying the Nabduino demo application	3
Creating a custom HTML device driver	4
Firewall troubleshooting	5

Getting started

Your Nabduino board comes preinstalled with a demo application that among other things let you toggle the on-board LED and read the state of the button. To see this in action, connect the Nabduino board to your local network and a power supply. Once the LED on the Nabduino stops flashing the board is connected to the Nabto network and is ready to accept connections from clients. To connect to the device simply enter its ID in the address field of your web browser¹.

If this is the first time you access a Nabto device you will be redirected to the plugin download page at Nabto.com. This plugin is required before the browser can communicate with Nabto devices.

For a general overview of what uNabto is and how the uNabto protocol works please visit http://embedded.nabto.com/starterkit_unpacked/unabto_starterkit.html.

Updating the Nabduino firmware

Updates to the demo application will be posted on the Nabduino web site. To download these updates to the board please follow the steps below.

- Go to www.nabduino.com and download:
 - the newest firmware image, and
 - the Nabduino updater PC application
- Reset the device either by disconnecting and reconnecting the power source or by pressing the reset button.
- While the LED on the Nabduino board is flashing execute the Nabduino updater and supply it the name of the firmware image as its sole argument.
 - On Windows you can do this either from a command prompt or simply by dragging and dropping the firmware image file onto the NabduinoUpdater.exe file.
 - On Linux open a terminal and type “mono NabduinoUpdater.exe nabduino.hex”. This requires that Mono has been installed. See <http://mono-project.com> for more information on this.
- When the update process completes the board will automatically restart and execute the new firmware.

Modifying the Nabduino demo application

To modify the Nabduino demo application please download and install the MPLAB C18 compiler and the MPLAB X IDE from Microchips website at <http://www.microchip.com/pagehandler/en-us/family/mplabx/>.

Once MPLAB X is up and running, download and open the Nabduino project from www.nabduino.com. This download includes the uNabto framework, the application files used for the Nabduino project and the MPLAB.X

¹ Firefox and Internet Explorer are supported for Windows. Firefox is supported for Linux. iPhone and iPad requires the Nabto app.

project files.

Simply open MPLAB X, click “Open Project” from the “File” menu and navigate to the folder containing the MPLAB.X folder and double click it.

The demo application handles initialization of the uNabto framework and the hardware so the only file that has to be modified is “application.c”. This file contains the three functions that make up a Nabduino application:

- `void setup(url)`. Insert your applications start-up logic here. By default the standard HTML device driver from nabduino.net will be used for the Nabduino board but by setting the parameter `url` to a valid URL a custom device driver may be used. As an example, to use a file on the local computer write:

```
void setup(char** url)
{
    static char urlBuffer[30];
    sprintf(url, "file://c:/my_html_dd.zip");
    *url = urlBuffer;
}
```

- `void loop(void)`. This function is called from the Nabto framework main loop. Custom logic can be implemented here but be sure not to stay in this function for too long as this will prevent uNabto from doing its work properly.
- `int application_event(request, read_buffer, write_buffer)`. This function is where the queries defined for your application must be implemented. See the Nabduino demo application for an example of this function.

Again, use the Nabduino updater application to download the new firmware image to the Nabduino board. The new firmware image file will be located in the folder “MPLAB.X\dist\bootloader\production\” and will be named “MPLAB.X.production.hex”.

Creating a custom HTML device driver

To add your own queries or to change the look of the device’s web site a custom HTML device driver must be created and deployed. The easiest way to do this is to download the default Nabduino HTML device driver and use that as a template. You can download it from http://nabduino.net/html_device_driver.zip.

For information on how to edit the HTML device driver please refer to the “uNabto Developer’s Documentation” in the uNabto starter kit linked to earlier in this document.

To test it out you must have a Nabduino device request that specific HTML device driver instead of the default one. This is done by moving the new zip file to a location where your browser may reach it and then make the Nabduino board point to that location by setting the `url` parameter in the `Setup()` function as shown above. The location can be somewhere on your local computer as shown in the example above or on a web site accessible from anywhere in the world.

OBS: The browser plugin by default only checks for an updated HTML device driver every 60 seconds. This is usually fine but when developing a new HTML device driver it’s advantageous to remove this timeout period. To

do this the configuration file for the plugin - nabto_config.ini - must be modified. On Windows this file is located at "%userprofile%\AppData\LocalLow\Nabto". On Linux it is found at "~/nabto" instead.

Find the line that reads

```
#deviceDriverInstallation=automatic
```

and change it to

```
deviceDriverInstallation=always
```

and save the file and restart your browser. The HTML device driver is now checked every time an action is performed on the device web site and you will see your changes immediately.

Firewall troubleshooting

NAT traversal for easy access to devices deployed behind firewalls is a key ability of Nabto. Typically you don't have to configure anything, you just connect your device to the Internet (e.g. through a router) and it is instantly accessible from a client (smart phone or browser). However, both the client and embedded device need to be able to access the Internet - *outgoing* access to certain services through the firewall is necessary:

- UDP port 3478
- UDP port 5566
- UDP port 5591
- TCP port 5568 (not strictly necessary)

In typical residential configurations, firewalls are typically open for outgoing access. But in some corporate environments you might have to contact your systems administrator for opening for outgoing access. General support for tunneling through HTTP will be made available in Q2 2012 - as well as a base station hosted client, accessible through HTTPS.