Tutorial – Build your own TTN front-end

Introduction

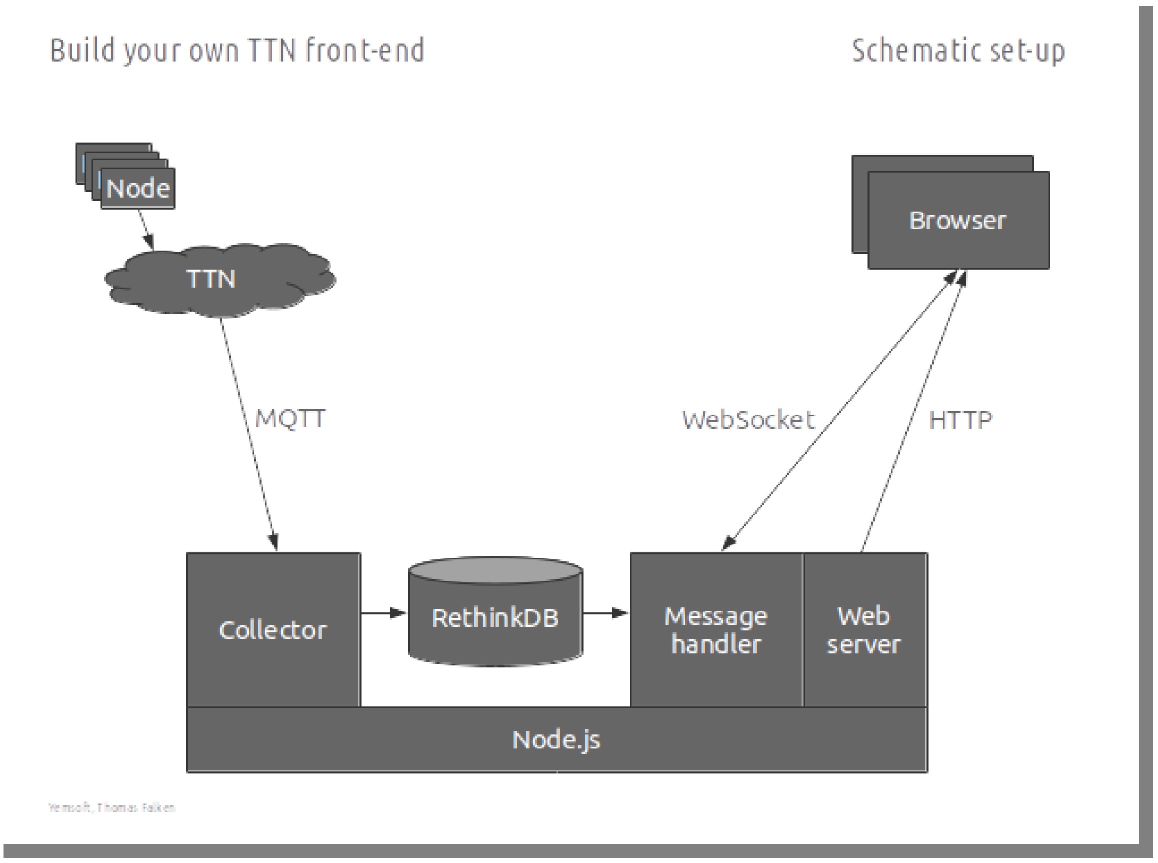
When you visit one of the workshops about building your own sensor nodes you get a great piece of hardware to get creative with. Once all works, you see the data in the TTN console. But then? What to do with all the data? Well there a quite some great things you can do with standard tools like ITTT. No real programming involved.
If you do not mind the programming, want a real customizable solution, want to be the boss of your own data, then this tutorial is a great starting point.

We will build an application that saves all node data in a database and a web front-end to visualize the existing and incoming data.

We will use scalable, modern, open source tools to make it work. So it can be the basis for a larger project. What are the ingredients:

- Node.js
  The basis under the application. Has great support by TTN.
- RethinkDB
  A NoSQL database, small footprint, scalable, easy to store JSON and we can subscribe on changes per table.
- Websockets
  The best way to get incoming node data real-time to the browser.

