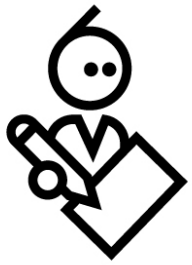
Building uNabto Demos

NABTO/001/TEN/031



Contents

1 Abstract

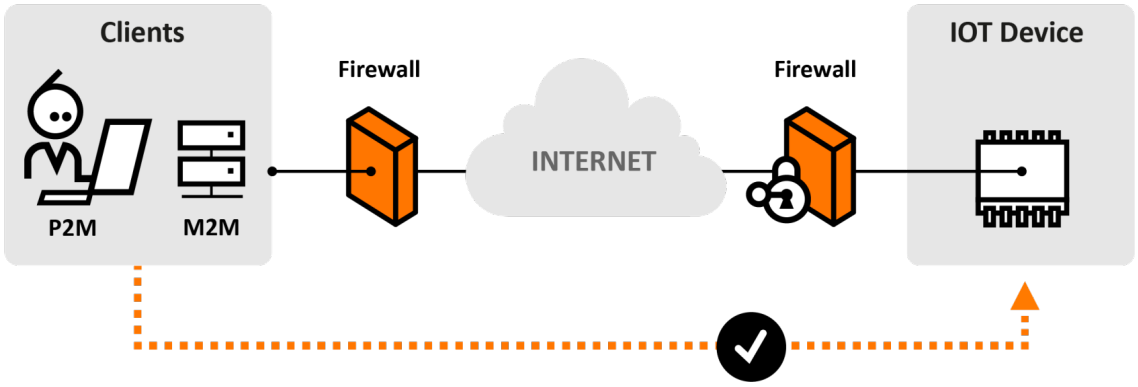
This document describes how to build and run the various demos supplied with the uNabto Device SDK.

2 Bibliography

TEN017	NABTO/001/TEN/017: uNabto SDK - Compiling Source Code
TEN023	NABTO/001/TEN/023: uNabto SDK - Writing a uNabto device application
TEN024	NABTO/001/TEN/024: uNabto SDK - Writing a uNabto HTML client application
TEN025	NABTO/001/TEN/025: uNabto SDK - Writing a Nabto API client application

3 What is Nabto?

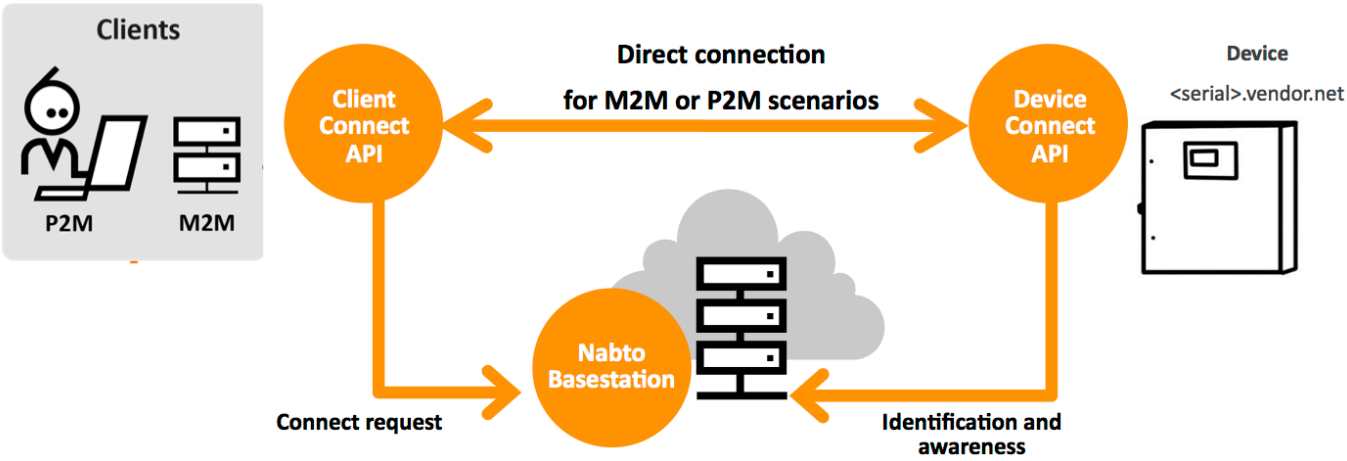
Nabto provides a full communication infrastructure to allow direct, encrypted communication between clients and even very resource limited devices – the Nabto communication platform. The platform supports direct peer-to-peer connectivity through NAT traversal. If either peer's firewall does not allow this, a transparent relay (TURN) is used to establish the connection.



