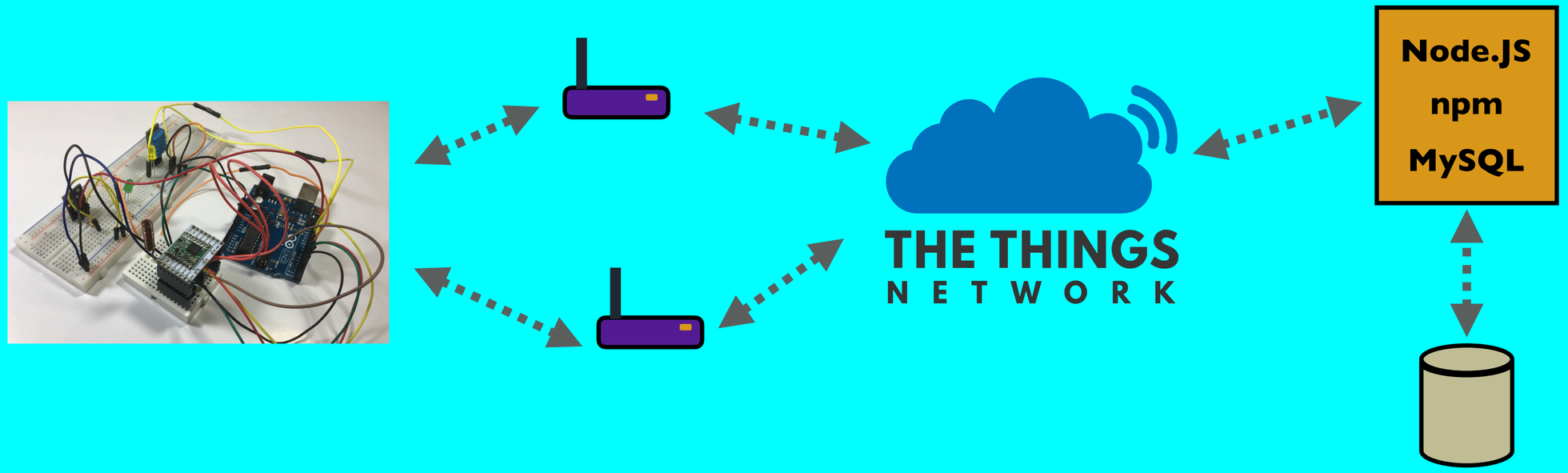
mobilefish.com

LORA / LORAWAN TUTORIAL 27

Retrieve and Store Sensor Data From The Things Network



v1.0.1

INTRO

- In this tutorial I will demonstrate:
  - how to retrieve sensor data from The Things Network,
  - how to store it in a database,
  - how to retrieve this data from the database and display it in a browser,
  - and how to create a downlink, using a NodeJS script, sending data from my computer to my LoRa end node.

