goo.gl/VdqqXC

LoRaWAN IoT with Arduino Uno, Dragino v1.3 & TheThingsNetwork

for /ch/open, September 2017
& make-munich.de, May 2017
& /ch/open, September 2016

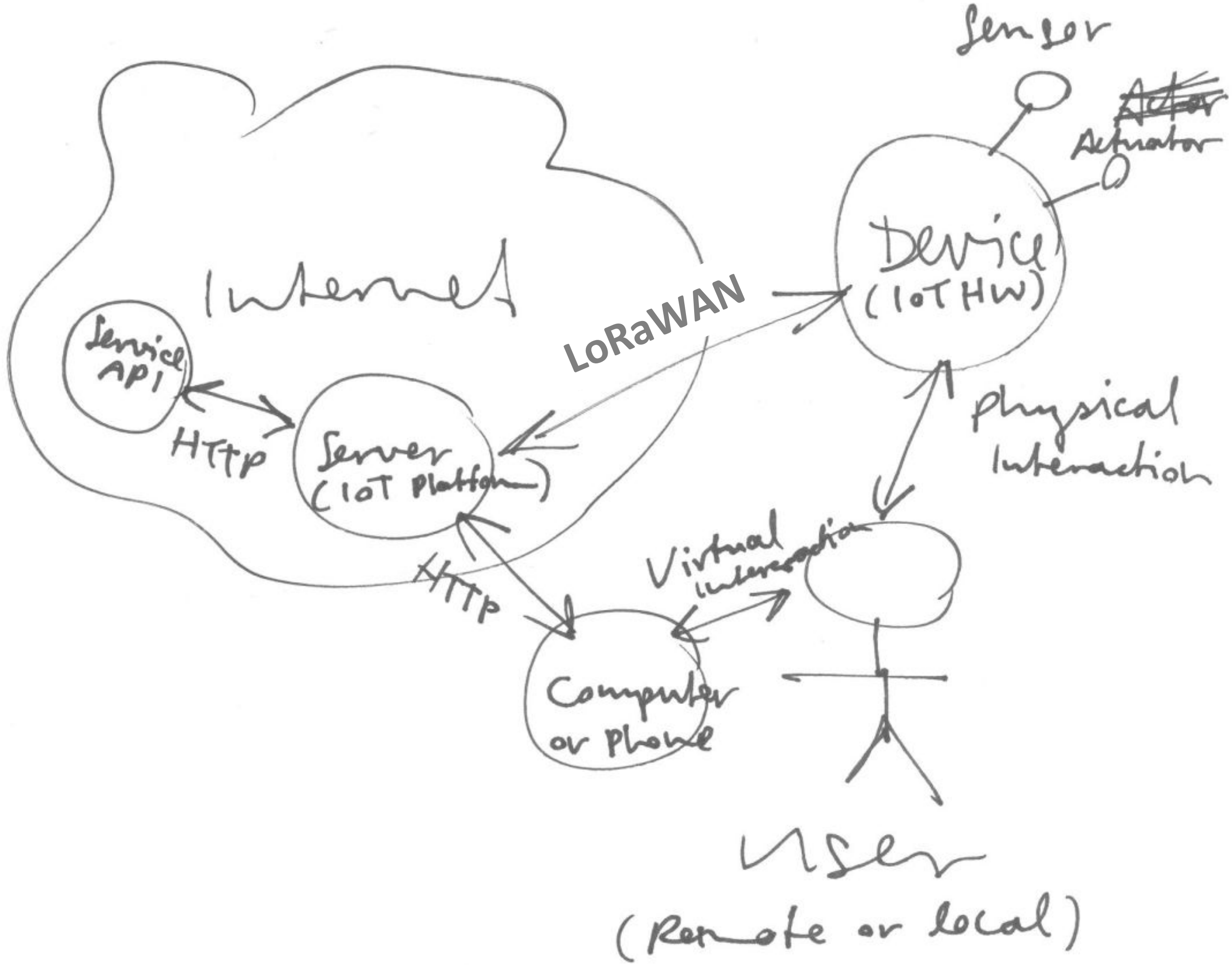
Internet of Things (IoT)

Computers with **sensors** and **actuators**, **connected** through Internet protocols.

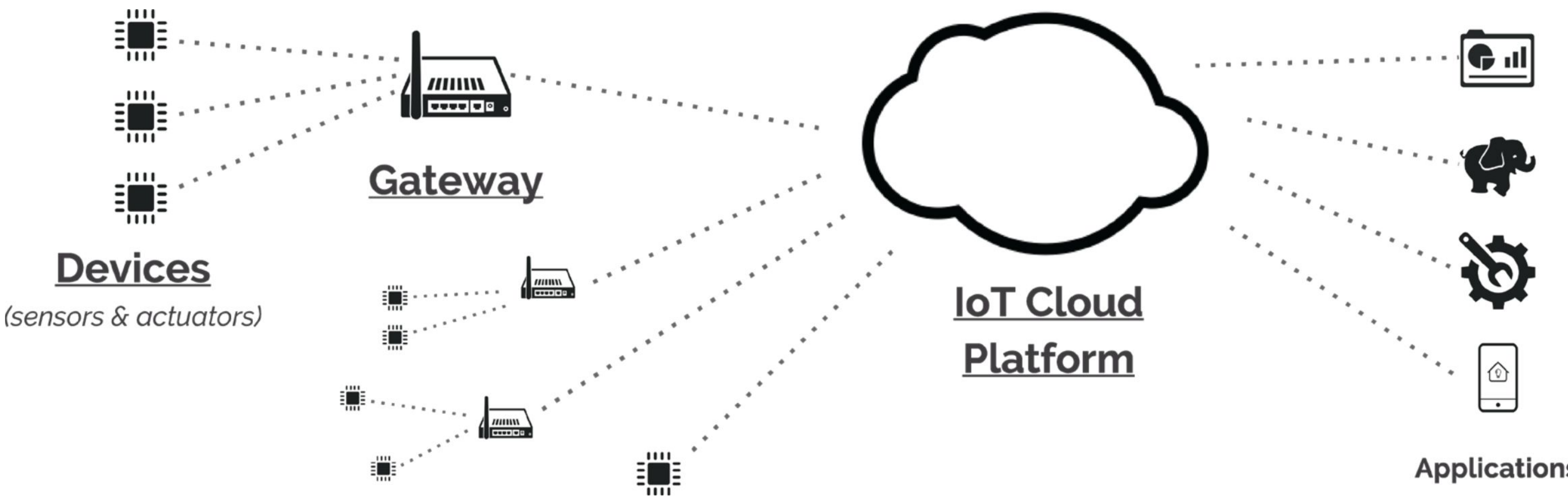
Instead of just accessing and editing virtual resources, we can now measure and manipulate **physical properties**.

For developers: IoT = physical objects with an API.

IoT reference model

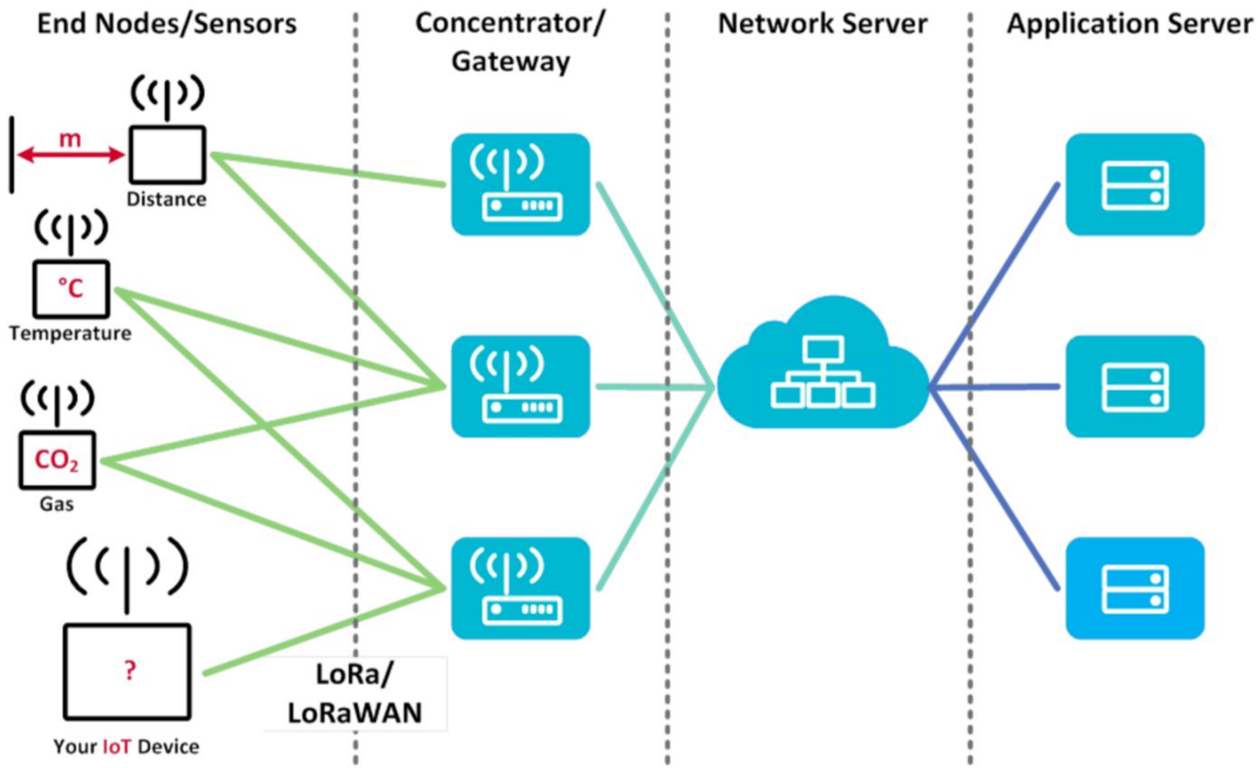


IoT reference model



LoRa Sensors Dynatronics ×

www.dynatronics.ch/lora/



- **End Nodes/Sensors** measure values and transmit the data
- **Concentrator/Gateway** receive the radio packets and trasmit them to the network server over LAN, Cell, …
- **Network Server** forwards and/or collects data from the end nodes/sensors
- **Application Server** makes something useful out of the data. Usually your server or application, that simplyfies your customers life

LoRa & LoRaWAN

Wireless communication network focused on **low power**, **long range** and **low cost**.

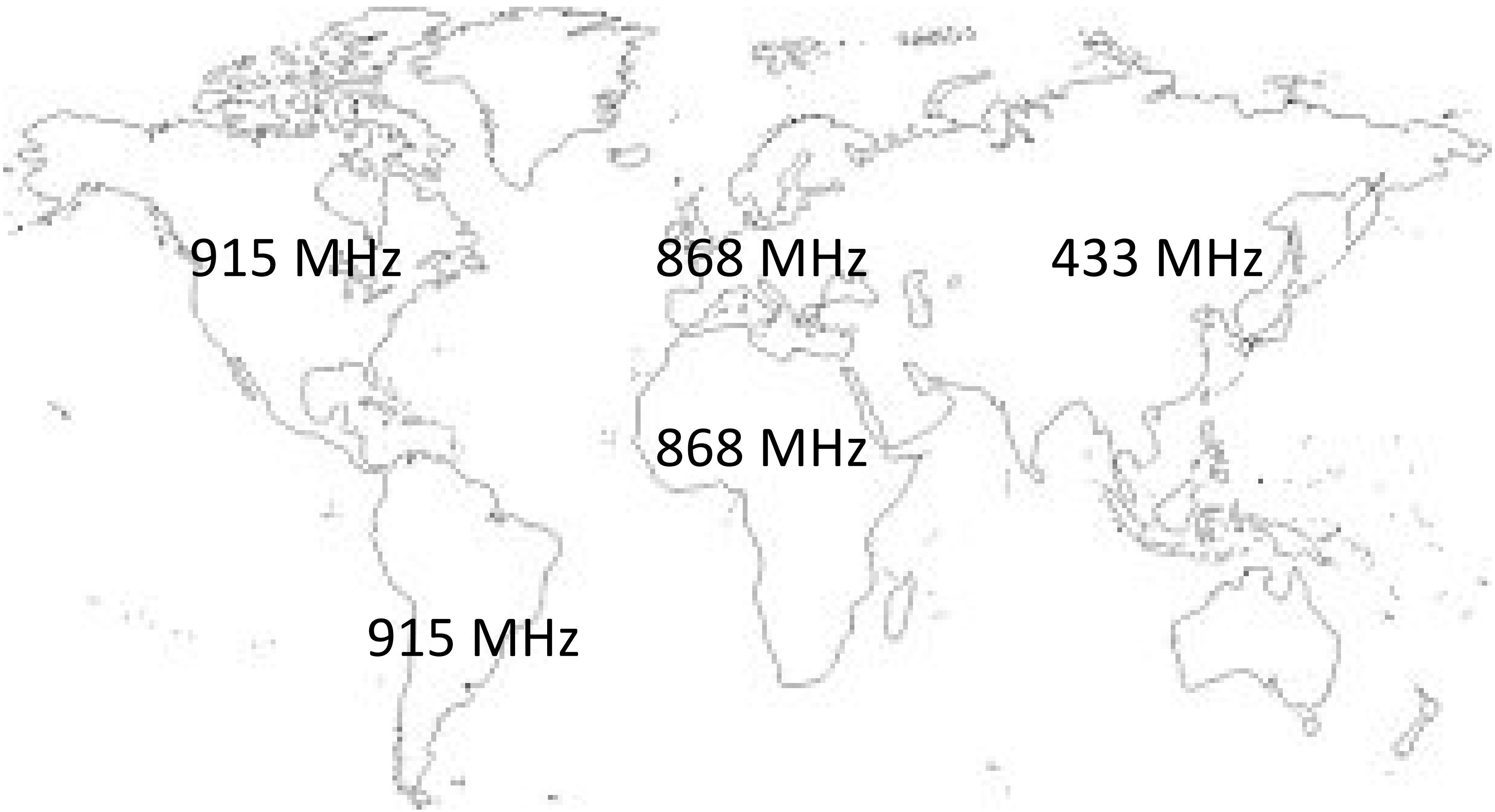
Power: ~14mA (RX), ~28mA (TX)