- Vendor integrates Nabto's highly optimized embedded software: 10 kB flash, 2kB RAM needed (see section "Device Requirements")[1]
- Each device is given a unique identity in DNS: e.g., <serial>.<vendordomain>.net
- Ability to seamlessly and securely "call" the device and request data, issue commands or start a data stream – regardless of its location
- Both online and offline environments: Skype™ style through cloud or Bonjour™ style local communication
- Only runtime data stored in cloud: Ensures high privacy and extreme scalability
- Interactive web experience: Nabto is fully integrated into browsers for ease of use, high adoption and simple development
- Secure, transparent tunneling of data between existing client and device applications

[1] The uNabto web protocol can be implemented in less than 1 kB flash if hand-coding the implementation and bypassing the abstractions provided by the SDK

4 Components in the Nabto Platform



The Nabto platform consists of 3 components:

- Nabto **client**: Binaries supplied by Nabto, used by the customer's HTML or native application
- Nabto **device**: The uNabto SDK - an open source framework supplied by Nabto, integrated with the customer's device application
- Nabto **basestation**: Services supplied by Nabto (Nabto- or self-hosted) that mediates connections between Nabto clients and devices. Also supplies the user interface to Nabto HTML clients.

The Nabto client initiates a direct, encrypted connection to the Nabto enabled device – the Nabto basestation mediates this direct connection: The device's unique name, e.g. <serial>.vendor.net, is mapped to the IP address of the Nabto basestation – this is where devices register when online and where clients look for available devices. After connection establishment, the basestation is out of the loop – no data is stored on the basestation, it only knows about currently available Nabto enabled devices.

The client can also discover the device if located on the same LAN and communicate directly without the basestation – useful for bootstrap scenarios or for offline use.

Integrating Nabto on the customer's device is the topic of [TEN023].

The customer's client application may use the Nabto client in different ways: The customer application can be an HTML application that uses the Nabto client to retrieve JSON data in a web application – in this scenario the Nabto client is typically a web browser plugin or mobile app, hosting the customer application. The latter is distributed from the basestation to the client and is denoted an HTML device driver bundle. Writing such an application is the topic of [TEN024].

The customer's client application can also be a native (non-HTML) application, linked with a Nabto client API library. The native client application can use the same request/response mechanism to invoke the device as HTML applications do. Additionally, the native client can establish streaming data connections with the device – this is a popular way of adding seamless, secure remote access capabilities to legacy client and device applications. Native client applications are the topic of [TEN025].

5 uNabto Demos

The unabto SDK's `unabto/demo` directory contains demonstration projects for some of the currently supported platforms. This chapter gives a short introduction on some of the demos to get you started.

This document will not walk you through all the specific board and PC setups. The documentation is only focused on getting the uNabto framework up and running.

The easiest network setup is to give the development board internet access through a router. Though the platform also support other configurations e.g. where the board is directly connected to the client using local link addresses. For all configurations it is a requirement that the client has internet access the first time it tries to connect to the device, so that the HTML device driver can be fetched from the base station.

All demos (except "Weather Station" and "Nabduino") are targeted the same simple HTML device driver, "Remote Light Switch Demo", which simulates the use-case of turning the lights on and off in the clients living room from a remote location. The Weather Station demo presents simulated wind speeds and temperatures to the user, using the older TPT template approach. The Nabduino demo is a standalone project from Nabto, that has all the newest features implemented on Nabto's own development board.