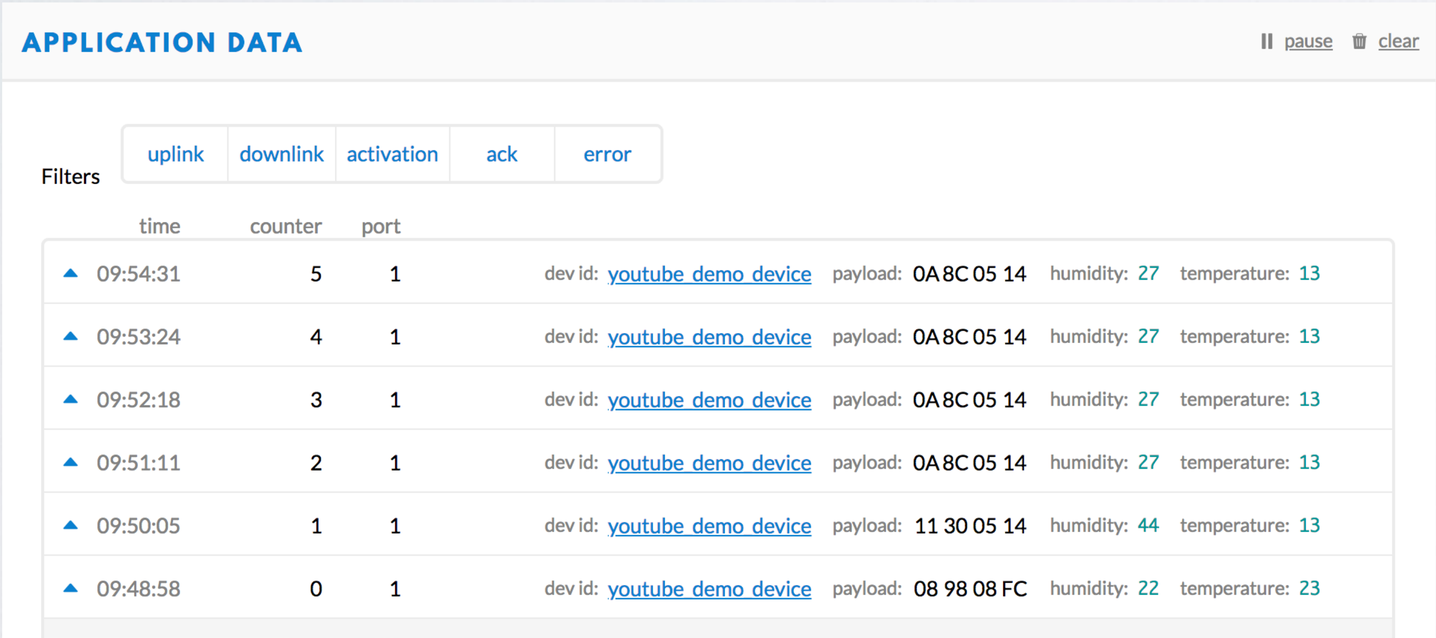
TTN_APP_SERVER GITHUB REPOSITORY

- All code used in this tutorial can be found in the following Github repository: https://github.com/robertlie/ttn_app_server

SENDING SENSOR DATA FROM END NODE TO TTN

- I highly recommend that you first watch tutorial 26 if you have not done so. https://youtu.be/EMoZ9taGZRs

- In tutorial 26 I have demonstrated how sensor data is send to The Things Network. This tutorial (tutorial 27) uses this data.



SDK RETRIEVING SENSOR DATA FROM TTN

- The Things Network community developers created several Software Development Kits (SDK) to receive activations and messages from IoT devices via The Things Network to your server.
  It also allows you to send messages back to the IoT devices from your server.

- The SDK's are available in Go, Java, Python and Node.JS
  Go: https://github.com/TheThingsNetwork/go-app-sdk
  Java: https://github.com/TheThingsNetwork/java-app-sdk
  Python: https://github.com/TheThingsNetwork/python-app-sdk
  Node.JS: https://github.com/TheThingsNetwork/node-app-sdk

- In this tutorial I will use the Node.JS SDK.

PREREQUISITES

- This tutorial assumes you have installed the following software packages and know how these packages works.
  - Node.JS (JavaScript server environment) and npm (node package manager)
    https://nodejs.org/en/download/package-manager/
  - MySQL (Relational Database Management System)
    In this tutorial MySQL Community Server is used.
    https://www.mysql.com/downloads/
  - phpMyAdmin (Web based administration tool for MySQL)
    https://www.phpmyadmin.net/

PREREQUISITES

- As a demonstration a PHP program "read_table.php" is written to display the sensor data in a browser.
- To make this PHP program work on your computer you need to install a web server (for example Apache) in conjunction with PHP and MySQL.
- In this tutorial I will not explain how these packages are installed or configured.