Cost: ~$10 (modem), ~$300 (gateway)

Range: 5km (urban) to +40km (rural)

https://www.lora-alliance.org/What-Is-LoRa

LoRa Spektrum



The Things Network

A **global community**, building **open source** software and hardware to operate a **crowd-sourced** LoRa network.

https://www.thethingsnetwork.org/

Topics of this workshop

1) **Getting started** with Arduino
2) **Using sensors and actuators** with Arduino
3) **Connecting to LoRaWAN** w/ TheThingsNetwork
4) **Sending sensor data** via LoRaWAN with lmic
5) **Forwarding sensor data** to ThingSpeak, IFTTT
6) **Sending downlink messages** to the Arduino
7) **Deploying a node** with a weatherproof case
8) **Deploying a TTN gateway** walk through

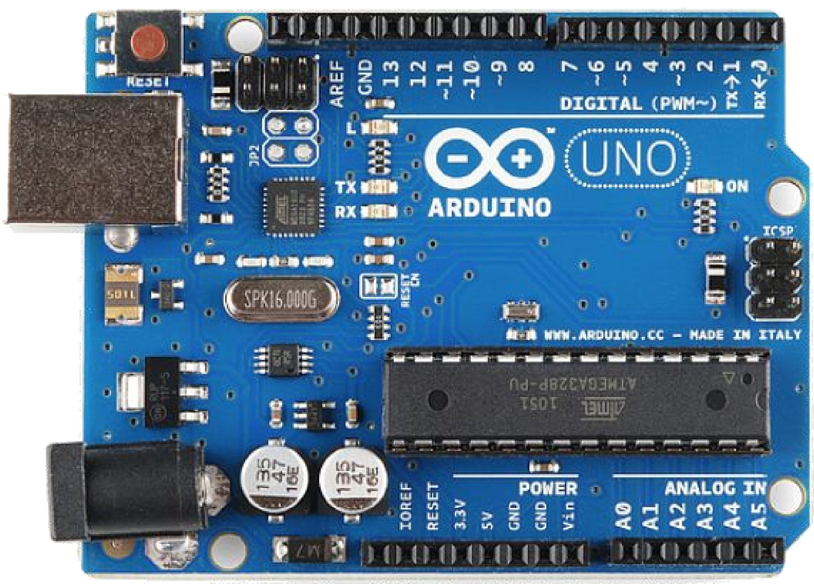
Questions? Just ask / Use Google / Help each other

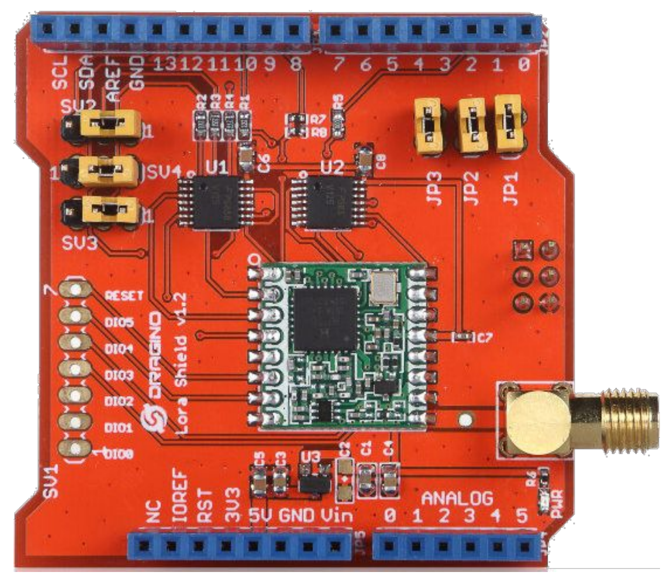
1) Getting Started

How to set up the Arduino – the basics of embedded programming, step by step.

Hardware

This tutorial is based on the **Arduino** Uno board and the Dragino v1.3 **LoRa shield**





Note: For the first part we just need the Arduino Uno

Getting started

The **IDE** (**I**ntegrated **D**evelopment **E**nvironment) allows you to **program** your board, i.e. “make it do something new”

You **edit** a program on your computer, then **upload** it to your board where it’s stored in the program memory (flash) and **executed** in RAM

Note: Once it has been programmed, your board can run on its own, without another computer

Download and install Arduino.cc

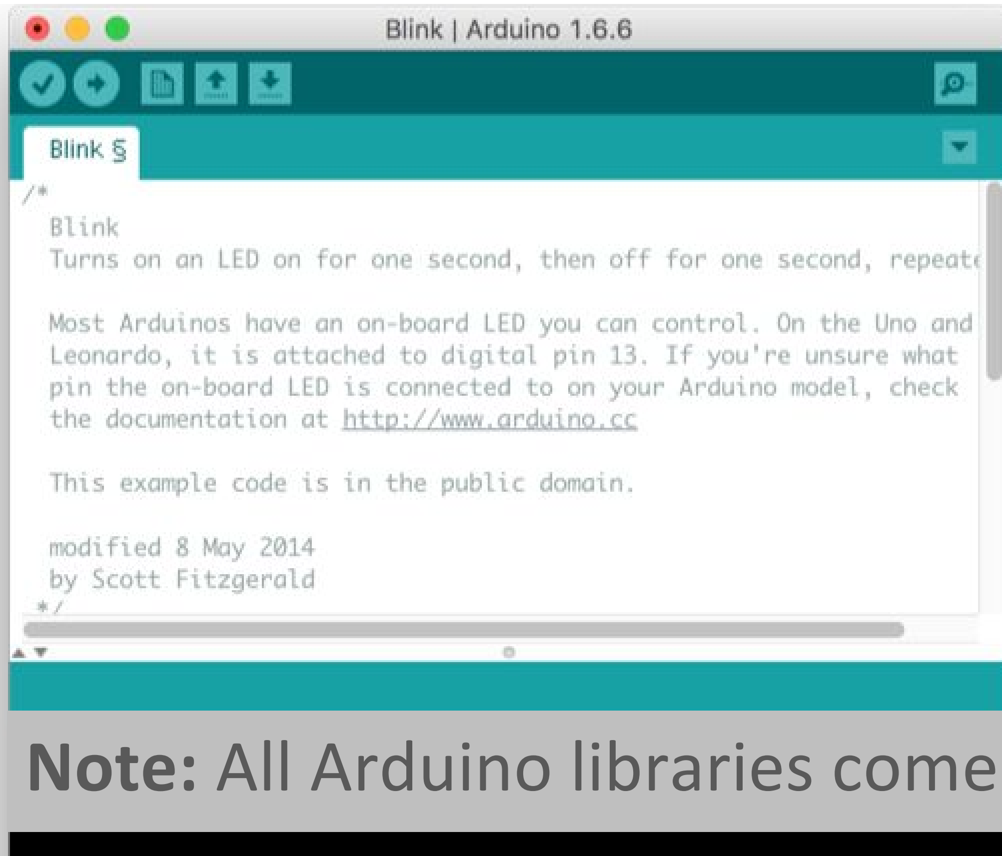
To install the **Arduino IDE** and connect your Arduino board to your computer via USB, see http://arduino.cc/en/Guide/**MacOSX** or http://arduino.cc/en/Guide/**Windows** or http://arduino.cc/playground/Learning/**Linux**

Or install https://codebender.cc/static/plugin and use the https://codebender.cc/ online IDE

Note: Codebender is great, but has some limitations

Examples included with Arduino

Go to *File > Examples > Basics* to open the *Blink* example, scroll down to see the actual code



Note: All Arduino libraries come with examples, too

Upload code to your Arduino

Select *Tools > Board > Arduino/Genuino Uno* and set the right USB port in the *Port* menu, then click *Upload*

Note: clicking *Upload* also compiles the source code

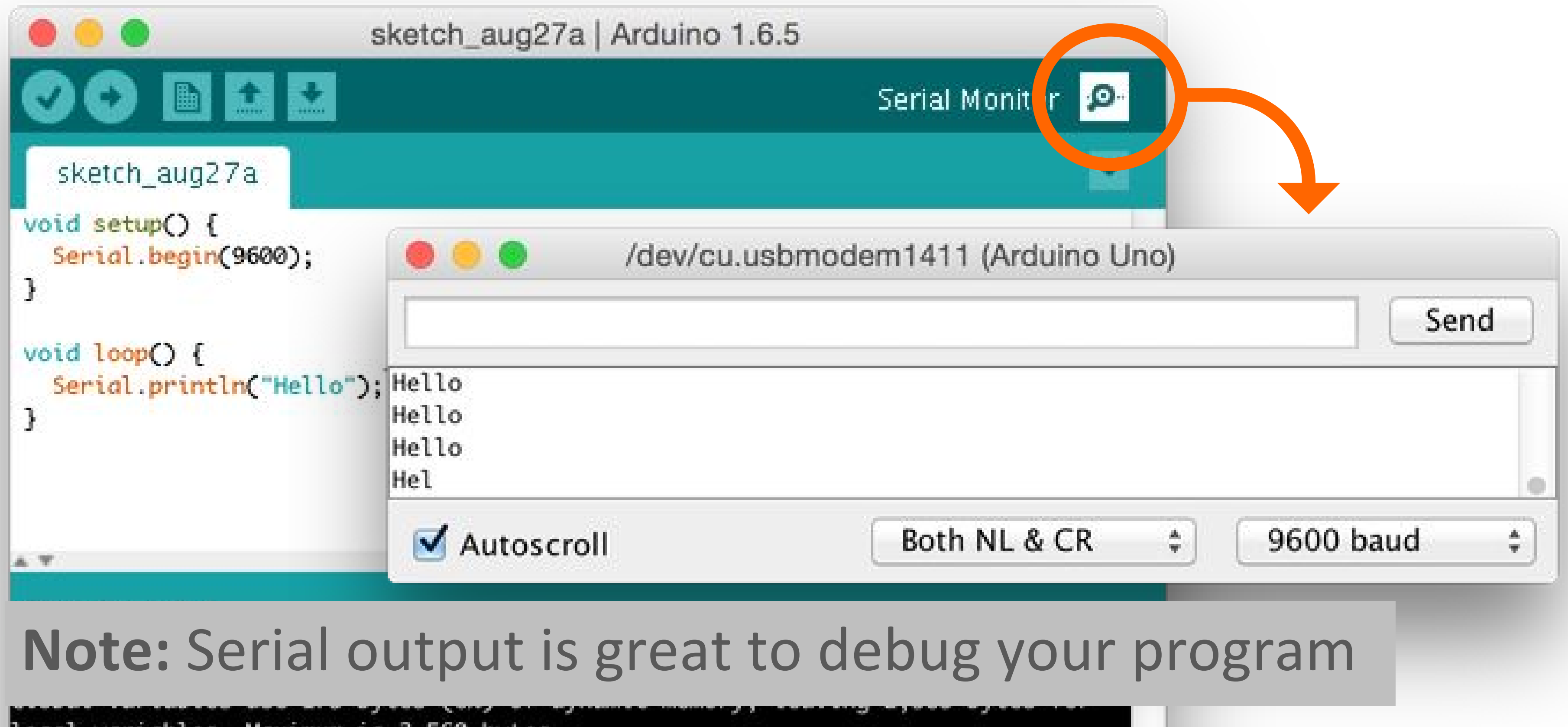
Hello, World! (serial output)

```
void setup () { // run once
  Serial.begin(9600); // set baud rate
}

void loop () { // run again and again
  Serial.println("Hello, World!"); // print output
}
```

Note: type this program code into your IDE and upload it to the device, then check the next slide

Serial output with Arduino

Click the *Serial Monitor* icon to see serial output, and make sure the baud rate (e.g. *9600*) matches your code



Note: Serial output is great to debug your program

Examples available online

Use it!

