LoRaWAN

Tutorial

Raspberry Pi Single Gateway
and Arduino Node

1 Introduction

1.1 Preface

LoRa (Long Range) and LoRaWAN (LoRa Wide Area Network) are treated in this book.

In the current world, more and more devices are linked together. Often via WiFi or a mobile connections (3G / 4G). Mobile connections generally cost money and Wi-Fi is not available everywhere. There are also situations where devices are completely out of the reach of any network.

In addition, it is possible that there is no electricity available in the places where these devices are placed. If this is the case, the equipment is dependent on batteries.

LoRa communication was introduced for these situations. Features are low energy consumption and a wide long range.

A fully-fledged LoRaWAN compatible network costs hundreds to thousands of euros. Usually too expensive for an "normal" hobbyist.

In this book we will make a simple, relatively inexpensive LoRa network with a Raspberry Pi and Arduino Mega. This solution has the necessary limitations but offers a good introduction to LoRa and LoRaWAN.

In this solution we will also use The Things Network (TTN).

1.2 Accountability

The texts and images included in this book come as far as I as a writer could verify from the so-called free domain (free pics, public text). In the unlikely event that this is not the case, I hereby offer my sincere apologies.

This book has been compiled with the utmost care and the solutions shown have been extensively tested. Should errors nevertheless have crept into the text or sources, this has been done without any intention. Despite all the care devoted to the composition of this book, the author can not be held liable for any damage resulting from any error in this publication.

The starting point is that the reader is in possession of a PC, laptop or tablet with preferably Windows 10.

Working with the necessary hardware and voltage adapters can be a risk if careless handling is involved. The solution shown works with low voltages (max 5 volts), which of course limits the risks. Wrong connection can of course damage certain hardware irreparably! The author can not be held liable for any damage resulting from this. It is all entirely at your own risk.

The content is translated from Dutch to English so there may be some flaws in the translation.

Also some software may use Dutch menus, buttons, messages and texts.

1.3 Abbreviations

Afkorting	Betekenis
ABP	Activation By Personalisation
CN	China
DR	Data Rate
EU	Europe
EUI	Extended Unique Identifier
IoT	Internet of Things
JDK	Java Development Kit
LAN	Local Area Network
LMIC	LoraMAC-In-C libary
LoRa	Long Range
LoRaWAN	Long Range Wide Area Network
M2M	Machine to Machine
MQTT	Message Queuing Telemetry Transport
OTAA	Over The Air Activation
RF	Radio Frequency
RPI	Raspberry Pi
SDK	Software Development Kit
SF	Spreading Factor
SSH	Secure Shell
TTN	The Things Network
US	United Stated
WAN	Wide Area Network
Wi-Fi	Wireless Fidelity

2 Content

Content

3 What is a LoRaWan?

To get a good understanding of what this book is about, first a piece of theory. A number of terms that are discussed are LoRa, LoRaWAN, nodes and frequencies.

3.1 LoRa

LoRa stands for Long Range. It is a patented technology from the company SemTech (see: https://www.semtech.com/technology/lora).

Features are:

- low energy use
- a wide range (up to 15 kilometers in rural areas and up to 2 kilometers in urban areas)
- secure communication in both directions
- a limited data rate (at least 300, to a maximum of 50,000 bytes per second)

LoRa uses a technology called spread spectrum. Spread spectrum is a form of wireless communication in which the energy of the transmitted signal is deliberately distributed over a certain frequency domain. Such scattered signals have a much larger bandwidth than the information they contain, creating a noisy signal that is difficult to detect or to intercept. Furthermore, it is also very difficult to disturb a spread spectrum signal with a different signal. Because of these properties, spread spectrum is ideal for military applications and other environments where a high reliability of the signal is desired.

For broadcasting a spread spectrum we use so-called RF (Radio Frequency) chips. This tutorial uses a the RFM95W HopeRF transceiver.

3.1.1 Things (sensors and actors)

LoRa is therefore also ideal for Internet-of-Things (IoT) applications. With LoRa all sorts of things can talk to the internet without having to use mobile communication (3G / 4G) or Wi-Fi. Ideal for places where no WiFi is available or where no mobile communication is possible. This requires a so-called gateway to the internet. In other words thing talks to a gateway. Gateways then communicate with the internet. Things can be sensors or actors.

A sensor reports information about for example as the temperature and humidity.
An actor can provide control from a given assignment such as turning a light on or off.

Sensors and actors are relatively inexpensive and they can be used for all kinds of applications. A few examples:

- Remote opening of a garage door.
- Locating a pet.
- Measuring the air quality.
- Measuring the water level in a ditch.
- Registering whether or not a parking space is occupied.

In principle things don’t constantly seek connection with the gateway, but only when needed.

Things uses the so-called Wide Area Network (WAN) technology and therefore are ideal for all kinds of applications that are beyond the reach of WiFi network or Local Area Network (LAN).

For example, LoRa could be used for smart cities. In such cities all kinds of devices are connected to each other that are normally not connect. Examples are lampposts, parking meters, traffic control installations, location monitors, air quality monitors and so on.

With LoRa, communities have a good opportunity to bring their countless IoT applications together at an affordable rate.

Another characteristic of LoRa is that there is seamless interoperability between smart things without the need for locally complex installations. Since the costs are relatively low (certainly compared to other technologies) it is possible to install secure networks. Not only for the professionals but also for the hobbyist.

3.2 LoRaWAN

LoRaWAN (Long Range Wide Area Network) is a specification for a telecommunication network suitable for long-distance communication with little power. The technology is used for machine-to-machine communication (M2M) and is therefore ideal for the internet of things (IoT).

LoRaWAN is designed to allow devices with low power to communicate with applications connected to the Internet via long-distance wireless connections. The LoRaWAN protocols are defined by the LoRa Alliance and formalized in the LoRaWAN specification that can be requested on the LoRa Alliance website (https://lora-alliance.org/lorawan-for-developers).

3.3 The Things Network (TTN)

TTN started in 2015 in Amsterdam. The city of Amsterdam has been covered since July 2015 with the open LoRa (Long Range) network that is open and free to use. The Things Network is a community to create an internet-of-things data network with the LoRaWAN technology. With a simple 1000-euro LoRa router you can connect 10,000 devices, explains founder Wienke Giezeman. That was the start of the IoT data network in Amsterdam that The Things Network set up together with ten parties in the capital. All these parties purchased a lora gateway and on 21 August the city coverage network was live.

The roll-out of the TTN network is therefore financed by users that install gateways (antennas). At the moment, people from more than 600 cities from more than 90 countries worldwide are working on realizing the project.

The technique is specifically designed for low-cost, energy-efficient sensors that can communicate over a large distance. The technology is widely used for applications such as smart parking systems, automation in agriculture and smart water management. Bicycles, boats and pets can be also connected to the internet.

3.4 Architecture

3.4.1 Nodes

A Node or end device is a small device or microcontroller that is connected to LoRaWAN, for example an Arduino or Raspberry Pi. Nodes are used to collect data (i.e. temperature) or to receive commands (i.e. turn on the light). The devices use LoRa modules to communicate with the LoRaWAN. The nodes send encrypted messages.

3.4.2 Gateways

Gateways receive messages from nodes. A message sent by a node is received by all gateways within the range of that node. A gateway is connected to the internet and forwards the messages to network servers. In this book the messages are sent to the TTN.

3.4.3 Application

An application uses the messages that are available on TTN. The application can use these messages to execute certain actions, for example sending an SMS to warn the owner of a certain event. This book uses a TTN client application that sends alerts to the PushBullet mobile app. But you also could send an email or sms.

3.4.4 Uplink and downlink messages

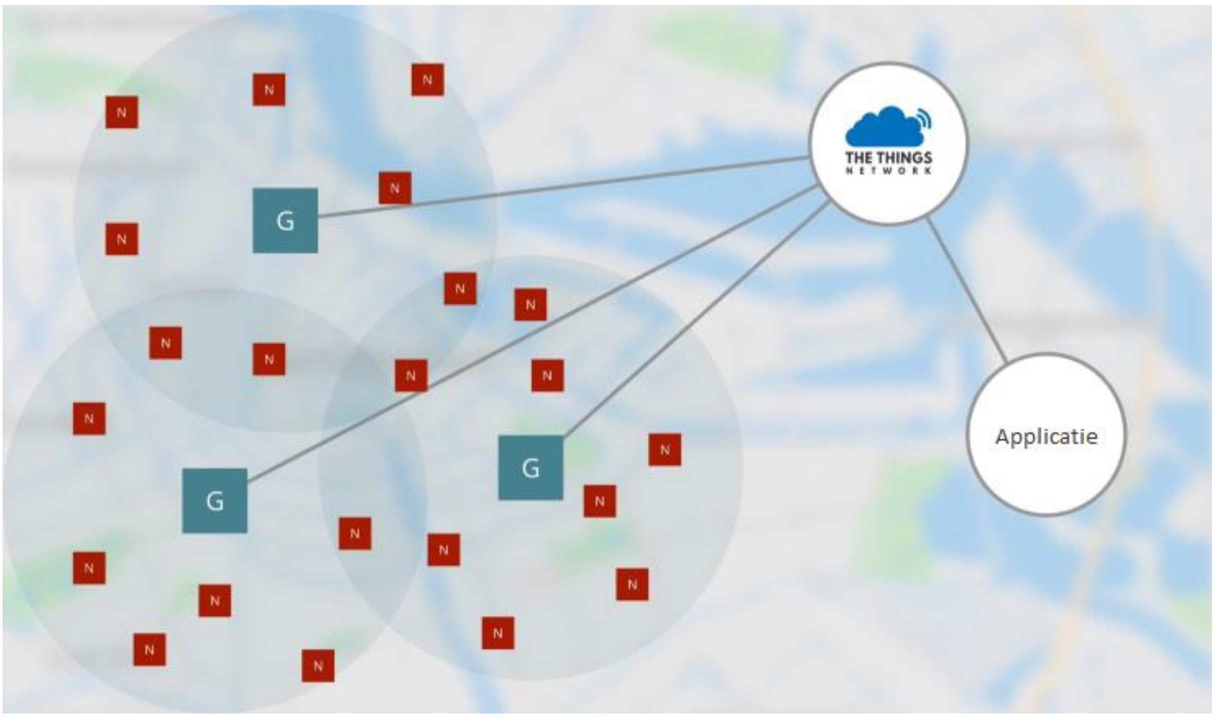
An uplink message is a message that is sent via the node to the application that then can process the message.

A downlink message is a message that is sent via the application to the node that then can process the message.

This book only uses uplink messages. Downlink messages are not (yet) supported by the software that is used.

The solution therefore is LoRaWAN compatible, but the solution does NOT fully comply with the LoRaWAN specifications. This does not have to be an objection in gaining experience with LoRaWAN.

3.4.5 Architecure map



N = Node
G = Gateway

By TTN - https://www.thethingsnetwork.org/wiki/Home, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=55457490

3.4.6 Frequencies

LoRaWAN uses so-called free radio frequencies that are currently specified as follows:

Europa	863-870 MHz and 433-434 MHz	EU868 en EU433
United Status	902-928 MHz	US915
China	470-510 MHz and 779-787 MHz	CN780

All frequencies are listed in the PDF document:
https://lora-alliance.org/sites/default/files/2018-04/lorawantm_regional_parameters_v1.1rb_-_final.pdf

Gateways and nodes in Europe operate at 868 MHz and in the United States at 915 MHz. It is a good thing to pay attention to this when purchasing hardware.

TTN uses three channels with the following frequencies:
- Channel 0: 868.1 MHz
- Channel 1: 868.3 MHz
- Channel 2: 868.5 MHz

3.4.7 Data frequency

The data frequency relies on:
- Transmission power (tx value),
- Bandwidth and
- spreading factor (SF).

If you lower the tx value, you save on the battery, but the range of the signal is shorter.

Bandwidth and spreading factor together form the data rate. This determines how quickly the bytes are sent.

With LoRa, the signal (chrip) is transmitted over the entire bandwidth. How long this signal is transmitted is called a spreading factor (SF). This is the shortest with SF7 and with SF12 this is the longest.

If you increase the data rate (making the bandwidth wider or the spreading factor lower), the bytes are sent in a shorter time.

By making the bandwidth 2x wider (for example from 125 KHz to 250 KHz), you can send two extra bytes in one go.

By making the spreading factor 1 step lower (for example from SF10 to SF9), you can send two extra bytes in one go. By reducing the spreading factor, however, it becomes more difficult for the gateway to receive a transmissions because it is more sensitive to noise. You could compare this with two people who occupy a noisy place (for example in a bar). If you are far apart, you have to talk slowly (SF12), but if you are close, you can talk faster (SF7)

TTN has three bandwidths: 125, 250 and 500 KHz. This depends on the frequency plan. In Europe the bandwidth 500 KHz is not used. In addition, TTN has six spreading factors: SF7 to SF12.

Overview for EU868 and EU433:

Data Rate	Configuratie	bits/seconde	Max payload (bytes)
DR0	SF12/125kHz	250	59
DR1	SF11/125kHz	440	59
DR2	SF10/125kHz	980	59
DR3	SF9/125kHz	1760	123
DR4	SF8/125kHz	3125	230
DR5	SF7/125kHz	5470	230
DR6	SF7/250kHz	11000	230

4 Raspberry Pi Single Channel Gateway

4.1 Introduction

To set up a single channel gateway we use a Raspberry Pi Model 3B (RPI) and a Dragino Lora / GPS HAT v1.4.

4.2 Hardware

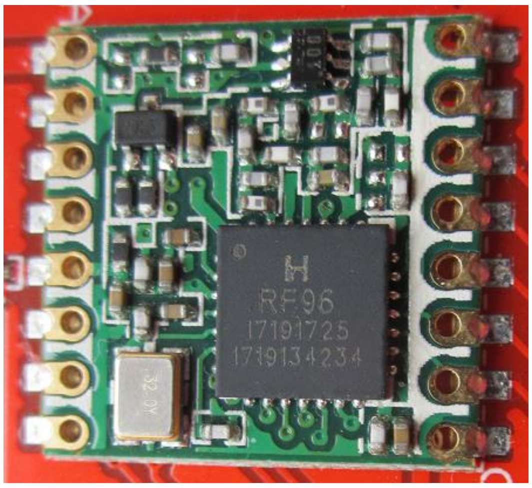
Hardware used:

- Raspberry Pi 3(b) (RPI)
- Dragino v1.4 hat

The Dragino can be placed on the RPI without any problems. No further installation is required.

Note: This Dragino also has a GPS, but this is not used in this book.

Our Dragino has a Hope RF96 chip which uses SX1276 chipset.



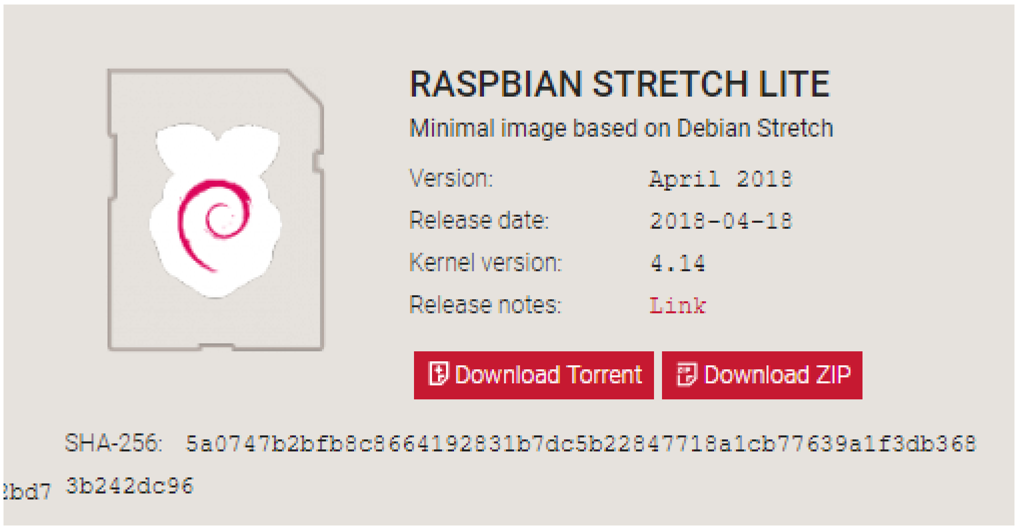
It works on 868MHz:

4.3 Headless installation

4.3.1 Download Raspbian Stretch

Download the latest version of Raspbian Stretch Lite.

Download link: https://www.raspberrypi.org/downloads/raspbian/

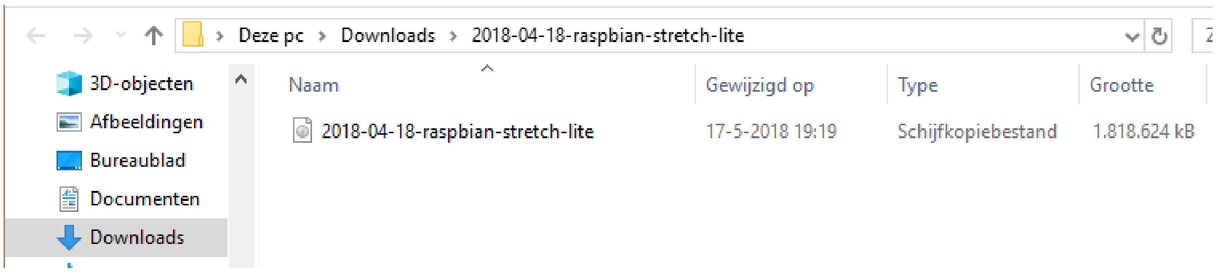

RASPBIAN STRETCH LITE

Minimal image based on Debian Stretch

Version:	April 2018
Release date:	2018-04-18
Kernel version:	4.14
Release notes:	Link

SHA-256: 5a0747b2bfb8c8664192831b7dc5b22847718a1cb77639a1f3db368
2bd7 3b242dc96

Unzip the ZIP file:



Naam	Gewijzigd op	Type	Grootte
2018-04-18-raspbian-stretch-lite	17-5-2018 19:19	Schijfkopiebestand	1.818.624 kB

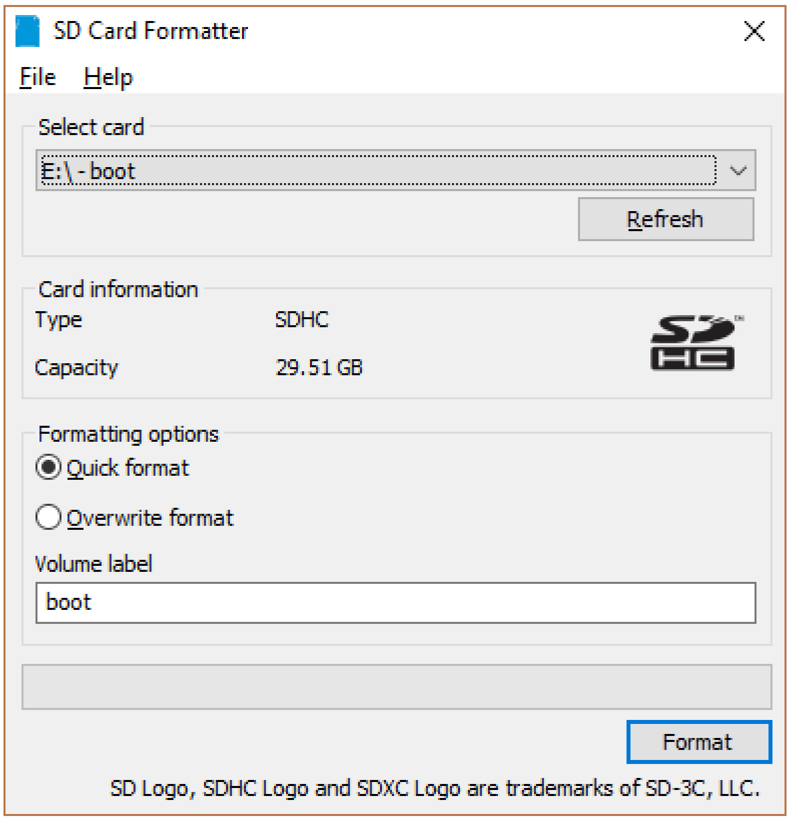
4.3.2 Prepare SD card

Use a Micro SD card of at least 8 GB. Use a Micro SD adapter if necessary.

Download and install the SD Card formatter.

Download link: https://www.sdcard.org/downloads/index.html

Start SD Card formatter, select the SD card → Format



Select card: E:\ - boot

Card information

Type SDHC

Capacity 29.51 GB

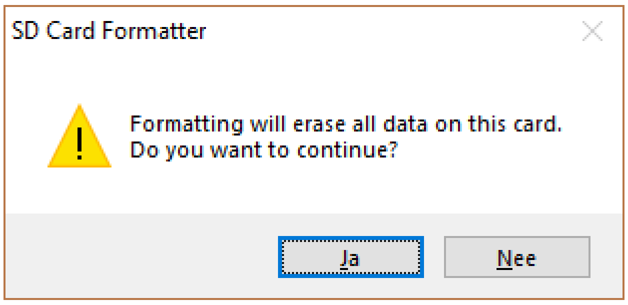
Formatting options

[✓ Quick format] Overwrite format

Volume label: boot

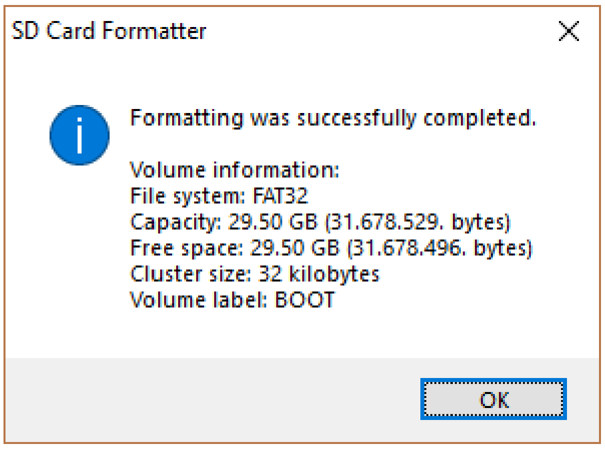
SD Logo, SDHC Logo and SDXC Logo are trademarks of SD-3C, LLC.

→ Yes (Ja)


SD Card Formatter

Formatting will erase all data on this card.
Do you want to continue?

→ OK


SD Card Formatter

Formatting was successfully completed.

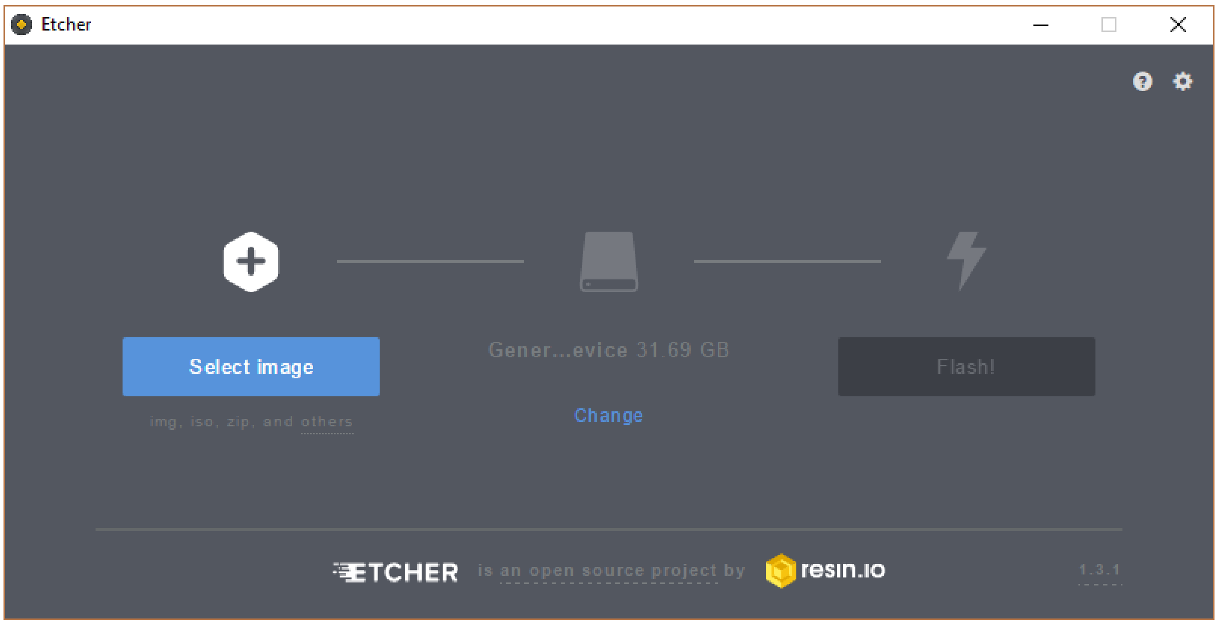
Volume information:
File system: FAT32
Capacity: 29.50 GB (31.678.529. bytes)
Free space: 29.50 GB (31.678.496. bytes)
Cluster size: 32 kilobytes
Volume label: BOOT

4.3.3 Transfer operation system

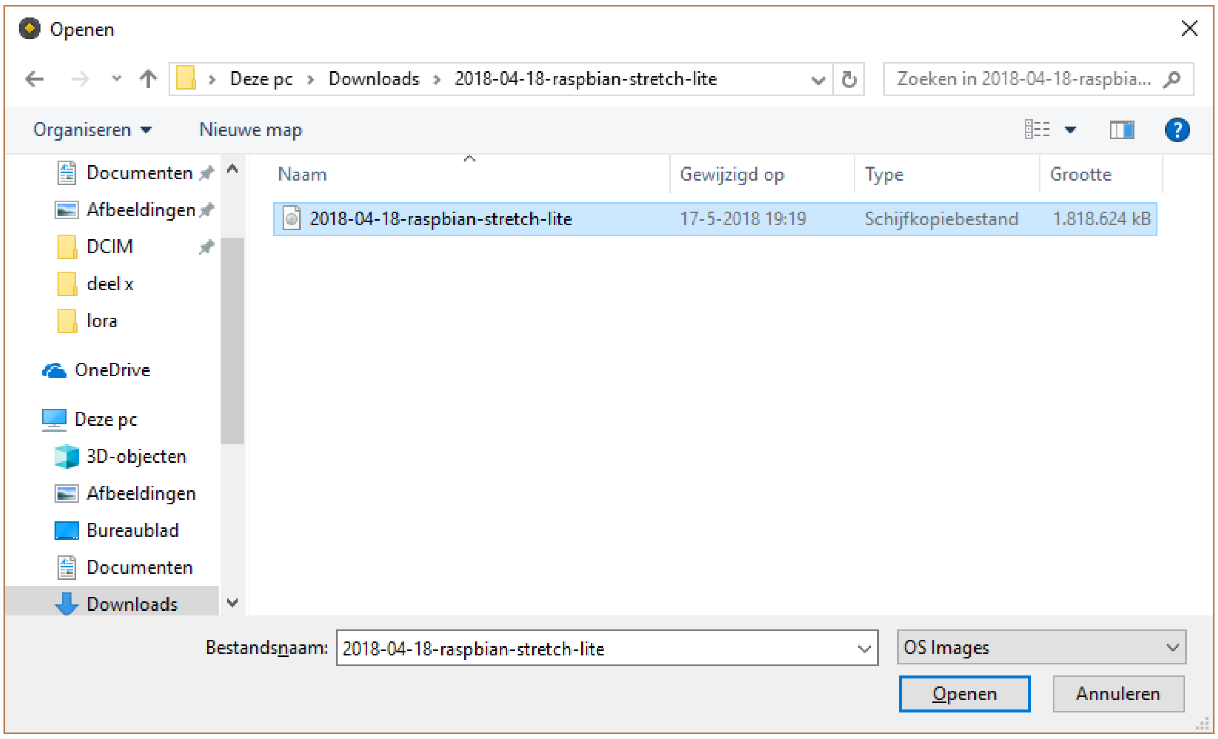
Download and install Etcher.

Download link: https://etcher.io/

Start Etcher.


Etcher
Select image
img, iso, zip, and others
Gener...evice 31.69 GB
Change
Flash!
ETCHER is an open source project by resin.io
1.3.1

Select the Raspbian Strecth Lite image.

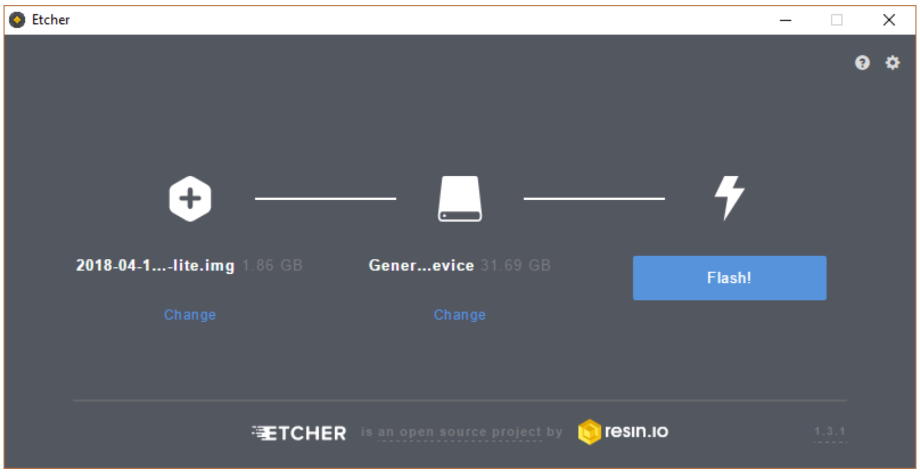


Naam	Gewijzigd op	Type	Grootte
2018-04-18-raspbian-stretch-lite	17-5-2018 19:19	Schijfkopiebestand	1.818.624 kB

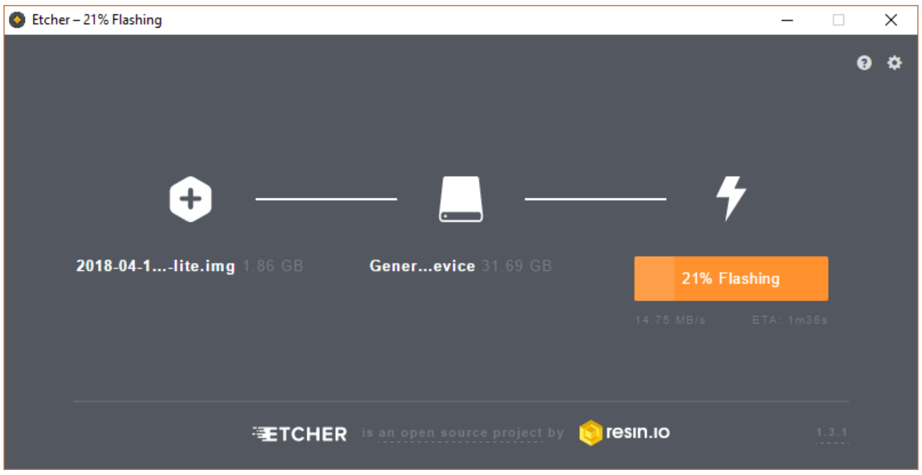
Bestandsnaam: 2018-04-18-raspbian-stretch-lite

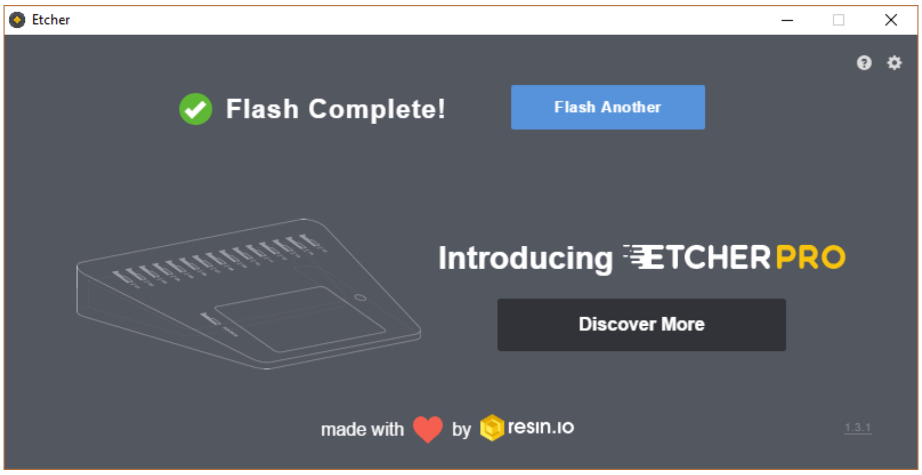
OS Images

→ Flash!


Etcher
2018-04-1...-lite.img 1.86 GB
Change
Gener...evice 31.69 GB
Change
Flash!
ETCHER is an open source project by resin.io
1.3.1

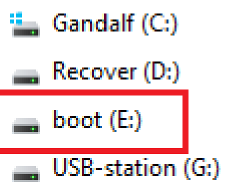
This can take some time!





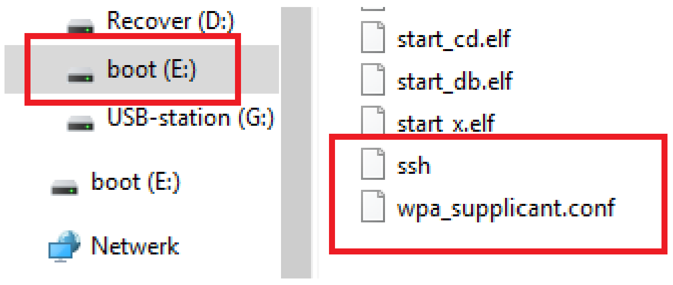
4.3.4 Headless installation

To set up SSH and WiFi, two files must be created in the boot part of the SD card.



Create two files:

- ssh
- wpa_supplicant.conf



The ssh files should be an empy file.

Content of the wpa_supplicant.conf file:

```
country=NL
ctrl_interface=DIR=/var/run/wpa_supplicant
update_config=1

network={
    ssid="MySSID"
    psk="MySecretPassword"
    scan_ssid=1
}
```

Change country, ssid and psk to fit your settings.