APPLICATION ID AND ACCESS KEY

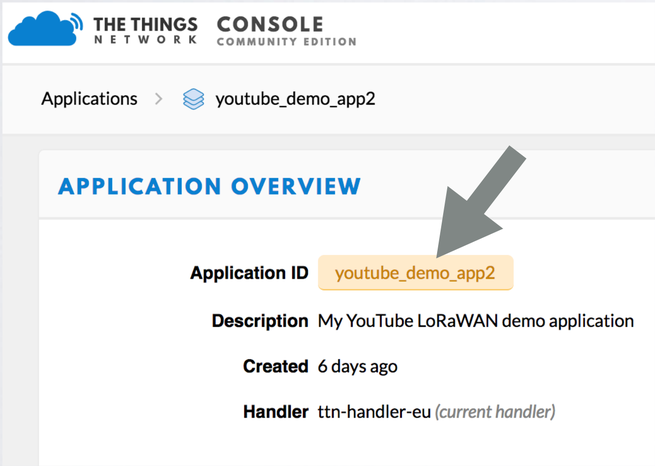
- Goto The Things Network console.

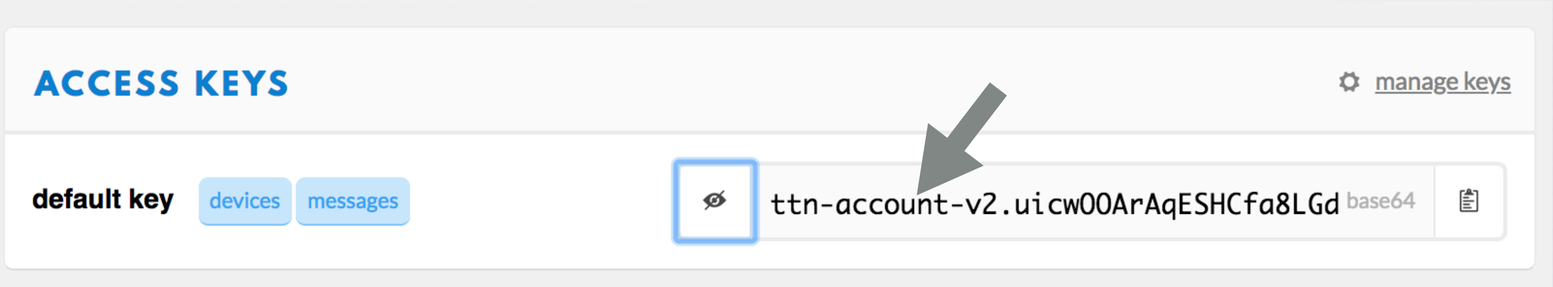
- Goto the applications page and select the application which receives the sensor data.
  In this demo the application ID is "youtube_demo_app2".

- WARNING:
  In tutorial 26 the application ID was "youtube_demo_app".
  For tutorial 27, as a test, I deleted the application ID "youtube_demo_app" in the assumption I could recreate the application ID again but this was a wrong assumption!
  Once you delete an application ID you can NOT recreate it again.
  So be aware of this!

APPLICATION ID AND ACCESS KEY

- To retrieve sensor data from The Things Network to your server, you need:
  - The application ID
    Example: youtube_demo_app2
  - Access key
    Example: ttn-account-v2.uicwOOArAqESHCfa8LGdftBSM6IZWjCdv4Art14iKtc





MODIFY FILE CONFIG.JS

- Download the Git repository:
  https://github.com/robertlie/ttn_app_server

- Goto folder ttn_app_server

- Install the node modules, type: npm install

- Modify file config.js

```
user: 'ENTER_MYSQL_ACCOUNT_NAME_HERE'
password: 'ENTER_MYSQL_PASSWORD_HERE'
appID: ‘ENTER_TTN_APP_ID_HERE'
accessKey: 'ENTER_TTN_ACCESSKEY_HERE'
```

MODIFY FILE READ_TABLE.PHP

- Modify file read_table.php

```
$username = "ENTER_MYSQL_ACCOUNT_NAME_HERE";
$password = “ENTER_MYSQL_PASSWORD_HERE";
```

END NODE SENDS SENSOR DATA TO TTN

- Make sure the end node sends sensor data to TTN, see tutorial 26.