The **source code** of slides with a blue ribbon is available online, just click the link to display it.

Or download the ZIP from https://bitbucket.org/tamberg/iotworkshop/get/tip.zip

Or browse code online at https://bitbucket.org/tamberg/iotworkshop/src/tip

Note: use the *Raw* button to see files as plain text

2) Using sensors and actuators

How to measure and manipulate physical properties with sensors and actuators – the basics of electronics, interactive systems and physical computing, in a few easy examples.

Inputs and outputs

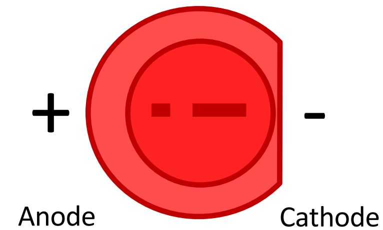
IoT hardware has an **interface to the real world**.

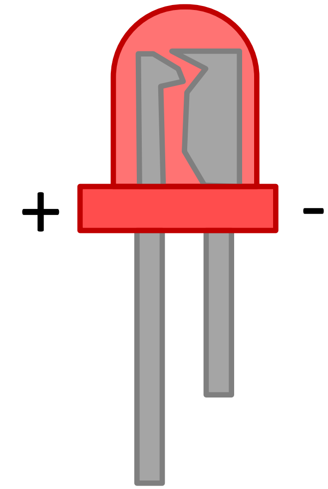
GPIO (**G**eneral **P**urpose **I**nput/**O**utput) pins.

Measure: **read** sensor value from **input** pin
Manipulate: **write** actuator value to **output** pin.

Inputs and outputs can be **digital or analog**.

The LED





The **LED** (**L**ight **E**mitting **D**iode)
is a simple digital **actuator**

LEDs have a **short leg** (**-**) and a **long leg** (**+**)
and it matters how they are oriented in a circuit

To prevent damage, LEDs are used together with
a 1KΩ **resistor** (or anything from 300Ω to 2KΩ)

The resistor

Resistors are the **workhorse of electronics**

Resistance is **measured in Ω** (Ohm)

A resistors orientation does not matter

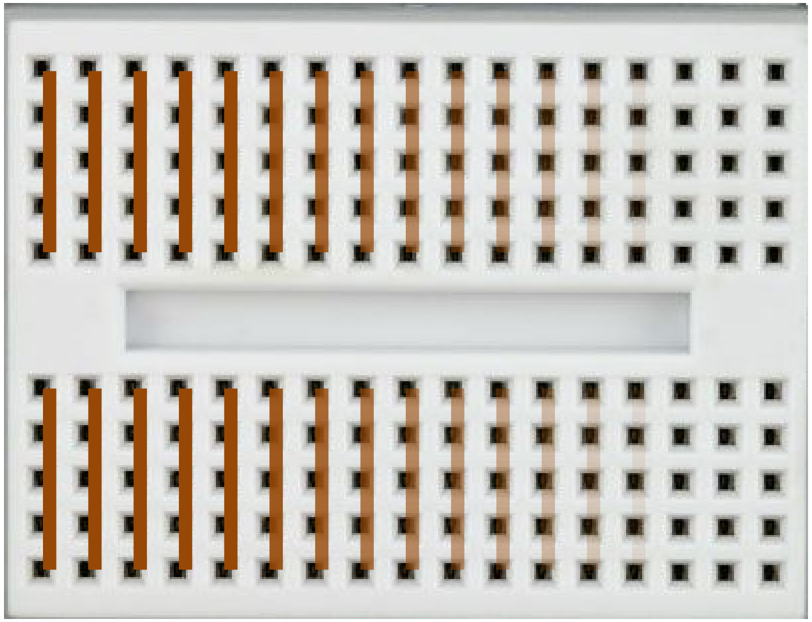
A resistors Ω value is **color-coded** right on it

Note: color codes are great, but it's easier to use a multi-meter if you've got one, and just measure Ω

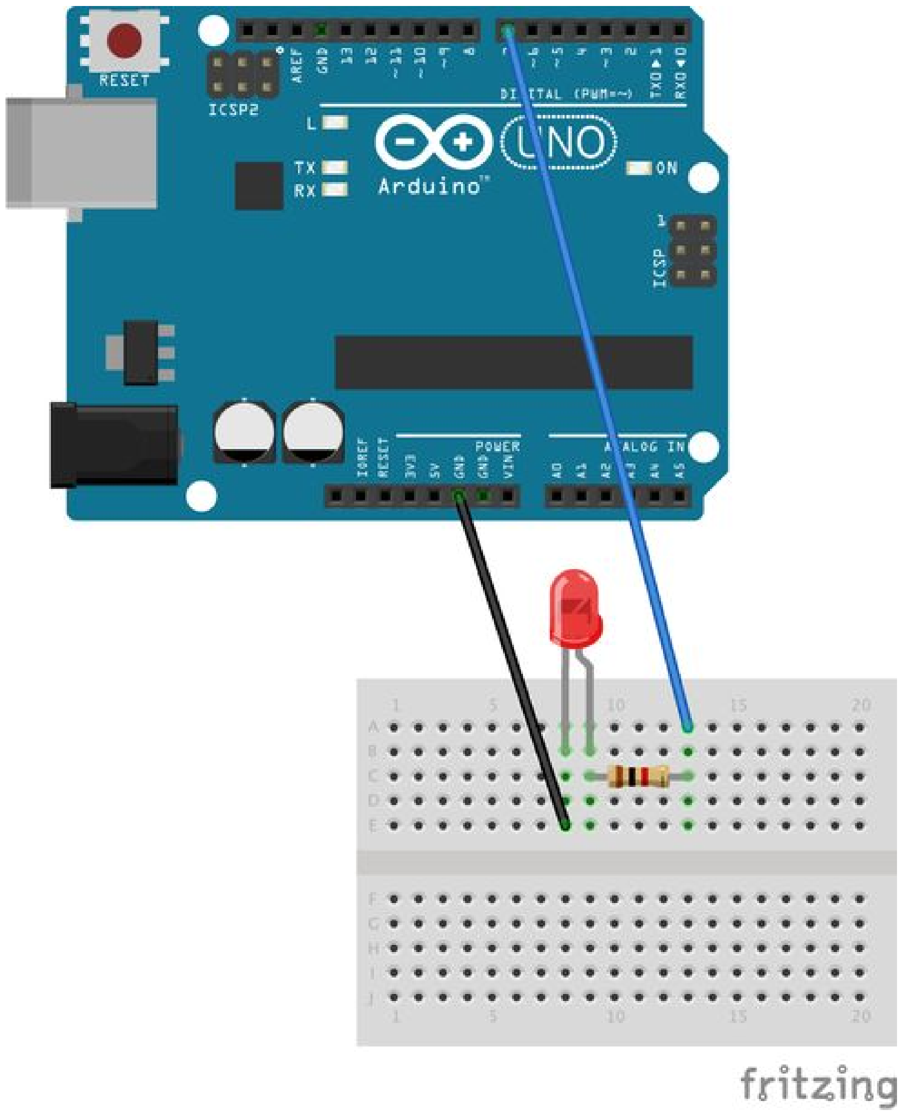
The breadboard

A breadboard lets you wire electronic components without any soldering

Its holes are connected “under the hood” as shown here



Wiring a LED with Arduino



Note: the additional **1K Ω** resistor should be used to prevent damage to the pins / LED if it's reversed

The long leg of the LED is connected to **pin D7**, the short leg to ground (**GND**)

Blinking a LED (digital output)

Digital Output.ino

Get code here ↗

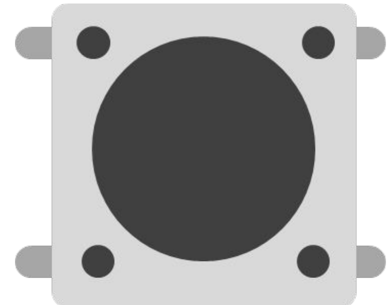
```
int ledPin = 7;

void setup () {
  pinMode(ledPin, OUTPUT);
}
void loop () {
  digitalWrite(ledPin, HIGH);
  delay(500); // wait 500ms
  digitalWrite(ledPin, LOW);
  delay(500);
}
```

Blinking an LED is the *Hello World* of embedded software

Set *ledPin* as wired in your LED circuit

HIGH = digital 1 (5V) means LED is **on**, **LOW** = digital 0 (0V) means LED is **off**

The switch



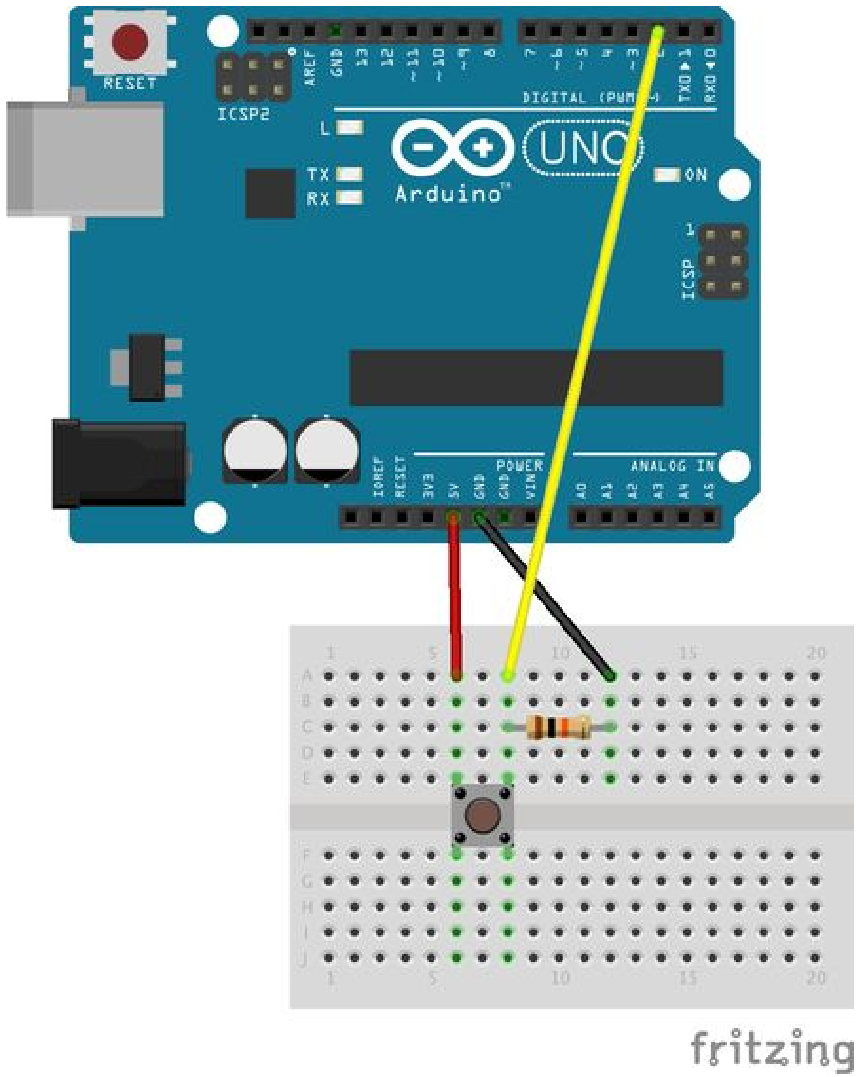
A switch is a simple, digital **sensor**

Switches come in different forms, but all of them in some way **open** or **close** a gap in a wire

The **pushbutton** switch has four legs for easier mounting, but only two of them are needed

Note: you can also easily build your own switches, for inspiration see e.g. http://vimeo.com/2286673

Wiring a switch with Arduino



Note: the resistor in this setup is called *pull-down* 'cause it pulls the pin voltage down to GND (0V) if the switch is open

Pushbutton **switch**

10K Ω resistor

5V

GND

D2 (max input 5V!)