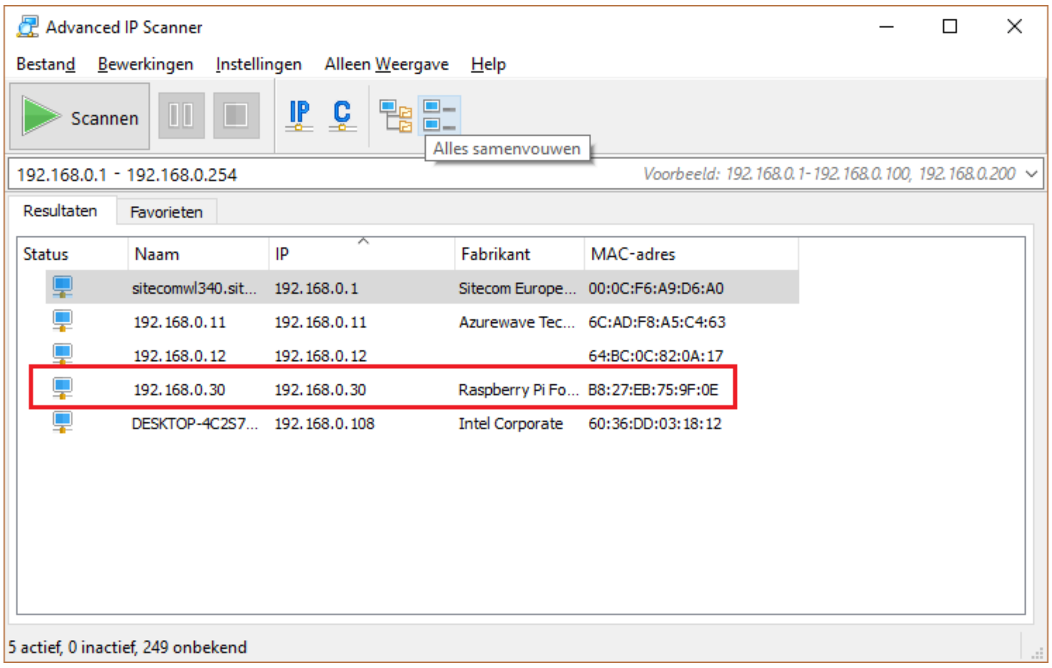
4.3.5 Start Raspberry Pi and determine IP-address

Start the Raspberry Pi with just prepared SD card. The start-up can take a few minutes.

Download IP Scan.

Download link: https://www.advanced-ip-scanner.com/nl/

Start (or install and start) IP Scan. Click Scan and wait for the scan to finish. It may take some time before all IP addresses become visible. If necessary, start a new scan if the IP address of the Raspberry Pi is not yet visible.



192.168.0.1 - 192.168.0.254 | Voorbeeld: 192.168.0.1-192.168.0.100, 192.168.0.200

Status	Naam	IP	Fabrikant	MAC-adres
	sitecomwl340.sit...	192.168.0.1	Sitecom Europe...	00:0C:F6:A9:D6:A0
	192.168.0.11	192.168.0.11	Azurewave Tec...	6C:AD:F8:A5:C4:63
	192.168.0.12	192.168.0.12		64:BC:0C:82:0A:17
	192.168.0.30	192.168.0.30	Raspberry Pi Fo...	B8:27:EB:75:9F:0E
	DESKTOP-4C2S7...	192.168.0.108	Intel Corporate	60:36:DD:03:18:12

5 actief, 0 inactief, 249 onbekend

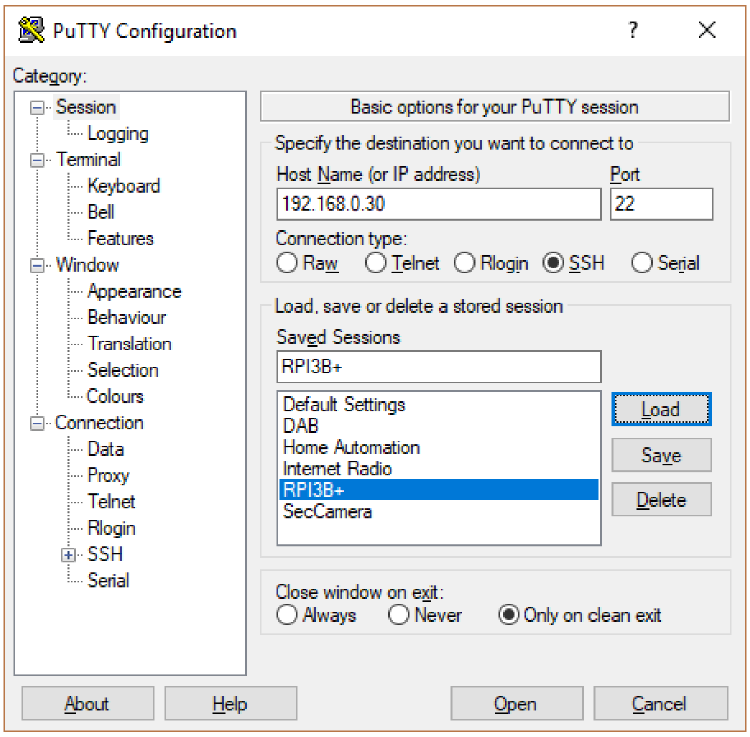
4.3.6 SSH via PuTTY

Download and install PuTTY.

Download link: https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

Start PuTTY

Host Name: IP-adres


PuTTY Configuration

Category:

Session
Logging
Terminal
Keyboard
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
Telnet
Rlogin
SSH
Serial

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): 192.168.0.30
Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

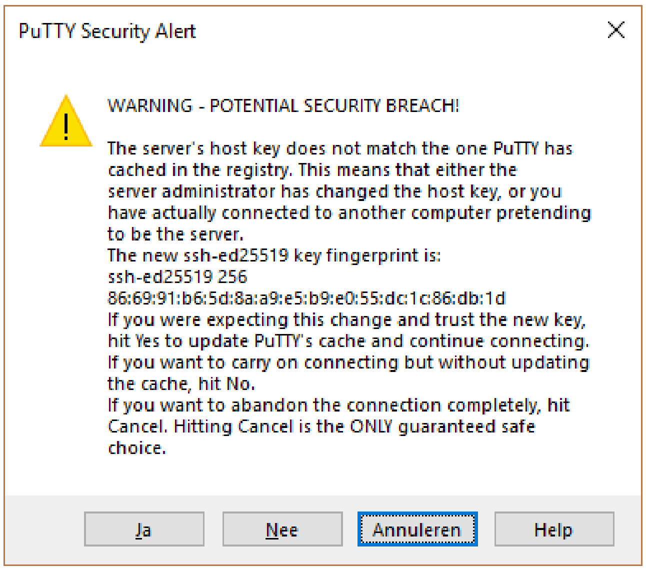
Load, save or delete a stored session

Saved Sessions: RPI3B+

Default Settings
DAB
Home Automation
Internet Radio
RPI3B+
SecCamera

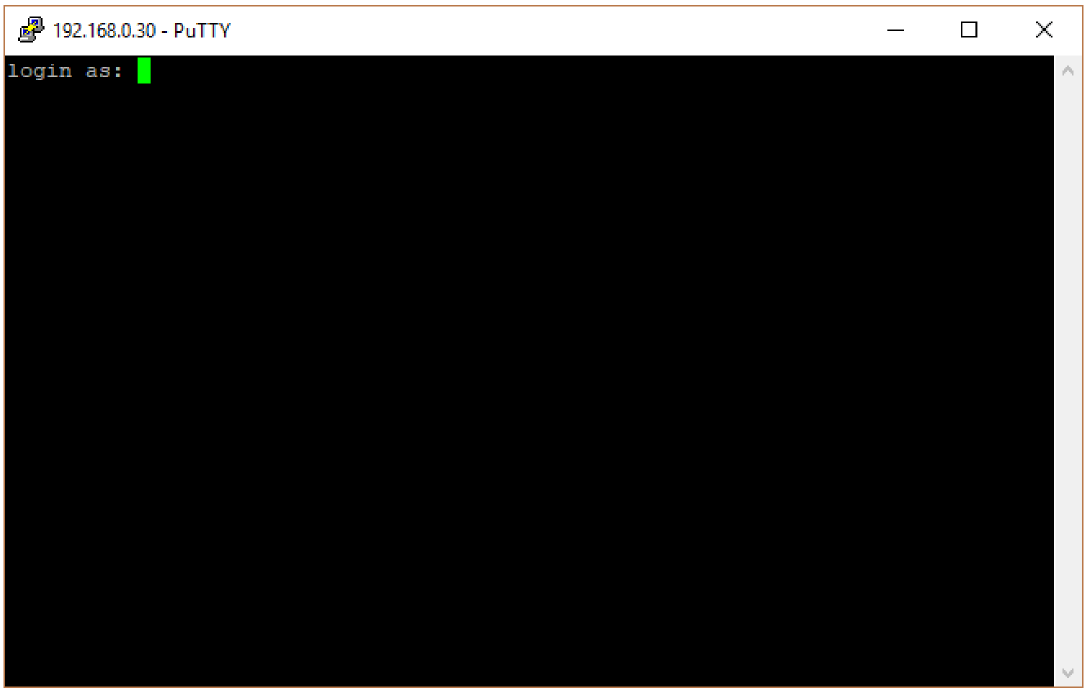
Close window on exit: Always Never [✓ Only on clean exit]

Yes (Ja)


PuTTY Security Alert

WARNING - POTENTIAL SECURITY BREACH!

The server's host key does not match the one PuTTY has cached in the registry. This means that either the server administrator has changed the host key, or you have actually connected to another computer pretending to be the server.
The new ssh-ed25519 key fingerprint is:
ssh-ed25519 256
86:69:91:b6:5d:8a:a9:e5:b9:e0:55:dc:1c:86:db:1d
If you were expecting this change and trust the new key, hit Yes to update PuTTY's cache and continue connecting.
If you want to carry on connecting but without updating the cache, hit No.
If you want to abandon the connection completely, hit Cancel. Hitting Cancel is the ONLY guaranteed safe choice.

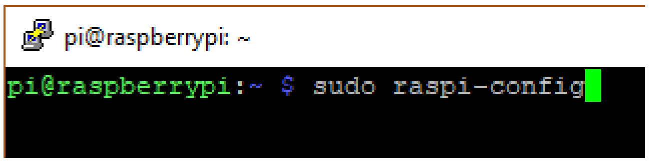


```
login as:
```

Log in:
- Login as: pi
- Password: raspberrry

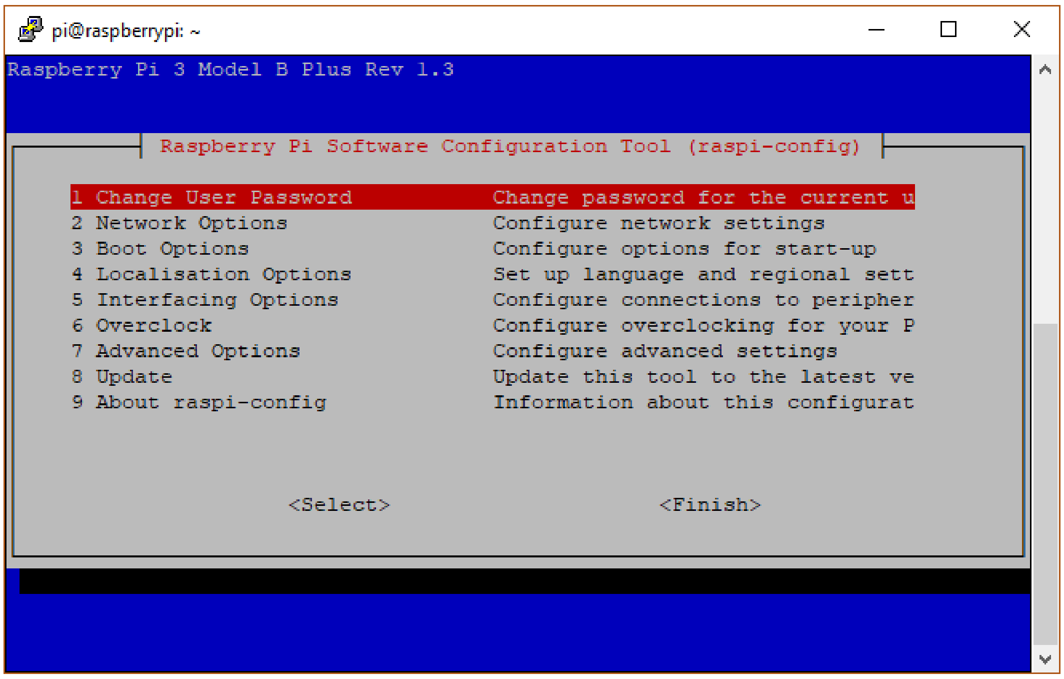
4.3.7 Configure Raspbian Stretch

Command: sudo raspi-config



```
pi@raspberrypi: ~
pi@raspberrypi:~ $ sudo raspi-config
```

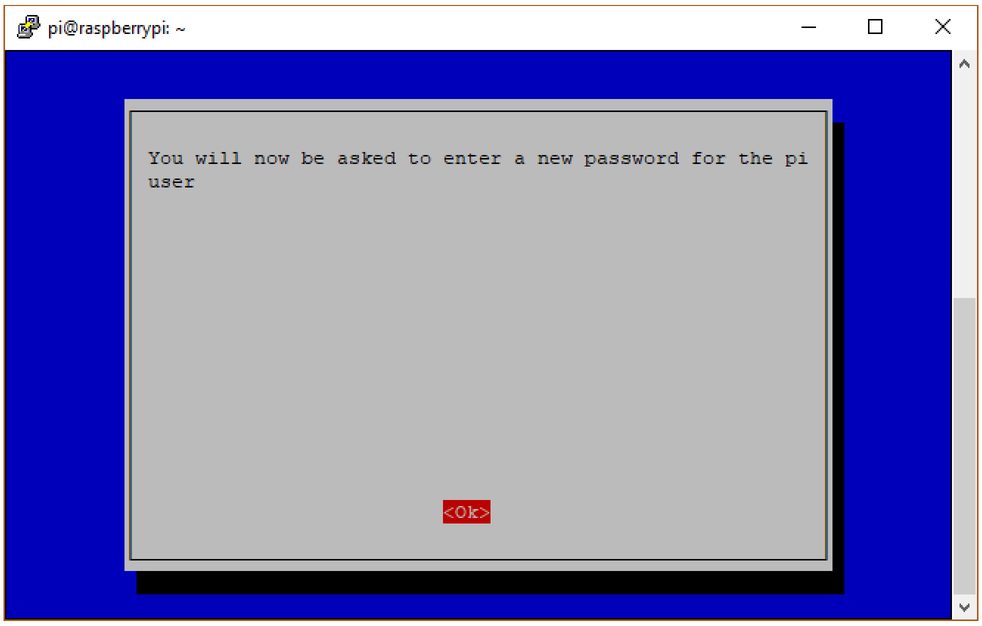
Result:



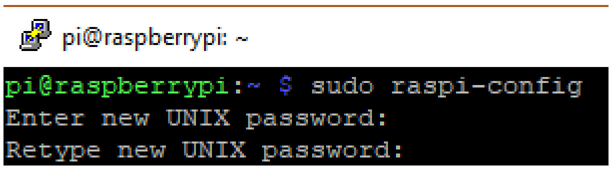
```
pi@raspberrypi: ~
Raspberry Pi 3 Model B Plus Rev 1.3

        Raspberry Pi Software Configuration Tool (raspi-config)

    1 Change User Password           Change password for the current u
    2 Network Options                Configure network settings
    3 Boot Options                   Configure options for start-up
    4 Localisation Options           Set up language and regional sett
    5 Interfacing Options            Configure connections to peripher
    6 Overclock                      Configure overclocking for your P
    7 Advanced Options               Configure advanced settings
    8 Update                         Update this tool to the latest ve
    9 About raspi-config             Information about this configurat


                 <Select>                     <Finish>
```

4.3.7.1 Change password

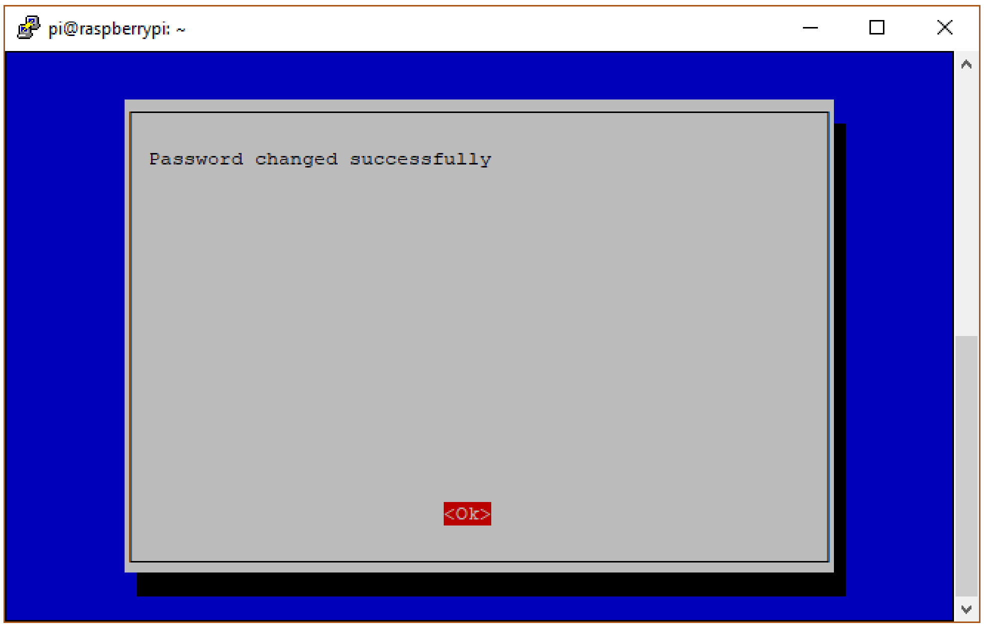
To prevent unauthorized login, the password must be changed with option 1 Change User Password.



```
pi@raspberrypi: ~

You will now be asked to enter a new password for the pi
user

<Ok>
```



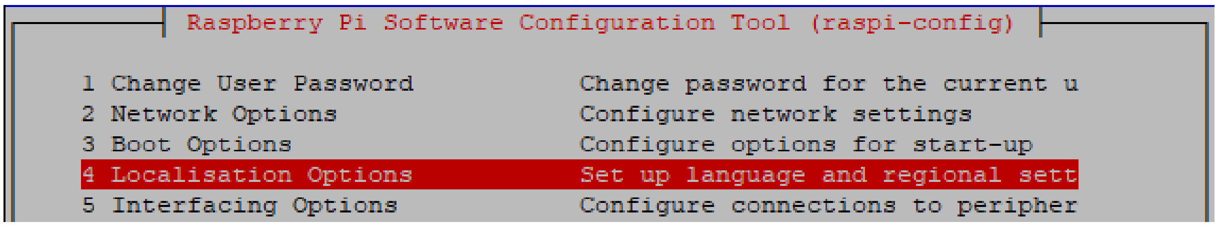
```
pi@raspberrypi: ~
pi@raspberrypi:~ $ sudo raspi-config
Enter new UNIX password:
Retype new UNIX password:
```



```
pi@raspberrypi: ~

Password changed successfully

<Ok>
```

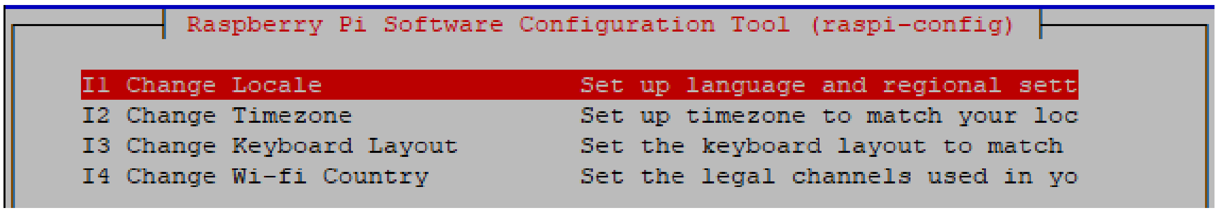
4.3.7.2 Change localisation settings

With option 4 Localization Options the localization options can be set.



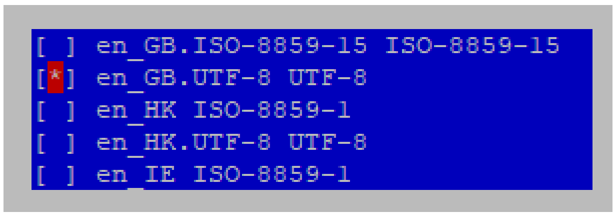
```
Raspberry Pi Software Configuration Tool (raspi-config)

1 Change User Password          Change password for the current u
2 Network Options               Configure network settings
3 Boot Options                  Configure options for start-up
4 Localisation Options          Set up language and regional sett
5 Interfacing Options           Configure connections to peripher
```

Option I1 Change Locale and <Select> with the tab key:



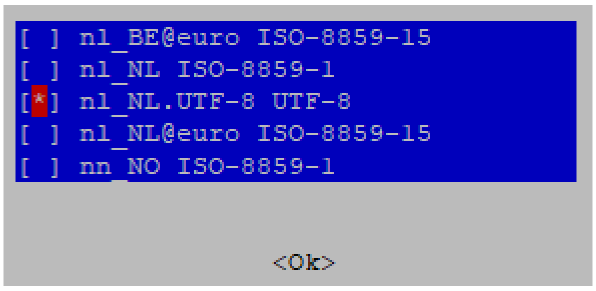
```
Raspberry Pi Software Configuration Tool (raspi-config)

I1 Change Locale                Set up language and regional sett
I2 Change Timezone              Set up timezone to match your loc
I3 Change Keyboard Layout       Set the keyboard layout to match
I4 Change Wi-fi Country         Set the legal channels used in yo
```

Scroll down with the cursor key and delete selection at en_GB.UTF-8 UTF-8 with the space key:



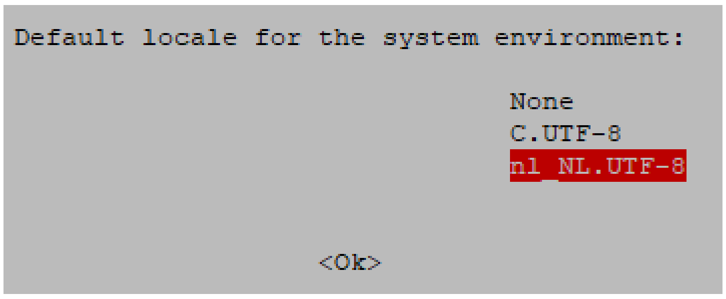
```
[ ] en_GB.ISO-8859-15 ISO-8859-15
[*] en_GB.UTF-8 UTF-8
[ ] en_HK ISO-8859-1
[ ] en_HK.UTF-8 UTF-8
[ ] en_IE ISO-8859-1
```

Scroll down and select nl_NL.UTF-8 UTF-8 (or your own country) with the space key and <Ok> (tab key).



```
[ ] nl_BE@euro ISO-8859-15
[ ] nl_NL ISO-8859-1
[*] nl_NL.UTF-8 UTF-8
[ ] nl_NL@euro ISO-8859-15
[ ] nn_NO ISO-8859-1

                 <Ok>
```

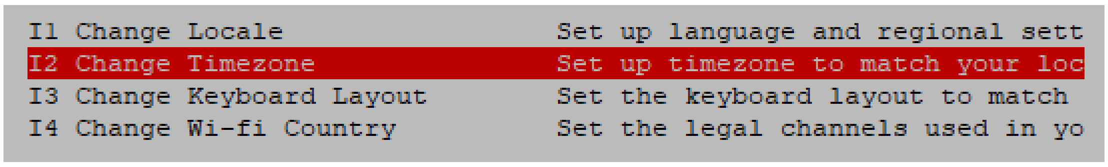
Select nl_NL.UTF-8 (or your selection) and <Ok>



```
Default locale for the system environment:

                               None
                               C.UTF-8
                               nl_NL.UTF-8

                 <Ok>
```

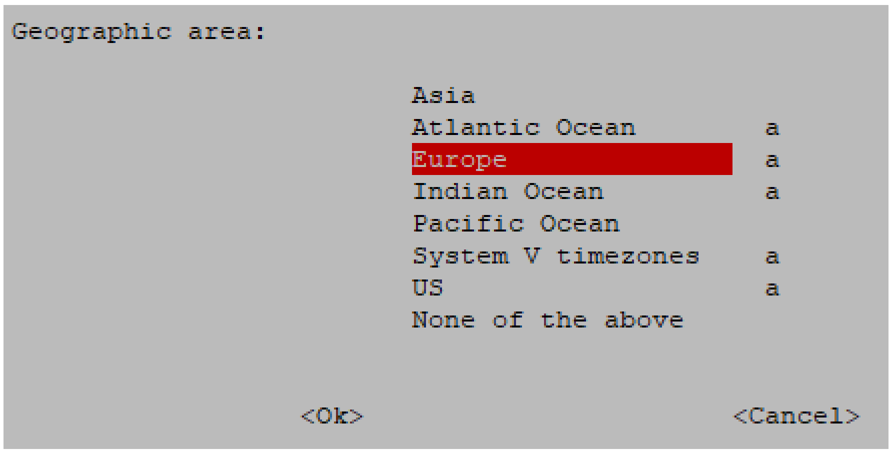
4.3.7.3 Change timezone

Select option 4 and I2 Change Timezone and <Select>



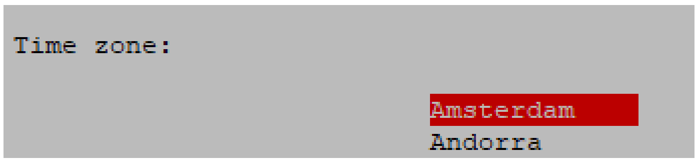
```
I1 Change Locale               Set up language and regional sett
I2 Change Timezone             Set up timezone to match your loc
I3 Change Keyboard Layout      Set the keyboard layout to match
I4 Change Wi-fi Country        Set the legal channels used in yo
```

Select Europe (or your own geograhic ares) and <Ok>



```
Geographic area:

                         Asia
                         Atlantic Ocean       a
                         Europe               a
                         Indian Ocean         a
                         Pacific Ocean
                         System V timezones   a
                         US                   a
                         None of the above


                  <Ok>                       <Cancel>
```

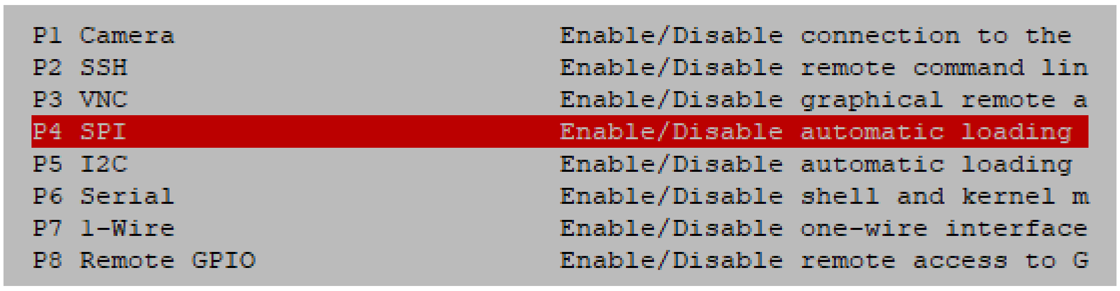
Select Amsterdam (or your own time zone) and <Ok>



```
Time zone:

                          Amsterdam
                          Andorra
```

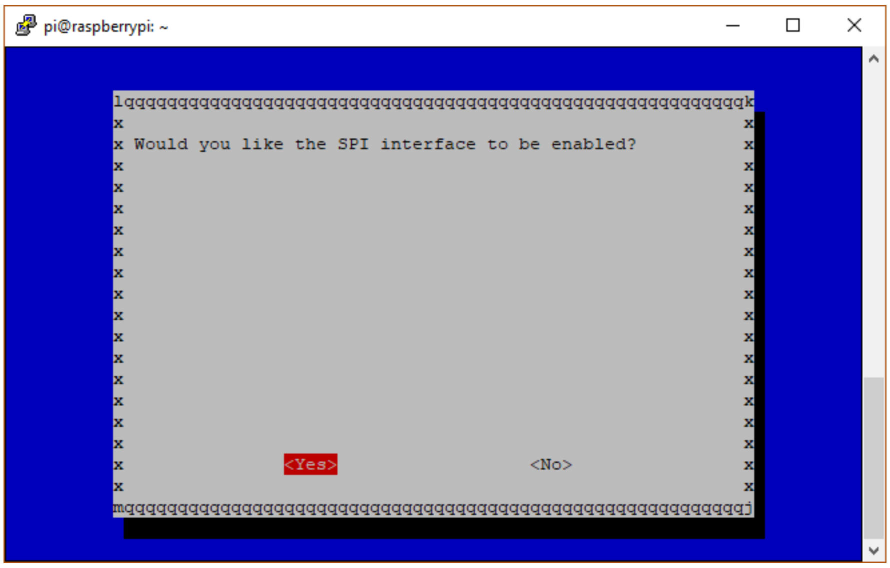
4.3.7.4 Change interfacing options

Select option 5 Interfacing Options, P4 SPI and <Select>



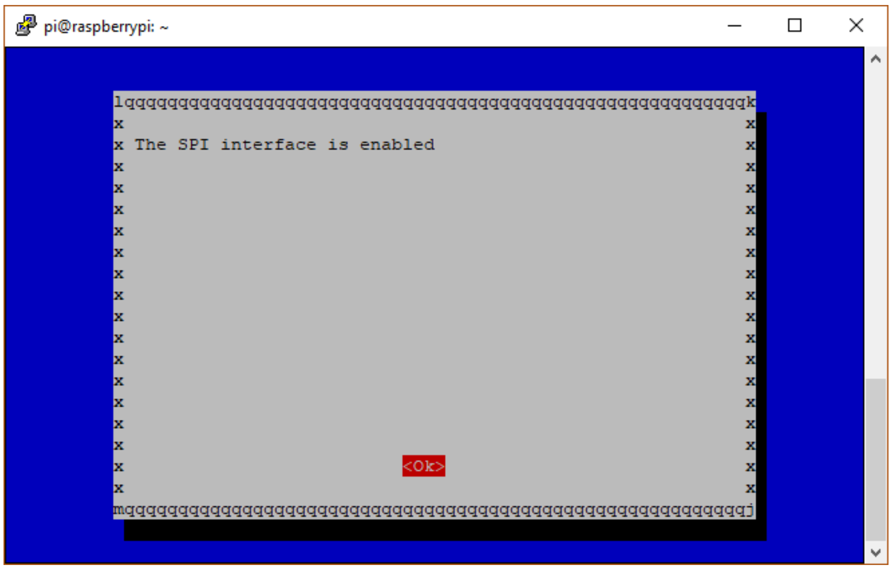
```
P1 Camera              Enable/Disable connection to the
P2 SSH                 Enable/Disable remote command lin
P3 VNC                 Enable/Disable graphical remote a
P4 SPI                 Enable/Disable automatic loading
P5 I2C                 Enable/Disable automatic loading
P6 Serial              Enable/Disable shell and kernel m
P7 1-Wire              Enable/Disable one-wire interface
P8 Remote GPIO         Enable/Disable remote access to G
```

Select <Yes>



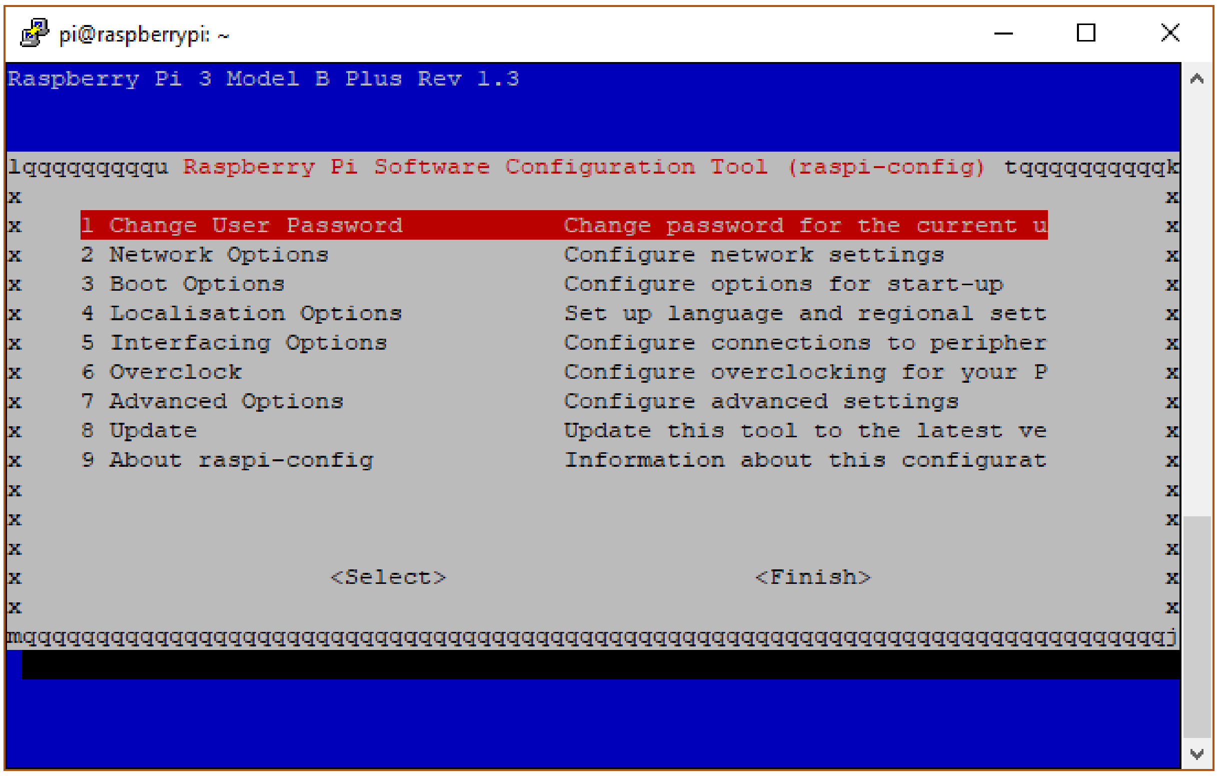
```
pi@raspberrypi: ~

Would you like the SPI interface to be enabled?

        <Yes>                    <No>
```

Select OK



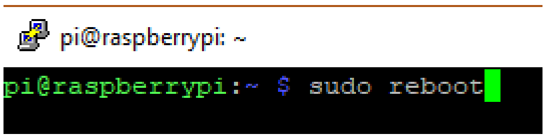
```
pi@raspberrypi: ~

The SPI interface is enabled

                  <Ok>
```

Select <Finish>



```
pi@raspberrypi: ~
Raspberry Pi 3 Model B Plus Rev 1.3

lqqqqqqqqqu Raspberry Pi Software Configuration Tool (raspi-config) tqqqqqqqqqqk
x                                                                               x
x    1 Change User Password           Change password for the current u        x
x    2 Network Options                Configure network settings               x
x    3 Boot Options                   Configure options for start-up           x
x    4 Localisation Options           Set up language and regional sett        x
x    5 Interfacing Options            Configure connections to peripher        x
x    6 Overclock                      Configure overclocking for your P        x
x    7 Advanced Options               Configure advanced settings              x
x    8 Update                         Update this tool to the latest ve        x
x    9 About raspi-config             Information about this configurat        x
x                                                                               x
x                                                                               x
x                                                                               x
x                     <Select>                     <Finish>                     x
x                                                                               x
mqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqj
```

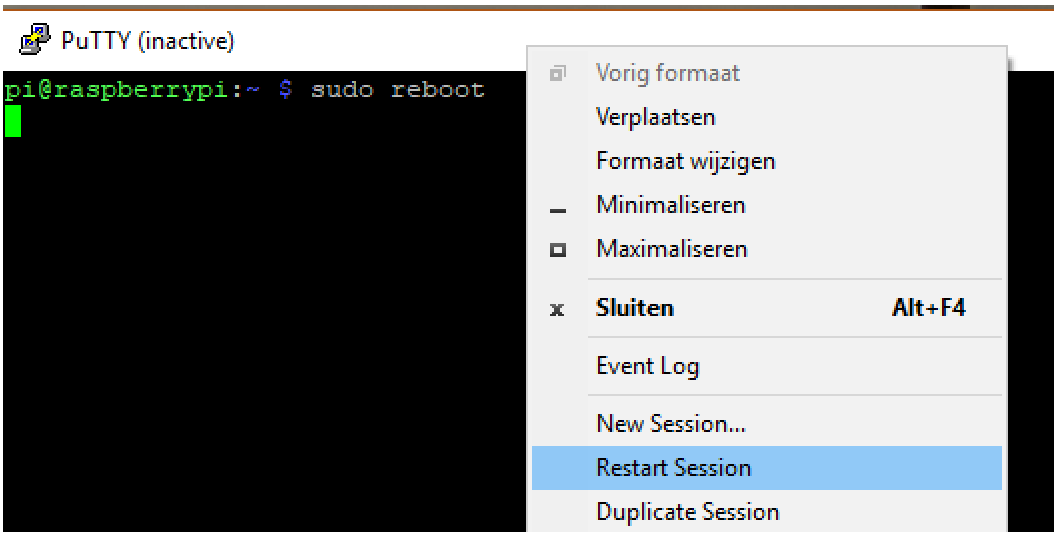
Reboot the Raspberry Pi with the command: sudo reboot



```
pi@raspberrypi: ~
pi@raspberrypi:~ $ sudo reboot
```

4.3.7.5 Update Raspbian Stretch

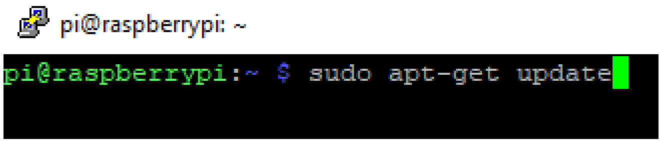
Restart PuTTY and login with the new password.