APPLICATION DATA

pause clear

Filters: uplink | downlink | activation | ack | error

time	counter	port					
09:54:31	5	1	dev id: youtube demo device	payload: 0A 8C 05 14	humidity: 27	temperature: 13
09:53:24	4	1	dev id: youtube demo device	payload: 0A 8C 05 14	humidity: 27	temperature: 13
09:52:18	3	1	dev id: youtube demo device	payload: 0A 8C 05 14	humidity: 27	temperature: 13
09:51:11	2	1	dev id: youtube demo device	payload: 0A 8C 05 14	humidity: 27	temperature: 13
09:50:05	1	1	dev id: youtube demo device	payload: 11 30 05 14	humidity: 44	temperature: 13
09:48:58	0	1	dev id: youtube demo device	payload: 08 98 08 FC	humidity: 22	temperature: 23

RETRIEVE.JS

- Run the script retrieve.js, type: node retrieve.js
  This script only retrieves sensor data from TTN and displays it in the terminal.

SEND.JS

- It is possible to create a downlink by sending data to the end node using script send.js.
- Modify file send.js:
  client.send("youtube_demo_device", Buffer.alloc(1, 0x00, 'binary'));
- Depending on the hex value send, the yellow and green leds can be On or Off.
- Run the script, type:
  node send.js

Hex value	Yellow Led	Green Led
00	Off	Off
01	On	Off
02	Off	On
03	On	On

CREATE_DB.JS AND CREATE_TABLE.JS

- The retrieved sensor data from TTN can be stored in a MySQL database.
  A database and corresponding table needs to be created.

- First create the database **ttn_demo_db**, type: node create_db.js

- Next create the table **sensor_data**, type: node create_table.js

- Use the web application phpMyAdmin, to check if the database and table are created.
  http://localhost/~username/phpmyadmin/index.php

CREATE_DB.JS AND CREATE_TABLE.JS

Server: localhost » Database: ttn_demo_db » Table: sensor_data

Browse | Structure | SQL | Search | Insert | Export | Import | Privileges | Operations | Triggers

Table structure | Relation view

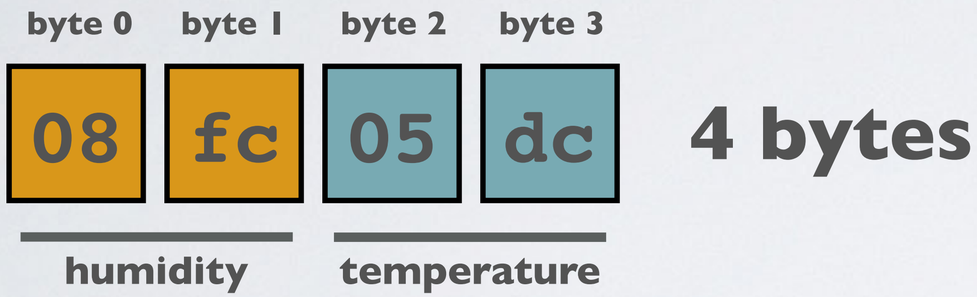
	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	**id**	int(11)			No	*None*		AUTO_INCREMENT	Change Drop More
☐	2	**hardware_serial**	varchar(50)	latin1_swedish_ci		Yes	*NULL*			Change Drop More
☐	3	**port**	tinyint(4)			Yes	*NULL*			Change Drop More
☐	4	**counter**	bigint(20)			Yes	*NULL*			Change Drop More
☐	5	**payload_raw**	tinyblob			Yes	*NULL*			Change Drop More
☐	6	**time**	varchar(30)	latin1_swedish_ci		Yes	*NULL*			Change Drop More
☐	7	**frequency**	float(6,3)			Yes	*NULL*			Change Drop More
☐	8	**modulation**	varchar(255)	latin1_swedish_ci		Yes	*NULL*			Change Drop More
☐	9	**data_rate**	varchar(255)	latin1_swedish_ci		Yes	*NULL*			Change Drop More
☐	10	**airtime**	int(11)			Yes	*NULL*			Change Drop More
☐	11	**coding_rate**	varchar(3)	latin1_swedish_ci		Yes	*NULL*			Change Drop More
☐	12	**gateways**	text	latin1_swedish_ci		Yes	*NULL*			Change Drop More