The demo directory also contains a *kitchen sink demo* (demonstration of all communication functionality) between uNabto and the HTML Device Driver. This is a good place to look for a demonstration on sending or receiving e.g. a raw string or a list.

For simplicity reasons, crypto is not enabled in most of the uNabto SDK demos. Embedded certificate based authentication and encryption is currently only supported out-of-the-box in the Nabduino project.

5.1 Microsoft Windows

Located in `unabto/demo/pc_demo/unabto_win32`.

The Windows demo uses Visual Studio to build and run a small WIN32 application that simulates a light bulb in a prompt.

5.1.1 Requirements

- PC running Microsoft Windows XP or newer.
- Microsoft Visual Studio 2005 or newer.

5.1.2 How-To

1. Install Microsoft Visual Studio 2005 or newer.
2. Open `unabto/demo/pc_demo/unabto_win32/unabto_win32.sln` in Visual Studio (convert the project solution if a newer version then 2005 is being used).

3. Change the command arguments to match your device id and cryptographic key obtained through https://portal.nabto.com, for instance: `-d mydevice.demo.nab.to -s -k 720b825ba2c24cb65a1c140662aa99ab`
4. Build and run the project.
5. The uNabto stub is now running and can be accessed from any browser with the plugin by visiting its ID.

Note! Visual Studio stores by default the build executable as `pc_demo/unabto_win32/Debug/unabto_win32`, which can be run directly with arguments in the command prompt. Use "`unabto_win32.exe -h`" to see which arguments are valid.

Note! Older versions of Visual Studio have been seen to throw a "conversion to COFF" error when building the uNabto demo. To fix this disable incremental linking by going to the following menu:

```
Project Properties
   -> Configuration Properties
       -> Linker (General)
          -> Enable Incremental Linking -> "No (/INCREMENTAL:NO)"
```

5.2 Linux (General)

Located in `unabto/demo/pc_demo/unabto_unix`.

The Unix demo uses `CMake` and `make` to build for almost all unix platforms, including Linux and Mac OSX. The demo simulates a light bulb by printing it's state in the console and uses the same source files as the Windows demo.

5.2.1 Requirements

- PC or device running a unix distribution.
- CMake and make.

5.2.2 How-To

1. Install CMake 2.8 or newer.
2. Install "`build-essential`".
3. Install "`libssl-dev`".
4. Run "`cmake .`" in the `unabto_unix` directory.
5. Run "`make`" to make the executable.
6. Use "`./unabto_unix -h`" to see the available optional settings.

7. Run the newly made unabto_unix with arguments that match your device id and cryptographic key obtained through https://portal.nabto.com, for instance: `./unabto_unix -d mydevice.demo.nab.to -s -k 720b825ba2c24cb65a1c140662aa99ab`
8. The uNabto stub is now running and can be accessed from any browser with the plugin by visiting its ID.

Source files and settings are located in `unabto/demo/pc_demo/src`.

5.3 HTML-DD Demo

The HTML Device Driver demo shows how to use the different uNabto queries. It demonstrates how to send and receive multiple integers, raws and deep lists of data.

This project is only directly buildable on a unix platform, but the uNabto application_event logic is the same for all platforms.

5.3.1 How-To

1. Go to `unabto/demo/html_demo`.
2. Run "`cmake .`" and "`make`".
3. Run the resulting binary with command arguments that matches your device id and cryptographic key obtained through https://portal.nabto.com, for instance: `./html_demo -d mydevice.demo.nab.to -s -k 720b825ba2c24cb65a1c140662aa99ab`
4. Navigate to `mydevice.demo.nab.to` using your browser.

5.4 Raspberry Pi

The Raspberry Pi demo uses the normal unix version of nabto to run on the ARM processor. It demonstrates toggling the onboard ACT LED by using system commands that manipulate `/sys/class/leds/led0/brightness`.