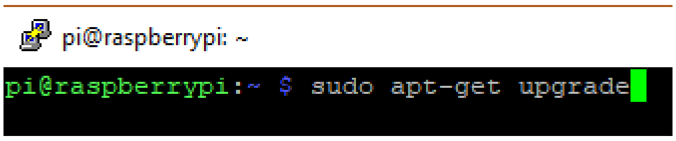
```
pi@raspberrypi:~ $ sudo reboot
```

Update the operation system with the commands:

- sudo apt-get update
- sudo apt-get upgrade

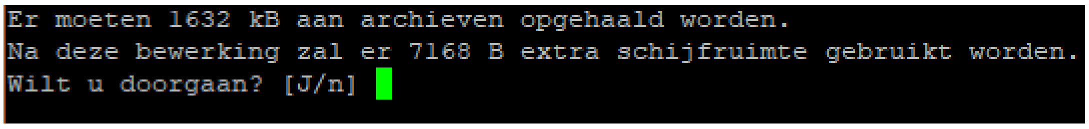


```
pi@raspberrypi:~ $ sudo apt-get update
```



```
pi@raspberrypi:~ $ sudo apt-get upgrade
```

This will take some time. Answer Y (J) on the displayed question:



```
Er moeten 1632 kB aan archieven opgehaald worden.
Na deze bewerking zal er 7168 B extra schijfruimte gebruikt worden.
Wilt u doorgaan? [J/n]
```

4.3.8 Install Gateway software

You need the following installations on your RPI:

- WiringPi
- Git
- Single Channel Packet Forwarder software

WiringPi is a so-called GPIO interface library which is used for interfacing with the Dragino LoRa shield.

Git is required for retrieving the single channel packet forwarder software.

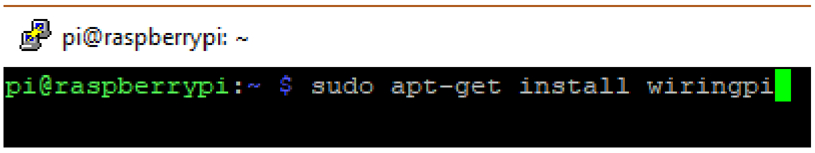
The single channel packet software is required for setting up the gateway. At the end of 2017 it seemed that TTN would no longer allow single channel packet software.

In a TTN statement:
Single-Channel gateways are not LoRaWAN compliant and are currently not officially supported by The Things Network.

Fortunately, an alternative has come (for the time being?). Which is discussed in this book.

4.3.8.1 WiringPi

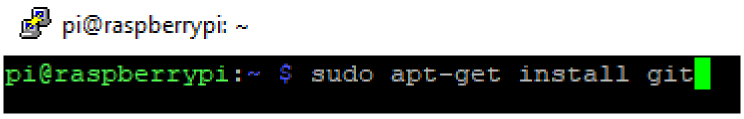
Command: sudo apt-get install wiringpi



```
pi@raspberrypi: ~
pi@raspberrypi:~ $ sudo apt-get install wiringpi
```

4.3.8.2 Git

Command: sudo apt-get install git



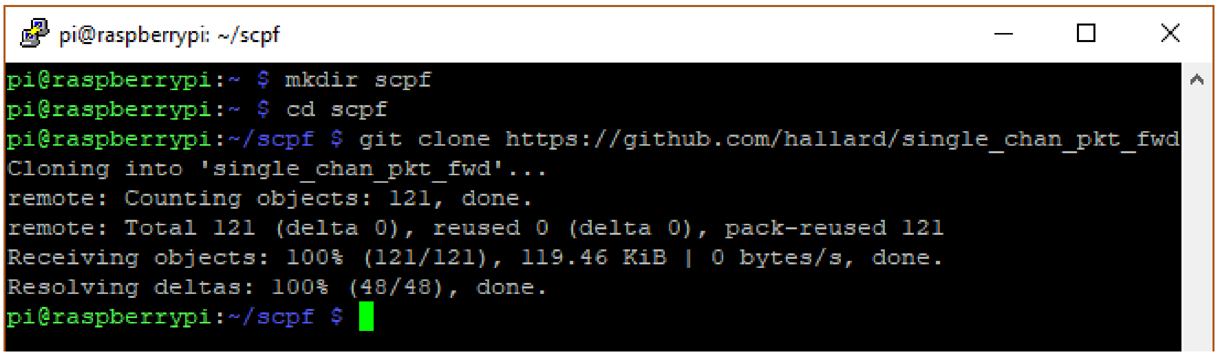
```
pi@raspberrypi: ~
pi@raspberrypi:~ $ sudo apt-get install git
```

Answer Y (J) on the asked question.

4.3.8.3 Single channel packet software

Commands:

- cd ~
- mkdir scpf
- cd scpf
- git clone https://github.com/hallard/single_chan_pkt_fwd

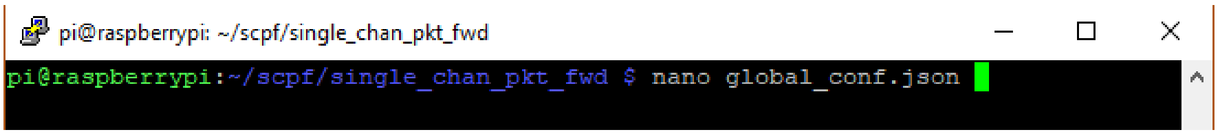


pi@raspberrypi: ~/scpf

```
pi@raspberrypi:~ $ mkdir scpf
pi@raspberrypi:~ $ cd scpf
pi@raspberrypi:~/scpf $ git clone https://github.com/hallard/single_chan_pkt_fwd
Cloning into 'single_chan_pkt_fwd'...
remote: Counting objects: 121, done.
remote: Total 121 (delta 0), reused 0 (delta 0), pack-reused 121
Receiving objects: 100% (121/121), 119.46 KiB | 0 bytes/s, done.
Resolving deltas: 100% (48/48), done.
pi@raspberrypi:~/scpf $
```

Change global.conf

Command: nano global_conf.json



pi@raspberrypi: ~/scpf/single_chan_pkt_fwd

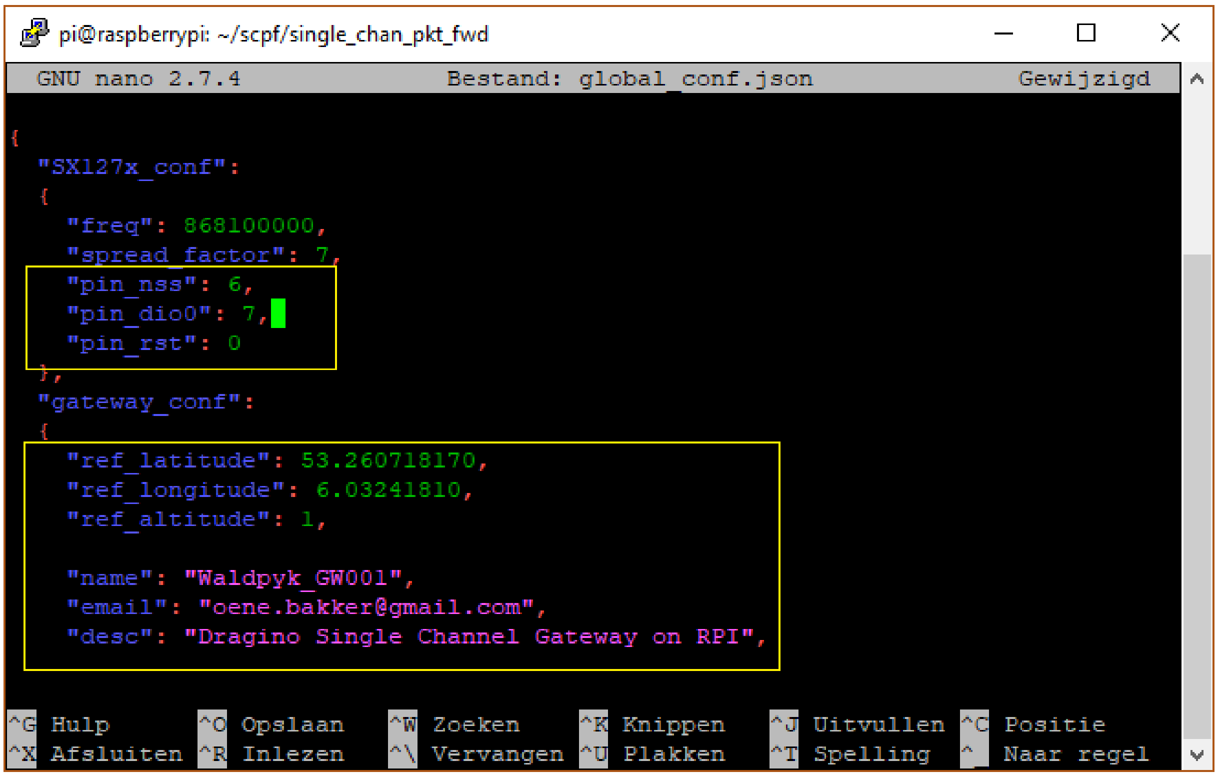
```
pi@raspberrypi:~/scpf/single_chan_pkt_fwd $ nano global_conf.json
```

For the Dragino, the following changes must be made to the pin configuration:

- "pin_nss": 6,
- "pin_dio0": 7,
- "pin_rst": 0

It is also useful to adjust the gateway configuration. Here we are a bit ahead on the TTN configuration (chapter 5).

If you know the GPS position of the gateway, it can be entered here (optional).



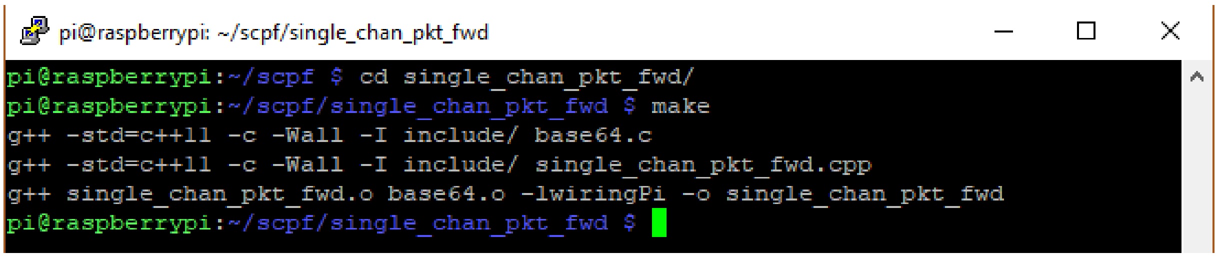
```
pi@raspberrypi: ~/scpf/single_chan_pkt_fwd
  GNU nano 2.7.4            Bestand: global_conf.json            Gewijzigd

{
  "SX127x_conf":
  {
    "freq": 868100000,
    "spread_factor": 7,
    "pin_nss": 6,
    "pin_dio0": 7,
    "pin_rst": 0
  },
  "gateway_conf":
  {
    "ref_latitude": 53.260718170,
    "ref_longitude": 6.03241810,
    "ref_altitude": 1,

    "name": "Waldpyk_GW001",
    "email": "oene.bakker@gmail.com",
    "desc": "Dragino Single Channel Gateway on RPI",

^G Hulp      ^O Opslaan  ^W Zoeken     ^K Knippen  ^J Uitvullen ^C Positie
^X Afsluiten ^R Inlezen  ^\ Vervangen  ^U Plakken  ^T Spelling  ^_ Naar regel
```

Save the file with Ctrl + X, J and Enter.

The software now has to be compiled and linked.

Commands:

- cd single_chan_pkt_fwd
- make



```
pi@raspberrypi: ~/scpf/single_chan_pkt_fwd
pi@raspberrypi:~/scpf $ cd single_chan_pkt_fwd/
pi@raspberrypi:~/scpf/single_chan_pkt_fwd $ make
g++ -std=c++11 -c -Wall -I include/ base64.c
g++ -std=c++11 -c -Wall -I include/ single_chan_pkt_fwd.cpp
g++ single_chan_pkt_fwd.o base64.o -lwiringPi -o single_chan_pkt_fwd
pi@raspberrypi:~/scpf/single_chan_pkt_fwd $
```

Note:
For TTN you can use the following three channels (channels):

- Channel 0 = “freq” : 868100000
- Channel 1 = “freq” : 868300000
- Channel 2 = “freq” : 868500000

4.3.9 Limitations

This gateway solution has the following limitations:

1. The gateway has only one channel. A LoRaWAN compatible gateway has 8.
2. The gateway works with only one spreading factor (SF).
3. The gateway does not (yet) have a downlink function *)

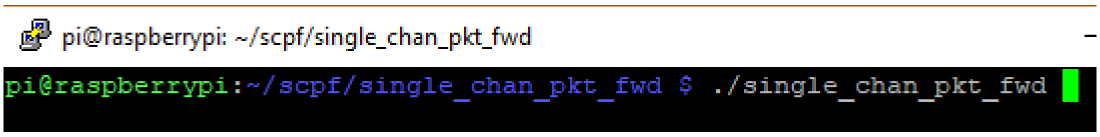
*) There is a downlink solution but for that you need two Raspberry Pi's and two Draginos.

Normally nodes (end devices) will use all three channels. This means that in principle 1 in 3 messages are picked up and processed by the Gateway. Our gateway will only receive messages on channel 0. Changing channels is called frequency hopping. In this book this is solved by having the node only send messages channel 0 (so channels 1 and 2 are skipped).

In addition, the gateway and node must use the same spreading factor. In this book we use SF7.

4.4 Start gateway

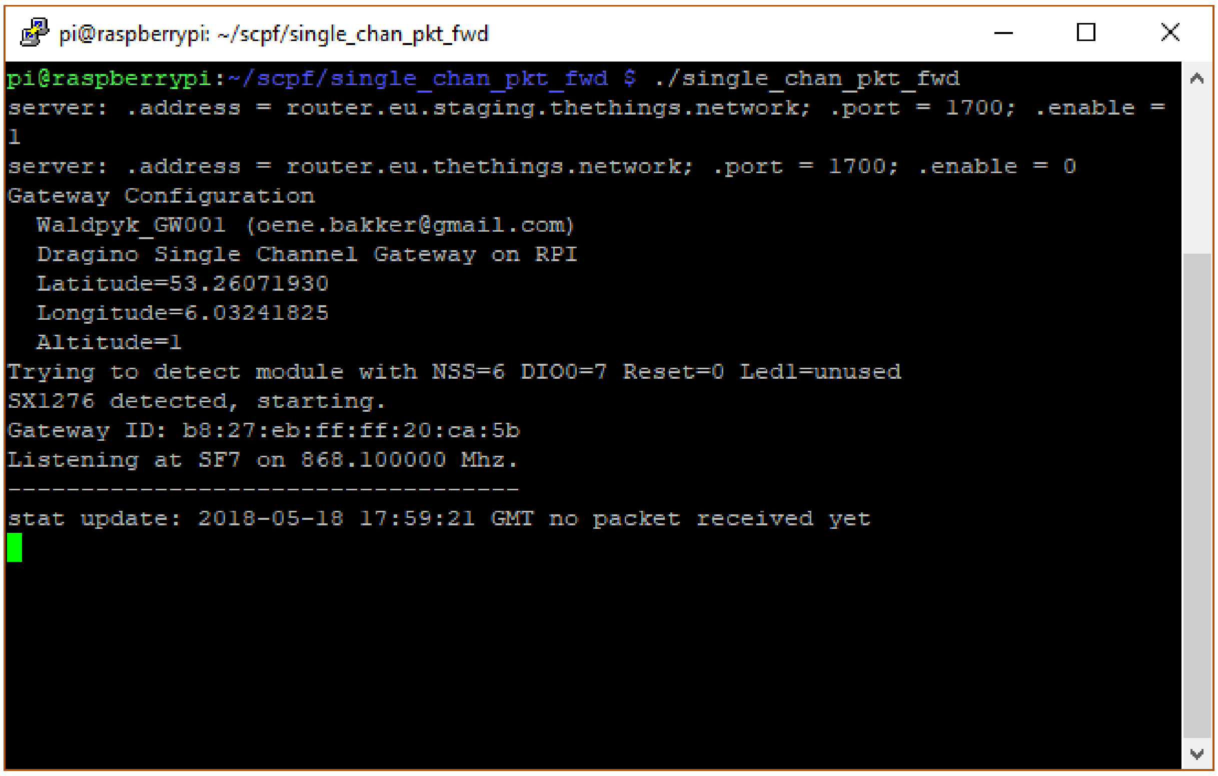
The gateway is started with the command: ./single_chan_pkt_fwd



pi@raspberrypi: ~/scpf/single_chan_pkt_fwd

```
pi@raspberrypi:~/scpf/single_chan_pkt_fwd $ ./single_chan_pkt_fwd
```

Result:



pi@raspberrypi: ~/scpf/single_chan_pkt_fwd

```
pi@raspberrypi:~/scpf/single_chan_pkt_fwd $ ./single_chan_pkt_fwd
server: .address = router.eu.staging.thethings.network; .port = 1700; .enable =
1
server: .address = router.eu.thethings.network; .port = 1700; .enable = 0
Gateway Configuration
  Waldpyk_GW001 (oene.bakker@gmail.com)
  Dragino Single Channel Gateway on RPI
  Latitude=53.26071930
  Longitude=6.03241825
  Altitude=1
Trying to detect module with NSS=6 DIO0=7 Reset=0 Led1=unused
SX1276 detected, starting.
Gateway ID: b8:27:eb:ff:ff:20:ca:5b
Listening at SF7 on 868.100000 Mhz.
-----------------------------------
stat update: 2018-05-18 17:59:21 GMT no packet received yet
```

Please make a note of the Gateway ID (we need it later on):

```
SX1276 detected, starting.
Gateway ID: b8:27:eb:ff:ff:20:ca:5b
Listening at SF7 on 868.100000 MHz.
```

Remove the ":" by a space and convert the lowercase letters to uppercase letters. In this example, the gateway is declared as: B8 27 EB FF FF 20 CA 5B

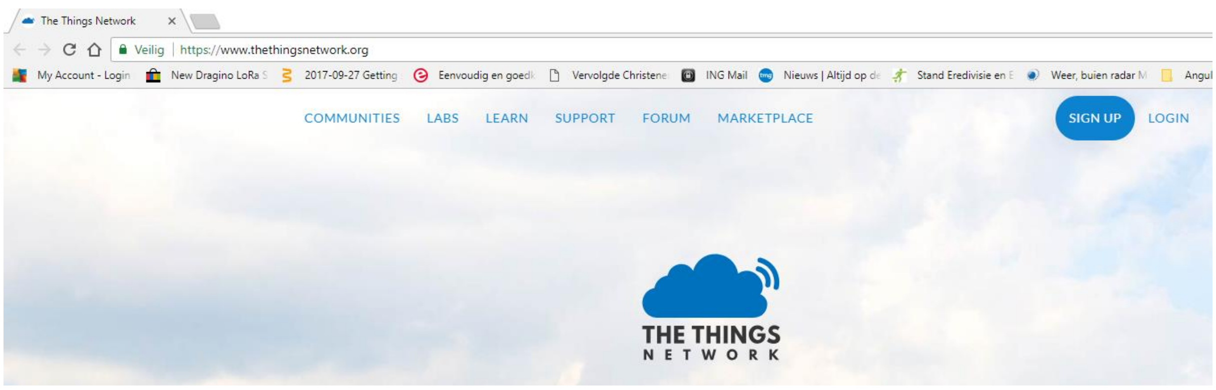
The gateway is now ready for use. In the next step, the gateway will be announced in TTN.

5 Add the gateway to TTN

5.1 Create a TTN account

Open a browser and go to: https://www.thethingsnetwork.org/

Click SIGN UP and create an account.

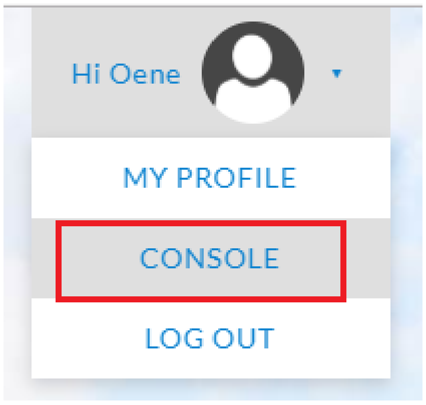


5.2 TTN add a gateway

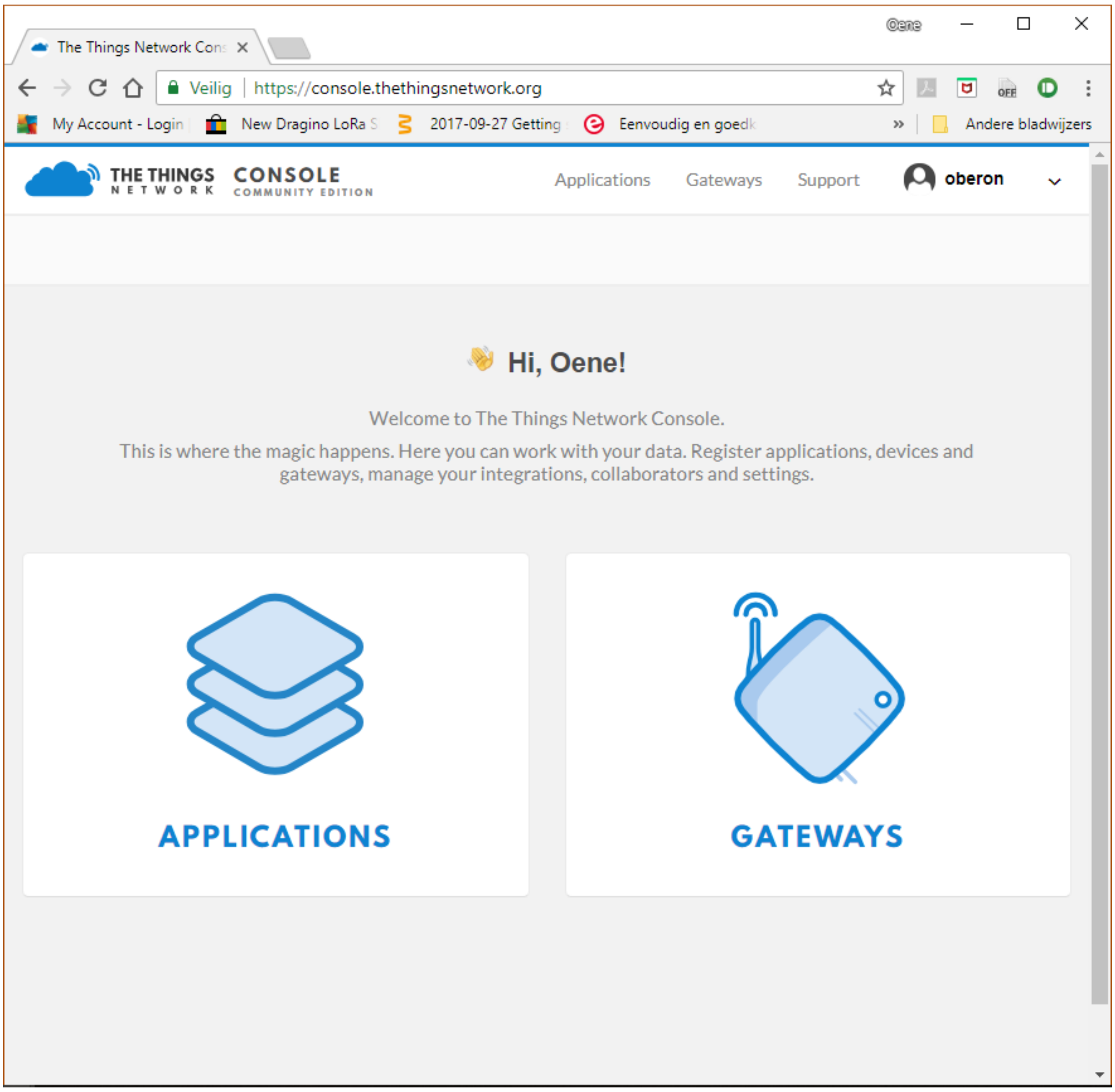
Open a browser and go to: https://www.thethingsnetwork.org/

Click LOGIN.

Select CONSOLE from the dropdown (right top of the page):



Click GATEWAYS to create a new gateway:

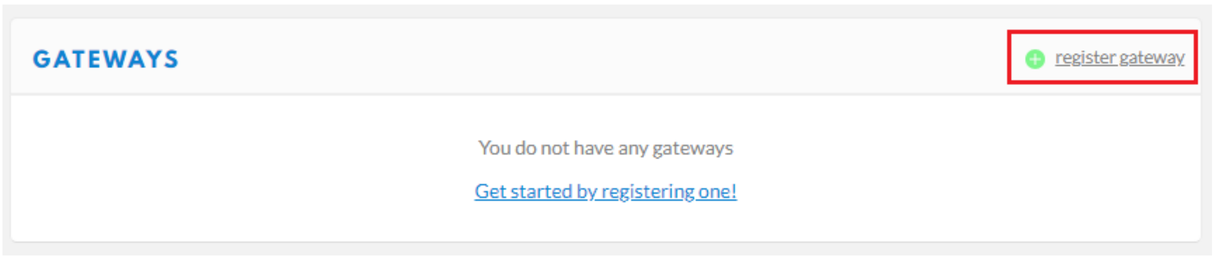


Hi, Oene!

Welcome to The Things Network Console.

This is where the magic happens. Here you can work with your data. Register applications, devices and gateways, manage your integrations, collaborators and settings.

Click register gateway:

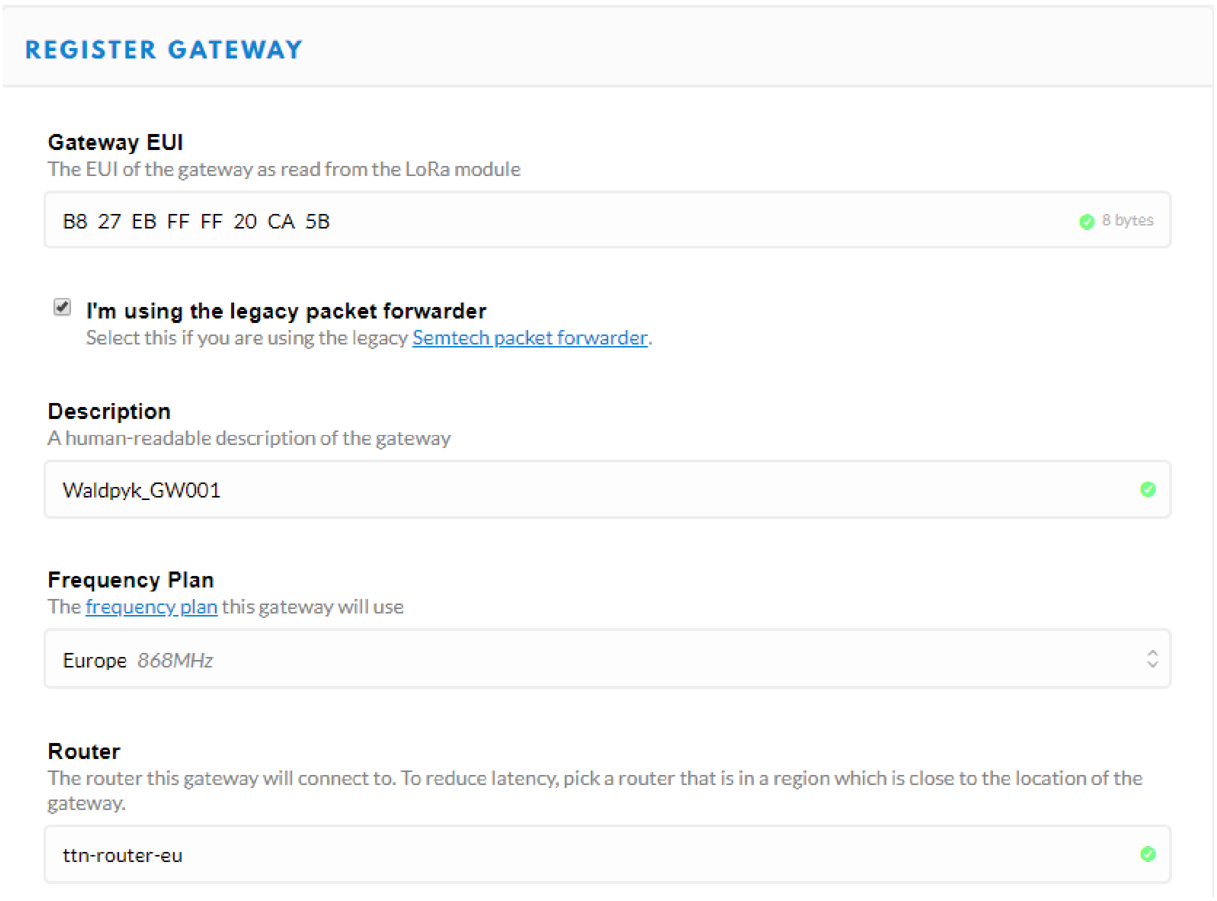


You do not have any gateways

Get started by registering one!

Input fields:

- Gateway ID: B8 27 EB FF FF 20 CA 5B
- Check I'm using the legacy packet forwarder
- Description: choose your own description
- Frequency Plan: Europe 868MHz
- Router: ttn-router-eu (wordt automatisch gevuld)



REGISTER GATEWAY

Gateway EUI
The EUI of the gateway as read from the LoRa module

B8 27 EB FF FF 20 CA 5B ✓ 8 bytes

☑ **I'm using the legacy packet forwarder**
Select this if you are using the legacy Semtech packet forwarder.