Reading a switch (digital input)

Digital Input.ino

```
int sensorPin = 2; // e.g. button switch

void setup () {
    Serial.begin(9600); // set baud rate
    pinMode(sensorPin, INPUT);
}

void loop () {
    int sensorValue = digitalRead(sensorPin);
    Serial.println(sensorValue); // print 0 or 1
}
```

Open the IDE serial monitor or terminal to see log output

Switching a LED

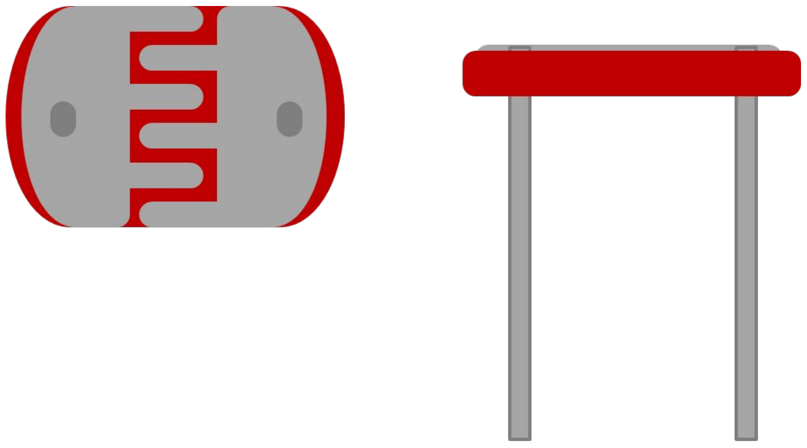
Switched LED.ino

```
int switchPin = 2;
int ledPin = 7; // or 13
void setup () {
  pinMode(switchPin, INPUT);
  pinMode(ledPin, OUTPUT);
}
void loop () {
  int switchValue = digitalRead(switchPin);
  if (switchValue == 0) {
    digitalWrite(ledPin, LOW);
  } else { // switchValue == 1
    digitalWrite(ledPin, HIGH);
  }
}
```

Note: figure out the wiring or just use the built-in LED, i.e. pin 13 on Arduino

The code inside an *if* statement is only executed if the con-dition is true, *else* is executed otherwise

The LDR

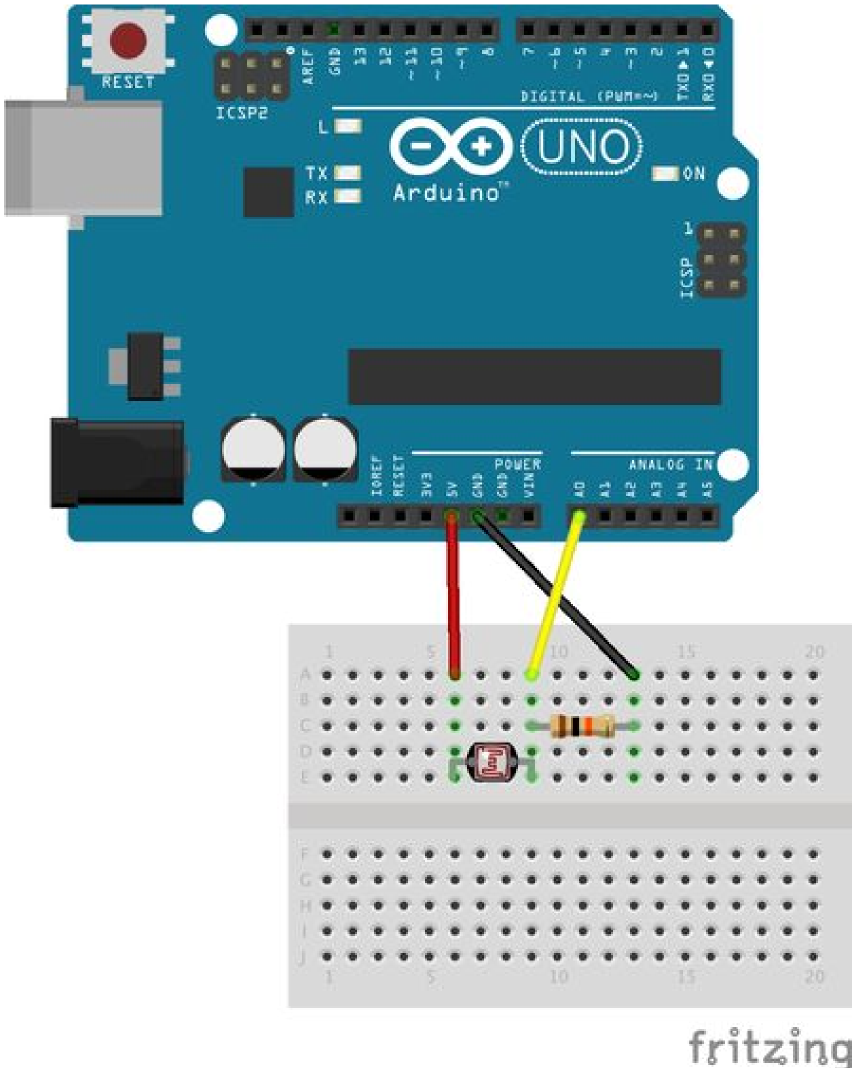


A photoresistor or **LDR** (**l**ight **d**ependent **r**esistor) is a resistor whose resistance depends on light intensity

An LDR can be used as a simple, **analog sensor**

The orientation of an LDR does not matter

Wiring an LDR with Arduino



Note: this setup is a *voltage-divider, as* the 5V total voltage is divided between LDR and resistor to keep 0V < **A0** < 2.5V

Photoresistor (LDR)

10K Ω resistor

5V

GND

A0

Reading an LDR (analog input)

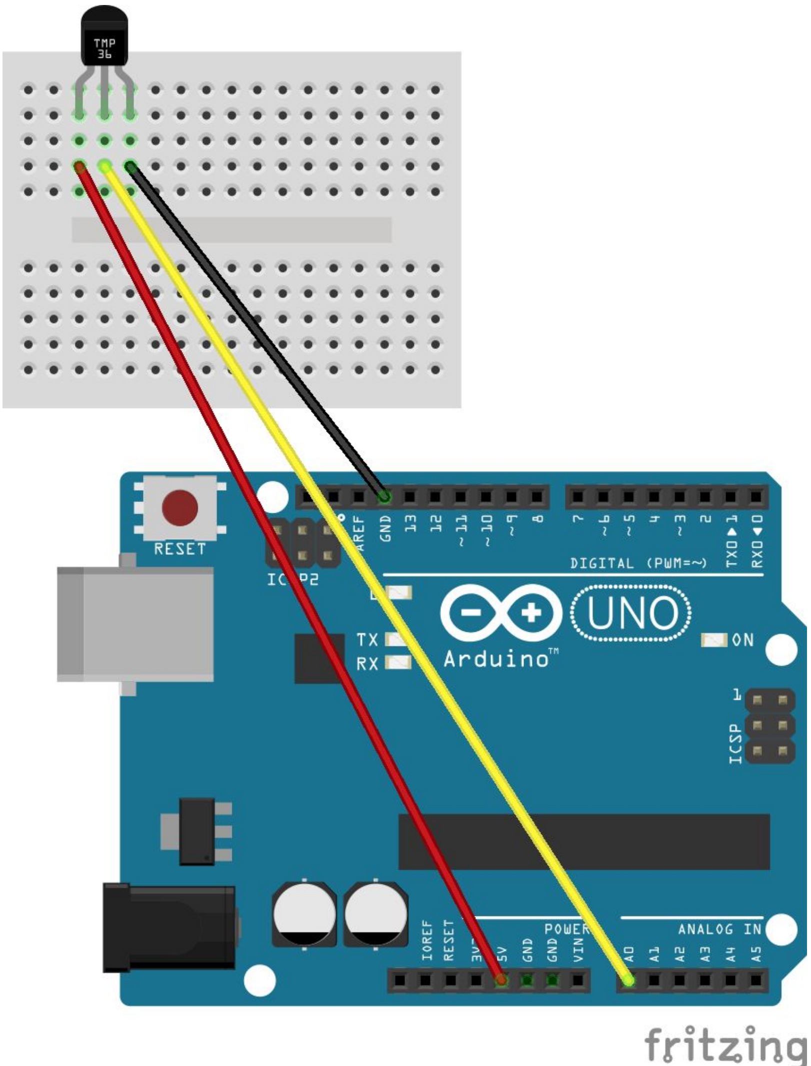
Analog Input.ino

```
int sensorPin = A0; // LDR or other analog sensor

void setup () {
    Serial.begin(9600); // set baud rate
}

void loop () {
    int sensorValue = analogRead(sensorPin);
    Serial.println(sensorValue); // print value
}
```

Open the IDE serial monitor or terminal to see log output

Note: use e.g. Excel to visualize values over time

Wiring a TMP36 with Arduino



Note: TMP36 is a cheap temperature sensor, so it is not very accurate. But a qualitative value can be read from it

Sensor (TMP36)
5V
GND
A0

Reading a TMP36 (analog input)

Analog InputTmp36.ino

```
int sensorPin = A0; // TMP36

void setup () {
  Serial.begin(9600); // set baud rate
}

void loop () {
  int sensorValue = analogRead(sensorPin);
  float voltage = (sensorValue * 5.0) / 1024.0;
  float tempCelsius = (voltage - 0.5) * 100;
  Serial.println(tempCelsius);
}
```

Open the IDE serial monitor or terminal to see log output

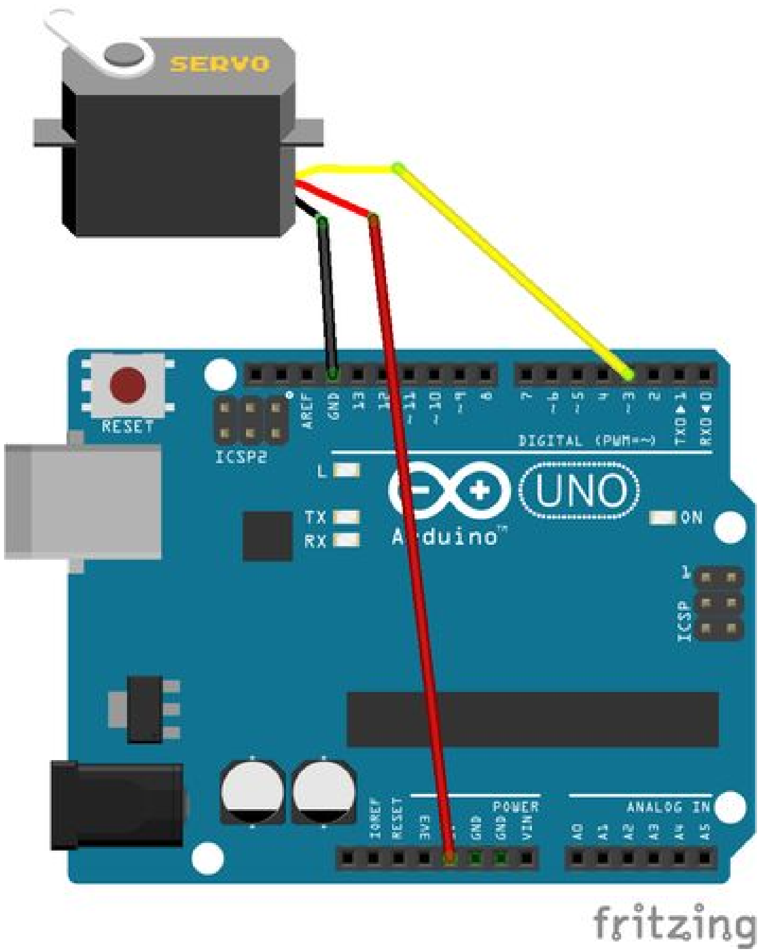
The Servo

A **servo** motor takes an input between 0 and 180 which is translated into a motor position in degrees

A servo is an **analog actuator**

To create an analog output for the servo, the device uses pulse width modulation (**PWM**)

Wiring a Servo with Arduino



Note: PWM pins on Arduino are those with a ~ symbol

5V
GND
D3 (PWM)

Controlling a Servo (PWM output)

Servo.ino

```
#include <Servo.h> // remove this line on the Photon
Servo servo; // create a new Servo object
int servoPin = 3; // a PWM pin

void setup () {
  servo.attach(servoPin);
}

void loop () {
  for (int pos = 0; pos <= 180; pos += 10) {
    servo.write(pos);
    delay(100);
  }
}
```