The following is tested on a 2011 Raspberry Pi board running wheezy-raspbian with a TrendNet WiFi dongle. It requires that the board has Internet, SSH access and CMake is working.

5.4.1 Requirements

- Raspberry Pi board
- Ethernet cable or a WiFi dongle

5.4.2 How-To

1. Copy the uNabto SDK to the device (`scp -r unabto_sdk pi@<ip>`).

2. SSH into the Raspberry Pi (`ssh pi@<ip>`).
3. cd into `unabto/demo/pc_demo/unabto_unix`.
4. Run "`cmake .`" and "`make`".
5. Run the newly made unabto_unix with arguments that match your device id and cryptographic key obtained through https://portal.nabto.com, for instance: `./unabto_unix -d mydevice.demo.nab.to -s -k 720b825ba2c24cb65a1c140662aa99ab`

Note! Start the application with "`nohup ./unabto_unix … &`" to run it in the background. Afterwards you can close the SSH connection and the nabto application will still be running.

5.5 Nabduino

Located in `unabto/demo/nabduino`.

The Nabduino board is Nabto's own development platform. The Nabduino is using a Microchip PIC18 processor and the implementation is therefore somewhat similar to the `pic_demo` (see Microchip PIC).

The concept is that the Nabduino application can be updated over an ethernet connection, and includes all the newest features of uNabto. See Nabduino homepage for further details on how it works and what it is capable of.

The Nabduino uses a separate HTML device driver from the other demos.

5.5.1 Requirements

- Nabduino board.
- Microchips MPLAB IDE, compiler and TCP/IP Stack (All three packages are available in free versions).
- Nabto Updater.

5.6 Microchip PIC

Located in `unabto/demo/pic_demo`.

The Microchip PIC demo project demonstrates toggling an LED remotely. It is tested on the PICDEM.net 2 development board, PIC Internet Radio and the PIC32 Ethernet kit.

The demo project also contains a simple integration with FreeRTOS for the PIC32 Ethernet board. For this port to work, it is necessary to include the FreeRTOS source.

5.6.1 Requirements

- A PIC development board with ethernet.
- ICD 3 In-circuit Debugger for the PIC18 demos (the PIC32 Ethernet kit has an onboard debugger).

- MPLAB IDE, compiler and TCP/IP Stack (All three packages are available in free versions).

The uNabto PIC demos should be compatible with Microchips newest software releases, and has been tested thoroughly with the following configuration:

- MPLAB X IDE v1.41, with the C18 v3.43 and XC32 v1.11 compilers.
- TCP/IP Stack v5.42.
- FreeRTOS v7.2.0.

5.6.2 How-To

1. Download and install the MPLAB X IDE.
2. Download and install the C18 compiler for the PIC18 projects or XC32 compiler for the PIC32 projects, from the same page.
3. Download and unpack the Microchip TCPIP Stack and make a link to it in the `unabto/external` directory (further instructions are located in the directory).
4. Do the same procedure with the FreeRTOS source if needed.
5. Connect all the hardware.
6. Open the `unabto/demo/pic_demo/MPLAB.X` project in MPLAB.

Note! On the PICDEM.net 2 board it is important to use the network interface closest to the onboard LCD since uNabto use the integrated ethernet controller, not the onboard ENC28J60.

The project main is located in `main.c`. This is where the MAC-address and unique ID is set. `application.c` contains the application logic where the functionality is implemented and can be changed to the users needs.

In `unabto_config.h` the uNabto related settings can be changed, otherwise they are set to default values according to `unabto/src/unabto_config_defaults.h`.

5.7 Arduino

The uNabto demo for the Arduino platform demonstrates toggling an LED remotely, but is easily extended to the client's specific needs.

5.7.1 Requirements

- Arduino board (it has been tested on Duemilanove and Uno)
- Arduino Ethernet shield (tested on Wiznet)
- Arduino software
- LED to do the blinking.