Description
A human-readable description of the gateway

Waldpyk_GW001 ✓

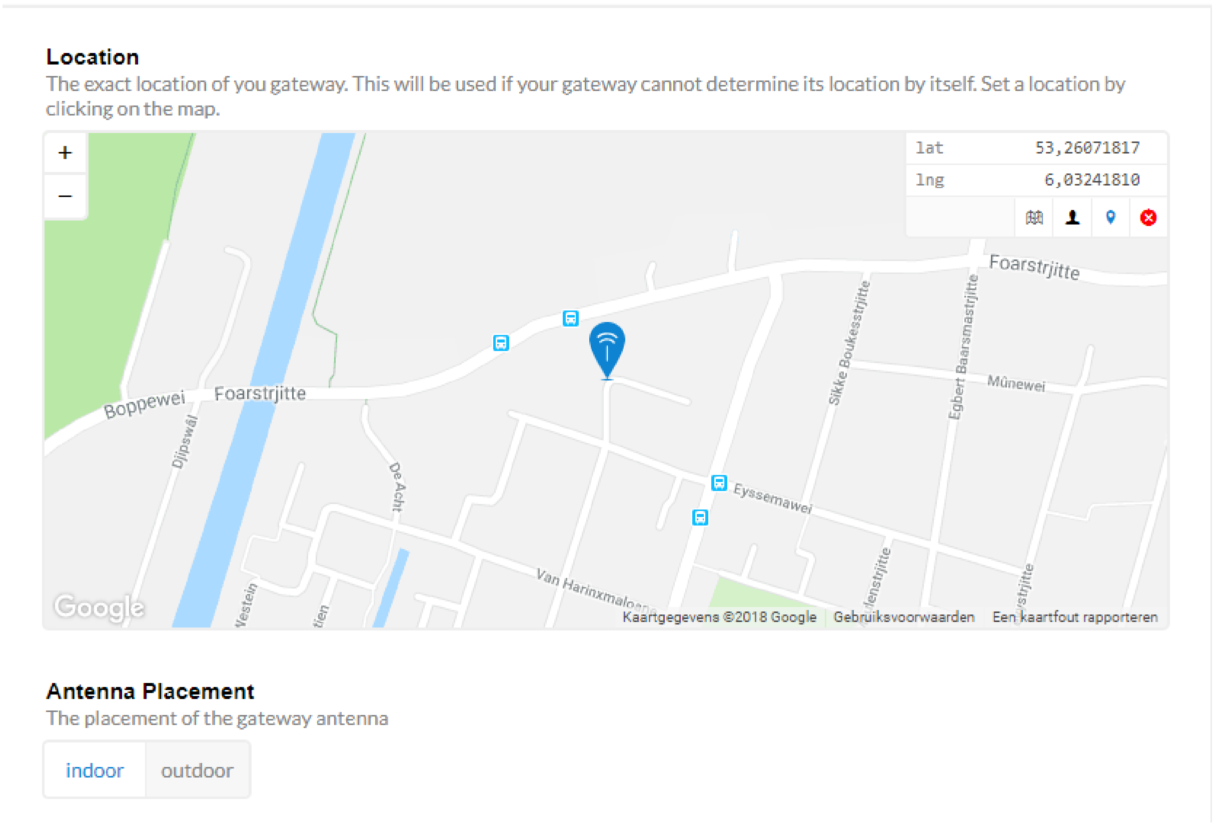
Frequency Plan
The frequency plan this gateway will use

Europe *868MHz*

Router
The router this gateway will connect to. To reduce latency, pick a router that is in a region which is close to the location of the gateway.

ttn-router-eu ✓

Note: Some of the values have already been discussed in section 4.3.8.3 (global_conf.json).


Location

The exact location of you gateway. This will be used if your gateway cannot determine its location by itself. Set a location by clicking on the map.

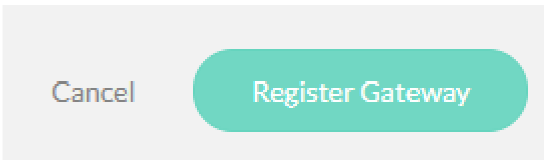
lat	53,26071817
lng	6,03241810

Antenna Placement

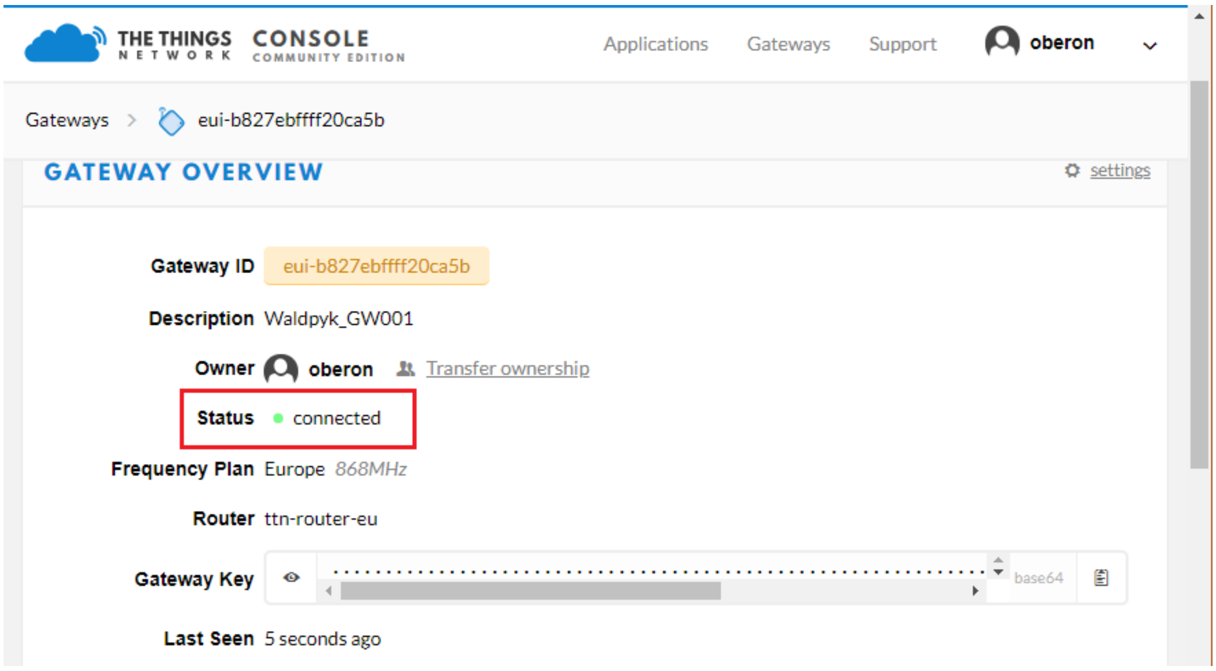
The placement of the gateway antenna

indoor | outdoor

Click Register Gateway at the bottom of the page:



If the Raspberry Pi Gateway is still active, the following Overview will be shown in TTN:



Gateway ID eui-b827ebffff20ca5b

Description Waldpyk_GW001

Owner oberon Transfer ownership

Status ● connected

Frequency Plan Europe 868MHz

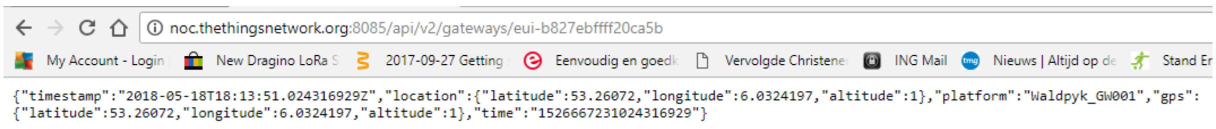
Router ttn-router-eu

Gateway Key base64

Last Seen 5 seconds ago

You also can check the status in the browser with the link:
http://noc.thethingsnetwork.org:8085/api/v2/gateways/eui-xxxxxxxxxxxxxxxx

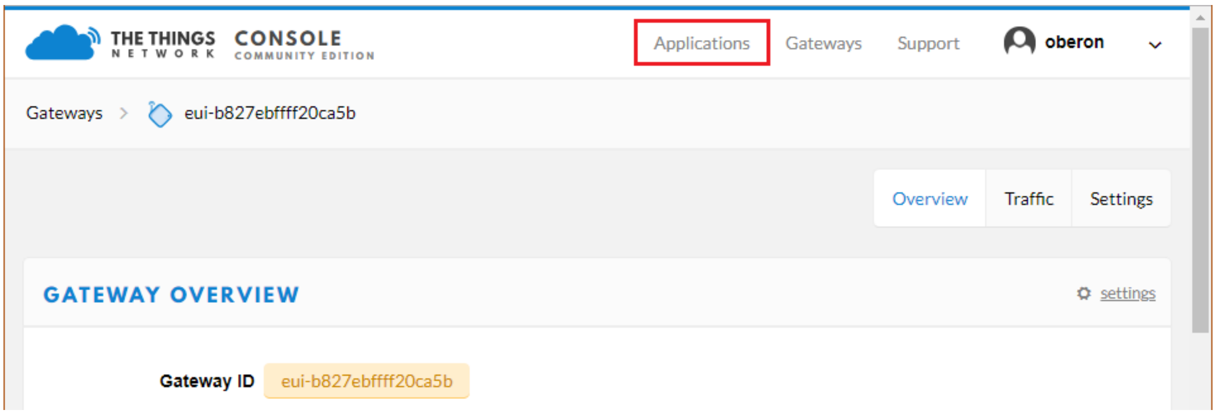
In this example:
http://noc.thethingsnetwork.org:8085/api/v2/gateways/eui-b827ebffff20ca5b



```
{"timestamp":"2018-05-18T18:13:51.024316929Z","location":{"latitude":53.26072,"longitude":6.0324197,"altitude":1},"platform":"Waldpyk_GW001","gps":
{"latitude":53.26072,"longitude":6.0324197,"altitude":1},"time":"1526667231024316929"}
```

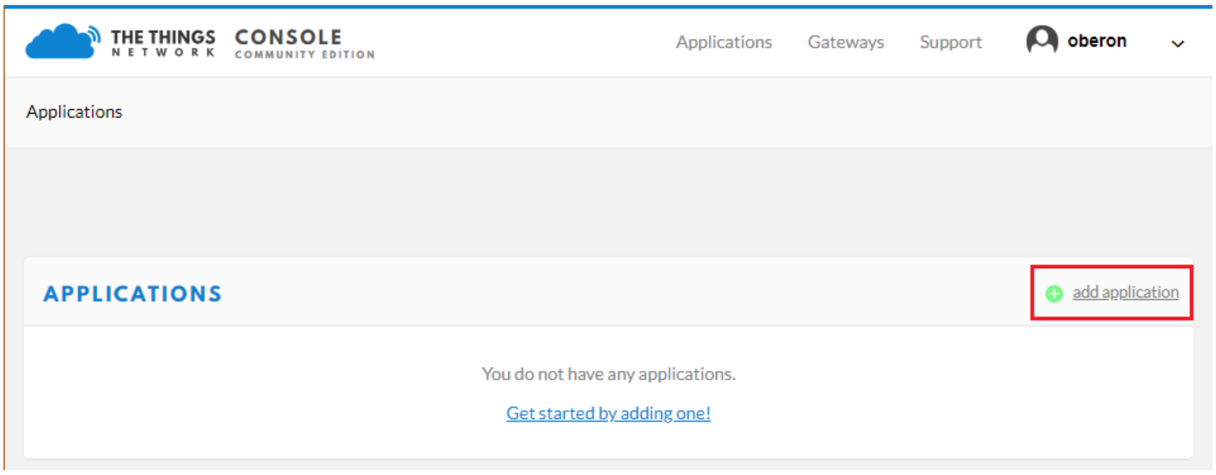
5.3 Add application

Click Applications:



Gateway ID eui-b827ebffff20ca5b

Click add application:

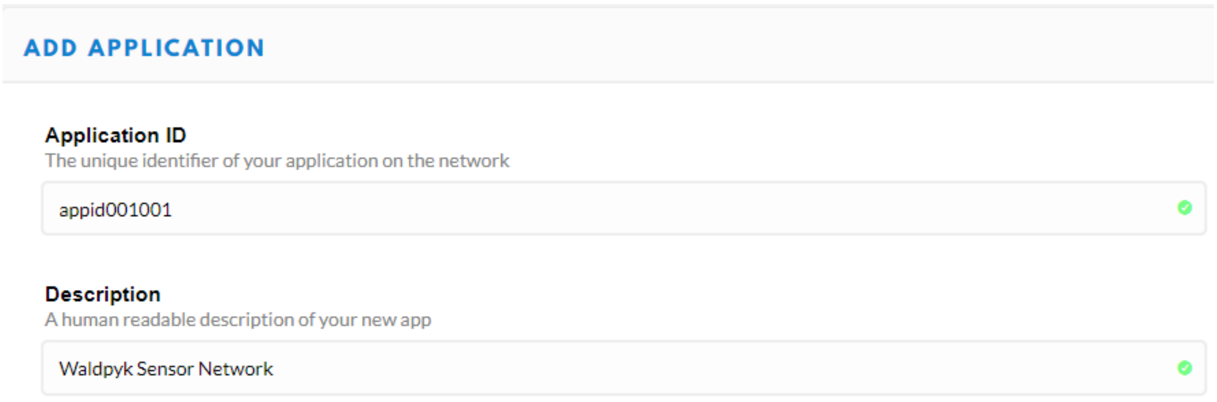


You do not have any applications.

Get started by adding one!

Input fields:

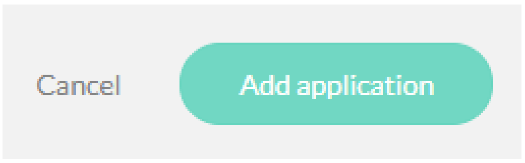
- Application ID: choose your own, but only use alphanumerics, no uppercase letters and use only – and _.
- Description: choose your own.



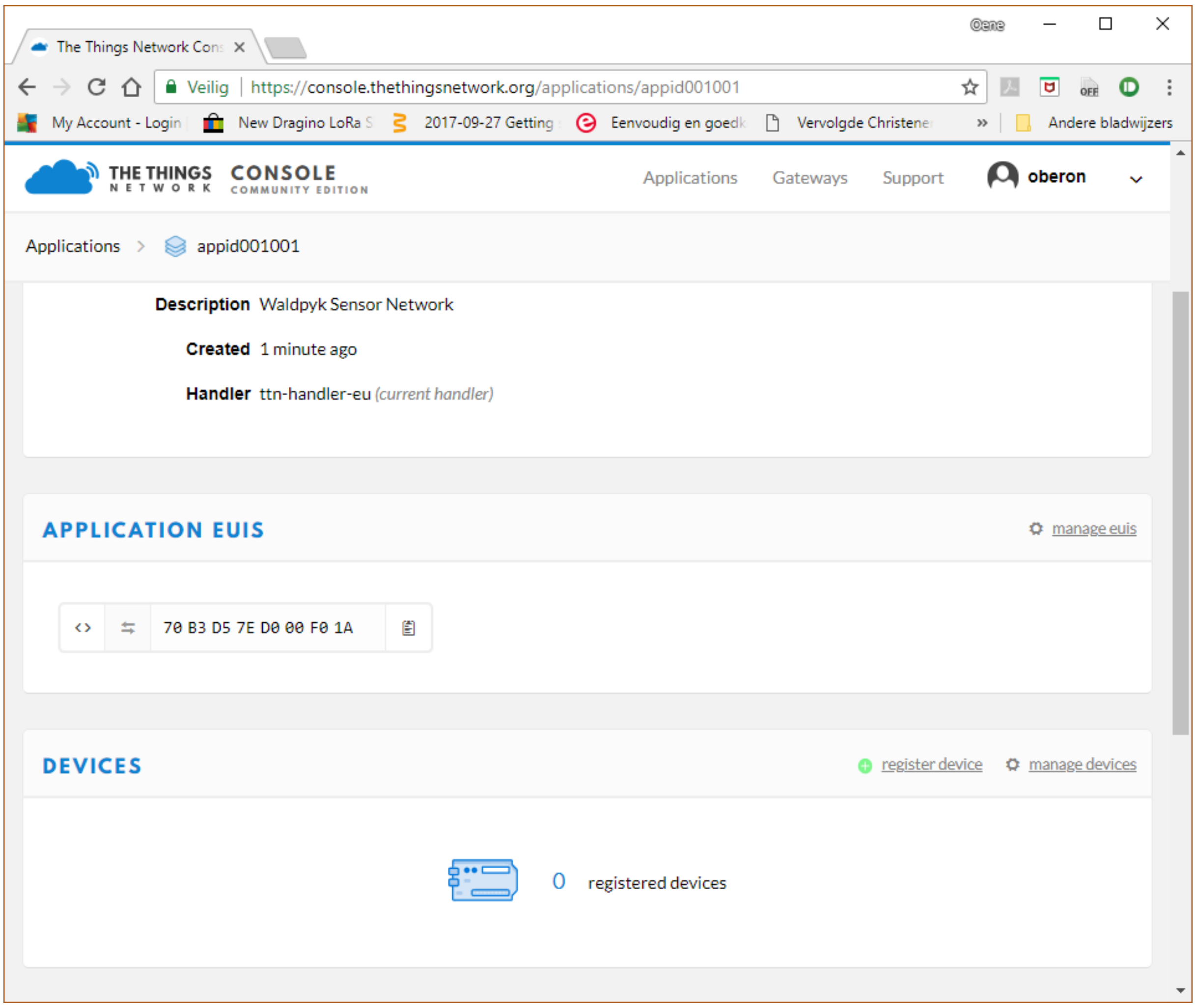
Application ID
The unique identifier of your application on the network
appid001001

Description
A human readable description of your new app
Waldpyk Sensor Network

Scroll down and click Add application.



The application is added:



Description Waldpyk Sensor Network

Created 1 minute ago

Handler ttn-handler-eu *(current handler)*

APPLICATION EUIS

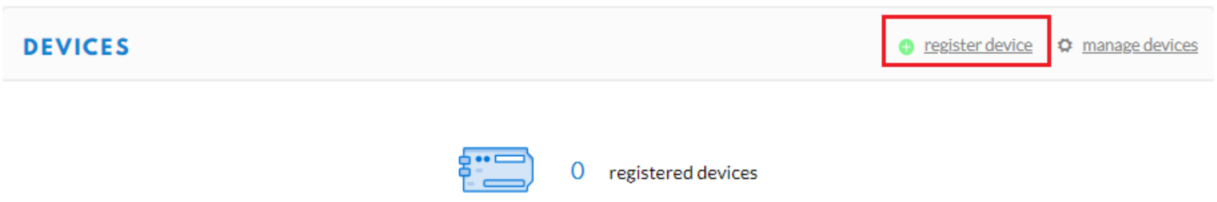
```
70 B3 D5 7E D0 00 F0 1A
```

DEVICES

0 registered devices

5.4 Add device

Click register device:



DEVICES

register device

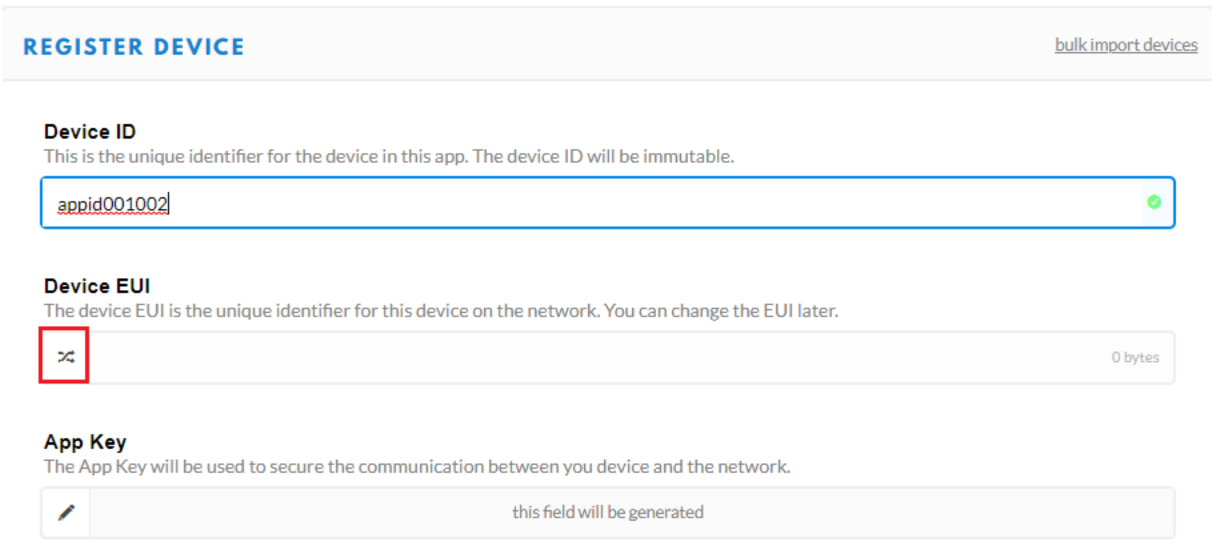
manage devices

0 registered devices

Input field:

- Device ID: Choose your own.

Klik op om het Device EUI automatisch te genereren.



REGISTER DEVICE

bulk import devices

Device ID
This is the unique identifier for the device in this app. The device ID will be immutable.

appid001002

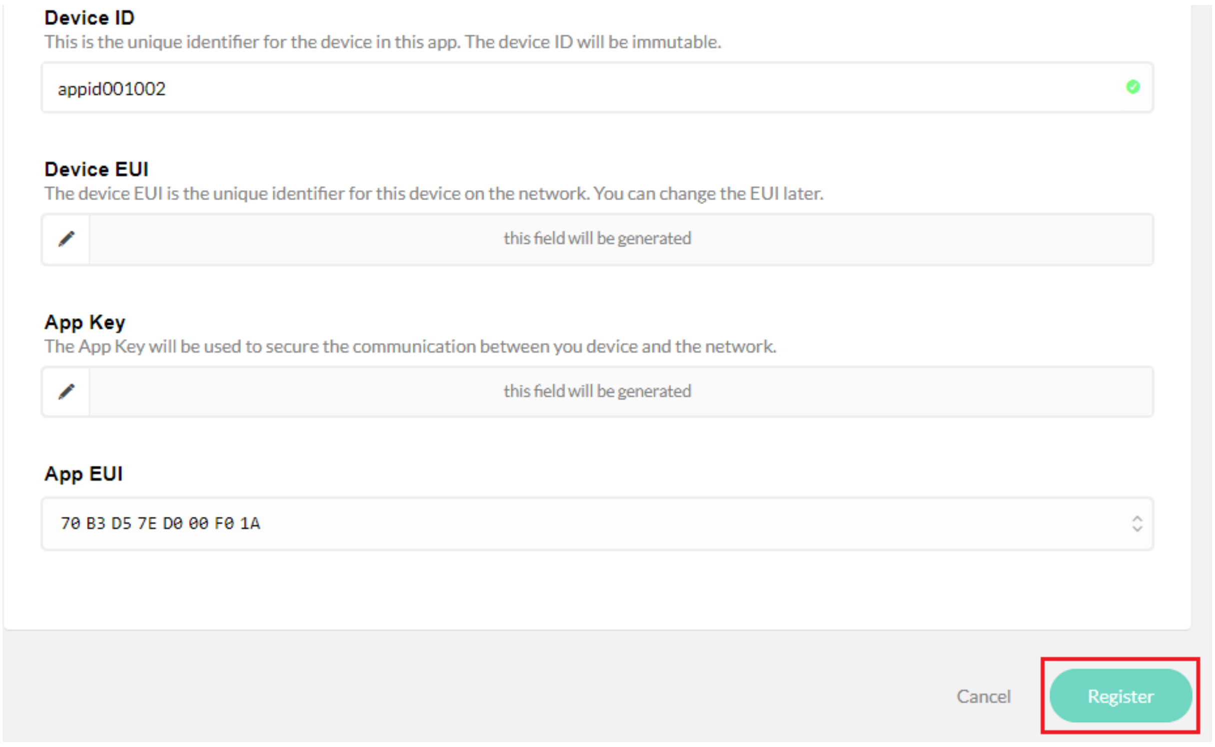
Device EUI
The device EUI is the unique identifier for this device on the network. You can change the EUI later.

0 bytes

App Key
The App Key will be used to secure the communication between you device and the network.

this field will be generated

Click Register:

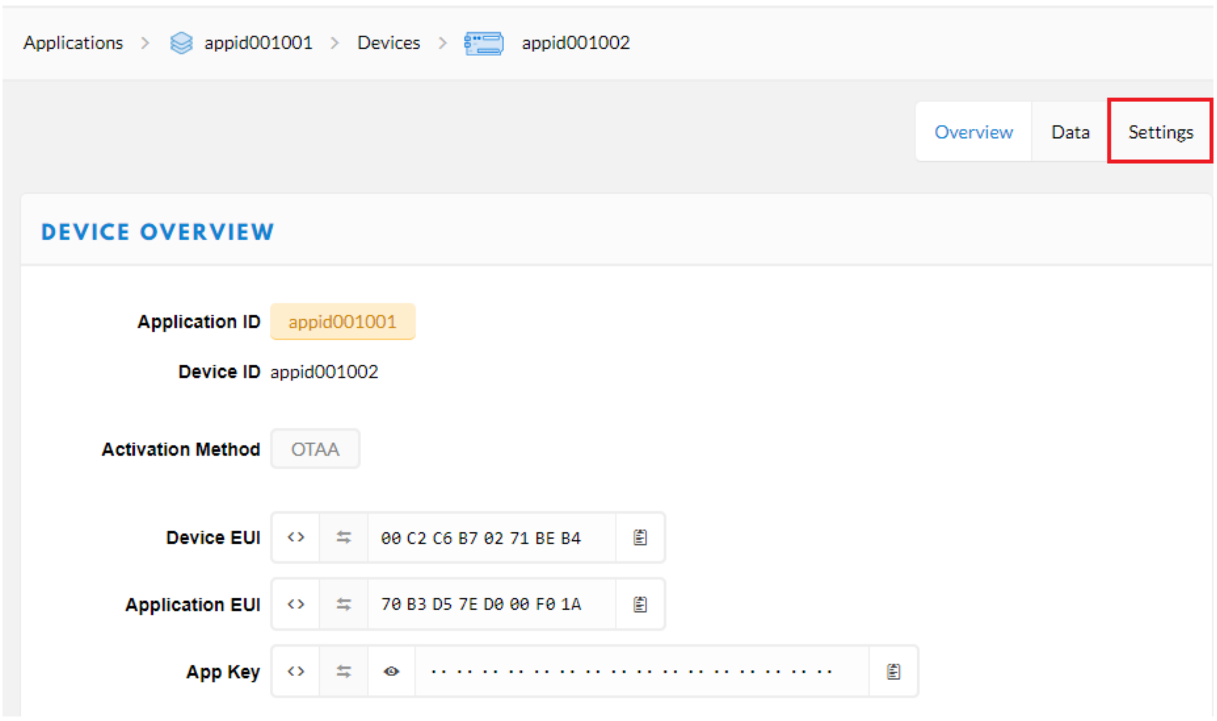

Device ID
This is the unique identifier for the device in this app. The device ID will be immutable.
appid001002

Device EUI
The device EUI is the unique identifier for this device on the network. You can change the EUI later.
this field will be generated

App Key
The App Key will be used to secure the communication between you device and the network.
this field will be generated

App EUI
70 B3 D5 7E D0 00 F0 1A

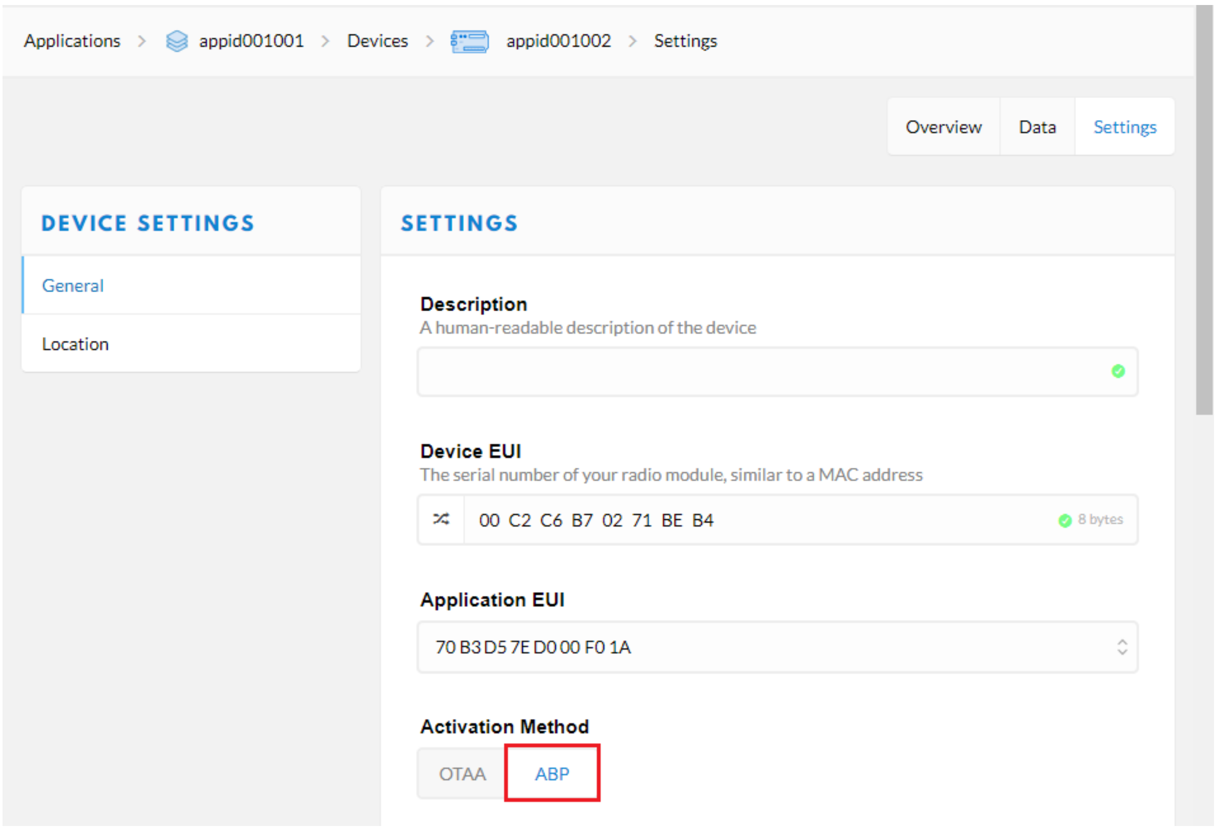
Click Settings:



DEVICE OVERVIEW

Application ID appid001001
Device ID appid001002
Activation Method OTAA
Device EUI 00 C2 C6 B7 02 71 BE B4
Application EUI 70 B3 D5 7E D0 00 F0 1A
App Key ································

Select ABP (Activation by Personalization)


Applications > appid001001 > Devices > appid001002 > Settings

Overview | Data | Settings

DEVICE SETTINGS

General

Location

SETTINGS

Description
A human-readable description of the device

Device EUI
The serial number of your radio module, similar to a MAC address

00 C2 C6 B7 02 71 BE B4 — 8 bytes

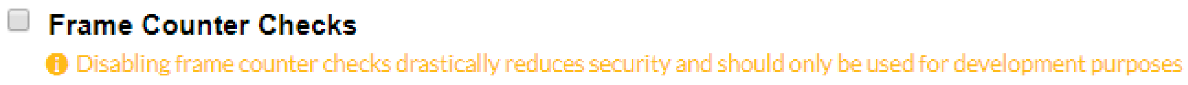
Application EUI

70 B3 D5 7E D0 00 F0 1A

Activation Method

OTAA [✓ ABP]

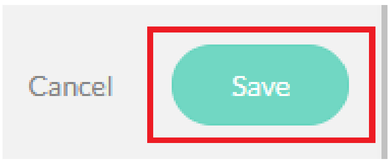
In ABP mode it is advisable to switch off the Frame Counter Checks:


☐ **Frame Counter Checks**

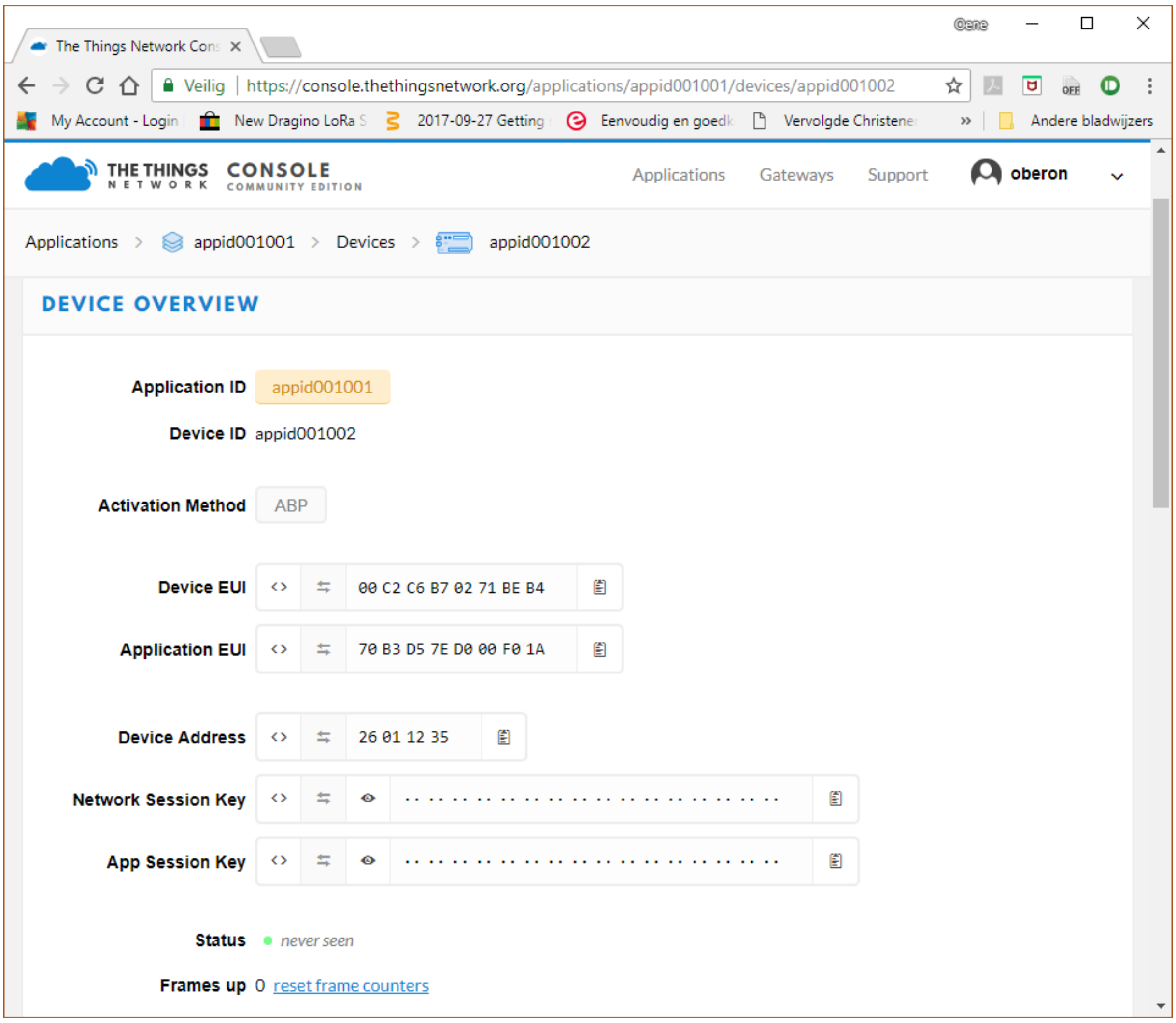
Disabling frame counter checks drastically reduces security and should only be used for development purposes

If the Frame Counter Checks is not turned off, data may not be shown on the console.