Note: *Servo* objects let you use Servos without PWM skills

The *for* loop repeats from pos 0 until pos is 180, in steps of 10

Controlling a Servo with an LDR

ServoWith
LDR.ino

```
#include <Servo.h> // remove this line on the Photon
Servo servo; // create a new Servo
int servoPin = 3; // a PWM pin
int sensorPin = A0; // LDR

void setup () {
   servo.attach(servoPin);
}

void loop () {
   int val = analogRead(sensorPin);
   int pos = map(val, 0, 255, 0, 180);
   servo.write(pos);
}
```

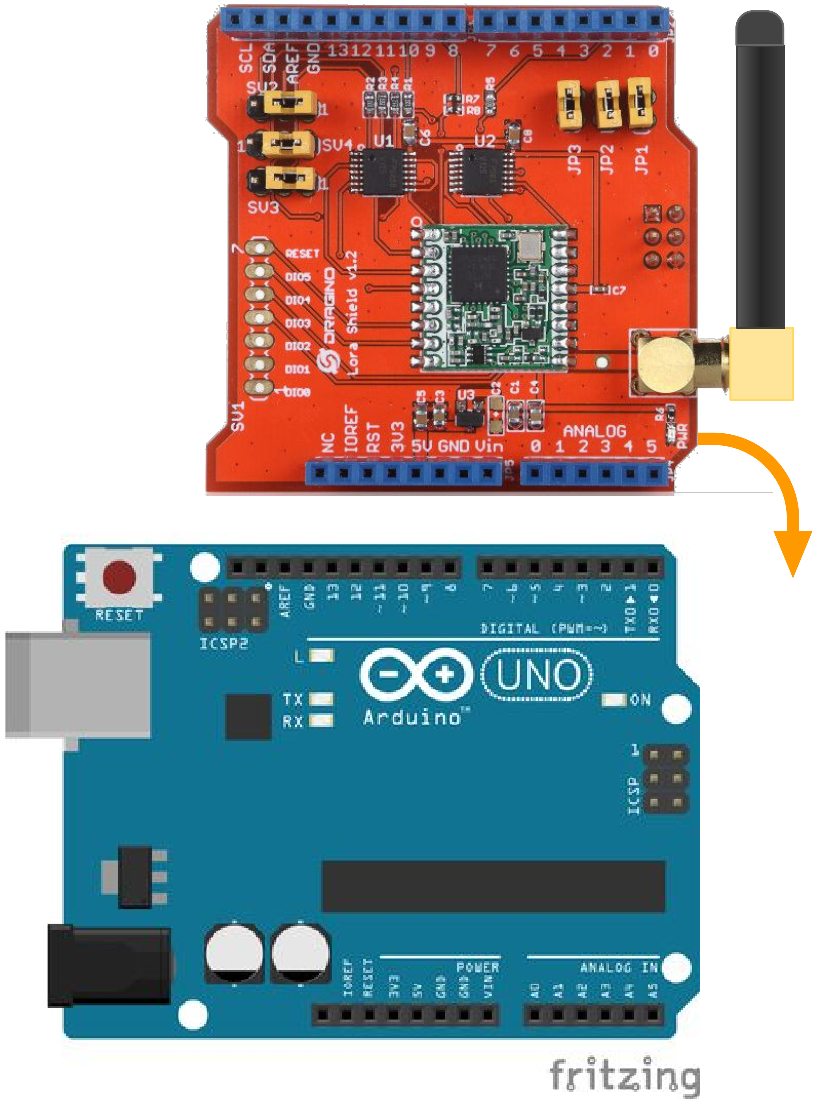
Note: combine the wiring diagrams of both, Servo & LDR

The *map* function is useful to map one range onto another

3) Connecting to LoRaWAN

How to get network and application keys and connect your Arduino to the LoRa wide area network provided by TheThingsNetwork.

Adding the Dragino LoRa shield



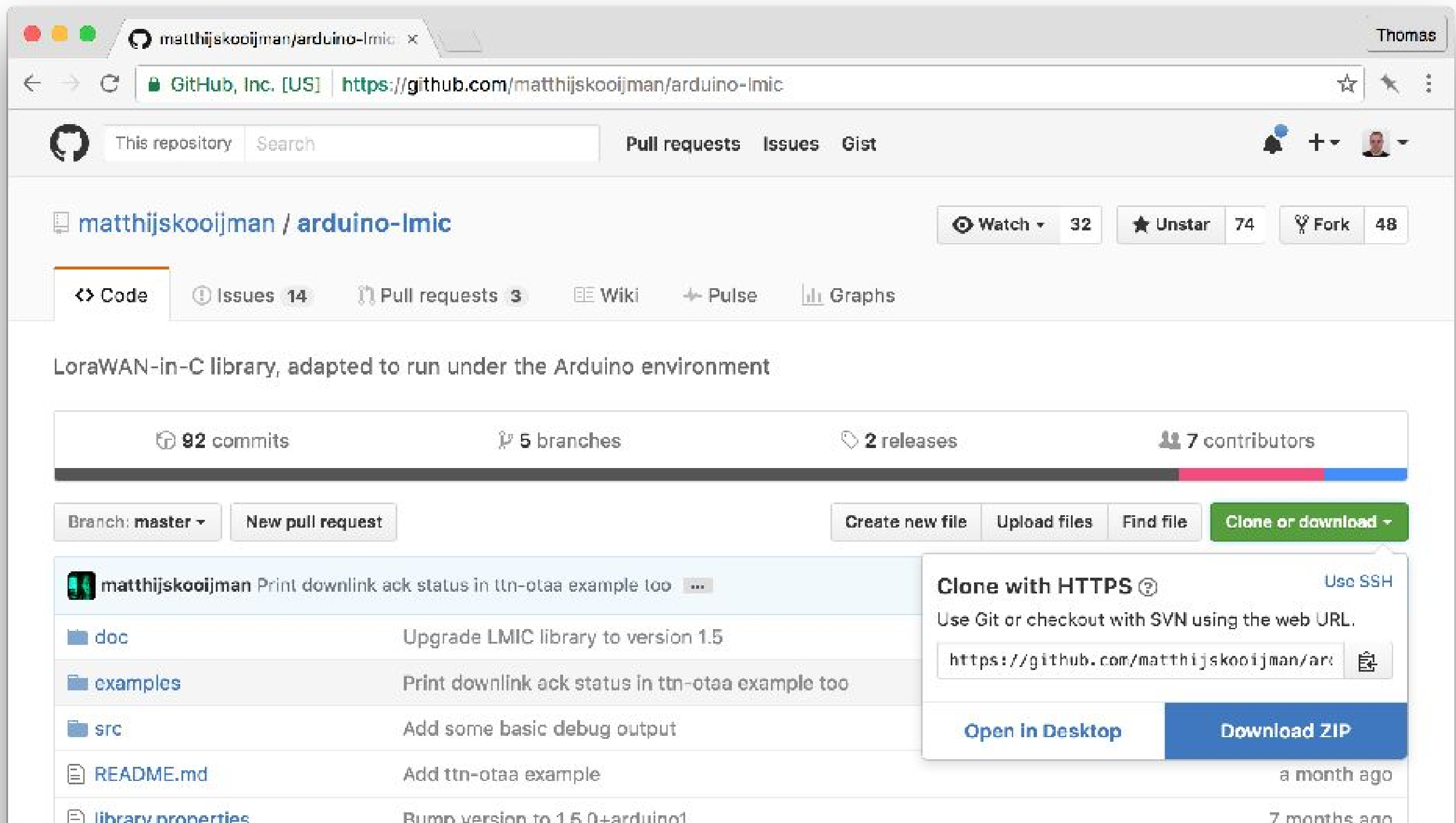
Note: the LoRa shield stacks onto the Arduino - just make sure the pins line up properly

Pin 2 and pins 6 - 13 are used by the Lora shield according to http://playground.arduino.cc/Main/ShieldPinUsage

Installing the lmic Arduino library

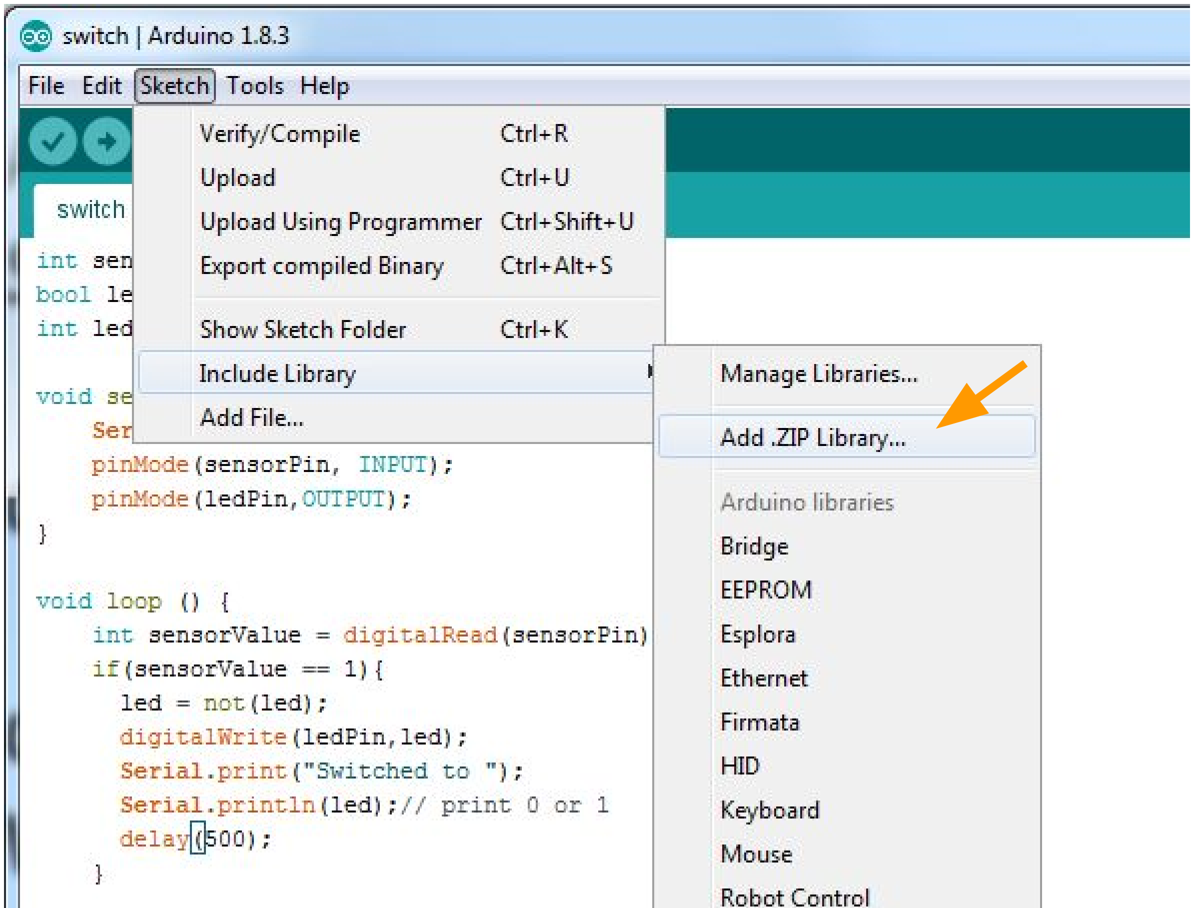
arduino-lmic on Github

Download ZIP from github:



Installing the lmic Arduino library

Add library to IDE:
Sketch > Include Library > Add ZIP Library



Installing the lmic Arduino library

OR

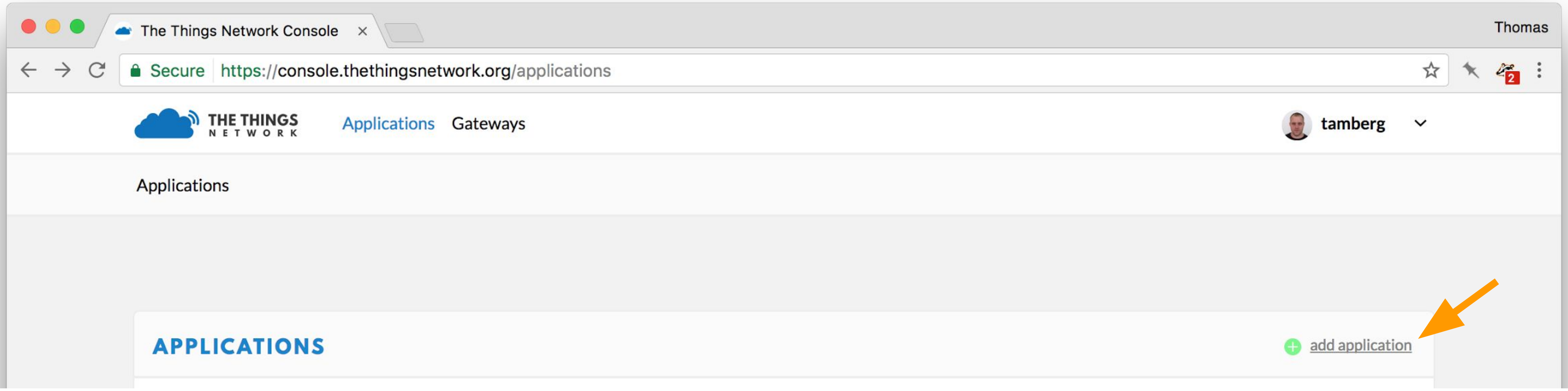
Mac:
extract ZIP to ~/Documents/Arduino/libraries/lmic