5.7.2 How-To

1. Copy the `unabto/demo/arduino/Nabto` directory from the SDK to the Arduino `libraries` directory. On Windows it is normally located in `My Documents\Arduino\libraries\` and on Mac/Linux it is located in `~/Documents/Arduino/libraries/`.
2. Open Arduino.
3. Open `Files → Examples → Nabto → Demo`.
4. Type in the MAC-address located on the bottom of the Ethernet shield.
5. Specify an unique ID for the Arduino demo, e.g `<macaddress>.sdk.u.nabto.net`.
6. Connect the LED to pin `A0` (anode) and `ground` (cathode).
7. Click `on Tools → Board` and make sure you have the right board chosen.
8. Click upload.
9. Open Firefox or Internet Explorer and type in the ID.

5.8 Renesas RL78

Located in `unabto/contrib/renesas_rl78`.

The Renesas project demonstrates running uNabto on a Renesas RL78 WiFi Evaluation board (YRDKRL78G14). Warning: This project is not actively maintained, this means that the project file probably needs to be updated before the project are able to compile and run.

The demo shows accelerometer, temperature, light and potentiometer graphs in a custom HTML-DD named `<ID>.renesas.u.nabto.net`.

5.8.1 Requirements

- The Renesas RL78 Evaluation board with programming interface.
- IAR Embedded Workbench IDE (available from http://www.am.renesas.com/products/mpumcu/rl78/rl78g1x/rl78g14/soft_tools_index.jsp)

5.8.2 How-To

1. Download and install the IAR Embedded Workbench IDE.
2. Open the uNabto Renesas demo project located in `unabto/contrib/renesas_rl78/YRDKRL78G14/`.
3. Setup the board jumpers as descripted in the evaluation documentation.
4. Click “Compile” and “Run”.
5. Follow the instructions displayed on the Renesas board screen.

5.9 RTX41XX

Located in `unabto/demo/rtx4100`.

The RTX project demonstrates the use of Nabto over WiFi on a small embedded device - the low power RTX4100 or RTX4140 WSAB board.

In this demo the user can toggle the onboard LEDs and read button status, plus graphical readouts of temperature and accelerometer data.

Note! The programming procedure currently only works on Windows.

5.9.1 Requirements

- The WSAB RTX4100 or RTX4140 module
- The WSAB Docking Station (for programming)
- Serial cables
- AmelieSdk (RTX’s SDK)
- GCC ARM Embedded tools

5.9.2 How-To

There are two ways to get uNabto running on your RTX module.

Using uNabto source:

- Plug the RTX41XX module into the WSAB Docking Station.
- Download GCC ARM Embedded tools and AmilieSdk.
- Copy the unabto folder to `Projects/Amelie/COLApps/Apps/Nabto`.

- Go to `unabto/demo/rtx4100/Build/RTX4100_WSAB` and run ba.bat.
- On success, upload the Nabto.fws to the module with COLa Controller.

Using RTX Nabto library:

- Plug the RTX41XX module into the WSAB Docking Station.
- Download GCC ARM Embedded tools and AmilieSdk.
- Go to `Projects/Amelie/COLApps/Apps/Nabto/Build/RTX4100_WSAB` and upload the precompiled .fws file with COLa Controller.

Note! See the RTX documentation located in `AmelieSdk/v1.xxx/Documents/RTX4100_Quick_Start_Guide_Nabto.pdf` for further details on installation.