Scroll down and click Save



Scroll down where the various keys are shown:



https://console.thethingsnetwork.org/applications/appid001001/devices/appid001002

Applications > appid001001 > Devices > appid001002

DEVICE OVERVIEW

Application ID appid001001

Device ID appid001002

Activation Method ABP

Device EUI 00 C2 C6 B7 02 71 BE B4

Application EUI 70 B3 D5 7E D0 00 F0 1A

Device Address 26 01 12 35

Network Session Key ··································

App Session Key ··································

Status never seen

Frames up 0 reset frame counters

Make a copy of the keys with 📋 :

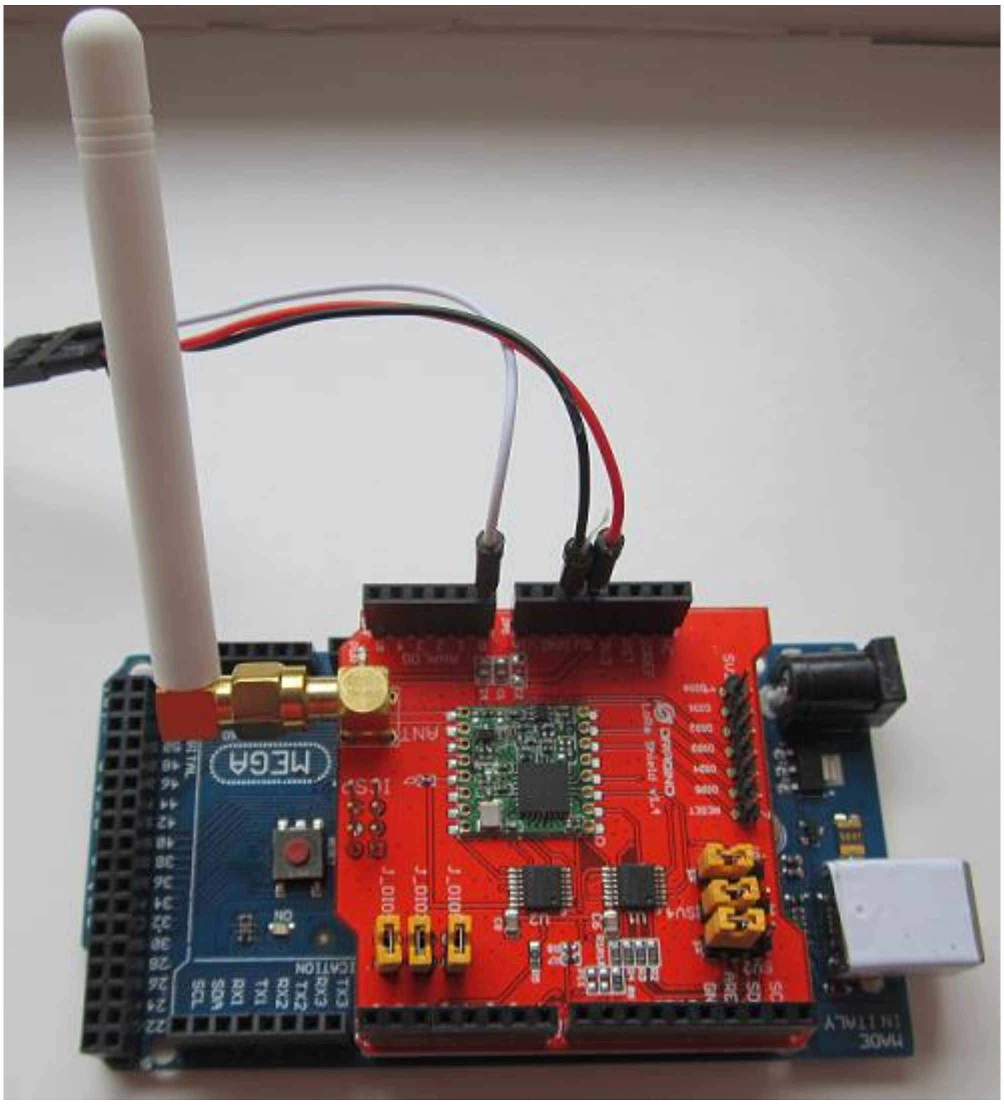
Device EUI:	00C2C6B70271BEB4
Application EUI:	70B3D57ED000F01A
Device Address:	26011235
Network Session Key:	FD 19 C6 3D 0D A9 78 E3 8D C7 A7 5D 9C 3A D7 2D
App Session Key:	93 4C 4E 59 1C A9 6C 20 EB C8 F4 C8 55 99 5A 40

6 Node

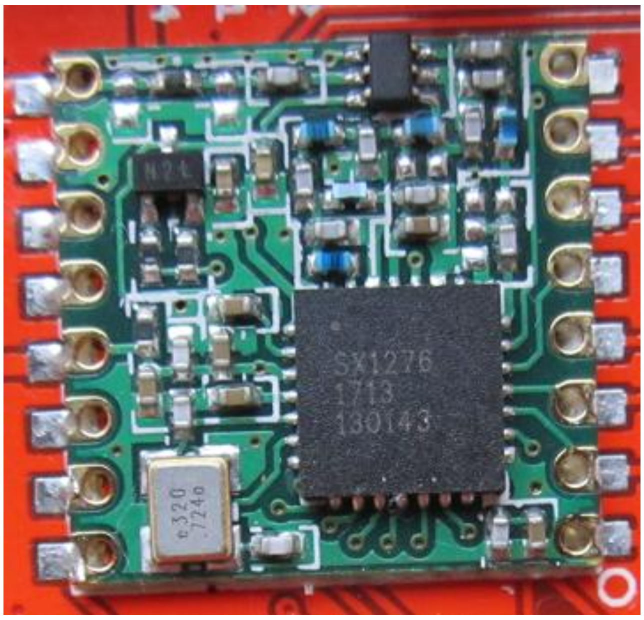
6.1 Arduino hardware setup

Required:
• Arduino Mega
• Dragino LoRa hat

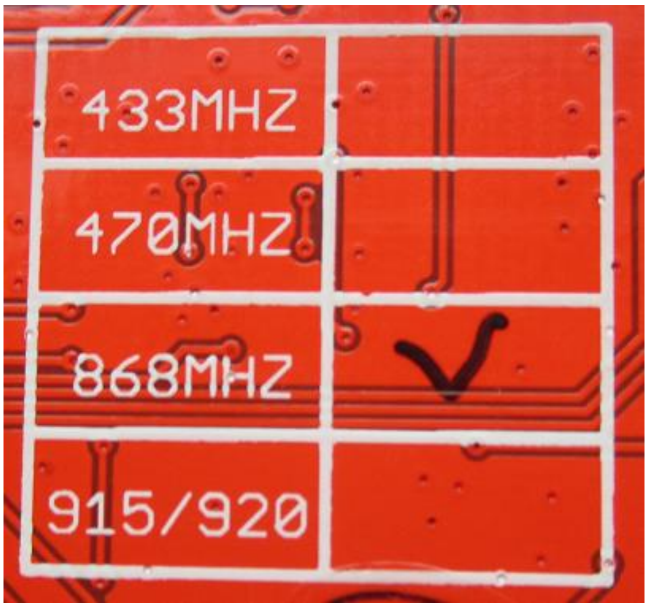
The installation is simple, the Dragino can be placed on the Arduino Mega without any further installation.



This Dragino uses a SX1276 chipset.



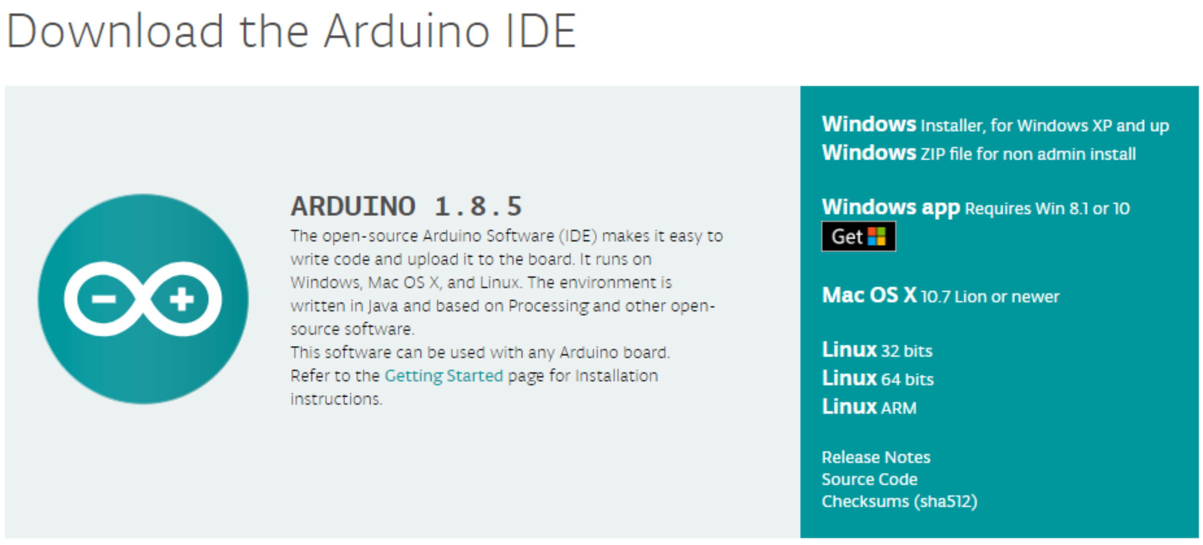
And works on 868 MHz.



6.2 Arduino IDE

Download and install the Arduino IDE if necessary.

Download link: https://www.arduino.cc/en/Main/Software



Download the Arduino IDE

ARDUINO 1.8.5

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.
This software can be used with any Arduino board.
Refer to the Getting Started page for Installation instructions.

Windows Installer, for Windows XP and up
Windows ZIP file for non admin install

Windows app Requires Win 8.1 or 10
Get

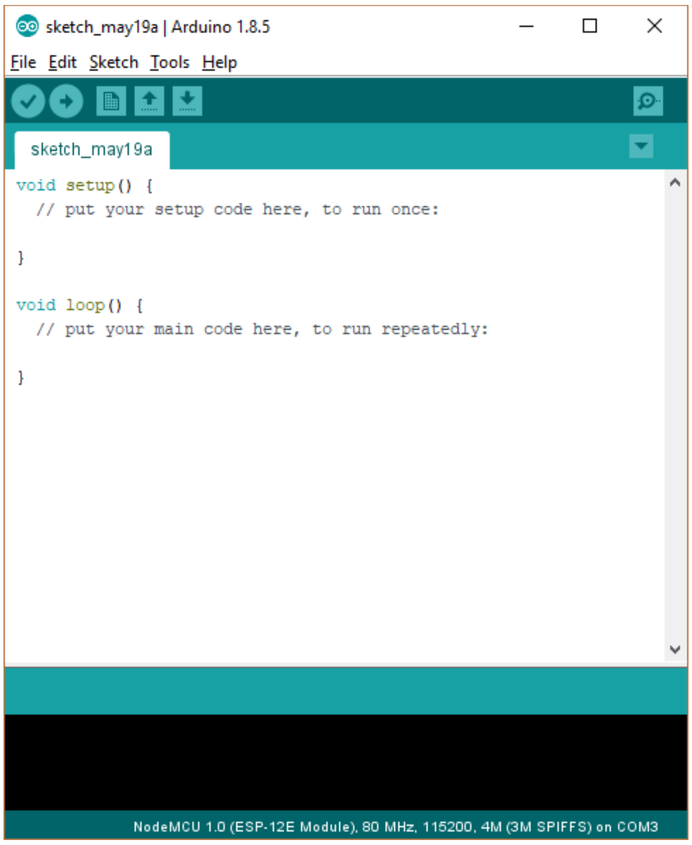
Mac OS X 10.7 Lion or newer

Linux 32 bits
Linux 64 bits
Linux ARM

Release Notes
Source Code
Checksums (sha512)

6.3 TTN library

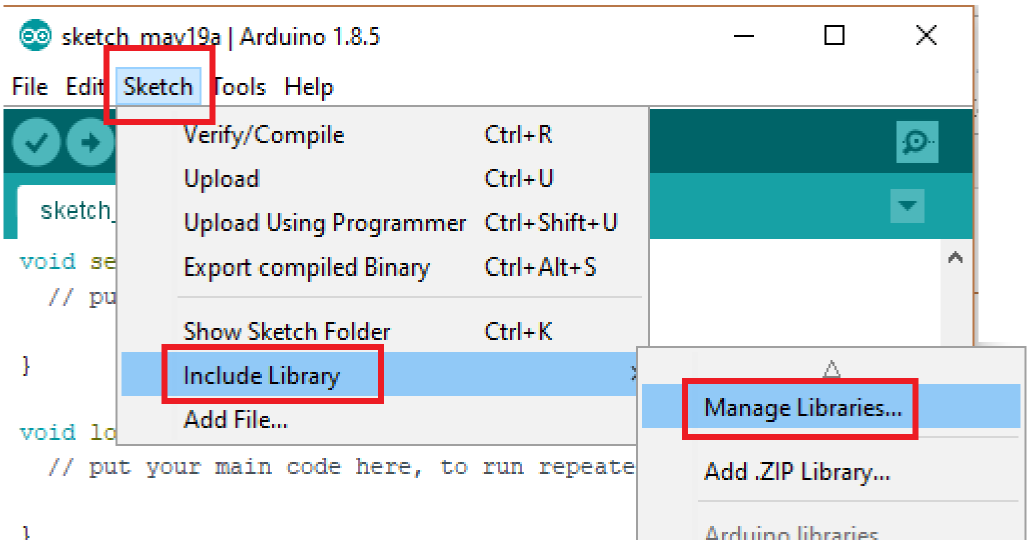
Start the Arduino IDE.



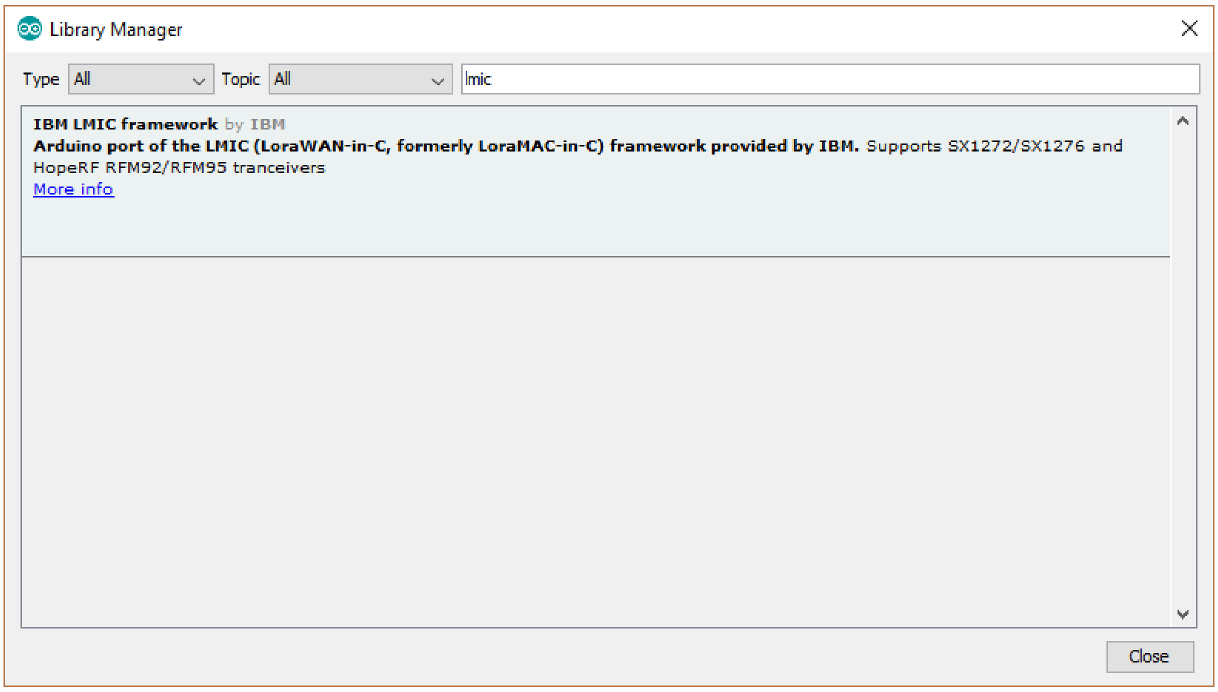
```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

NodeMCU 1.0 (ESP-12E Module), 80 MHz, 115200, 4M (3M SPIFFS) on COM3

Select menu, Sketch, Include Library and Manage Libraries

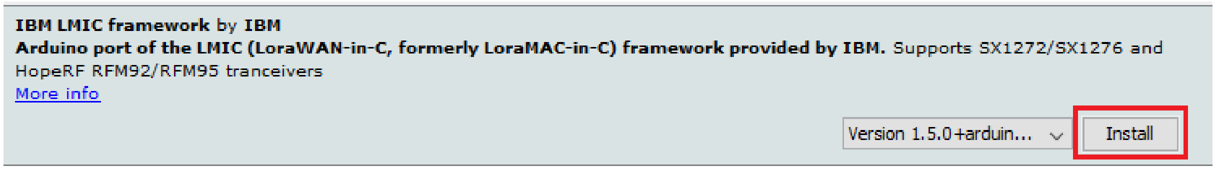

sketch_may19a | Arduino 1.8.5
File Edit Sketch Tools Help
Verify/Compile Ctrl+R
Upload Ctrl+U
Upload Using Programmer Ctrl+Shift+U
Export compiled Binary Ctrl+Alt+S
Show Sketch Folder Ctrl+K
Include Library
Add File...
Manage Libraries...
Add .ZIP Library...
Arduino libraries

Look for lmic



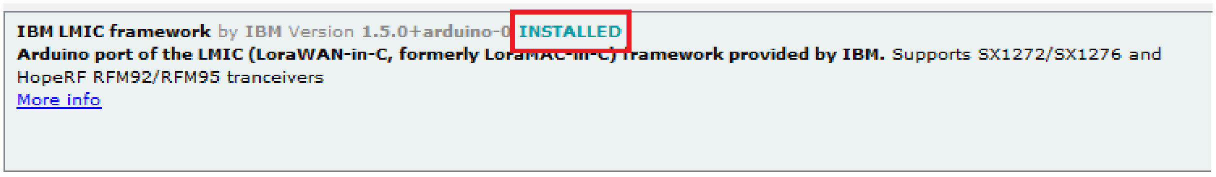
IBM LMIC framework by IBM
Arduino port of the LMIC (LoraWAN-in-C, formerly LoraMAC-in-C) framework provided by IBM. Supports SX1272/SX1276 and HopeRF RFM92/RFM95 tranceivers

Select IBM LMIC framework and Install



IBM LMIC framework by IBM
Arduino port of the LMIC (LoraWAN-in-C, formerly LoraMAC-in-C) framework provided by IBM. Supports SX1272/SX1276 and HopeRF RFM92/RFM95 tranceivers

Result:



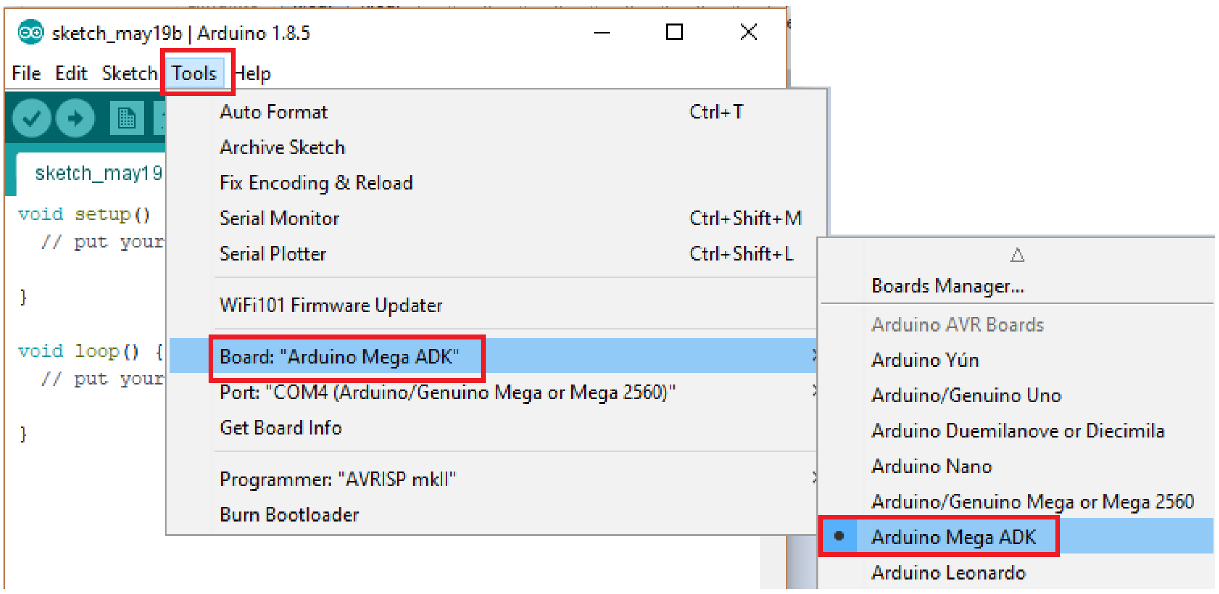
IBM LMIC framework by IBM Version 1.5.0+arduino-0 INSTALLED
Arduino port of the LMIC (LoraWAN-in-C, formerly LoraMAC-in-C) framework provided by IBM. Supports SX1272/SX1276 and HopeRF RFM92/RFM95 tranceivers

6.4 Hello World sketch

With the Arduino sketch in this example a LoRaWAN message is sent with the text:
`Hello, world!`

Connect the Arduino Mega to the PC or laptop.

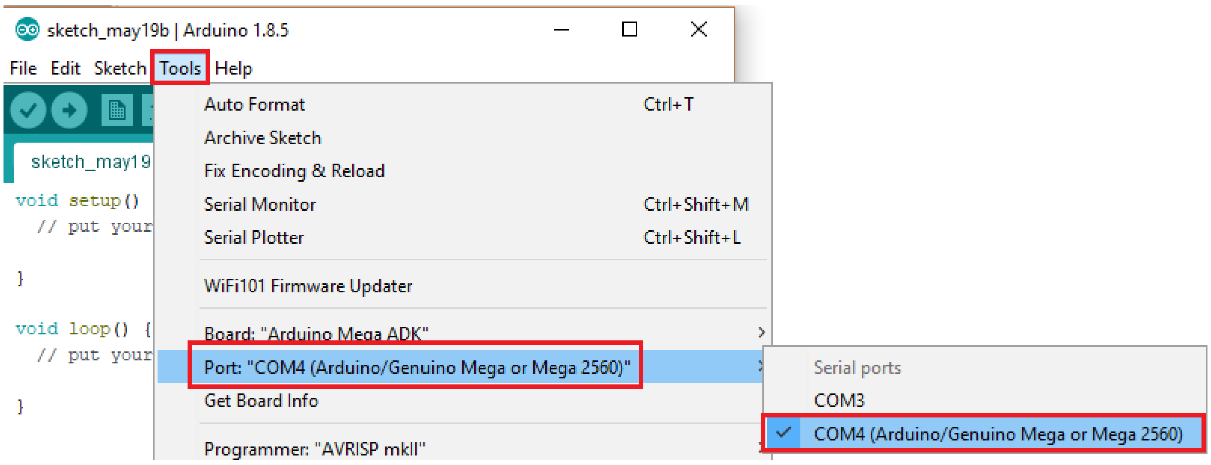
Select Tools in the menu, Board, Arduino Mega ADK



sketch_may19b | Arduino 1.8.5
File Edit Sketch Tools Help
sketch_may19
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
Board: "Arduino Mega ADK"
Port: "COM4 (Arduino/Genuino Mega or Mega 2560)"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Boards Manager...
Arduino AVR Boards
Arduino Yún
Arduino/Genuino Uno
Arduino Duemilanove or Diecimila
Arduino Nano
Arduino/Genuino Mega or Mega 2560
● Arduino Mega ADK
Arduino Leonardo

```
void setup()
  // put your

}

void loop() {
  // put your

}
```

Select menu, Tools, Port and the active com port, in this example COM4.



sketch_may19b | Arduino 1.8.5
File Edit Sketch Tools Help
sketch_may19
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
Board: "Arduino Mega ADK"
Port: "COM4 (Arduino/Genuino Mega or Mega 2560)"
Get Board Info
Programmer: "AVRISP mkII"
Serial ports
COM3
✓ COM4 (Arduino/Genuino Mega or Mega 2560)

```
void setup()
  // put your

}

void loop() {
  // put your

}
```

6.4.1 Arduino sketch

Copy the following source code from Thomas Telkamp and Matthijs Kooijman and change the following variables:

- NWKSKEY
- APPSKEY and
- DEVADDR

```
/*******************************************************************************
 * Copyright (c) 2015 Thomas Telkamp and Matthijs Kooijman
 *
 * Permission is hereby granted, free of charge, to anyone
 * obtaining a copy of this document and accompanying files,
 * to do whatever they want with them without any restriction,
 * including, but not limited to, copying, modification and redistribution.
 * NO WARRANTY OF ANY KIND IS PROVIDED.
 *
 * This example sends a valid LoRaWAN packet with payload "Hello,
 * world!", using frequency and encryption settings matching those of
 * the (early prototype version of) The Things Network.
 *
 * Note: LoRaWAN per sub-band duty-cycle limitation is enforced (1% in g1,
 *  0.1% in g2).
 *
 * Change DEVADDR to a unique address!
 * See http://thethingsnetwork.org/wiki/AddressSpace
 *
 * Do not forget to define the radio type correctly in config.h.
 *
 *******************************************************************************/

#include <lmic.h>
#include <hal/hal.h>
#include <SPI.h>

// LoRaWAN NwkSKey, network session key
// This is the default Semtech key, which is used by the prototype TTN
// network initially.
static const PROGMEM u1_t NWKSKEY[16] = { 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX,
0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX };

// LoRaWAN AppSKey, application session key
// This is the default Semtech key, which is used by the prototype TTN
// network initially.
static const u1_t PROGMEM APPSKEY[16] = { 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX,
0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX };

// LoRaWAN end-device address (DevAddr)
// See http://thethingsnetwork.org/wiki/AddressSpace
static const u4_t DEVADDR = 0xXXXXXXXX;

// These callbacks are only used in over-the-air activation, so they are
// left empty here (we cannot leave them out completely unless
// DISABLE_JOIN is set in config.h, otherwise the linker will complain).
void os_getArtEui (u1_t* buf) { }
void os_getDevEui (u1_t* buf) { }
void os_getDevKey (u1_t* buf) { }

static uint8_t mydata[] = "Hello, world!";
static osjob_t sendjob;

// Schedule TX every this many seconds (might become longer due to duty
// cycle limitations).
const unsigned TX_INTERVAL = 60;
```

```
// Pin mapping
const lmic_pinmap lmic_pins = {
    .nss = 10,
    .rxtx = LMIC_UNUSED_PIN,
    .rst = 9,
    .dio = {2, 6, 7},
};

void onEvent (ev_t ev) {
    Serial.print(os_getTime());
    Serial.print(": ");
    switch(ev) {
        case EV_SCAN_TIMEOUT:
            Serial.println(F("EV_SCAN_TIMEOUT"));
            break;
        case EV_BEACON_FOUND:
            Serial.println(F("EV_BEACON_FOUND"));
            break;
        case EV_BEACON_MISSED:
            Serial.println(F("EV_BEACON_MISSED"));
            break;
        case EV_BEACON_TRACKED:
            Serial.println(F("EV_BEACON_TRACKED"));
            break;
        case EV_JOINING:
            Serial.println(F("EV_JOINING"));
            break;
        case EV_JOINED:
            Serial.println(F("EV_JOINED"));
            break;
        case EV_RFU1:
            Serial.println(F("EV_RFU1"));
            break;
        case EV_JOIN_FAILED:
            Serial.println(F("EV_JOIN_FAILED"));
            break;
        case EV_REJOIN_FAILED:
            Serial.println(F("EV_REJOIN_FAILED"));
            break;
            break;
        case EV_TXCOMPLETE:
            Serial.println(F("EV_TXCOMPLETE (includes waiting for RX windows)"));
            if(LMIC.dataLen) {
                // data received in rx slot after tx
                Serial.print(F("Data Received: "));
                Serial.write(LMIC.frame+LMIC.dataBeg, LMIC.dataLen);
                Serial.println();
            }
            // Schedule next transmission
            os_setTimedCallback(&sendjob, os_getTime()+sec2osticks(TX_INTERVAL), do_send);
            break;
        case EV_LOST_TSYNC:
            Serial.println(F("EV_LOST_TSYNC"));
            break;
        case EV_RESET:
            Serial.println(F("EV_RESET"));
            break;
        case EV_RXCOMPLETE:
            // data received in ping slot
            Serial.println(F("EV_RXCOMPLETE"));
            break;
        case EV_LINK_DEAD:
            Serial.println(F("EV_LINK_DEAD"));
            break;
        case EV_LINK_ALIVE:
            Serial.println(F("EV_LINK_ALIVE"));
            break;
         default:
            Serial.println(F("Unknown event"));
            break;
    }
}
```

```
void do_send(osjob_t* j){
    // Check if there is not a current TX/RX job running
    if (LMIC.opmode & OP_TXRXPEND) {
        Serial.println(F("OP_TXRXPEND, not sending"));
    } else {
        // Prepare upstream data transmission at the next possible time.
        LMIC_setTxData2(1, mydata, sizeof(mydata)-1, 0);
        Serial.println(F("Packet queued"));
    }
    // Next TX is scheduled after TX_COMPLETE event.
}

void setup() {
    Serial.begin(115200);
    Serial.println(F("Starting"));

    #ifdef VCC_ENABLE
    // For Pinoccio Scout boards
    pinMode(VCC_ENABLE, OUTPUT);
    digitalWrite(VCC_ENABLE, HIGH);
    delay(1000);
    #endif

    // LMIC init
    os_init();
    // Reset the MAC state. Session and pending data transfers will be discarded.
    LMIC_reset();

    // Set static session parameters. Instead of dynamically establishing a session
    // by joining the network, precomputed session parameters are be provided.
    #ifdef PROGMEM
    // On AVR, these values are stored in flash and only copied to RAM
    // once. Copy them to a temporary buffer here, LMIC_setSession will
    // copy them into a buffer of its own again.
    uint8_t appskey[sizeof(APPSKEY)];
    uint8_t nwkskey[sizeof(NWKSKEY)];
    memcpy_P(appskey, APPSKEY, sizeof(APPSKEY));
    memcpy_P(nwkskey, NWKSKEY, sizeof(NWKSKEY));
    LMIC_setSession (0x1, DEVADDR, nwkskey, appskey);
    #else
    // If not running an AVR with PROGMEM, just use the arrays directly
    LMIC_setSession (0x1, DEVADDR, NWKSKEY, APPSKEY);
    #endif

    // Set up the channels used by the Things Network, which corresponds
    // to the defaults of most gateways. Without this, only three base
    // channels from the LoRaWAN specification are used, which certainly
    // works, so it is good for debugging, but can overload those
    // frequencies, so be sure to configure the full frequency range of
    // your network here (unless your network autoconfigures them).
    // Setting up channels should happen after LMIC_setSession, as that
    // configures the minimal channel set.
    LMIC_setupChannel(0, 868100000, DR_RANGE_MAP(DR_SF12, DR_SF7),  BAND_CENTI);  // g-band
    LMIC_setupChannel(1, 868300000, DR_RANGE_MAP(DR_SF12, DR_SF7B), BAND_CENTI);  // g-band
    LMIC_setupChannel(2, 868500000, DR_RANGE_MAP(DR_SF12, DR_SF7),  BAND_CENTI);  // g-band
    LMIC_setupChannel(3, 867100000, DR_RANGE_MAP(DR_SF12, DR_SF7),  BAND_CENTI);  // g-band
    LMIC_setupChannel(4, 867300000, DR_RANGE_MAP(DR_SF12, DR_SF7),  BAND_CENTI);  // g-band
    LMIC_setupChannel(5, 867500000, DR_RANGE_MAP(DR_SF12, DR_SF7),  BAND_CENTI);  // g-band
    LMIC_setupChannel(6, 867700000, DR_RANGE_MAP(DR_SF12, DR_SF7),  BAND_CENTI);  // g-band
    LMIC_setupChannel(7, 867900000, DR_RANGE_MAP(DR_SF12, DR_SF7),  BAND_CENTI);  // g-band
    LMIC_setupChannel(8, 868800000, DR_RANGE_MAP(DR_FSK,  DR_FSK),  BAND_MILLI);  // g2-band
    // TTN defines an additional channel at 869.525MHz using SF9 for class B
    // devices' ping slots. LMIC does not have an easy way to define set this
    // frequency and support for class B is spotty and untested, so this
    // frequency is not configured here.