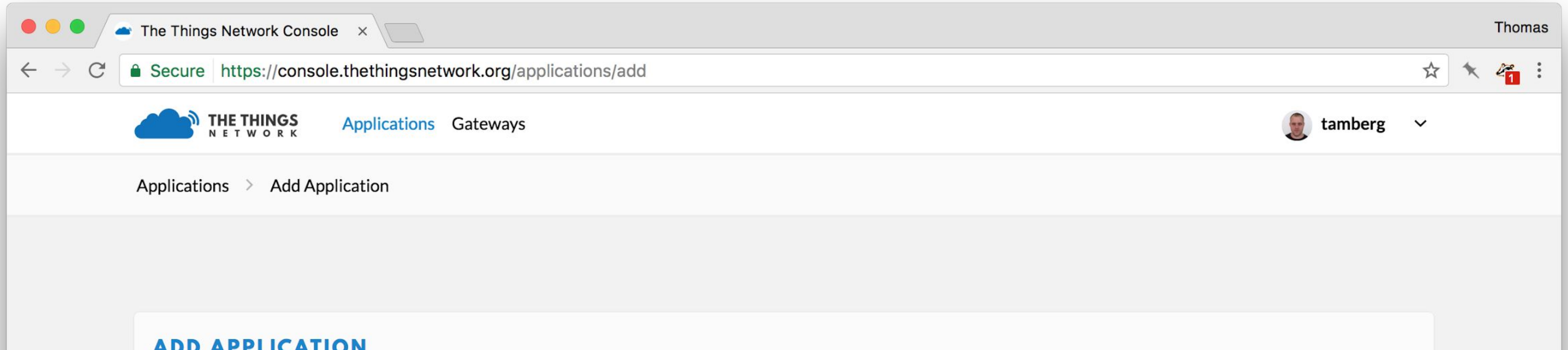
Windows:
extract ZIP to
C:\Users\USER_NAME\Documents\Arduino\Libraries\lmic

Linux:
cd ~/Arduino/Libraries/
git clone https://github.com/matthijskooijman/arduino-lmic.git

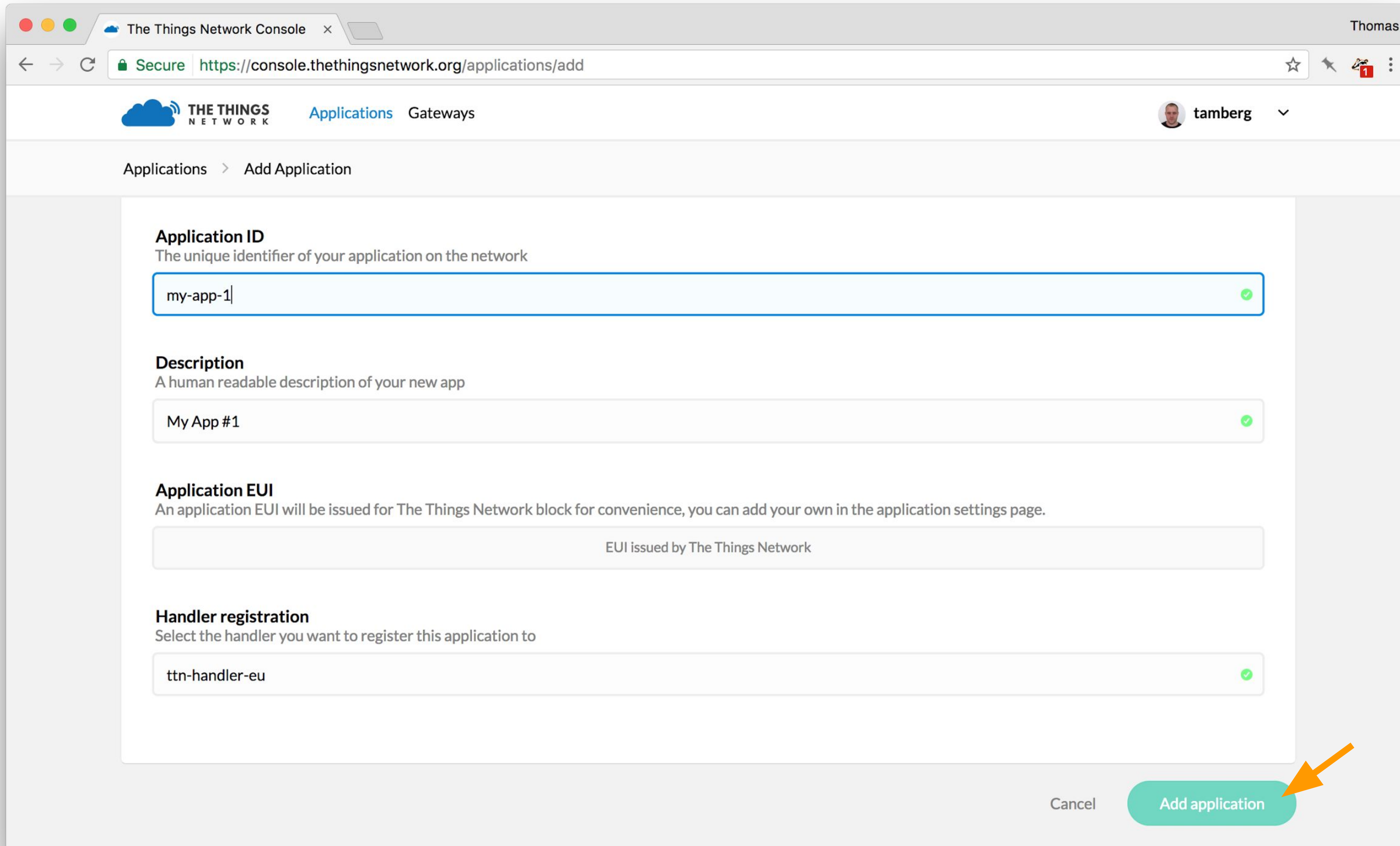
Adding a TTN application

https://console.thethingsnetwork.org/applications
(register to get an account, if needed)