For me this little project was a first acquaintance with NodeJS, MQTT and the NoSQL database. Specially NodeJS was harder than I expected, but now I understand why so many people are positive about it. I hope this information helps you to get a smooth start. Don't stop after reading, pull the project and start altering it, to fit your own idea's. You will learn a lot!



Application structure

The Things Network does everything to receive data from our nodes and provides it to us in end points, (almost) no storage provided. So our first task is to listen to one of these end points and store the incoming packets. MQTT is a nice standard for this. TTN support it and we could use it also for other sources.

MQTT will give us a large JSON for each packet, next to the payload it holds a lot of details about the connection. We store the complete JSON packet and later we can filter what we really want to use.

MQTT and TTN provide not only the data packets but also data on events like our node joining the network. We also store these, but in a different table.

Once the JSON datasets are stored we can do something useful with the collected data. In the tutorial we use Arduino nodes with the BMP280 sensor that collects temperature and air pressure data. The nodes use the code from the Amsterdam April workshop. The two values (for temperature and pressure) are compresses into 4 bytes and a decoder function in the TTN console produces an additional JSON element “payload_fields: {celcius:99.9,mbar:9999}”. That is the data we are interested in for our front-end.

Tutorial – Build your own TTN front-end

Our front-end will present the temperature and pressure as two gauges. When the page loads we will hand the average values over the last 24 hours to the gauges. Whenever new node data is collected (while the page is open) the web page gets the latest values and displays them in the gauges. Some additional messages are shown in a text section.

To do so our message handler will:

- Get 24 hours of payload and average the data for the initial gauge display.
- Subscribe to the RethinkDB for changes in our data table. Whenever that event fires we forward the payload to the front-end.

Next to the basic elements (collect, store, forward) we need a web server to serve the HTML page, JS and CSS files for our web page. Node.js can provide that, closely interlinked with the websocket server.

Let’s get hands-on

The following instructions are based on Ubuntu Linux, but all tools work also under Mac and Windows. If you install the Node.JS and RethinkDB Windows version then the rest should work.

In the appendix you find a short summary of steps to get the project running on Windows. The steps are quite similar on all OS, so if you are in a hurry skip to the last page.

Prepare your environment

Project files

Get the project from GitHub. The project is located in:

https://github.com/Yemsoft-Thomas/ttn-frontend-tutorial.git

Download ZIP or use git to put in your ‘ttn-frontend-tutorial’ folder.

RethinkDB

To install RethinkDB on recent Ubuntu (16.10/17.04) use the prebuild DEB from:

http://atnnn.github.io/rethinkdb/yakkety/rethinkdb_2.3.5~0yakkety_amd64.deb

and the commands:

```
sudo apt-get install libcurl3 libprotobuf10

sudo dpkg -i rethinkdb_2.3.5~0yakkety_amd64.deb
```

Tutorial – Build your own TTN front-end

Why? There are no official sources for Yakkerty and building manually breaks on my PC. See:

```
https://github.com/rethinkdb/rethinkdb/issues/6172
```

If you start rethinkdb for the first time it will create a DB data folder under your current folder. The tutorial runs RethinkDB from the main project folder, so our collected data goes into rethinkdb_data folder. *Don't include your local data folder in the git project!*

The RethinkDB is empty, running the "GetNodeDataInDB.js" for the first time will create a database called "ttn" and two tables. You can drop the tables / db and at next run a new / empty dataset will be build.

Node.js

Installation of Node.js is strait forward, just use apt. The tutorial uses the 6.10 LTS version.

The project uses the following node libraries:

- TTN client library <ttn@2.1.1>
- RethinkDB client <rethinkdb@2.3.3>
- Express framework <express@4.15.2>
- Socket.io <socket.io@1.7.3>