    // Disable link check validation
    LMIC_setLinkCheckMode(0);

    // Set data rate and transmit power (note: txpow seems to be ignored by the library)
    LMIC_setDrTxpow(DR_SF7,14);

    // Start job
    do_send(&sendjob);
}
```

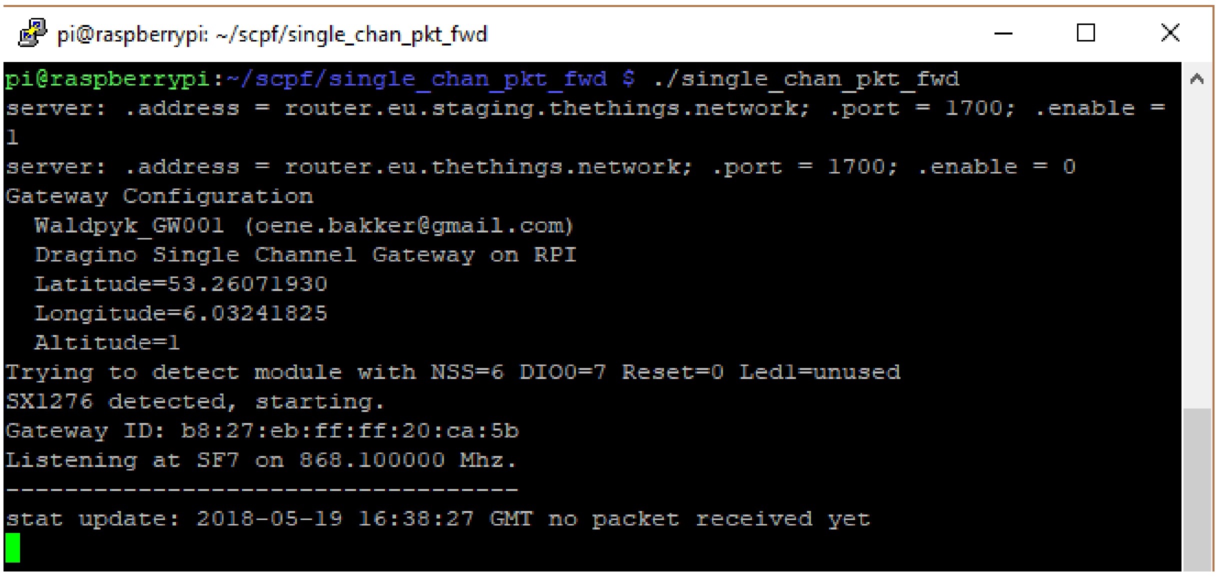
```
void loop() {
    os_runloop_once();
}
```

The values of the variables are determined in section 5.4:

Device Address:	26011235
Network Session Key:	FD 19 C6 3D 0D A9 78 E3 8D C7 A7 5D 9C 3A D7 2D
App Session Key:	93 4C 4E 59 1C A9 6C 20 EB C8 F4 C8 55 99 5A 40

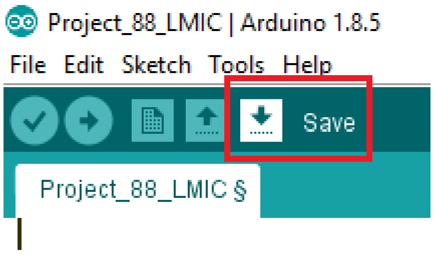
6.5 Testen Hello World sketch

Be sure the gateway is active:

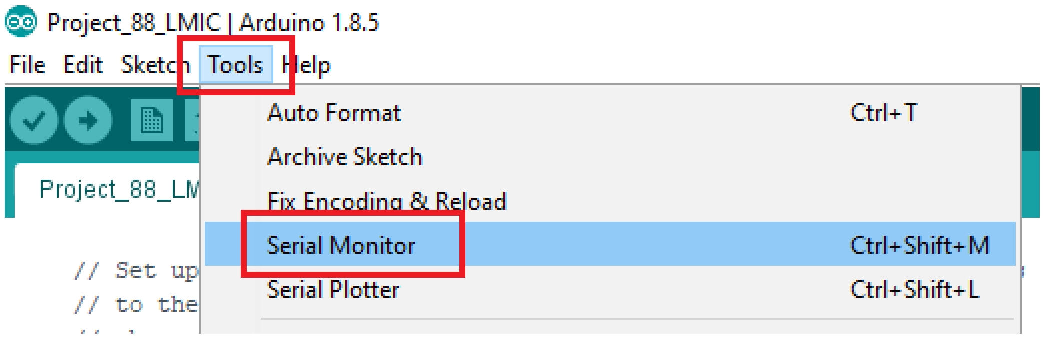


```
pi@raspberrypi: ~/scpf/single_chan_pkt_fwd
pi@raspberrypi:~/scpf/single_chan_pkt_fwd $ ./single_chan_pkt_fwd
server: .address = router.eu.staging.thethings.network; .port = 1700; .enable =
1
server: .address = router.eu.thethings.network; .port = 1700; .enable = 0
Gateway Configuration
  Waldpyk_GW001 (oene.bakker@gmail.com)
  Dragino Single Channel Gateway on RPI
  Latitude=53.26071930
  Longitude=6.03241825
  Altitude=1
Trying to detect module with NSS=6 DIO0=7 Reset=0 Led1=unused
SX1276 detected, starting.
Gateway ID: b8:27:eb:ff:ff:20:ca:5b
Listening at SF7 on 868.100000 Mhz.
-----------------------------------
stat update: 2018-05-19 16:38:27 GMT no packet received yet
```

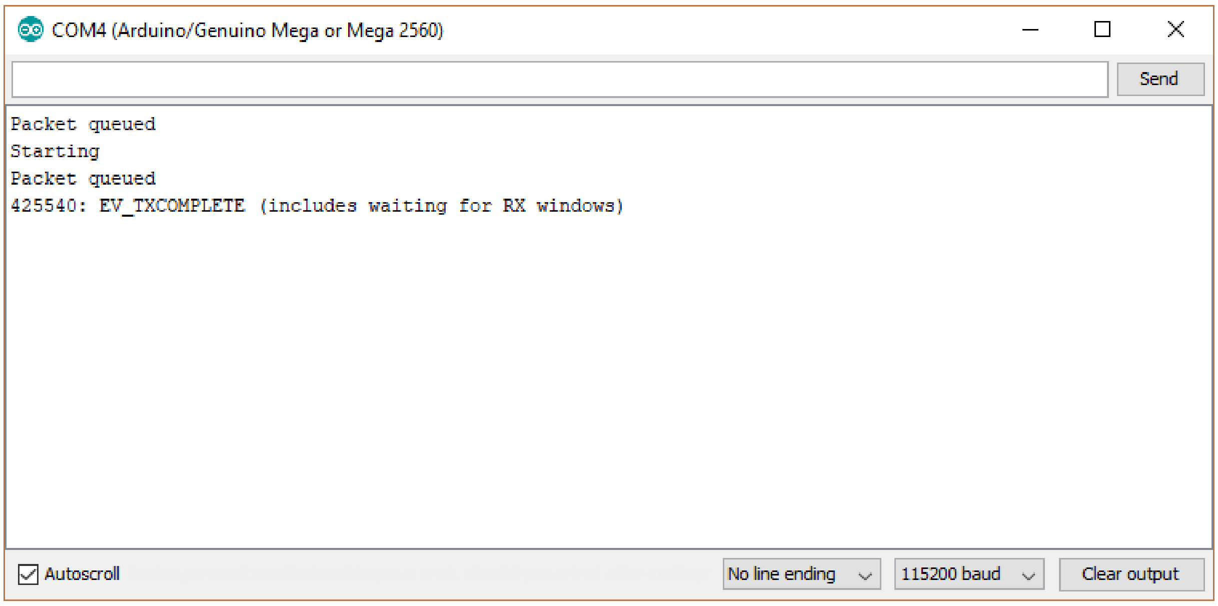
Compile and upload the sketch:



Open an terminal after the upload:

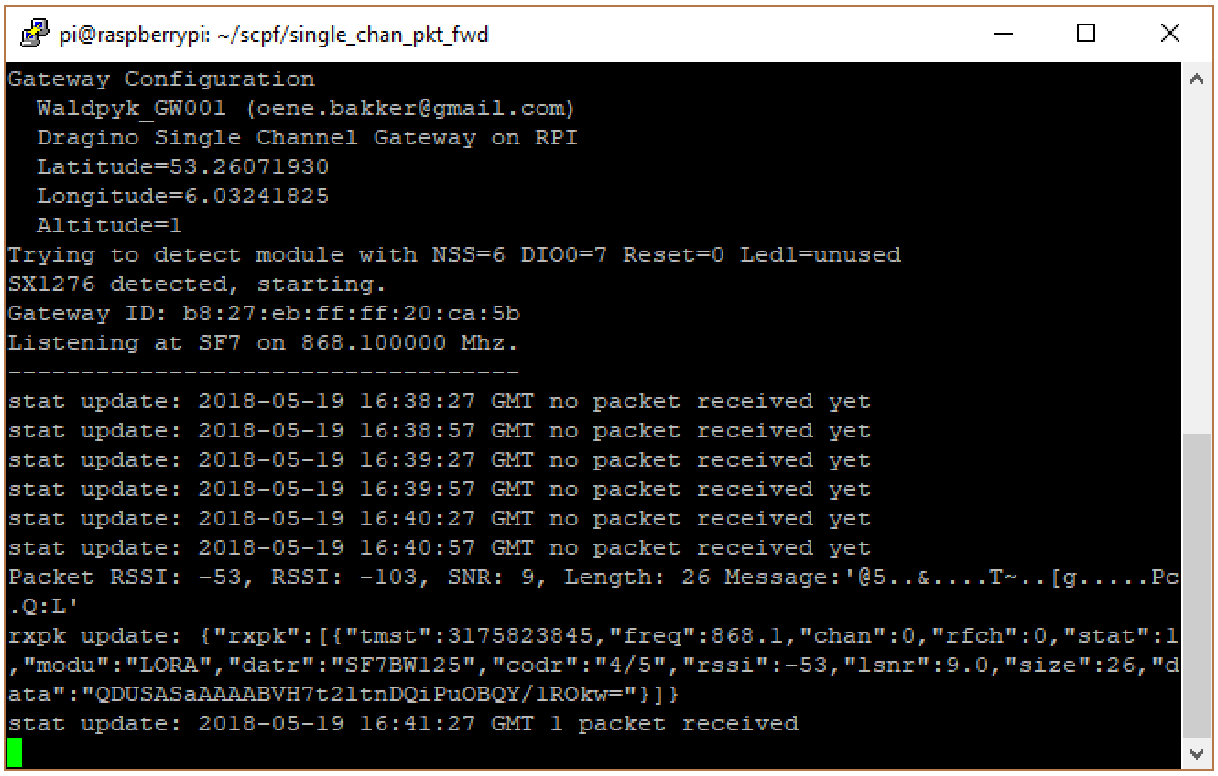

Project_88_LMIC | Arduino 1.8.5
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
Project_88_LM
// Set up
// to the

Result on the console (115.200 baud):



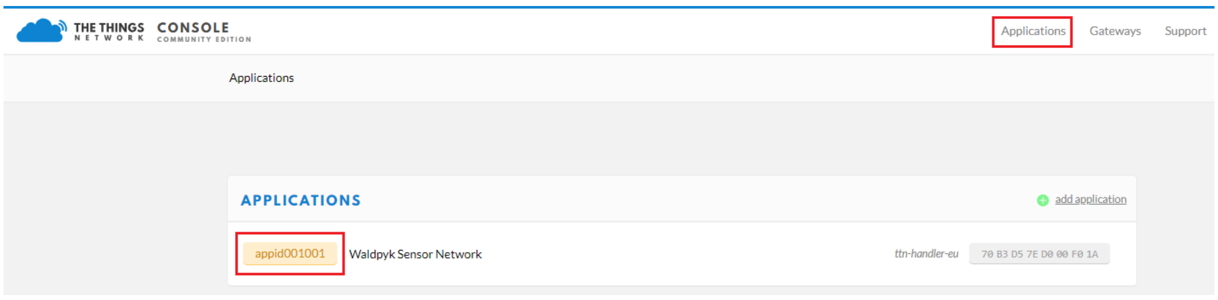
```
Packet queued
Starting
Packet queued
425540: EV_TXCOMPLETE (includes waiting for RX windows)
```

Result on the gateway console:



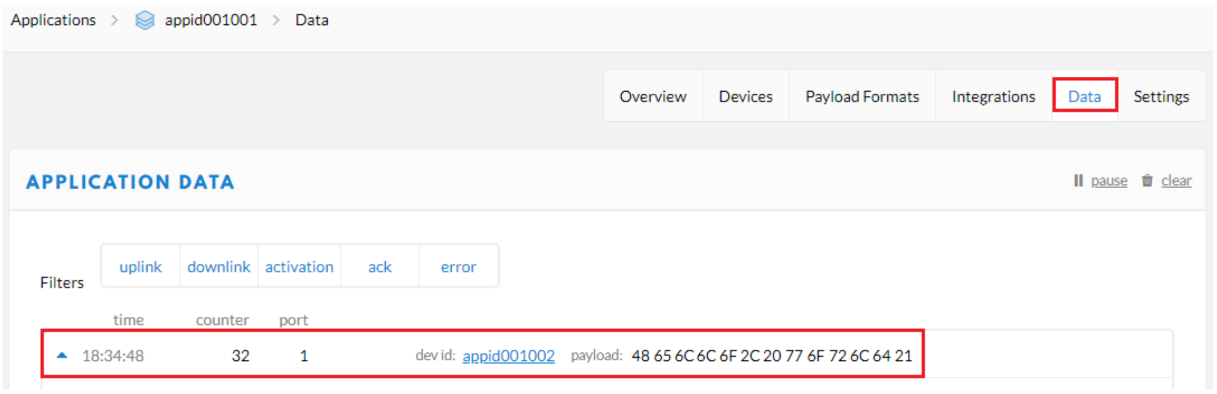
```
Gateway Configuration
  Waldpyk_GW001 (oene.bakker@gmail.com)
  Dragino Single Channel Gateway on RPI
  Latitude=53.26071930
  Longitude=6.03241825
  Altitude=1
Trying to detect module with NSS=6 DIO0=7 Reset=0 Led1=unused
SX1276 detected, starting.
Gateway ID: b8:27:eb:ff:ff:20:ca:5b
Listening at SF7 on 868.100000 Mhz.
-----------------------------------
stat update: 2018-05-19 16:38:27 GMT no packet received yet
stat update: 2018-05-19 16:38:57 GMT no packet received yet
stat update: 2018-05-19 16:39:27 GMT no packet received yet
stat update: 2018-05-19 16:39:57 GMT no packet received yet
stat update: 2018-05-19 16:40:27 GMT no packet received yet
stat update: 2018-05-19 16:40:57 GMT no packet received yet
Packet RSSI: -53, RSSI: -103, SNR: 9, Length: 26 Message:'@5..&....T~..[g.....Pc
.Q:L'
rxpk update: {"rxpk":[{"tmst":3175823845,"freq":868.1,"chan":0,"rfch":0,"stat":1
,"modu":"LORA","datr":"SF7BW125","codr":"4/5","rssi":-53,"lsnr":9.0,"size":26,"d
ata":"QDUSASaAAAABVH7t21tnDQiPuOBQY/1ROkw="}]}
stat update: 2018-05-19 16:41:27 GMT 1 packet received
```

Go to the TTN, click on Console, Applications and then click on the application.



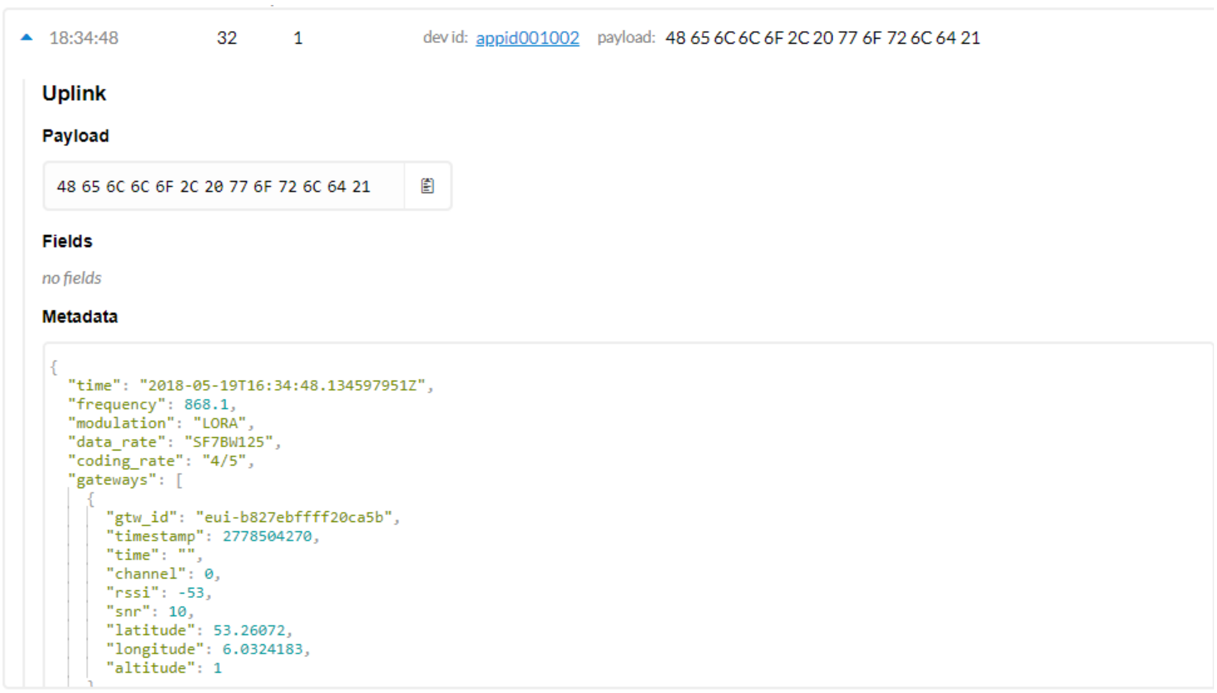
appid001001 Waldpyk Sensor Network ttn-handler-eu 70 B3 D5 7E D0 00 F0 1A

Click on Data and open the message:



time	counter	port		
18:34:48	32	1	dev id: appid001002	payload: 48 65 6C 6C 6F 2C 20 77 6F 72 6C 64 21

Result:



18:34:48 32 1 dev id: appid001002 payload: 48 65 6C 6C 6F 2C 20 77 6F 72 6C 64 21

Uplink

Payload

48 65 6C 6C 6F 2C 20 77 6F 72 6C 64 21

Fields

no fields

Metadata

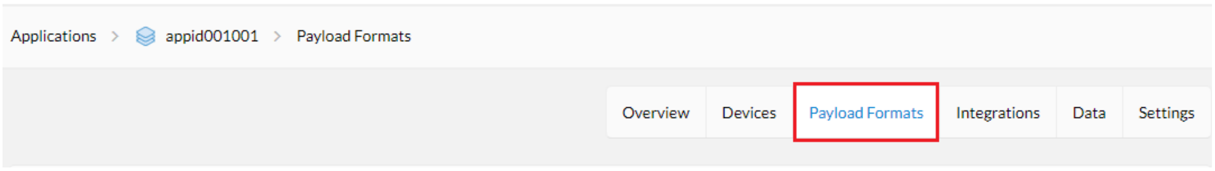
```
{
  "time": "2018-05-19T16:34:48.134597951Z",
  "frequency": 868.1,
  "modulation": "LORA",
  "data_rate": "SF7BW125",
  "coding_rate": "4/5",
  "gateways": [
    {
      "gtw_id": "eui-b827ebffff20ca5b",
      "timestamp": 2778504270,
      "time": "",
      "channel": 0,
      "rssi": -53,
      "snr": 10,
      "latitude": 53.26072,
      "longitude": 6.0324183,
      "altitude": 1
```

In the Payload the transmitted data is in hexadecimal format. Translated with the use of an ASCII table:

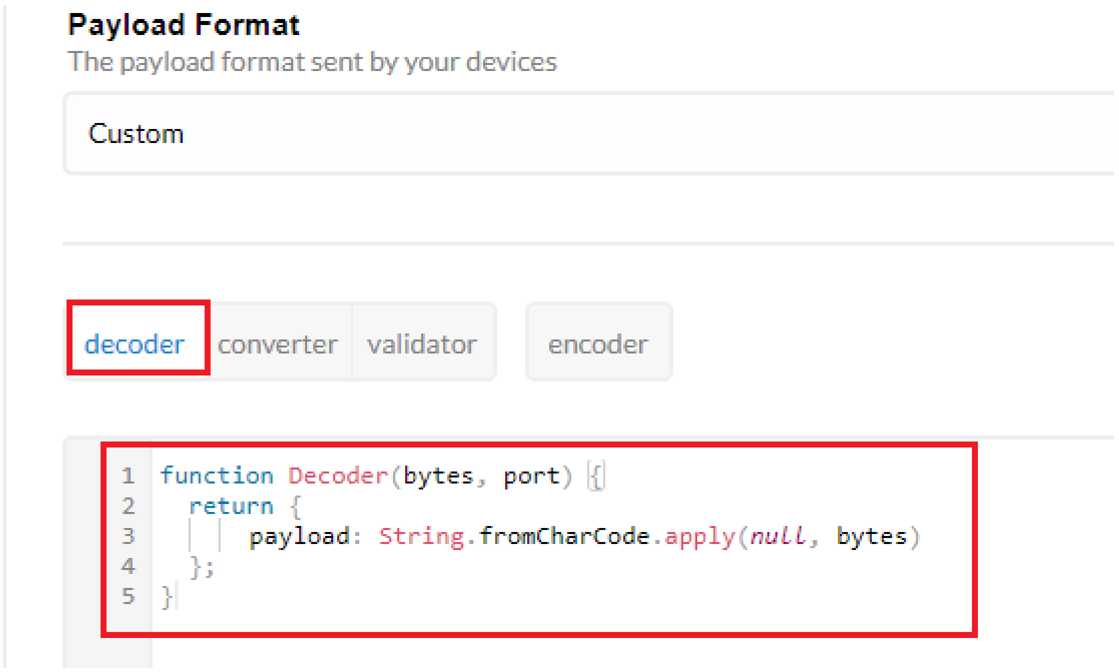
```
48 65 6C 6C 6F 2C 20 77 6F 72 6C 64 21
H  e  l  l  o  ,     w  o  r  l  d  !
```

Note:
A message is sent every minute. However, since we are dealing with a single channel gateway and there are 8 channels, there is one message every 8 minutes!!!

The translation of the payload can be done directly via a so called decoder. Select Payload Formats and create a decoder.

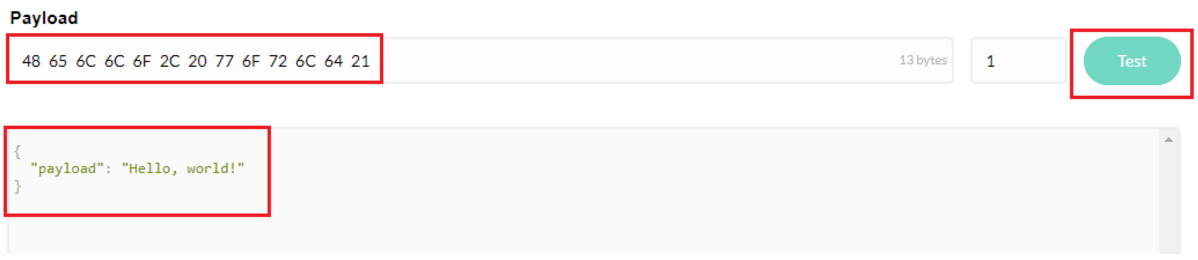


Various examples can be found using Google. For decoding we take the following decoder:



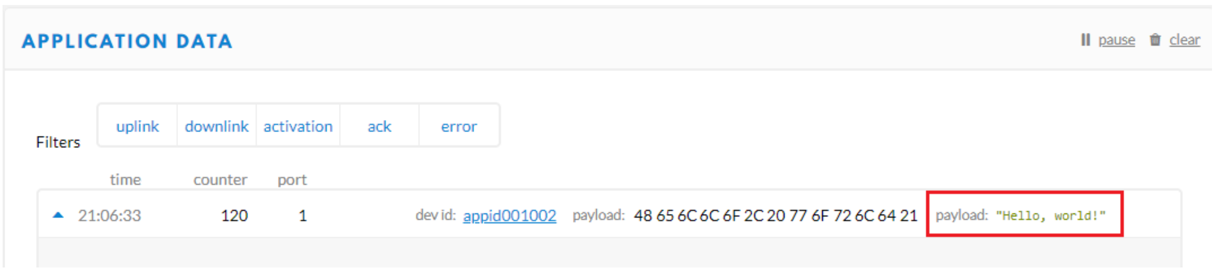
```
function Decoder(bytes, port) {
  return {
    payload: String.fromCharCode.apply(null, bytes)
  };
}
```

You can test the decoder directly:



```
{
  "payload": "Hello, world!"
}
```

The next time data is comming in the result will be:



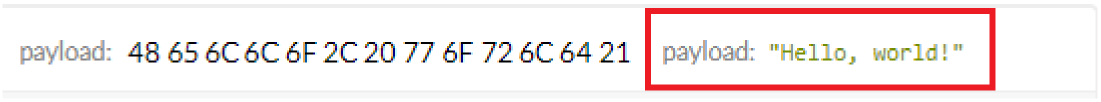
APPLICATION DATA

pause clear

Filters: uplink | downlink | activation | ack | error

time	counter	port			
21:06:33	120	1	dev id: appid001002	payload: 48 65 6C 6C 6F 2C 20 77 6F 72 6C 64 21	payload: "Hello, world!"

Message:



payload: 48 65 6C 6C 6F 2C 20 77 6F 72 6C 64 21 payload: "Hello, world!"

7 Python MQTT Client

7.1 Introduction

In this chapter, a Python MQTT client is used to read the node message sent to the gateway by the node.

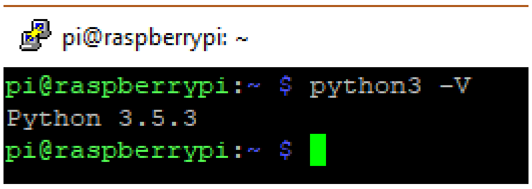
7.2 Python3 en pip3

In this chapter we use is python3 and pip3. Start a new PuTTY session and log on to the Raspberry Pi. Make sure that the gateway is not closed (otherwise start a new session).

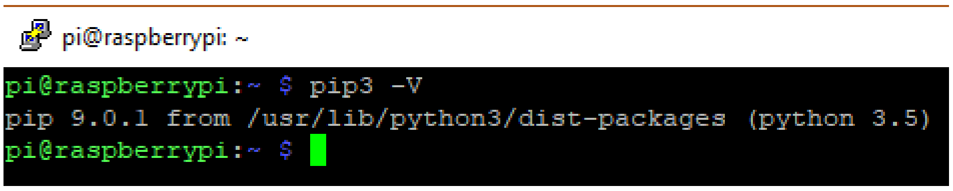
Python3 is already installed on the Raspberry Pi 3.

Open a new session with PuTTY.

The Python3 version can be checked with the command: python3 -V


```
pi@raspberrypi: ~
pi@raspberrypi:~ $ python3 -V
Python 3.5.3
pi@raspberrypi:~ $
```

And pip3 can be checked with the command: pip3 -V


```
pi@raspberrypi: ~
pi@raspberrypi:~ $ pip3 -V
pip 9.0.1 from /usr/lib/python3/dist-packages (python 3.5)
pi@raspberrypi:~ $
```

If Python3 and pip3 are not installed, you can use the following commands:
- sudo apt-get install python3
- sudo apt-get install python3-pip

7.3 Install TTN package

Use the following command to install the TTN package:
- sudo pip3 install ttn

7.4 Python TTN MQTT Client

The client consists of:

- A number of necessary imports
- Variables for the application id and the access key
- A callback function for handling an incoming uplink message
- Defining and initiating the TTN and MQTT handlers
- An endless loop (the client waits for incoming uplink messages)
- And finally handling the closing of the client (by pressing Ctrl + C)

Create the source with the commands:

- cd ~
- mkdir ttn
- cd ttn
- nano ttnclient.py

7.4.1 Source: ttnclient.py

```
# Imports
import sys
import time
import ttn
import base64

# Globals
app_id = "xxxxxxxx"
access_key = "ttn-account-v2.xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"

# Uplink callback function
def uplink_callback(data, client):
    print("*** Received uplink ***")

    # Show uplink message
    print(str(data))
    print()
    print("App_id    : ", str(data.app_id))
    print("Dev_id    : ", str(data.dev_id))
    print("HW serial : ", str(data.hardware_serial))

    print("Frequency : ", str(data.metadata.frequency))
    print("Gateway ID: ", str(data.metadata.gateways[0].gtw_id))

    print("Data      : ", str(data.payload_fields.payload))
    print()
    print("Raw data  : ", str(data.payload_raw))

    # Decode base64 message
    message = base64.b64decode(data.payload_raw)
    print("Decode B64: " , message.decode("utf-8"))
    print()


# Define TTN handler
print("TTN MQTT client started")
handler = ttn.HandlerClient(app_id, access_key)

# MQTT client
mqtt_client = handler.data()
mqtt_client.set_uplink_callback(uplink_callback)
mqtt_client.connect()
print("Client connected")
```

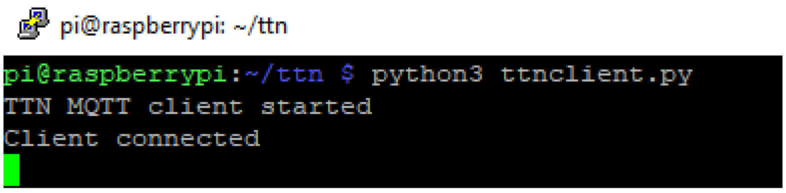
```
# Endless loop
try:
    # Wait for uplink message
    while True:
            time.sleep(1)
except KeyboardInterrupt:
    # Ctrl+C pressed
    print()
    print('Interrupted...')

# Graceful exit
mqtt_client.close()
print("Client closed")

sys.exit(0)
```

Sluit af met Ctrt+X en J.

7.5 Starten TTN MQTT Client

Start the client with the command:

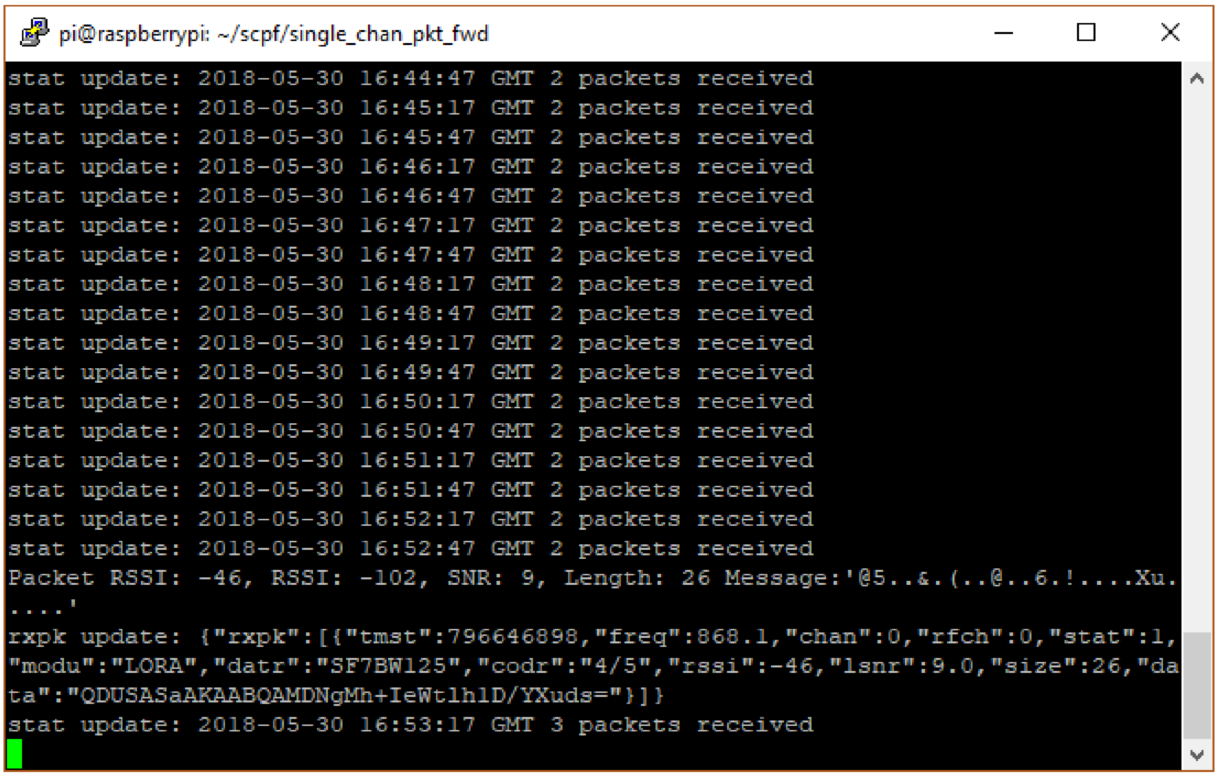
- python3 ttnclient.py



```
pi@raspberrypi:~/ttn $ python3 ttnclient.py
TTN MQTT client started
Client connected
```

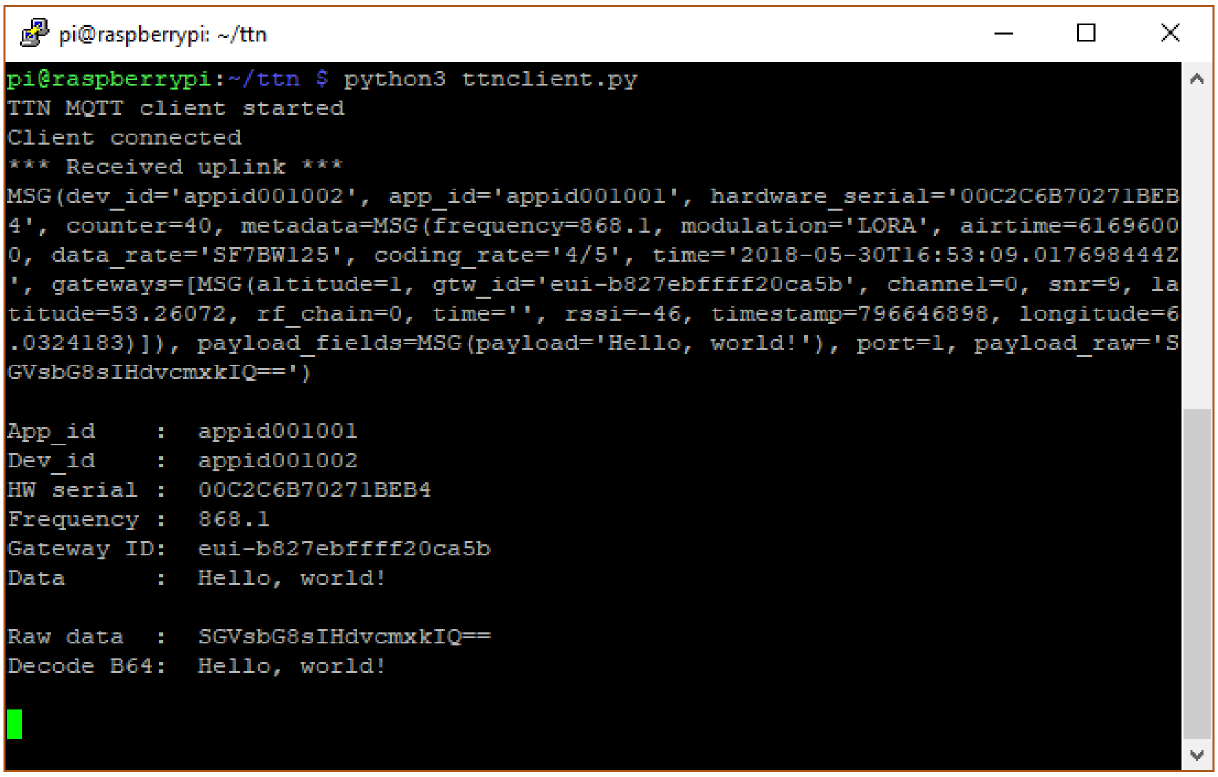
Wait for incoming messages.

Result on the gateway:



```
stat update: 2018-05-30 16:44:47 GMT 2 packets received
stat update: 2018-05-30 16:45:17 GMT 2 packets received
stat update: 2018-05-30 16:45:47 GMT 2 packets received
stat update: 2018-05-30 16:46:17 GMT 2 packets received
stat update: 2018-05-30 16:46:47 GMT 2 packets received
stat update: 2018-05-30 16:47:17 GMT 2 packets received
stat update: 2018-05-30 16:47:47 GMT 2 packets received
stat update: 2018-05-30 16:48:17 GMT 2 packets received
stat update: 2018-05-30 16:48:47 GMT 2 packets received
stat update: 2018-05-30 16:49:17 GMT 2 packets received
stat update: 2018-05-30 16:49:47 GMT 2 packets received
stat update: 2018-05-30 16:50:17 GMT 2 packets received
stat update: 2018-05-30 16:50:47 GMT 2 packets received
stat update: 2018-05-30 16:51:17 GMT 2 packets received
stat update: 2018-05-30 16:51:47 GMT 2 packets received
stat update: 2018-05-30 16:52:17 GMT 2 packets received
stat update: 2018-05-30 16:52:47 GMT 2 packets received
Packet RSSI: -46, RSSI: -102, SNR: 9, Length: 26 Message:'@5..&.(..@..6.!....Xu.
....'
rxpk update: {"rxpk":[{"tmst":796646898,"freq":868.1,"chan":0,"rfch":0,"stat":1,
"modu":"LORA","datr":"SF7BW125","codr":"4/5","rssi":-46,"lsnr":9.0,"size":26,"da
ta":"QDUSASaAKAABQAMDNgMh+IeWtlhlD/YXuds="}]}
stat update: 2018-05-30 16:53:17 GMT 3 packets received
```

Result on the TTN MQTT Client:


pi@raspberrypi: ~/ttn

```
pi@raspberrypi:~/ttn $ python3 ttnclient.py
TTN MQTT client started
Client connected
*** Received uplink ***
MSG(dev_id='appid001002', app_id='appid001001', hardware_serial='00C2C6B70271BEB
4', counter=40, metadata=MSG(frequency=868.1, modulation='LORA', airtime=6169600
0, data_rate='SF7BW125', coding_rate='4/5', time='2018-05-30T16:53:09.017698444Z
', gateways=[MSG(altitude=1, gtw_id='eui-b827ebffff20ca5b', channel=0, snr=9, la
titude=53.26072, rf_chain=0, time='', rssi=-46, timestamp=796646898, longitude=6
.0324183)]), payload_fields=MSG(payload='Hello, world!'), port=1, payload_raw='S
GVsbG8sIHdvcmxkIQ==')

App_id     :  appid001001
Dev_id     :  appid001002
HW serial  :  00C2C6B70271BEB4
Frequency  :  868.1
Gateway ID:   eui-b827ebffff20ca5b
Data       :  Hello, world!