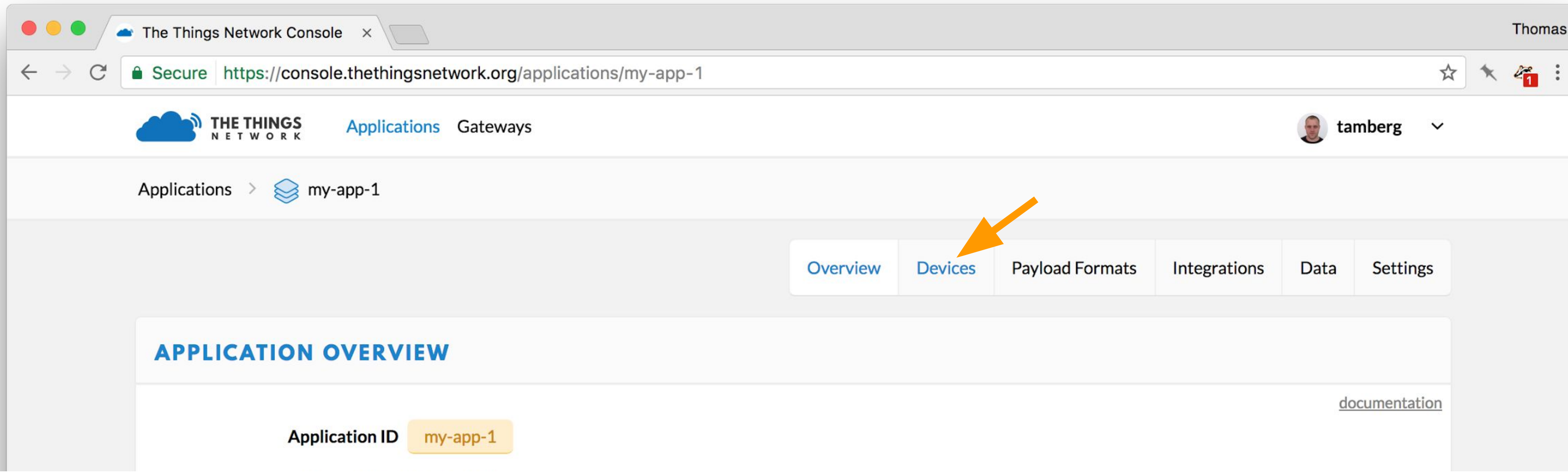


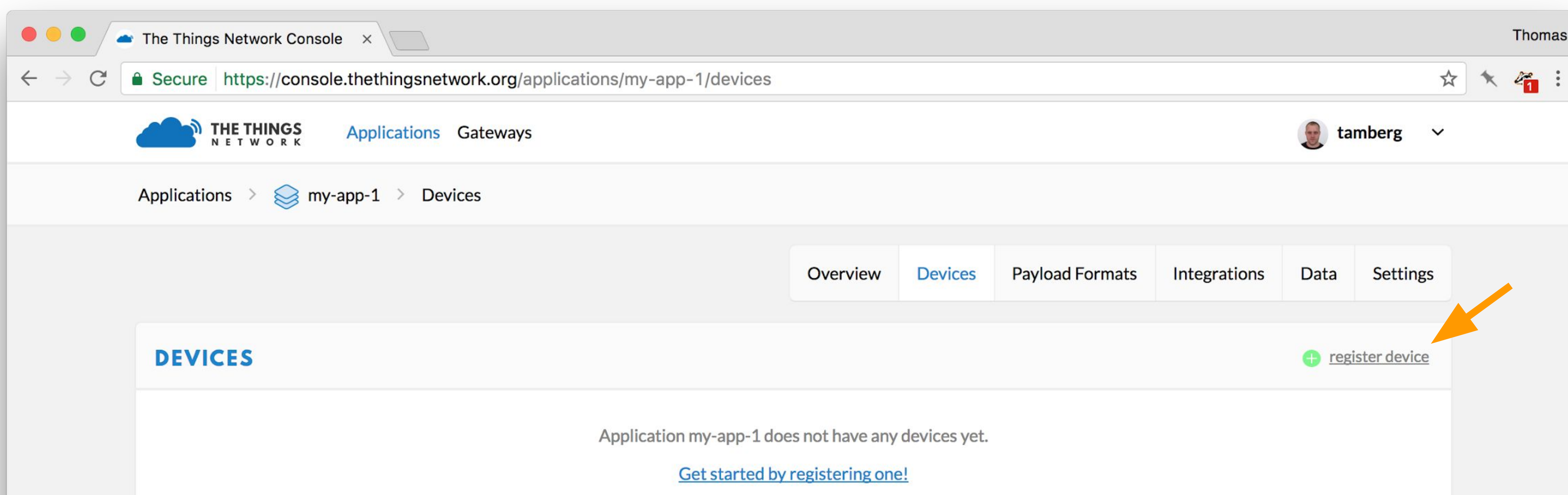


Adding a TTN application

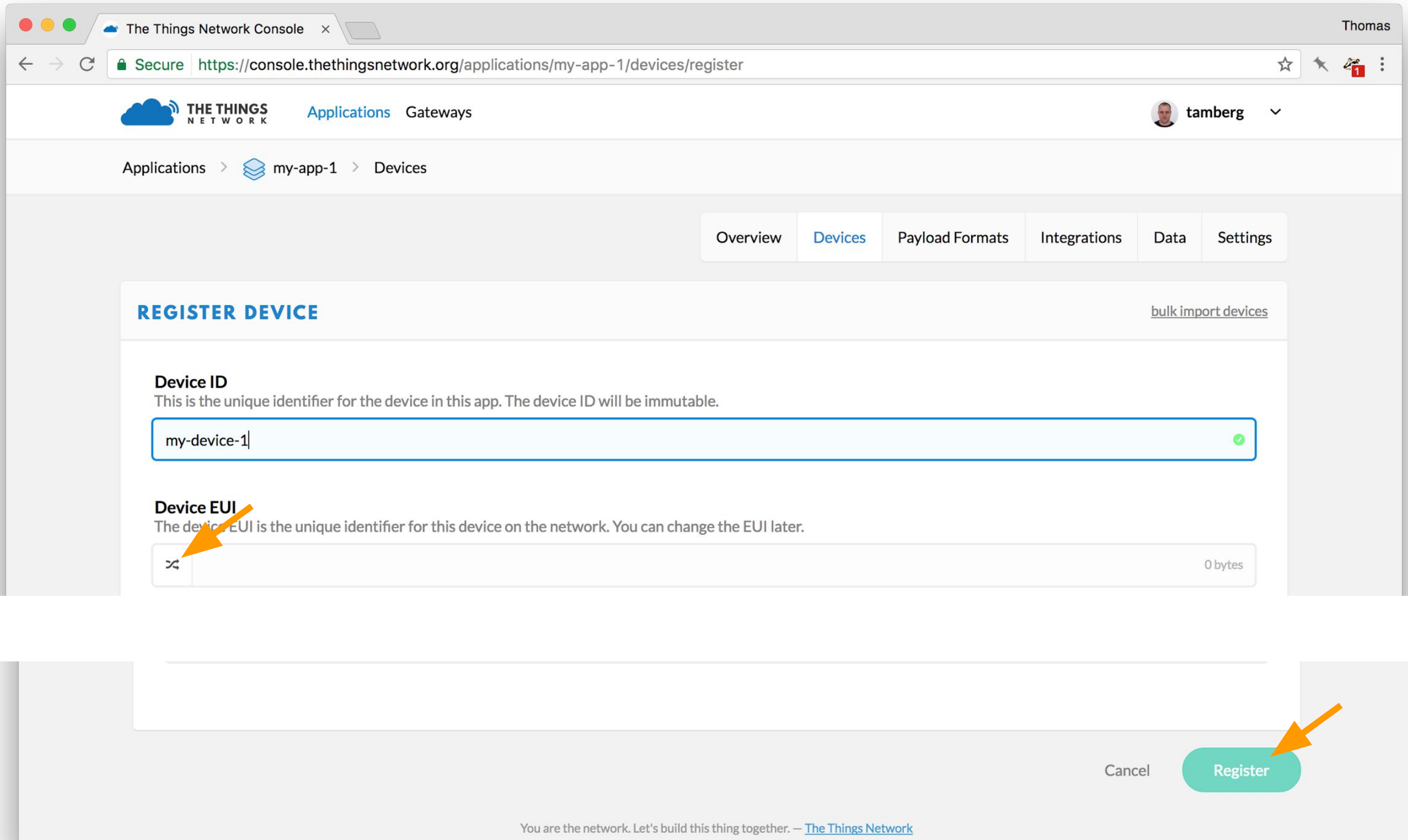


Registering a device

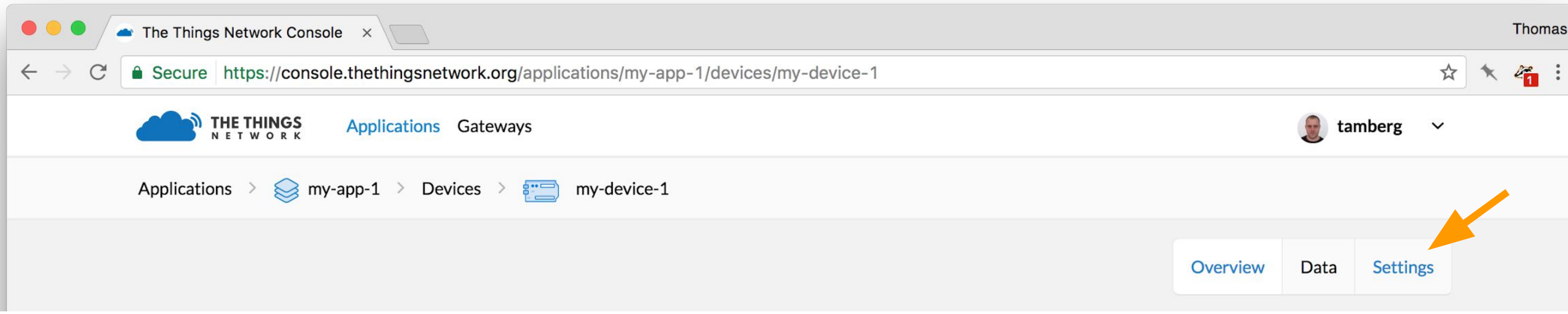


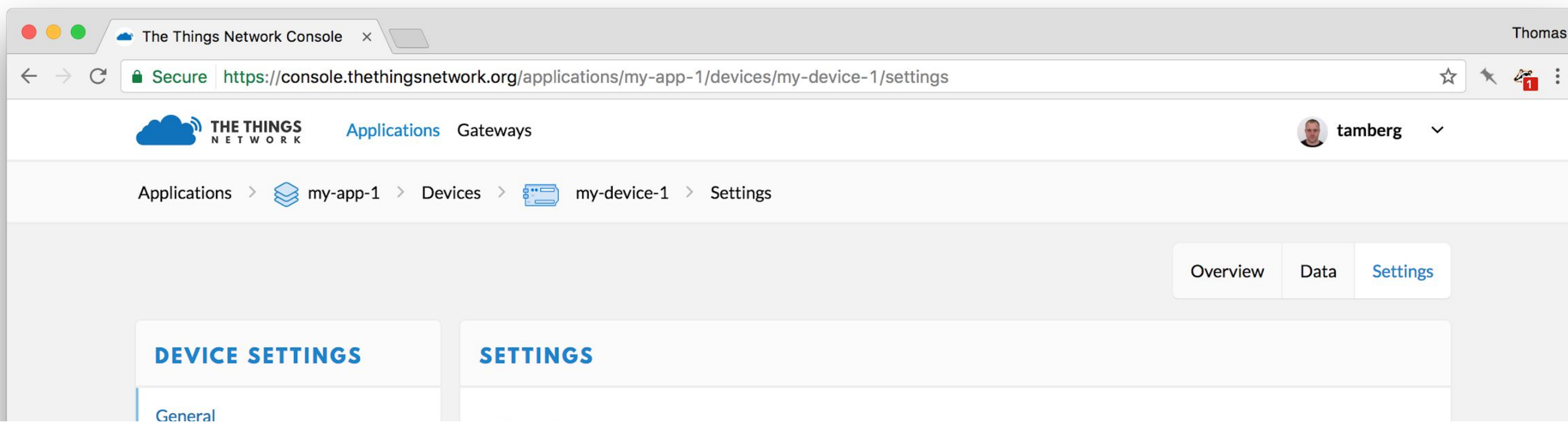


Registering a device

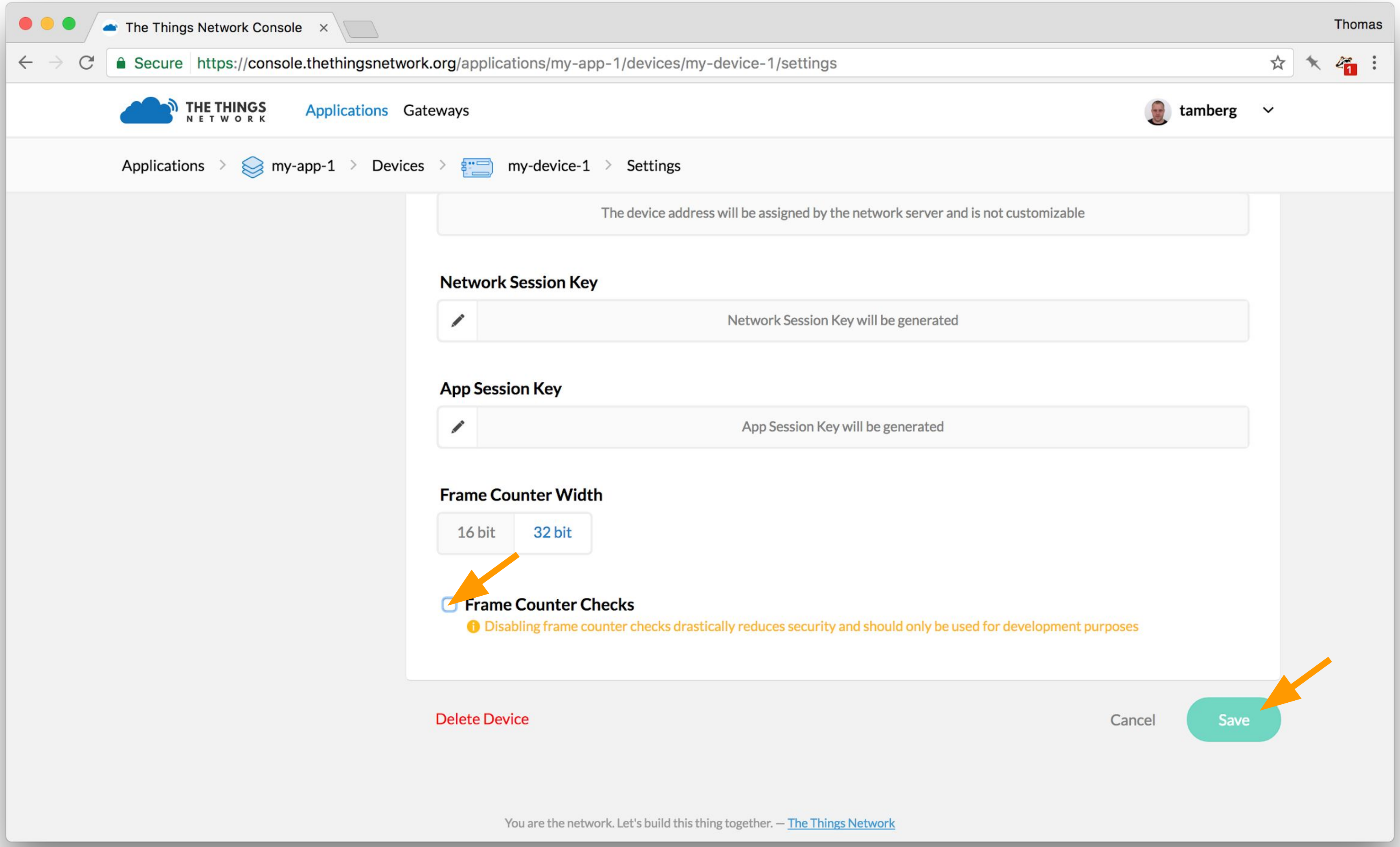


Registering a device with ABP

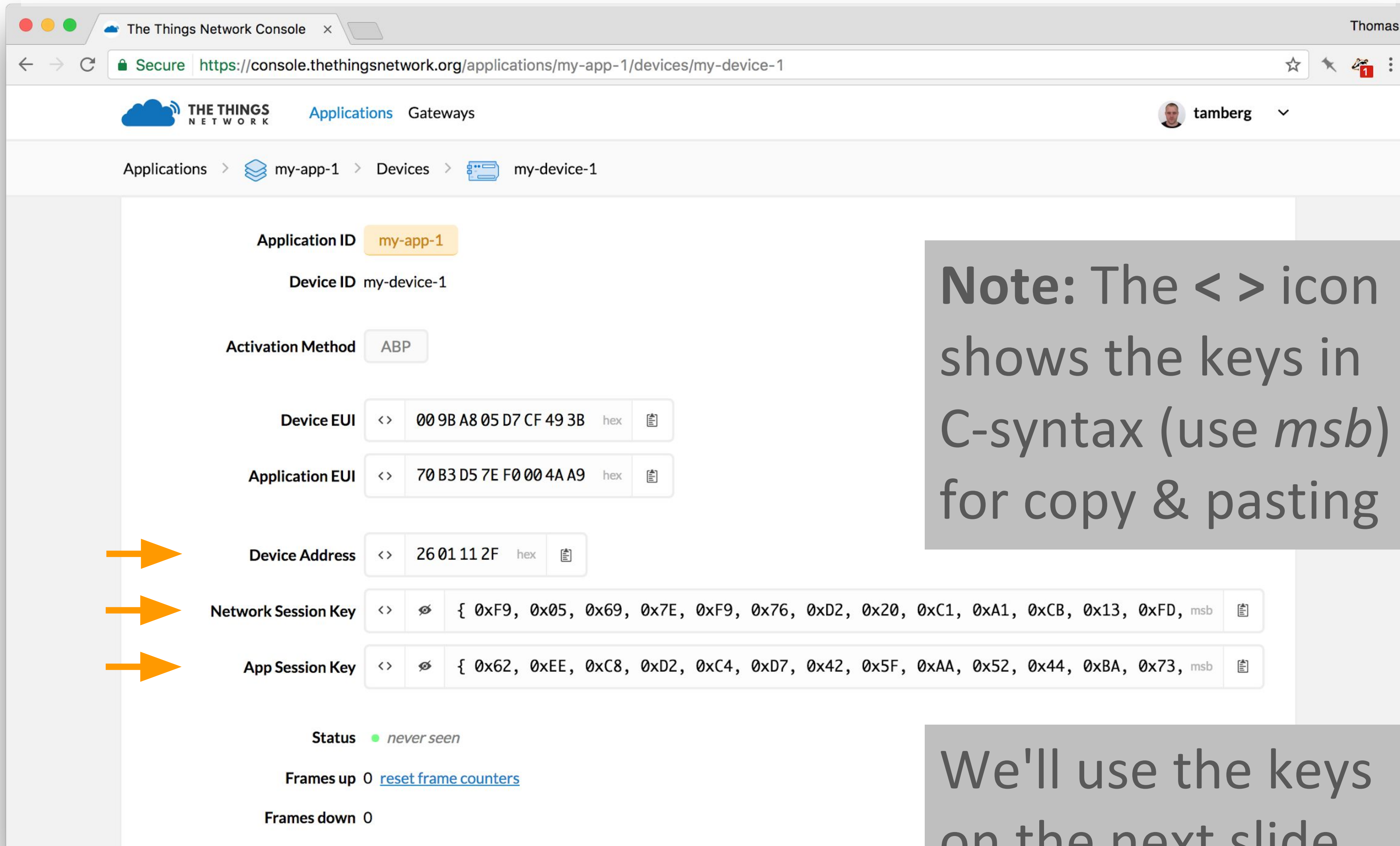




Registering a device with ABP



Getting your keys



Note: The < > icon shows the keys in C-syntax (use *msb*) for copy & pasting

We'll use the keys on the next slide...

Setting your keys in the code


```
#include <lmic.h> ...
static const u4_t DEVADDR = 0x01234567;
static const u1_t NWKSKEY[16] = {
  // e.g. for 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF
  0x00, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77,
  0x88, 0x99, 0xAA, 0xBB, 0xCC, 0xDD, 0xEE, 0xFF
};
static const u1_t APPSKEY[16] = {
  0x??, 0x??, 0x??, 0x??, 0x??, 0x??, 0x??, 0x??,
  0x??, 0x??, 0x??, 0x??, 0x??, 0x??, 0x??, 0x??
};
...
```

The **0x**.. is important

Same here

We will test this in the next chapter by sending text

4) Sending data

How to encode and transfer data packets to the TheThingsNetwork back-end server.

We'll be using the keys from the previous part for all examples.

Sending text with LoRaWAN

DraginoTtn
AbpTxText.ino

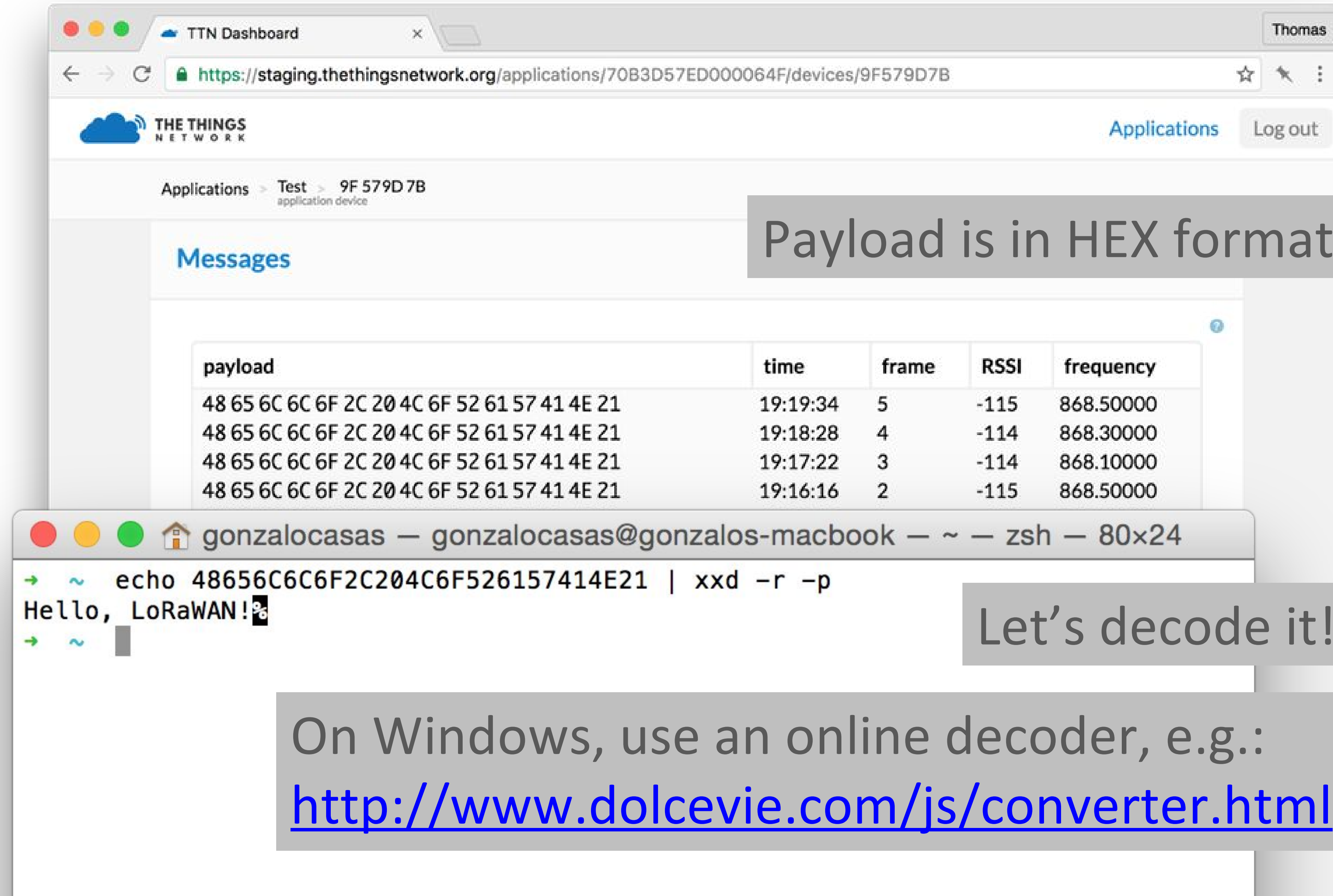
```
#include <lmic.h> ...
void do_send(osjob_t* j) {
  static uint8_t message[] = "Hello, LoRaWAN!"; // ASCII only