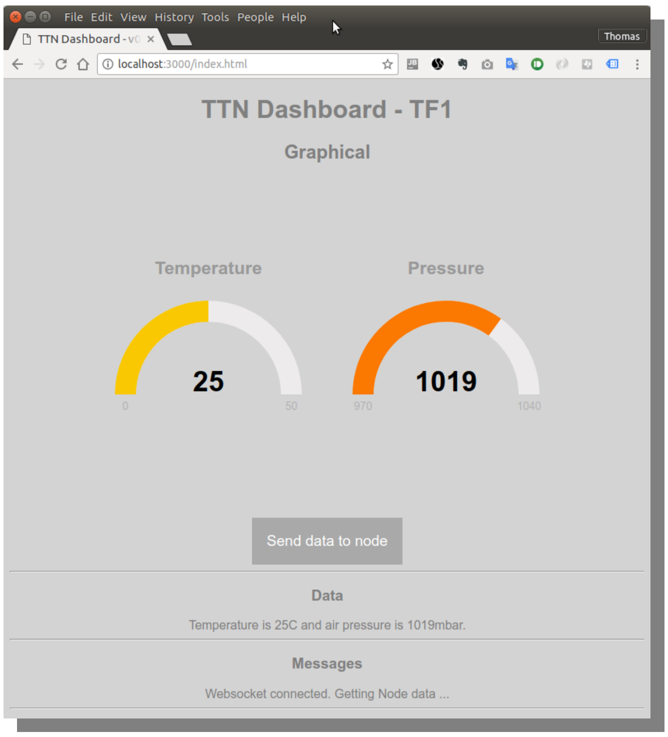
These are included in the projects 'package.json', so just run

```
npm update
```

from within the project folder.

Website

The website files are all included on GitHub, including a third party library to display the gauges. Jquery is loaded from an external CDN and the client's socket.io library is provided by the Node.JS socket.io module.


File Edit View History Tools People Help
TTN Dashboard - v0
Thomas
localhost:3000/index.html
Send data to node

TTN Dashboard - TF1

Graphical

Temperature 25 (0 – 50)

Pressure 1019 (970 – 1040)

Data

Temperature is 25C and air pressure is 1019mbar.

Messages

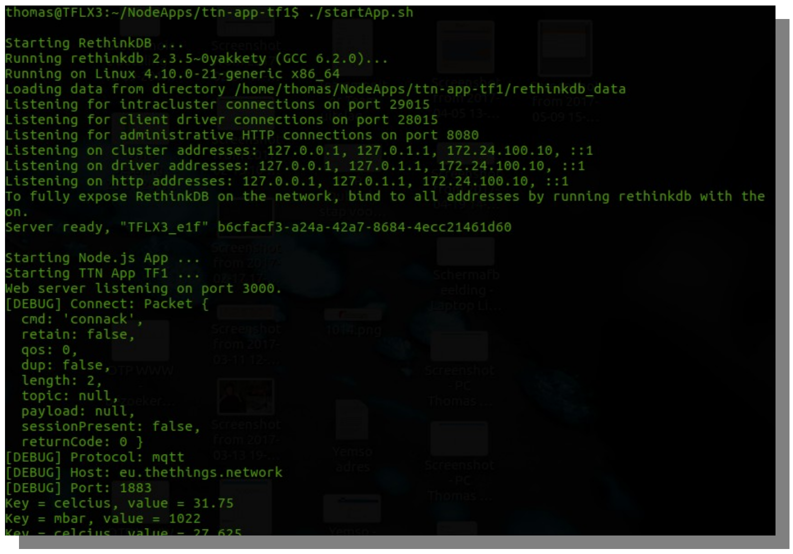
Websocket connected. Getting Node data ...

The project files

The project uses the following files for:

- index.js
  Starts the two main processes:
  - Getting the data from TTN into the RethinkDB,
  - Providing a web and websocket.io server for the front-end.
- GetNodeDataInDB.js
  Getting the TTN data into the RethinkDB.
  In your local version update the MQTT connection information by changing the AppName and AppKey variables. You find them in your TTN console.
  For the MQTT secure connection to TTN it also reads the certificate “mqtt-ca.pem” file.
- msgHandler.js
  Providing a web and websocket.io server for the front-end. The web server is quite a complete implementation that can serve all files from the “client” folder via HTTP.
  It also starts the websocket.io on port 3000 and serves RethinkDB update event data via the socket, once a client is connecting to it.
- startApp.sh
  A little script to start the RethinkDB and if successful the Node App. You probably need to make this work on your own machine.
- node_modules folder
  This is created when you install the NodeJS modules by “npm update”.
- client folder
  Here are all website files stored. The index.html with our simple front-end page. CSS and JS are in separate folders.
  - The web page uses a couple of JS libraries:
    - jquery
      Is loaded via Google CDN
    - JustGauge
      Is included in this GitHub project, the original can be found on:
      http://justgage.com/
      There you can find all documentation and examples.