Raw data   :  SGVsbG8sIHdvcmxkIQ==
Decode B64:   Hello, world!
```

The output is:

```
*** Received uplink ***
MSG(dev_id='appid001002', app_id='appid001001', hardware_serial='00C2C6B70271BEB
4', counter=40, metadata=MSG(frequency=868.1, modulation='LORA', airtime=6169600
0, data_rate='SF7BW125', coding_rate='4/5', time='2018-05-30T16:53:09.017698444Z
', gateways=[MSG(altitude=1, gtw_id='eui-b827ebffff20ca5b', channel=0, snr=9, la
titude=53.26072, rf_chain=0, time='', rssi=-46, timestamp=796646898, longitude=6
.0324183)]), payload_fields=MSG(payload='Hello, world!'), port=1, payload_raw='S
GVsbG8sIHdvcmxkIQ==')

App_id     :  appid001001
Dev_id     :  appid001002
HW serial  :  00C2C6B70271BEB4
Frequency  :  868.1
Gateway ID:   eui-b827ebffff20ca5b
Data       :  Hello, world!

Raw data   :  SGVsbG8sIHdvcmxkIQ==
Decode B64:   Hello, world!
```

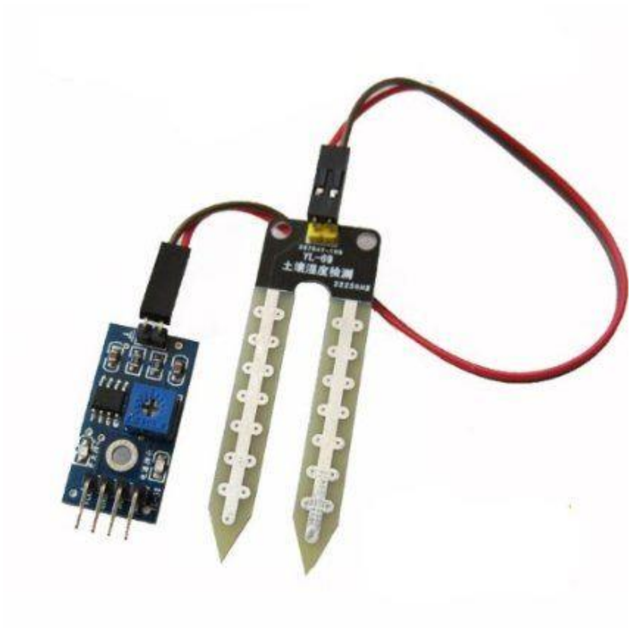
8 A practical example

8.1 Introduction

In this example, we will use the node to measure the soil moisture. Via a so-called Pushbullet alert on the mobile phone, the user is warned if the ground is too wet or too dry.

8.2 Hardware

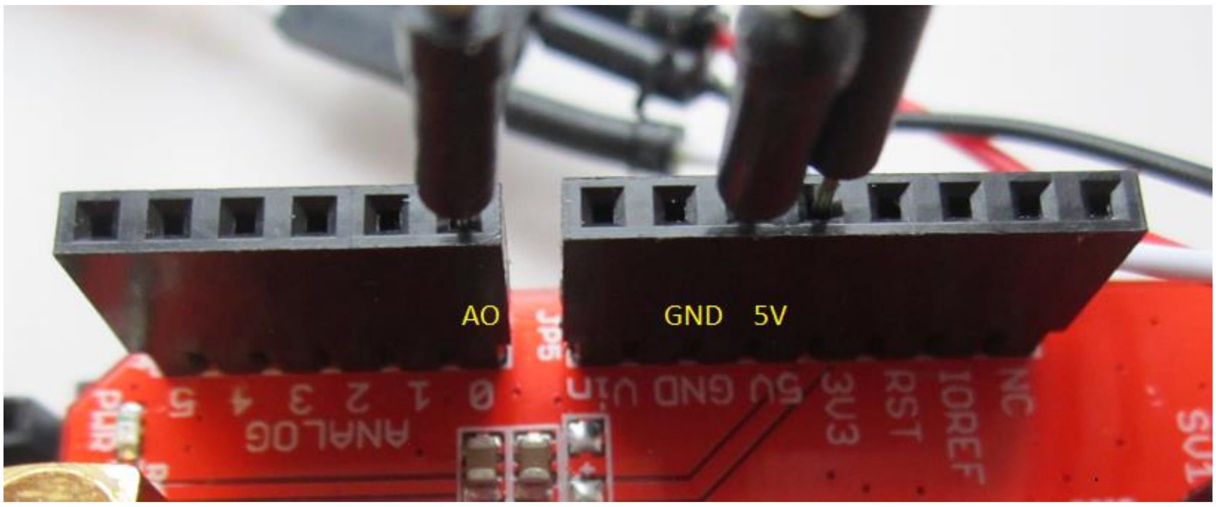
A ground humidity sensor is used to measure the humidity. This is a combination of a sensor (FC-28) and a so-called comparator chip module LM393. This can be used to set a limit value if the digital pin (D0) is used. The module contains a potentiometer with which the limit value can be set. Since we will use the analog output we will not use this module.



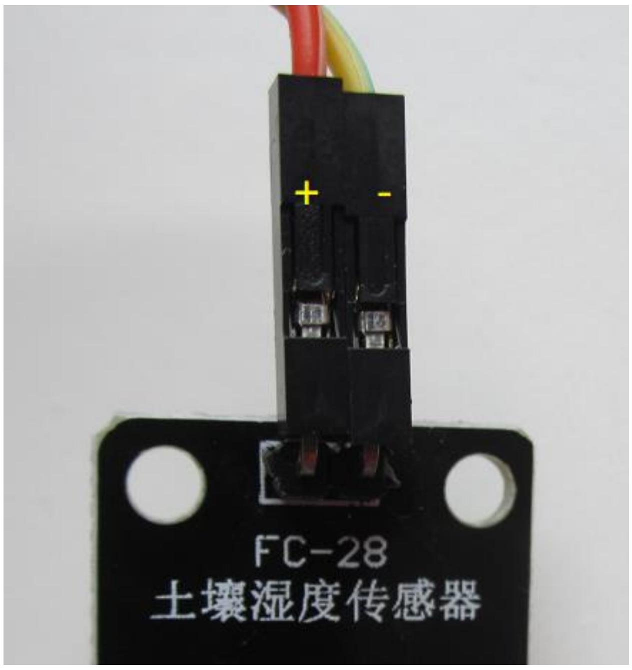
There are 4 connections that are connected to the Dragino as follows:

Sensor	Dragino
VCC	5V
GND	GND
A0	A0
D0	Not connected

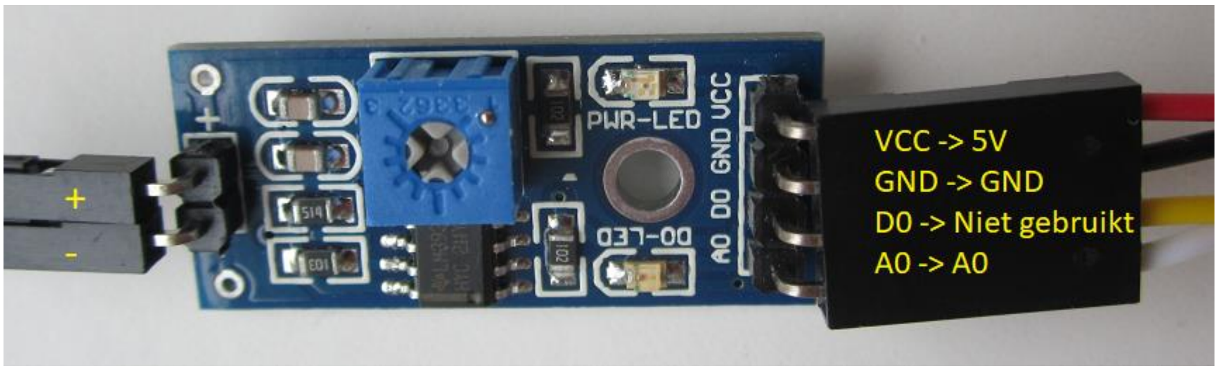
Connected to the Dragino:



FC-28 sensor connection:



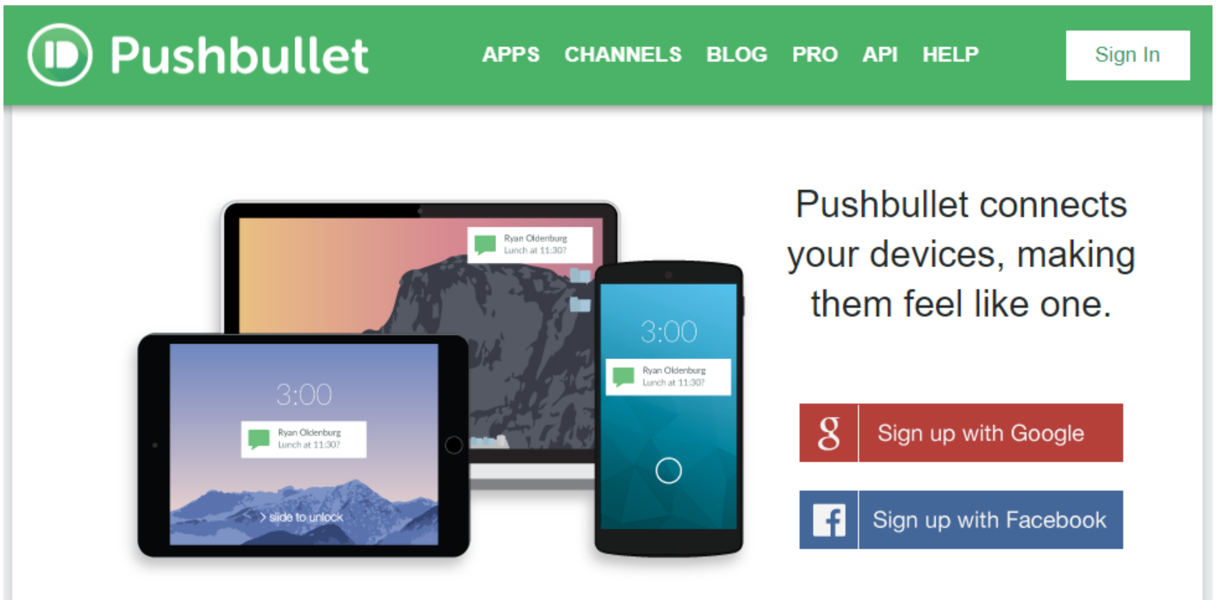
LM393 connection:



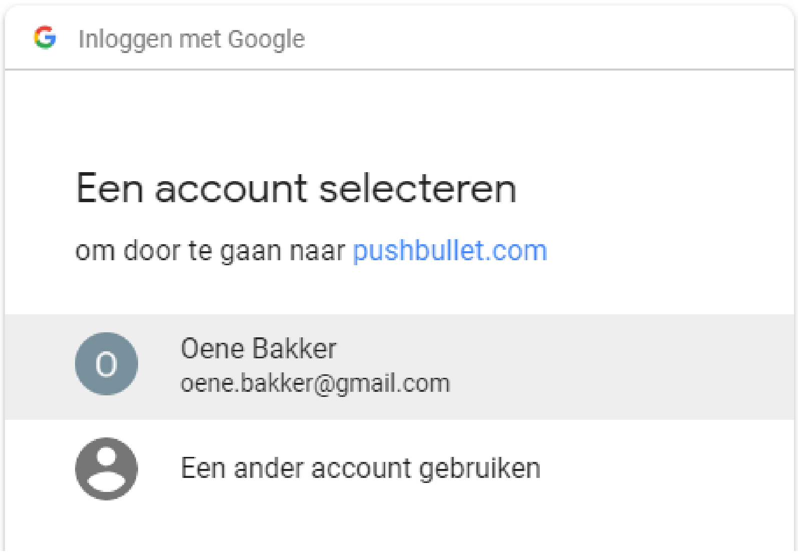
The sensor generates values between 0 (wet) and 1023 (dry). For the determination of the limit values to be used, you have to do a few experiments with dry and wet soil yourself. In this example I use the limit values:
• 250 (and lower) is too wet and
• 750 (and higher) is too dry.

8.3 PushBullet

To use PushBullet you need a Google or Facebook account. Go to:
https://www.pushbullet.com/



Log in with a Google or Facebook account.


Inloggen met Google

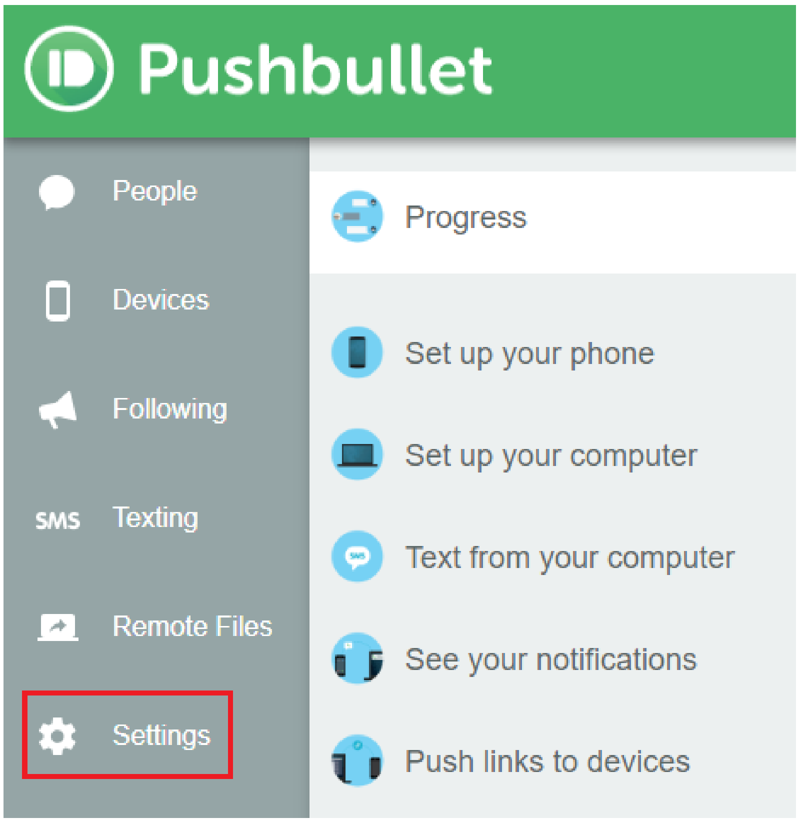
Een account selecteren

om door te gaan naar pushbullet.com

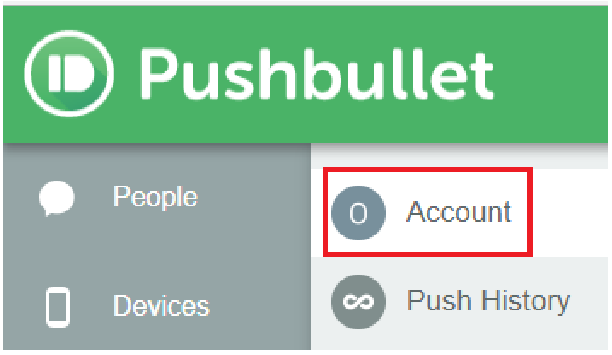
Oene Bakker
oene.bakker@gmail.com

Een ander account gebruiken

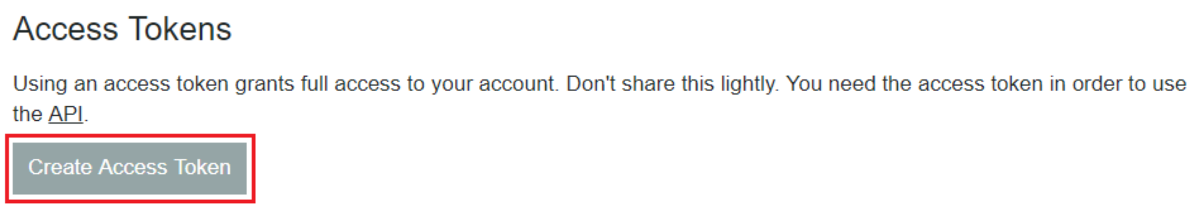
Select Settings:



Select Account:



Go to the right and scroll down a little and select Create Access Token:



Access Tokens

Using an access token grants full access to your account. Don't share this lightly. You need the access token in order to use the API.

Create Access Token

Make a copy of the access token.

8.3.1 PushBullet App

To receive PushBullet notifications, we need to install the PushBullet App. There are versions for Windows, Android and IOS.

8.4 Arduino Node

On the Arduino node the soil moisture is measured with the connected sensor. The TTN sensor part is suppressed. After all, we only use one channel. In default mode, 3 channels are used. This means that now a valid message is sent every 3 minutes (instead of the previous 9 minutes).

8.4.1 Arduino sketch

```
#include <lmic.h>
#include <hal/hal.h>
#include <SPI.h>
#include "moisturestate.h"


// LoRaWAN NwkSKey, network session key
// This is the default Semtech key, which is used by the prototype TTN
// network initially.
static const PROGMEM u1_t NWKSKEY[16] = { 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX,
0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX };

// LoRaWAN AppSKey, application session key
// This is the default Semtech key, which is used by the prototype TTN
// network initially.
static const u1_t PROGMEM APPSKEY[16] = { 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX,
0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX };

// LoRaWAN end-device address (DevAddr)
// See http://thethingsnetwork.org/wiki/AddressSpace
static const u4_t DEVADDR = 0xXXXXXXXX ;
```

```
// These callbacks are only used in over-the-air activation, so they are
// left empty here (we cannot leave them out completely unless
// DISABLE_JOIN is set in config.h, otherwise the linker will complain).
void os_getArtEui (u1_t* buf) { }
void os_getDevEui (u1_t* buf) { }
void os_getDevKey (u1_t* buf) { }

static uint8_t data_DRY[] = "DRY";
static uint8_t data_ACC[] = "ACC";
static uint8_t data_WET[] = "WET";

static osjob_t sendjob;

// Schedule TX every this many seconds (might become longer due to duty
// cycle limitations).
const unsigned TX_INTERVAL = 60;

// Soil moisture tresholds
const unsigned TRESSHOLD_DRY = 750;
const unsigned TRESSHOLD_WET = 250;

// Soil moisture state
MOISTURESTATE moistureState;

// Pin mapping Dragine hat
const lmic_pinmap lmic_pins = {
  .nss = 10,
  .rxtx = LMIC_UNUSED_PIN,
  .rst = 9,
  .dio = {2, 6, 7},
};

// Pin mapping soil moisture sensor
const int FC28_SENSOR_PIN = A0;


// Event handler
void onEvent (ev_t ev) {
  Serial.print(os_getTime());
  Serial.print(": ");
  switch (ev) {
    case EV_SCAN_TIMEOUT:
      Serial.println(F("EV_SCAN_TIMEOUT"));
      break;
    case EV_BEACON_FOUND:
      Serial.println(F("EV_BEACON_FOUND"));
      break;
    case EV_BEACON_MISSED:
      Serial.println(F("EV_BEACON_MISSED"));
      break;
    case EV_BEACON_TRACKED:
      Serial.println(F("EV_BEACON_TRACKED"));
      break;
    case EV_JOINING:
      Serial.println(F("EV_JOINING"));
      break;
    case EV_JOINED:
      Serial.println(F("EV_JOINED"));
      break;
    case EV_RFU1:
      Serial.println(F("EV_RFU1"));
      break;
    case EV_JOIN_FAILED:
      Serial.println(F("EV_JOIN_FAILED"));
      break;
    case EV_REJOIN_FAILED:
      Serial.println(F("EV_REJOIN_FAILED"));
      break;
      break;
    case EV_TXCOMPLETE:
      Serial.println(F("EV_TXCOMPLETE (includes waiting for RX windows)"));
      if (LMIC.dataLen) {
        // data received in rx slot after tx
        Serial.print(F("Data Received: "));
        Serial.write(LMIC.frame + LMIC.dataBeg, LMIC.dataLen);
        Serial.println();
      }
      // Schedule next transmission
      os_setTimedCallback(&sendjob, os_getTime() + sec2osticks(TX_INTERVAL),
                                  do_send);
      break;
    case EV_LOST_TSYNC:
      Serial.println(F("EV_LOST_TSYNC"));
      break;
    case EV_RESET:
      Serial.println(F("EV_RESET"));
      break;
    case EV_RXCOMPLETE:
      // data received in ping slot
      Serial.println(F("EV_RXCOMPLETE"));
      break;
    case EV_LINK_DEAD:
      Serial.println(F("EV_LINK_DEAD"));
      break;
    case EV_LINK_ALIVE:
      Serial.println(F("EV_LINK_ALIVE"));
      break;
    default:
      Serial.println(F("Unknown event"));
      break;
  }
}

void do_send(osjob_t* j) {
  // Check if there is not a current TX/RX job running
  if (LMIC.opmode & OP_TXRXPEND) {
    Serial.println(F("OP_TXRXPEND, not sending"));
  } else {
    readFC28Sensor();
    Serial.print(F("Read sensor: "));
    Serial.println(moistureState);
    // Prepare upstream data transmission at the next possible time.
    if (moistureState == TOO_DRY) {
      LMIC_setTxData2(1, data_DRY, sizeof(data_DRY) - 1, 0);
    } else if (moistureState == TOO_WET) {
      LMIC_setTxData2(1, data_WET, sizeof(data_WET) - 1, 0);
    } else {
      LMIC_setTxData2(1, data_ACC, sizeof(data_ACC) - 1, 0);
    }
    Serial.println(F("Packet queued"));
  }
  // Next TX is scheduled after TX_COMPLETE event.
}

// Read FC28 sensor value
void readFC28Sensor() {
  int value = analogRead(FC28_SENSOR_PIN);
  if (value >= TRESSHOLD_DRY) {
    moistureState = TOO_DRY;
  } else if (value <= TRESSHOLD_WET) {
    moistureState = TOO_WET;
  } else {
    moistureState = ACCEPTABLE;
  }
}
```

```
void setup() {
  Serial.begin(115200);
  Serial.println(F("Starting"));

#ifdef VCC_ENABLE
  // For Pinoccio Scout boards
  pinMode(VCC_ENABLE, OUTPUT);
  digitalWrite(VCC_ENABLE, HIGH);
  delay(1000);
#endif

  // LMIC init
  os_init();
  // Reset the MAC state. Session and pending data transfers will be discarded.
  LMIC_reset();

  // Set static session parameters. Instead of dynamically establishing a session
  // by joining the network, precomputed session parameters are be provided.
#ifdef PROGMEM
  // On AVR, these values are stored in flash and only copied to RAM
  // once. Copy them to a temporary buffer here, LMIC_setSession will
  // copy them into a buffer of its own again.
  uint8_t appskey[sizeof(APPSKEY)];
  uint8_t nwkskey[sizeof(NWKSKEY)];
  memcpy_P(appskey, APPSKEY, sizeof(APPSKEY));
  memcpy_P(nwkskey, NWKSKEY, sizeof(NWKSKEY));
  LMIC_setSession (0x1, DEVADDR, nwkskey, appskey);
#else
  // If not running an AVR with PROGMEM, just use the arrays directly
  LMIC_setSession (0x1, DEVADDR, NWKSKEY, APPSKEY);
#endif

  // Set up the channels used by the Things Network, which corresponds
  // to the defaults of most gateways. Without this, only three base
  // channels from the LoRaWAN specification are used, which certainly
  // works, so it is good for debugging, but can overload those
  // frequencies, so be sure to configure the full frequency range of
  // your network here (unless your network autoconfigures them).
  // Setting up channels should happen after LMIC_setSession, as that
  // configures the minimal channel set.
  /*
    LMIC_setupChannel(0, 868100000, DR_RANGE_MAP(DR_SF12, DR_SF7),  BAND_CENTI); // g-band
    LMIC_setupChannel(1, 868300000, DR_RANGE_MAP(DR_SF12, DR_SF7B), BAND_CENTI); // g-band
    LMIC_setupChannel(2, 868500000, DR_RANGE_MAP(DR_SF12, DR_SF7),  BAND_CENTI); // g-band
    LMIC_setupChannel(3, 867100000, DR_RANGE_MAP(DR_SF12, DR_SF7),  BAND_CENTI); // g-band
    LMIC_setupChannel(4, 867300000, DR_RANGE_MAP(DR_SF12, DR_SF7),  BAND_CENTI); // g-band
    LMIC_setupChannel(5, 867500000, DR_RANGE_MAP(DR_SF12, DR_SF7),  BAND_CENTI); // g-band
    LMIC_setupChannel(6, 867700000, DR_RANGE_MAP(DR_SF12, DR_SF7),  BAND_CENTI); // g-band
    LMIC_setupChannel(7, 867900000, DR_RANGE_MAP(DR_SF12, DR_SF7),  BAND_CENTI); // g-band
    LMIC_setupChannel(8, 868800000, DR_RANGE_MAP(DR_FSK,  DR_FSK),  BAND_MILLI); // g2-band
  */
  // TTN defines an additional channel at 869.525MHz using SF9 for class B
  // devices' ping slots. LMIC does not have an easy way to define set this
  // frequency and support for class B is spotty and untested, so this
  // frequency is not configured here.

  // Disable link check validation
  LMIC_setLinkCheckMode(0);

  // Set data rate and transmit power (note: txpow seems to be ignored by the library)
  LMIC_setDrTxpow(DR_SF7, 14);

  // Start job
  do_send(&sendjob);
}

void loop() {
  os_runloop_once();
}
```

8.4.2 Source: moisturestate.h

```
enum MOISTURESTATE {
  TOO_DRY,
  ACCEPTABLE,
  TOO_WET
};
```

8.5 RPI TTN MQTT PushBullet Client

The Python3 TTN Client handles node messages by sending a PushBullet message to the mobile app in case the soil is too dry or too wet. The client uses the previously generated PushBullet access token.

The content of the message is also dumped on the screen.

Commands:

- Open a new PuTTy session
- Log in
- cd ~
- mkdir ttn
- cd ttn
- nano ttnclient.py

8.5.1 Source: TTNClient.py

```
#!/usr/bin/python3

# Imports
import sys
import time
import ttn
import base64
import requests
import json

# Globals
ttn_app_id = "xxxxxxxxxxx"
ttn_access_key = "ttn-account-v2.xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
pushbullet_url = "https://api.pushbullet.com:443/v2/pushes"
pushbullet_token = "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"

# Send telegram notification
def send_notification(status):
    postData = {"type": "note", "title": "RPI Python Soil Alert!",
            "body": "Alert: Soil is too " + status + "!"}

    resp = requests.post(pushbullet_url, data=json.dumps(postData),
          headers={'Authorization': 'Bearer ' + pushbullet_token,
                   'Content-Type': 'application/json'})

    if resp.status_code != 200:
            raise Exception('Error sending telegram notification')
    else:
            print('Send telegram notification')
```

```
# Uplink callback function
def uplink_callback(data, client):
    print("*** Received uplink ***")

    # Show uplink message
    print(str(data))
    print()
    print("Data       : ")
    print("App_id     : ", str(data.app_id))
    print("Counter    : ", str(data.counter))
    print("Dev_id     : ", str(data.dev_id))
    print("Port       : ", str(data.port))
    print("HW serial  : ", str(data.hardware_serial))
    print("MetaData   : ")
    print("Airtime    : ", str(data.metadata.airtime))
    print("CodingRate: ", str(data.metadata.coding_rate))
    print("DataRate   : ", str(data.metadata.data_rate))
    print("Frequency  : ", str(data.metadata.frequency))
    print("Modulation: ", str(data.metadata.modulation))
    print("Time       : ", str(data.metadata.time))
    print("Gateways   : ")
    print("Altitude   : ", str(data.metadata.gateways[0].altitude))
    print("Channel    : ", str(data.metadata.gateways[0].channel))
    print("Gateway ID: ", str(data.metadata.gateways[0].gtw_id))
    print("Latitude   : ", str(data.metadata.gateways[0].latitude))
    print("Longitude  : ", str(data.metadata.gateways[0].longitude))
    print("RF_chain   : ", str(data.metadata.gateways[0].rf_chain))
    print("RSSI       : ", str(data.metadata.gateways[0].rssi))
    print("SNR        : ", str(data.metadata.gateways[0].snr))
    print("Time       : ", str(data.metadata.gateways[0].time))
    print("Timestamp  : ", str(data.metadata.gateways[0].timestamp))
    print("Payload    :")
    soilStatus = str(data.payload_fields.payload)

    print("Data       : ", soilStatus)
    print()
    print("Raw data   : ", str(data.payload_raw))

    # Send notification if soil is too dry or too wet
    if soilStatus == "DRY":
            send_notification("dry");
    if soilStatus == "WET":
            send_notification("wet");

    # Decode base64 message
    message = base64.b64decode(data.payload_raw)
    print("Decode B64: " , message.decode("utf-8"))
    print()

# Define TTN handler
print("TTN MQTT client started")
handler = ttn.HandlerClient(ttn_app_id, ttn_access_key)

# MQTT client
mqtt_client = handler.data()
mqtt_client.set_uplink_callback(uplink_callback)
mqtt_client.connect()
print("Client connected")

# Endless loop
try:
    # Wait for uplink message
    while True:
            time.sleep(1)
except KeyboardInterrupt:
    # Ctrl+C pressed
    print()
    print('Interrupted...')