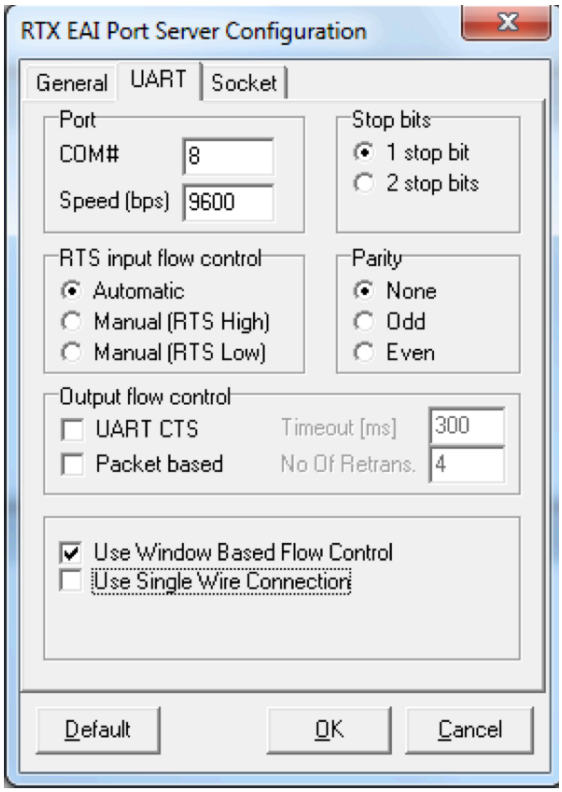


Figure 1. RTX EAI Programmer setup 1

Running the demo:

Provisining mode: . Press down switch 2 (SW2) at boot. . Connect to RTX's AP (the device's ID). . Goto 192.168.1.1. . Set AP the RTX41XX board should connect to. . After reboot the device will connect to the given AP.

Soft Access Point mode: . Press down switch 1 (SW1) at boot. . Connect to RTX41XX's AP (the device's ID).

Use Nabto: . Install the Nabto Plugin from http://download.nabto.com, or use the Nabto Android or iOS app. . Go to nabto://self to discover local devices, or type in the device's ID in your browser or app. . You are know connected to your Nabto RTX board.

5.10 Tunnel Demo

The Tunnel demo demonstrates the TCP tunnelling solution. The solution consists of a uNabto server which acts as a uNabto TCP proxy server. The other end of the tunnel is either a client built on our client api or the prebuilt Nabto Tunnel Manager Application, which can be downloaded from nabto.com. The demo is located in unabto/demo/tunnel.

The document [TEN030] available on https://nabto.com/downloads described in detail how to configure, build, run and troubleshoot the Nabto tunnels.

5.11 Stream echo demo

The stream echo demo shows a simple stream echo service. When after receiving the command echo\n it echoes all the data afterwards.

5.11.1 How-To build

The project is located in unabto/demo/stream_echo and uses CMake as project generator and build system. See the section Build projects using CMake for details about building a CMake project.

5.11.2 How-To run

The demo can be run using the stream_echo executable as a uNabto device and the StreamTerminal executable as client which can connect to the streaming echo device.

The stream_echo executable should be called as `stream_echo -d <Device ID> -s -k <Key>` with the key and device id obtained from portal.nabto.com

The StreamTerminal executable which can be found under downloads on nabto.com in the .Net demo package should be called as `StreamTerminal.exe <Device ID> --echo` when running type `echo` and press enter now everything typed should be echoed.

6 Build projects using CMake

All uNabto C projects which runs on Linux, Windows and Mac can be built with CMake as the build system. CMake is a project generator which means it can generate project files from a cross platform project description.

6.1 Prerequisites

6.1.1 Linux

- CMake
- Make
- Gcc

6.1.2 Windows

- CMake
- Visual studio 2010

6.1.3 Mac

- CMake
- Make
- Gcc/clang

6.2 How-To build a CMake project

When building a project it's recommended to create a new empty folder where the build will take place, this ensures the build will be an out of source build as opposed to an in source build which pollutes the source tree.

1. Open a terminal.
2. Create an empty folder.
3. Cd into newly created folder
4. Run cmake <path to uNabto project directory>

This will create a new project using the default generator for the current platform in use. On Linux and Mac a Makefile is generated which can be built using make. On Windows a Visual Studio project is generated which can be built using Visual Studio.