- websocket.io
  When installing the Node.JS Websocket.io library you automatically get it’s client JS library. No special installation.



```
thomas@TFLX3:~/NodeApps/ttn-app-tf1$ ./startApp.sh

Starting RethinkDB ...
Running rethinkdb 2.3.5~0yakkety (GCC 6.2.0)...
Running on Linux 4.10.0-21-generic x86_64
Loading data from directory /home/thomas/NodeApps/ttn-app-tf1/rethinkdb_data
Listening for intracluster connections on port 29015
Listening for client driver connections on port 28015
Listening for administrative HTTP connections on port 8080
Listening on cluster addresses: 127.0.0.1, 127.0.1.1, 172.24.100.10, ::1
Listening on driver addresses: 127.0.0.1, 127.0.1.1, 172.24.100.10, ::1
Listening on http addresses: 127.0.0.1, 127.0.1.1, 172.24.100.10, ::1
To fully expose RethinkDB on the network, bind to all addresses by running rethinkdb with the
on.
Server ready, "TFLX3_e1f" b6cfacf3-a24a-42a7-8684-4ecc21461d60

Starting Node.js App ...
Starting TTN App TF1 ...
Web server listening on port 3000.
[DEBUG] Connect: Packet {
  cmd: 'connack',
  retain: false,
  qos: 0,
  dup: false,
  length: 2,
  topic: null,
  payload: null,
  sessionPresent: false,
  returnCode: 0 }
[DEBUG] Protocol: mqtt
[DEBUG] Host: eu.thethings.network
[DEBUG] Port: 1883
Key = celcius, value = 31.75
Key = mbar, value = 1022
Key = celcius, value = 27.625
```

The Node.JS code produces a lot of output, just to give more insight in what it’s doing. It was very helpful to see when and in which order messages appeared. If you are new to Node.JS like me, give the log a close inspection.

Almost all scripts are based on basic samples only adding extra functionality to link all parts together and make our tutorial work. A lot of comments should help to make the code self explaining.

Sources of inspiration

Learning RethinkDB

- https://rethinkdb.com/docs/cookbook/javascript/

Node.JS TTN SDK (using MQTT to connect to TTN)

- https://www.thethingsnetwork.org/docs/applications/nodejs/quick-start.html
- https://www.thethingsnetwork.org/docs/applications/nodejs/live.html
- https://www.thethingsnetwork.org/docs/applications/nodejs/api.html

Socket.io info to get started (we use the Express examples as a basis)

- https://socket.io/docs/

The Gauge on the web page, see examples on

- http://justgage.com/

Windows installation notes

We had some time to test the tutorial on Windows, a Windows 8.1 Virtualbox was used.

- Created a project folder (d:\ttn-frontend-tutorial).
- RethinkDB ZIP downloaded and extracting the exe file into the project folder.
- Node.JS, 64bit LTS version MSI file downloaded.
  Run the installer, using defaults.
- Downloaded project ZIP from GitHub
  Extracted into the project directory.
- Alter your getNodeDataInDB.js
  You need to add your AppName and AppKey to connect via MQTT to TTN
- Open cmd.exe command line
  - Run `npm update` from within the project folder.
  - Started DB
  - Run `rethinkdb.exe` in project folder.
- Open DB dashboard in browser: `http://localhost:8080`
- Open a second cmd.exe, first is now busy running the DB.
  - Start the Node project from project folder: `node index.js`
  - *It failed!* Simultaneous I got an Windows Firewall message box asking if NodeJS was allowed to open private network resources. I agreed.
    Started Node script again.
- In RethinkDB dashboard I now found the new DB and tables. So the script works.
- In new browser Tab opened `http://localhost:3000` to view our project's front-end.

Great to see that this little project is OS independent, the steps are almost equal as for Ubuntu. Probably Mac will work too.