# Graceful exit
mqtt_client.close()
print("Client closed")

sys.exit(0)
```

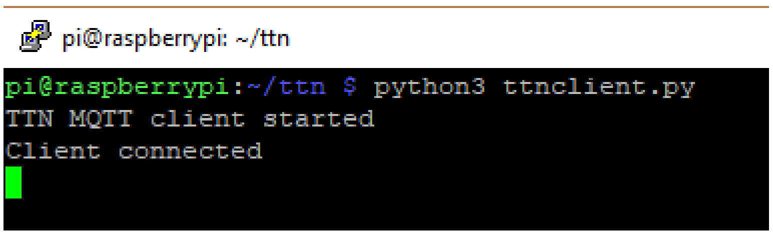
Sla het bestand op met Ctrl + X en J.

8.5.2 Testen

Start the client with the commands:

- cd ~
- cd ttn
- python3 ttnclient.py

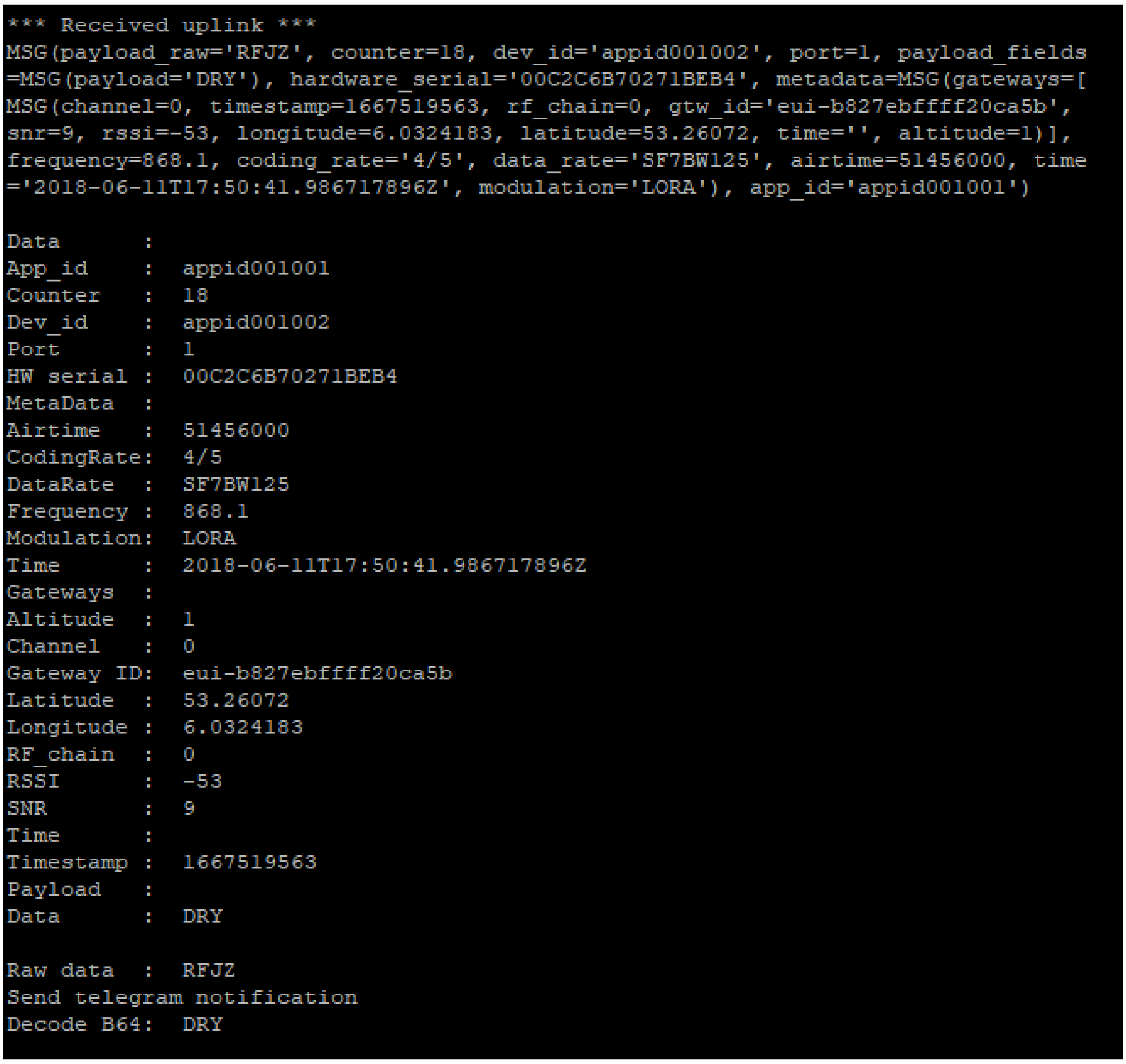
Result:



pi@raspberrypi: ~/ttn

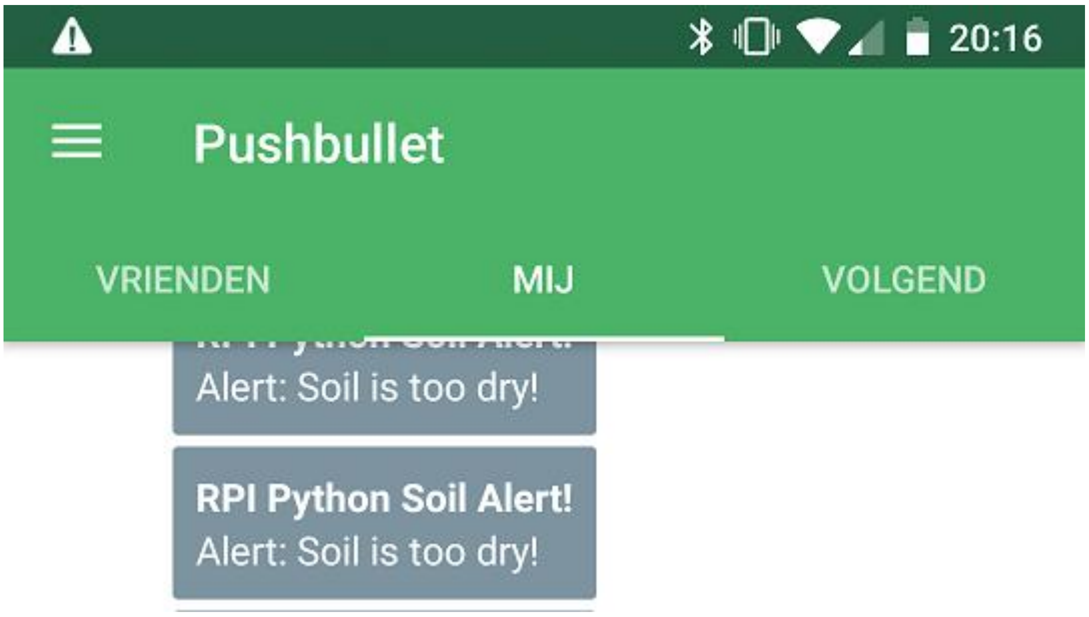
```
pi@raspberrypi:~/ttn $ python3 ttnclient.py
TTN MQTT client started
Client connected
```

As soon as a node sends a message this will be shown. A PushBullet message is also sent if the soil is too dry or too wet.



```
*** Received uplink ***
MSG(payload_raw='RFJZ', counter=18, dev_id='appid001002', port=1, payload_fields
=MSG(payload='DRY'), hardware_serial='00C2C6B70271BEB4', metadata=MSG(gateways=[
MSG(channel=0, timestamp=1667519563, rf_chain=0, gtw_id='eui-b827ebffff20ca5b',
snr=9, rssi=-53, longitude=6.0324183, latitude=53.26072, time='', altitude=1)],
frequency=868.1, coding_rate='4/5', data_rate='SF7BW125', airtime=51456000, time
='2018-06-11T17:50:41.986717896Z', modulation='LORA'), app_id='appid001001')

Data      :
App_id    :  appid001001
Counter   :  18
Dev_id    :  appid001002
Port      :  1
HW serial :  00C2C6B70271BEB4
MetaData  :
Airtime   :  51456000
CodingRate:  4/5
DataRate  :  SF7BW125
Frequency :  868.1
Modulation:  LORA
Time      :  2018-06-11T17:50:41.986717896Z
Gateways  :
Altitude  :  1
Channel   :  0
Gateway ID:  eui-b827ebffff20ca5b
Latitude  :  53.26072
Longitude :  6.0324183
RF_chain  :  0
RSSI      :  -53
SNR       :  9
Time      :
Timestamp :  1667519563
Payload   :
Data      :  DRY

Raw data  :  RFJZ
Send telegram notification
Decode B64:  DRY
```

A PushBullet message is shown on the mobile:



Alert: Soil is too dry!

RPI Python Soil Alert!
Alert: Soil is too dry!

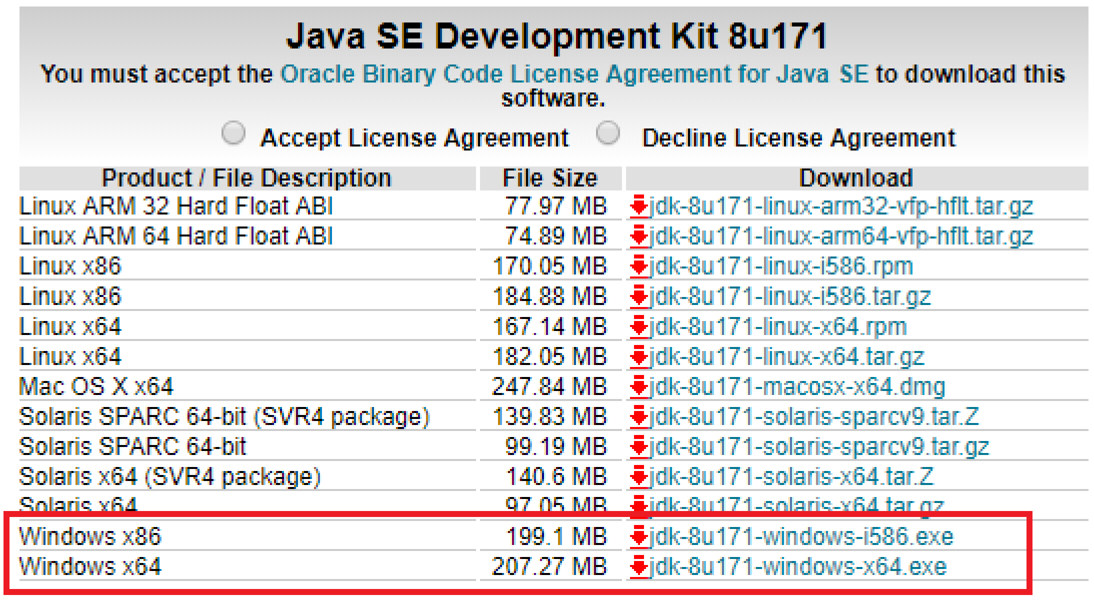
8.6 Windows 10 Java client

A client can run on any platform. In this example, we are going to create a Java client that runs under Windows 10.

8.6.1 JDK8 en Maven

8.6.1.1 Java SDK (JDK8)

If necessary, download the 32 or 64 bit version of the Java Software Development Kit. Go to:
http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html



Java SE Development Kit 8u171

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

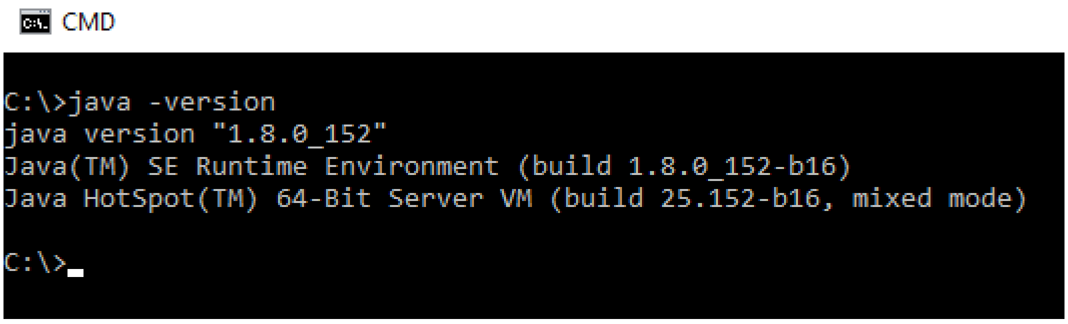
○ Accept License Agreement ○ Decline License Agreement

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.97 MB	jdk-8u171-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.89 MB	jdk-8u171-linux-arm64-vfp-hflt.tar.gz
Linux x86	170.05 MB	jdk-8u171-linux-i586.rpm
Linux x86	184.88 MB	jdk-8u171-linux-i586.tar.gz
Linux x64	167.14 MB	jdk-8u171-linux-x64.rpm
Linux x64	182.05 MB	jdk-8u171-linux-x64.tar.gz
Mac OS X x64	247.84 MB	jdk-8u171-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	139.83 MB	jdk-8u171-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	99.19 MB	jdk-8u171-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	140.6 MB	jdk-8u171-solaris-x64.tar.Z
Solaris x64	97.05 MB	jdk-8u171-solaris-x64.tar.gz
Windows x86	199.1 MB	jdk-8u171-windows-i586.exe
Windows x64	207.27 MB	jdk-8u171-windows-x64.exe

Install the JDK.

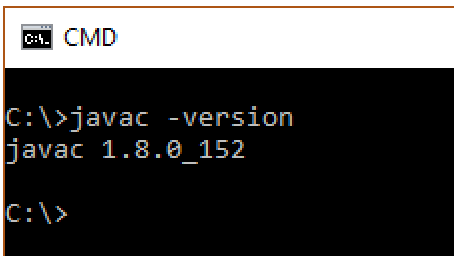
Create/change the following environment variables:
- JAVA_HOME: refer to the JDK installation directory
- PATH: add% JAVA_HOME% \ bin directory

Check the Java Runtime version with the command: java -version



```
C:\>java -version
java version "1.8.0_152"
Java(TM) SE Runtime Environment (build 1.8.0_152-b16)
Java HotSpot(TM) 64-Bit Server VM (build 25.152-b16, mixed mode)

C:\>_
```

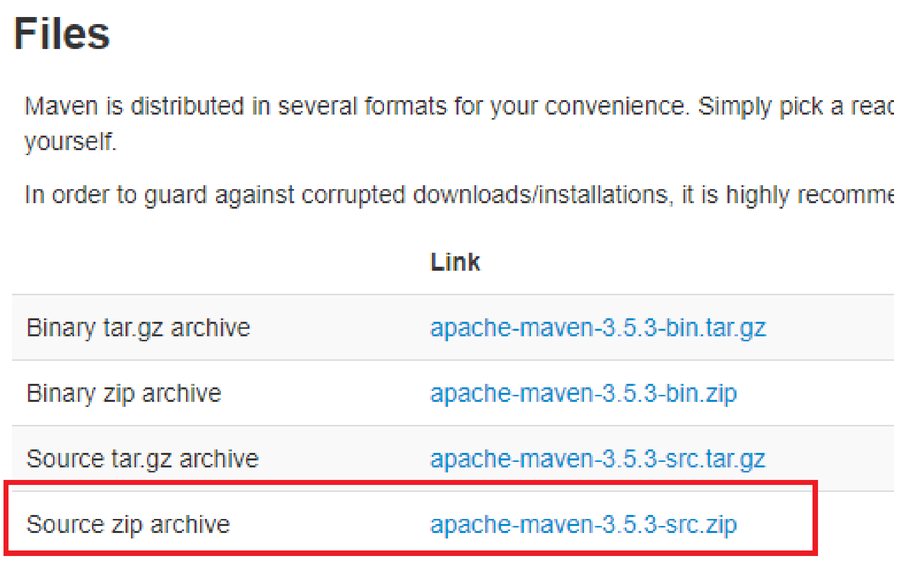
Check the Java compiler version with the command: javac -version



```
C:\>javac -version
javac 1.8.0_152

C:\>
```

8.6.1.2 Maven

If necessary, download the latest version of Maven. Go to:
https://maven.apache.org/download.cgi



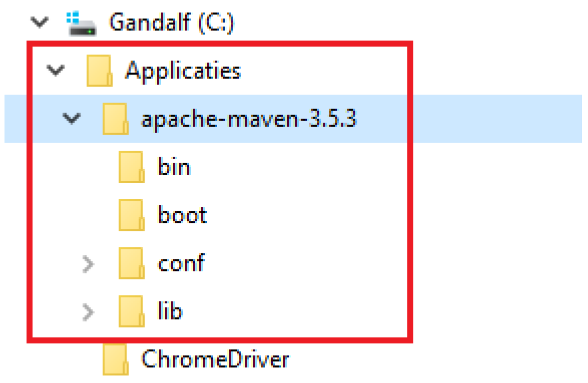
Files

Maven is distributed in several formats for your convenience. Simply pick a reac
yourself.

In order to guard against corrupted downloads/installations, it is highly recomme

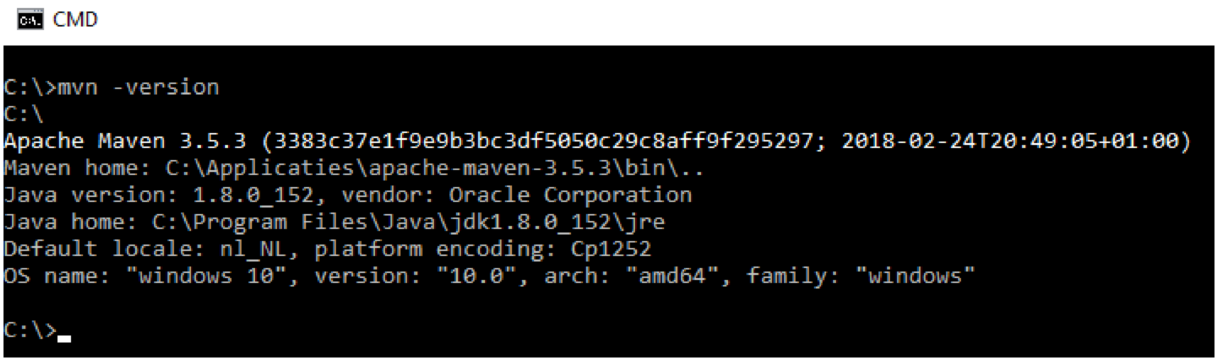
	Link
Binary tar.gz archive	apache-maven-3.5.3-bin.tar.gz
Binary zip archive	apache-maven-3.5.3-bin.zip
Source tar.gz archive	apache-maven-3.5.3-src.tar.gz
Source zip archive	apache-maven-3.5.3-src.zip

Unzip to a location of your choice, for example:
C:\Applicaties



Create/change the following environment variables:
- MAVEN_HOME: refer to the installation directory of Maven
- PATH: add %MAVEN_HOME%\bin

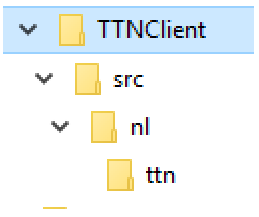
Check the installed Maven version: mvn -version



```
C:\>mvn -version
C:\
Apache Maven 3.5.3 (3383c37e1f9e9b3bc3df5050c29c8aff9f295297; 2018-02-24T20:49:05+01:00)
Maven home: C:\Applicaties\apache-maven-3.5.3\bin\..
Java version: 1.8.0_152, vendor: Oracle Corporation
Java home: C:\Program Files\Java\jdk1.8.0_152\jre
Default locale: nl_NL, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"

C:\>_
```

8.6.2 Directory structure

Create the following structure:



Commands example:

- C:
- mkdir projecten
- cd\projecten
- mkdir TTNClient
- cd TTNClient
- md src
- cd src
- md nl
- cd nl
- md ttn
- cd ttn

Create a pom.xml file in the map TTNClient.

Create the files:

- TTNClient.java and
- PushBullet.java

in the directory TTNClient\src\nl\ttn

Use a tekst editor like Notepad or Notepad++.

8.6.4 Dependencies: pom.xml

Content of the pom.xml file:

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>TTNClient</groupId>
  <artifactId>TTNClient</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <build>
    <sourceDirectory>src</sourceDirectory>
    <plugins>
      <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.7.0</version>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
    </plugins>
  </build>
    <dependencies>
          <dependency>
              <groupId>org.json</groupId>
              <artifactId>json</artifactId>
              <version>20180130</version>
          </dependency>
        <dependency>
            <groupId>org.thethingsnetwork</groupId>
            <artifactId>data-mqtt</artifactId>
            <version>2.1.2</version>
        </dependency>
    </dependencies>
</project>
```

8.6.5 PushBullet.java

Class PushBullet source:

```java
package nl.ttn;

import java.io.IOException;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.ProtocolException;
import java.net.URL;

public class PushBullet {
     // Constants
     final static String accessToken = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX";
     final static String host = "https://api.pushbullet.com:443/v2/pushes";

     public static void send(String status) {
             // Locals
             URL url = null;
             HttpURLConnection con = null;
             int responseCode;
             String postData = "{\"body\":\"Soil is too " + status + "!\",
                     \"title\":\"Java Soil Alert!\",\"type\":\"note\"}";

             // Send message
             try {
                     url = new URL(host);
                     con = (HttpURLConnection) url.openConnection();
                     con.setDoOutput(true);
                     con.setRequestProperty("Authorization", "Bearer " + accessToken);
                     con.setRequestProperty("Content-Type", "application/json");
                     con.setRequestProperty("Content-Length",
                                     Integer.toString(postData.length()));
                     con.getOutputStream().write(postData.getBytes("UTF8"));

                     // Check response
                     responseCode = con.getResponseCode();
                     System.out.println("\nSending 'POST' request to URL : " + url);
                     System.out.println("Response Code : " + responseCode);

             } catch (ProtocolException e) {
                     e.printStackTrace();
             } catch (MalformedURLException e) {
                     e.printStackTrace();
             } catch (IOException e) {
                     e.printStackTrace();
             }
     }
}
```

8.6.6 TTNClient.java

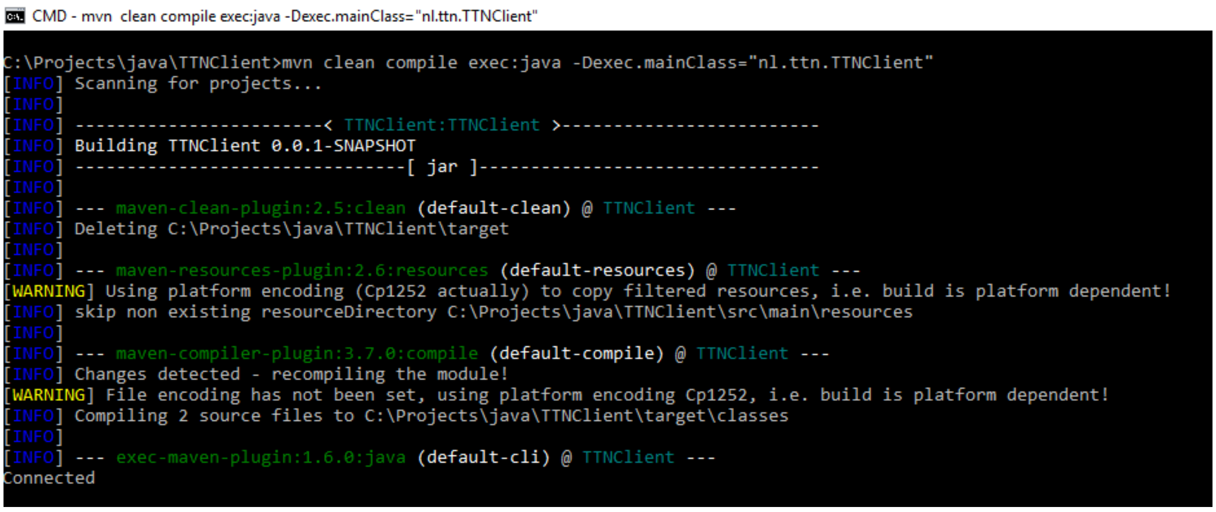
Class TTNClient source:

```
package nl.ttn;

import java.net.URISyntaxException;
import java.util.List;

import org.thethingsnetwork.data.common.Connection;
import org.thethingsnetwork.data.common.Metadata;
import org.thethingsnetwork.data.common.Metadata.Gateway;
import org.thethingsnetwork.data.common.messages.DataMessage;
import org.thethingsnetwork.data.common.messages.UplinkMessage;
import org.thethingsnetwork.data.mqtt.Client;

public class TTNClient {

    private static void showData(DataMessage data) {
           // Data
        System.out.println();
        System.out.println("Data      :");
        System.out.println("App_id    : " + ((UplinkMessage) data).getAppId());
        System.out.println("Counter   : " + ((UplinkMessage) data).getCounter());
        System.out.println("Dev_id    : " + ((UplinkMessage) data).getDevId());
        System.out.println("HW serial : " + ((UplinkMessage) data).getHardwareSerial());
        System.out.println("Port      : " + ((UplinkMessage) data).getPort());

        // Metdata
        Metadata md = ((UplinkMessage) data).getMetadata();
        System.out.println("Metadata  :");
        System.out.println("BitRate   : " + md.getBitRate());
        System.out.println("CodingRate: " + md.getCodingRate());
        System.out.println("DataRate  : " + md.getDataRate());
        System.out.println("Frequecy  : " + md.getFrequency());
        System.out.println("Modulation: " + md.getModulation());
        System.out.println("Time      : " + md.getTime());

        // Gateway data
        List<Gateway> gw  = ((UplinkMessage) data).getMetadata().getGateways();
        System.out.println("Gateways  : ");
        System.out.println("Altitude  : " + gw.get(0).getAltitude());
        System.out.println("Channel   : " + gw.get(0).getChannel());
        System.out.println("GatewayId : " + gw.get(0).getId());
        System.out.println("Latitude  : " + gw.get(0).getLatitude());
        System.out.println("Longitude : " + gw.get(0).getLongitude());
        System.out.println("RFChain   : " + gw.get(0).getRfChain());
        System.out.println("RSSI      : " + gw.get(0).getRssi());
        System.out.println("SNR       : " + gw.get(0).getSnr());
        System.out.println("Time      : " + gw.get(0).getTime());
        System.out.println("Timestamp : " + gw.get(0).getTimestamp());

        // Payload
        System.out.println("Payload  : ");
        System.out.println("Data     : " + ((UplinkMessage)
                            data).getPayloadFields().get("payload"));
        byte[] payload = ((UplinkMessage) data).getPayloadRaw();
        System.out.print(  "Raw data : ");
        for(int i=0; i<payload.length; i++) {
            char ch = (char) payload[i];
            System.out.print(ch);
        }
        System.out.println();
       System.out.println();
       alertTest((String) ((UplinkMessage) data).getPayloadFields().get("payload"));
       System.out.println();
       System.out.println();
    }
    public static void alertTest(String status) {
            if(status.equals("DRY")) {
                    System.out.println("+----------------------+");
                    System.out.println("| Status: Soil to dry! | ");
                    System.out.println("+----------------------+");
                    PushBullet.send("dry");
            } else if(status.equals("WET")) {
                    System.out.println("+----------------------+");
                    System.out.println("| Status: Soil to wet! |");
                    System.out.println("+----------------------+");
                    PushBullet.send("wet");
            } else {
                    System.out.println("+----------------------+");
                    System.out.println("| Status: OK           |");
                    System.out.println("+----------------------+");
            }
    }

    public static void main(String[] args) {
        String region = "eu";
        String appId = "appid001001";
        String accessKey = "ttn-account-v2.kYhYtD8v-1D13MEvSB-o4rNTgUceQprmyTK9Fnbpz9g";

        try {
                    Client client = new Client(region, appId, accessKey);
            client.onError((Throwable _error) -> System.err.println("Error: " +
                            _error.getMessage()));
            client.onConnected((Connection _client) -> System.out.println("Connected"));
            client.onMessage((String devId, DataMessage data) -> {
                System.out.println(
                    "------------------------------------------------------------------");
                    showData(data);
            });
                    client.start();
            } catch (URISyntaxException e1) {
                    System.err.println(e1.getMessage());
                    e1.printStackTrace();
            } catch (Exception e2) {
                    System.err.println(e2.getMessage());
                    e2.printStackTrace();
            }
    }
}
```

8.6.7 Compile and test

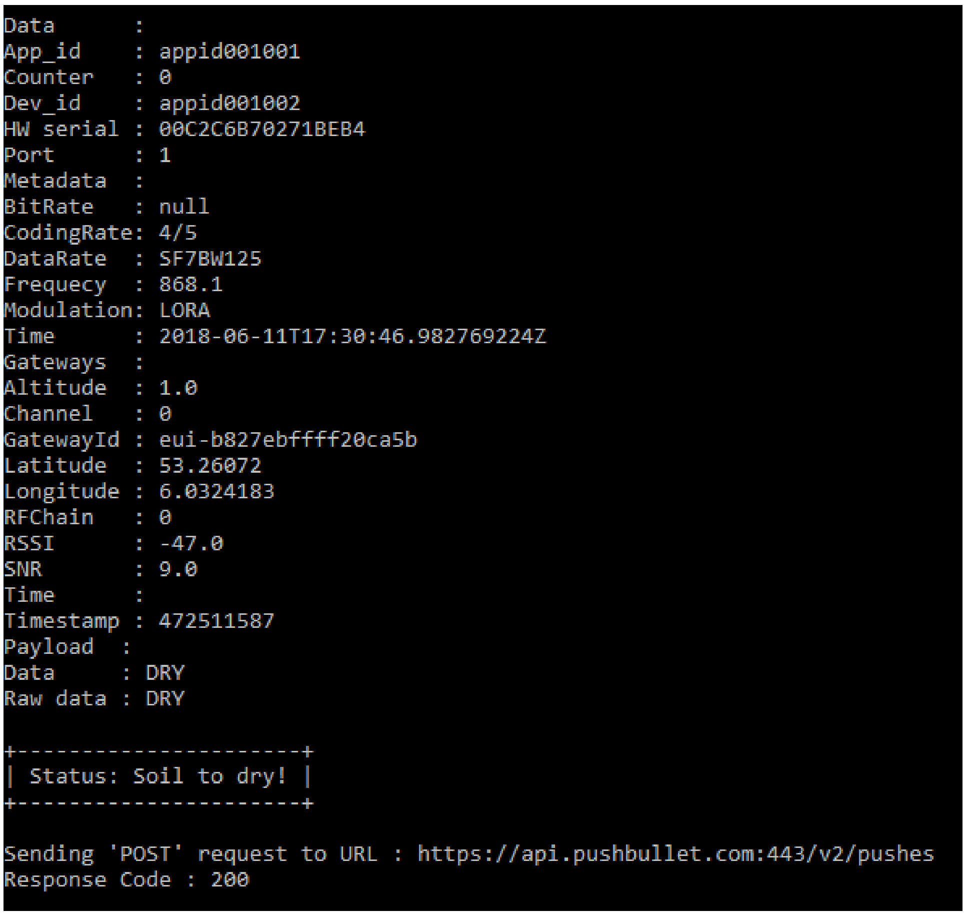
Open a command prompt and compile and run with the command:

```
mvn clean compile exec:java -Dexec.mainClass="nl.ttn.TTNClient"
```



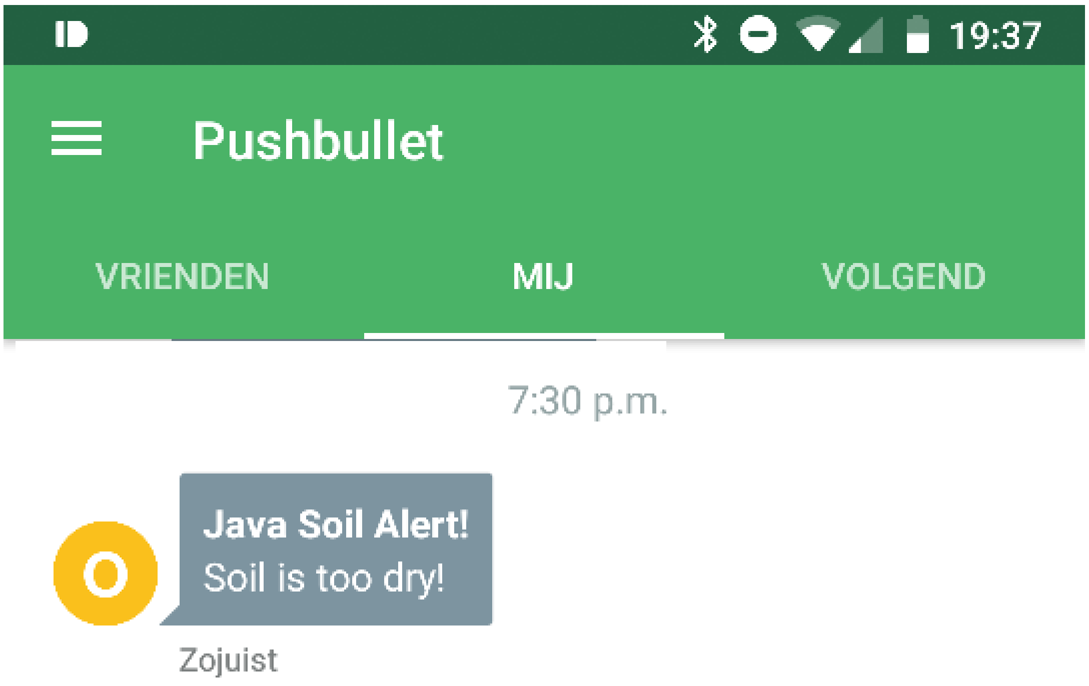
```
CMD - mvn  clean compile exec:java -Dexec.mainClass="nl.ttn.TTNClient"

C:\Projects\java\TTNClient>mvn clean compile exec:java -Dexec.mainClass="nl.ttn.TTNClient"
[INFO] Scanning for projects...
[INFO]
[INFO] -----------------------< TTNClient:TTNClient >------------------------
[INFO] Building TTNClient 0.0.1-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ TTNClient ---
[INFO] Deleting C:\Projects\java\TTNClient\target
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ TTNClient ---
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory C:\Projects\java\TTNClient\src\main\resources
[INFO]
[INFO] --- maven-compiler-plugin:3.7.0:compile (default-compile) @ TTNClient ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding Cp1252, i.e. build is platform dependent!
[INFO] Compiling 2 source files to C:\Projects\java\TTNClient\target\classes
[INFO]
[INFO] --- exec-maven-plugin:1.6.0:java (default-cli) @ TTNClient ---
Connected
```

As soon as a node message is sent, this will become visible on the console:



```
Data       :
App_id     : appid001001
Counter    : 0
Dev_id     : appid001002
HW serial  : 00C2C6B70271BEB4
Port       : 1
Metadata   :
BitRate    : null
CodingRate: 4/5
DataRate   : SF7BW125
Frequecy   : 868.1
Modulation: LORA
Time       : 2018-06-11T17:30:46.982769224Z
Gateways   :
Altitude   : 1.0
Channel    : 0
GatewayId  : eui-b827ebffff20ca5b
Latitude   : 53.26072
Longitude  : 6.0324183
RFChain    : 0
RSSI       : -47.0
SNR        : 9.0
Time       :
Timestamp  : 472511587
Payload   :
Data      : DRY
Raw data  : DRY

+----------------------+
| Status: Soil to dry! |
+----------------------+

Sending 'POST' request to URL : https://api.pushbullet.com:443/v2/pushes
Response Code : 200
```

A PushBullet message will appear on the mobile if the soil is too dry or too wet.



Java Soil Alert!
Soil is too dry!