TABLE COLUMN PAYLOAD_RAW

- The column "payload_raw" has data type tinyblob to store binary data.
- The payload_fields data is not stored because the payload_raw data can be used to recreate the payload_fields data.
- In tutorial 26 I have used this Arduino sketch: https://www.mobilefish.com/download/lora/ttn-otaa-sensors.ino.txt
- The DHT11 sensor measured the humidity and temperature.

TABLE COLUMN PAYLOAD_RAW

- The sketch transmits the humidity and temperature data as four bytes:



4 bytes

- humidity = 0x08fc = 2300
  temperature = 0x05dc = 1500

- humidity = 2300 / 100 = 23.00 % RH
  temperature = 1500 / 100 = 15.00 °C

TABLE COLUMN PAYLOAD_RAW

- If the button switch is pressed a single byte is transmitted.

byte 0

02

1 byte

Hex value	Yellow Led	Green Led
00	Off	Off
01	On	Off
02	Off	On
03	On	On

DECODER FUNCTION

```
function Decoder(bytes, port) {
    if(bytes.length == 1) {
        if(bytes[0] == 1) {
            return {
                'button': 'activated'
            }
        } else {
            return {
                'error': 'button action unknown'
            }
        }
    } else if(bytes.length == 4) {
        var humidity = (bytes[0]<<8) | bytes[1];
        var temperature = (bytes[2]<<8) | bytes[3];
        return {
            'humidity': humidity/ 100,
            'temperature': temperature/100
        }
    } else {
        return {
            'error': 'payload unknown'
        }
    }
}
```

code used for button switch

code used for DHT11

Decoder function used in tutorial 26

A modified version can be found in:
read_table.js
read_table.php
retrieve.js

TABLE COLUMN TIME

- The column "time" has data type varchar(30) and not datetime.
  Time example received from TTN: '**2018-12-27T14:39:12.420921047Z**'

- The time is measured with 9 digits fractional-seconds (420921047).

- I have not used the datetime data type because MySQL has fractional seconds support for datetime with up to 6 digits precision.

STORE_RECORDS.JS

- To retrieve sensor data from TTN and store it in a MySQL database, type: node store_records.js

- Use the web application phpMyAdmin, to check if sensor data are stored. http://localhost/~username/phpmyadmin/index.php

+ Options

←T→	id	hardware_serial	port	counter	payload_raw	time	frequency	modulation	data_rate	airtime	coding_rate	gateways
Edit Copy Delete	1	008943795813113F	1	2	[BLOB - 4 B]	2019-01-02T14:05:38.241454021Z	867.300	LORA	SF7BW125	51456000	4/5	[{"gtw_id":"eui-1dee008f7b8235d3"
Edit Copy Delete	2	008943795813113F	1	3	[BLOB - 4 B]	2019-01-02T14:06:45.164954047Z	868.500	LORA	SF7BW125	51456000	4/5	[{"gtw_id":"eui-1dee008f7b8235d3"
Edit Copy Delete	3	008943795813113F	1	4	[BLOB - 4 B]	2019-01-02T14:07:52.075216389Z	867.500	LORA	SF7BW125	51456000	4/5	[{"gtw_id":"eui-1dee008f7b8235d3"
Edit Copy Delete	4	008943795813113F	1	5	[BLOB - 4 B]	2019-01-02T14:08:59.411949524Z	868.100	LORA	SF7BW125	51456000	4/5	[{"gtw_id":"eui-1dee008f7b8235d3"
Edit Copy Delete	5	008943795813113F	1	6	[BLOB - 4 B]	2019-01-02T14:10:06.018479527Z	867.700	LORA	SF7BW125	51456000	4/5	[{"gtw_id":"eui-1dee008f7b8235d3"
Edit Copy Delete	6	008943795813113F	1	7	[BLOB - 4 B]	2019-01-02T14:11:11.417818963Z	868.300	LORA	SF7BW125	51456000	4/5	[{"gtw_id":"eui-1dee008f7b8235d3"
Edit Copy Delete	7	008943795813113F	1	8	[BLOB - 4 B]	2019-01-02T14:12:18.554328236Z	867.900	LORA	SF7BW125	51456000	4/5	[{"gtw_id":"mobilefish
Edit Copy Delete	8	008943795813113F	1	9	[BLOB - 4 B]	2019-01-02T14:13:24.134783357Z	868.500	LORA	SF7BW125	51456000	4/5	[{"gtw_id":"mobilefish

