|  |  |
| --- | --- |
|  | Francotyp-Postalia |

Windows IoT – Development Environment Setup

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Version | Date | Authors | Comments/Modifications |
|  | 1.0 | 19.09.2016 | Hendryk Irmischer | initial revision |
|  |  |  |  |  |
|  |  |  |  |  |

This document describes the steps to setup a development environment to develop a Node.js application on an IoT-Device (e.g.: Raspberry Pi 3) and how to start debugging.

Table of Contents

[1. Requirements 2](#_Toc462048963)

[1.1. Hardware 2](#_Toc462048964)

[1.2. Software 2](#_Toc462048965)

[2. Preparation 2](#_Toc462048966)

[3. Proxy Setup 2](#_Toc462048967)

[4. System configuration 3](#_Toc462048968)

[5. Setup a shared folder 4](#_Toc462048969)

[6. Verify USB serial port is available 4](#_Toc462048970)

[7. Node.js Installation 5](#_Toc462048971)

[8. Deployment 5](#_Toc462048972)

[9. Debugging 6](#_Toc462048973)

# Requirements

## Hardware

* PC
* IoT Device (supported by Debian, e.g.: Raspberry Pi 3)
* FM's, Scales etc

## Software

* Raspbian Jessy Lite (from [here](https://www.raspberrypi.org/downloads/raspbian/))
* [Win32 DiskImager](https://sourceforge.net/projects/win32diskimager/)
* SSH Client (e.g.: [Kitty](http://www.heise.de/download/product/kitty-75185))

# Preparation

1. Start Win32 DiskImager
2. Select Drive with SSD
3. Select Raspbian Image
4. Press ‘Write’
5. Insert SSD into IoT Device
6. Boot device
7. Now you should find out, which hostname your device will have after first boot
8. Ping the hostname to get its IP (for raspbian jessy lite the default hostname is: raspberrypi)

C:\> ping raspberrypi

# Proxy Setup

1. Open SSH-Session with the device (via Kitty or Putty)
   1. Default username: pi
   2. Default password: raspberry
2. Edit file for general proxy settings (see [here](http://www.gtkdb.de/index_36_2111.html))

$ sudo nano /etc/profile.d/proxy.sh

1. Paste following text into file:

export http\_proxy=http://192.168.2.96:8080

export https\_proxy=http://192.168.2.96:8080

export ftp\_proxy=http://192.168.2.96:8080

export HTTP\_PROXY=$http\_proxy

export HTTPS\_PROXY=$https\_proxy

export FTP\_PROXY=$ftp\_proxy

1. Edit file for apt-get proxy settings (see [here](http://askubuntu.com/questions/89437/how-to-install-packages-with-apt-get-on-a-system-connected-via-proxy))

$ sudo nano /etc/apt/apt.conf

1. Paste following text into file:

Acquire::http::proxy "http://192.168.2.96:8080/";

Acquire::ftp::proxy "ftp://192.168.2.96:8080/";

Acquire::https::proxy "https://192.168.2.96:8080/";

1. Reopen the SSH-Session
2. Type:

env

1. HTTP\_PROXY, HTTPS\_PROXY and FTP\_PROXY should be listed here

# System configuration

1. Update source list

$ sudo aptitude update

1. Upgrade installed packages

$ sudo aptitude upgrade

1. Create a development user and add him to needed groups

$ sudo adduser devel

$ sudo usermod -aG adm,dialout,cdrom,sudo,audio,video,plugdev,games,users,input,netdev,gpio,i2c,spi devel

1. Change password of user ‘pi’ (optional):

$ passwd

1. Close current SSH-Session
2. Open new SSH-Session and login as the newly created user (e.g.: devel)
3. Modify visudo to enable environment variables for sudo commands too:

$ sudo visudo

1. In the upper part of the file append the following lines:

Defaults env\_keep += "http\_proxy"

Defaults env\_keep += "https\_proxy"

Defaults env\_keep += "HTTP\_PROXY"

Defaults env\_keep += "HTTPS\_PROXY"

1. Below the line pi ALL=(ALL) NOPASSWD: ALL paste the following line:

devel ALL=(ALL) NOPASSWD: ALL

1. Close the file
2. Run raspi-config:

$ sudo raspi-config

1. Expand Filesystem (important)
2. Internationalisation Options
   1. Change Local 🡪 de\_DE (UTF8)
   2. Change Timezone 🡪 Berlin
3. Advanced Options
   1. Hostname 🡪 set to something useful
4. Finish
5. Reboot

# Setup a shared folder

1. Navigate to your user’s home directory (e.g.: for user devel):

$ cd ~/

1. Create a project folder in devel’s home directory:

mkdir ~/navigator-web

1. Install samba

$ sudo aptitude install samba

1. Open samba config:

$ sudo nano /etc/samba/smb.conf

1. Append following text to file (Adjust sharename, comment and path to your needs):

[Sharename]

comment=WebApp Deployment Folder

path=/home/devel/navigator-web

browseable=Yes

writeable=Yes

only guest=No

create mask=0777

directory mask=0777

public=no

1. Create a samba user for the share:

$ sudo smbpasswd -a devel

1. Enter desired password for access to the shared folder
2. Restart samba service to apply changes:

$ sudo /etc/init.d/samba restart

1. Verify access to the sahred folder from dev machine:
   1. Open Explorer 🡪 [\\[device-name-or-ip](file:///\\[device-name-or-ip)]
   2. The shared folder should be visible now
   3. Map shared folder as network drive (optional)

# Verify USB serial port is available

1. List all USB devices:

$ lsusb

1. Plug in franking machine into device’s USB port and list devices again:

$ lsusb

1. A new device should have shown up:

Bus 001 Device 004: ID 2311:0502

1. Check if the device has been mounted:

$ ls -al /dev/ttyACM\*

* 1. Should show the following:

crw-rw---- 1 root dialout 166, 0 Sep 19 09:28 /dev/ttyACM0

# Node.js Installation

1. Add ‘nodesource’ to sources.list so that we can simply install and update node.js via apt-get

$ curl -sL https://deb.nodesource.com/setup\_6.x | sudo -E bash –

1. Install Node.js

$ sudo aptitude install -y nodejs

1. Verify node has been installed correctly:

$ node –v

1. Verify node has been installed correctly:

$ node

> 1 + 3

4

> We can hit Ctrl-C twice to exit the REPL and get back to the bash (shell) prompt.

1. Install typings module globally

$ sudo npm install typings --global

1. Install typescript module globally

$ sudo npm install typescript –global

# Deployment

1. Copy necessary files to deployment folder
2. Install node modules

$ npm install

1. Every native module should now be compiled correctly on the target platform
2. Run the application

$ npm start

# Debugging

1. See [here](https://github.com/Microsoft/nodejstools/wiki/Remote-Debugging-Node.js-running-on-Linux) for instructions