READ_TABLE.JS

- The sensor data is stored in the table sensor_data. To display all records from table sensor_data in a terminal, type: node read_table.js

```
ttn_app_server — -bash — 54×36
Data retrieved from table sensor_data
counter= 2
hardware_serial= 008943795813113F
port= 1
payload_raw= <Buffer 0a 8c 05 14>
payload_fields= { humidity: 27, temperature: 13 }
time (UTC)= 2019-01-02T14:05:38.241454021Z
frequency= 867.3
modulation= LORA
data_rate= SF7BW125
airtime= 51456000
coding_rate= 4/5
******* Gateway *******
gtw_id= eui-1dee008f7b8235d3
timestamp= 2161854795
time=
channel= 4
rssi= -114
snr= -5.8
rf_chain= 0
latitude= 52.4509
longitude= 4.80436
altitude= 10
```

READ_TABLE.PHP

- The sensor data is stored in the table sensor_data.
- To display all records from table sensor_data in a browser:
  - First deploy file read_table.php in a web server (for example Apache supporting PHP and MySQL).
  - Open a browser and open the PHP file.

READ_TABLE.PHP

sand.mobilefish.com

Mobilefish.com - Read table sensor_data

Read table sensor_data

Id	hardware_serial	port	counter	payload_raw	time	frequency	modulation	data_rate	airtime	coding_rate	gateways
1	008943795813113F	1	2	{'humidity': 27, 'temperature': 13 }	2019-01-02T14:05:38.241454021Z	867.300	LORA	SF7BW125	51456000	4/5	[{"gtw_id":"eui-1dee008f7b8235d3","ti {"gtw_id":"mobilefish","g 02T14:05:38Z","channe
2	008943795813113F	1	3	{'humidity': 27, 'temperature': 13 }	2019-01-02T14:06:45.164954047Z	868.500	LORA	SF7BW125	51456000	4/5	[{"gtw_id":"eui-1dee008f7b8235d3","ti {"gtw_id":"mobilefish","g 02T14:06:45Z","channe
3	008943795813113F	1	4	{'humidity': 27, 'temperature': 13 }	2019-01-02T14:07:52.075216389Z	867.500	LORA	SF7BW125	51456000	4/5	[{"gtw_id":"eui-1dee008f7b8235d3","ti {"gtw_id":"mobilefish","g 02T14:07:52Z","channe
4	008943795813113F	1	5	{'humidity': 27, 'temperature': 13 }	2019-01-02T14:08:59.411949524Z	868.100	LORA	SF7BW125	51456000	4/5	[{"gtw_id":"eui-1dee008f7b8235d3","ti {"gtw_id":"mobilefish","g 02T14:09:00Z","channe
5	008943795813113F	1	6	{'humidity': 27, 'temperature': 13 }	2019-01-02T14:10:06.018479527Z	867.700	LORA	SF7BW125	51456000	4/5	[{"gtw_id":"eui-1dee008f7b8235d3","ti {"gtw_id":"mobilefish","g 02T14:10:06Z","channe

DROP_DB.JS

- To completely delete the database ttn_demo_db, type: node drop_db.js
  BE CAREFUL, ONCE DELETED ALL DATA IS LOST.