  if ((LMIC.opmode & OP_TXRXPEND) == 0) { // ok to send
    LMIC_setTxData2(1, message, sizeof(message) - 1, 0);
  }
}
void onEvent (ev_t ev) { ...
  case EV_TXCOMPLETE: os_setTimedCallback(..., do_send); ...
}
```

Details do not matter here.

Queue message to be sent after next TXCOMPLETE

Note: The **lmic** library is currently the most robust way to use the HopeRF chip, which is on the Dragino shield. A easier to use library might come. You're an early adopter.

Viewing data in the dashboard



Payload is in HEX format

Let's decode it!

On Windows, use an online decoder, e.g.:
http://www.dolcevie.com/js/converter.html

Sending numbers with LoRaWAN

DraginoTtnAbpTxInt.ino

```
#include <lmic.h> ...
void do_send(osjob_t* j) {
  int value = analogRead(A0);
  static uint8_t message[2];
  message[0] = highByte(value);
  message[1] = lowByte(value);
  if ((LMIC.opmode & OP_TXRXPEND) == 0) { // ok to send
    LMIC_setTxData2(1, message, sizeof(message), 0);
  }
}
void onEvent (ev_t ev) { ...
  case EV_TXCOMPLETE: os_setTimedCallback(..., do_send); ...
}
```

An integer is 2 bytes

Put the high and low byte into the message

Numeric payload in the dashboard

Applications > Demo Network (application) > 27 63 2F 50 (device)

Messages

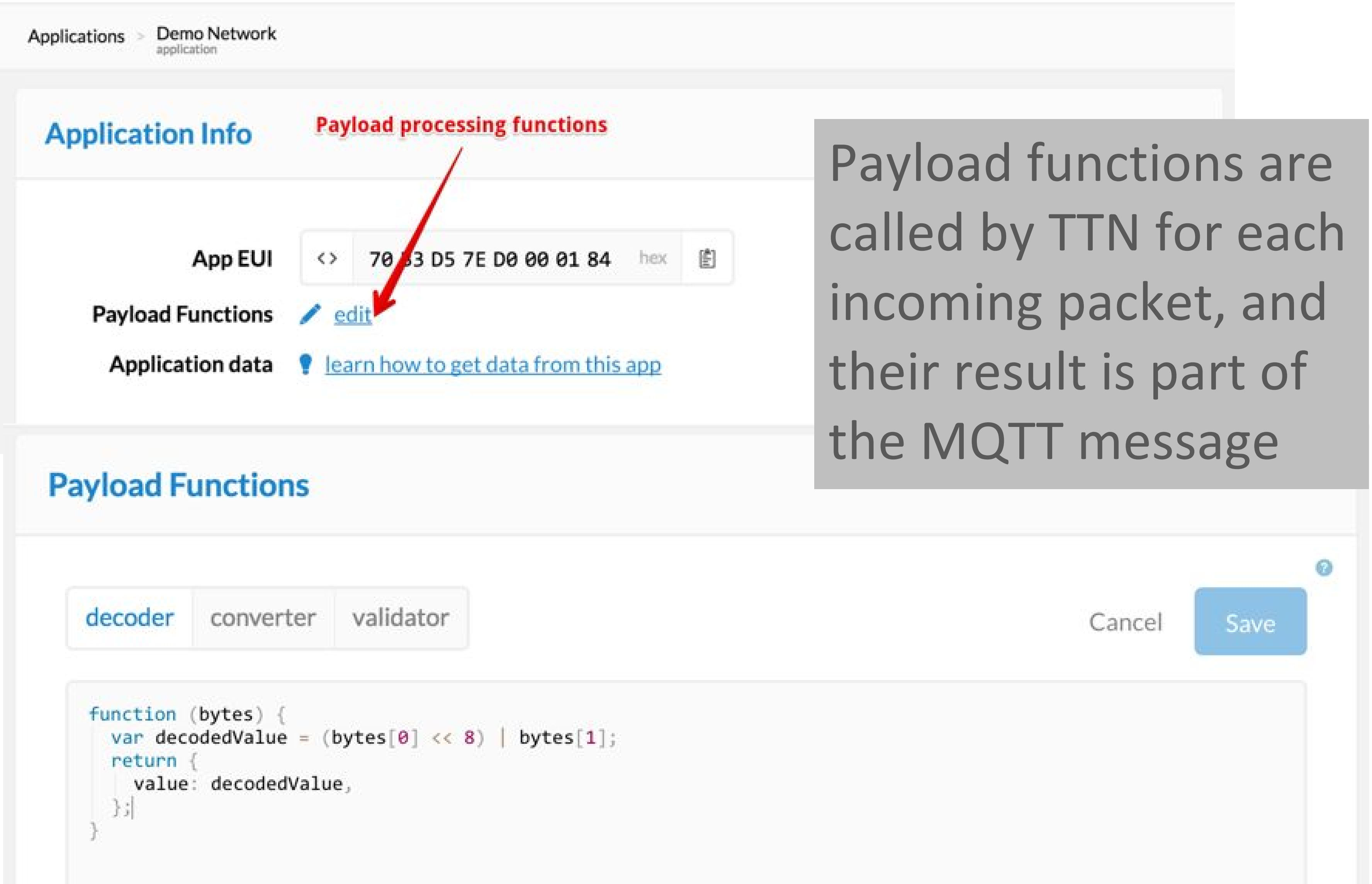
clear log

payload	time	frame	RSSI	frequency
01 C2	14:42:50	28	-29	868.50000
01 C5	14:42:18	26	-30	868.10000
01 80	14:42:03	25	-30	868.50000
01 87	14:41:47	24	-31	868.30000
01 9A	14:41:16	22	-30	868.50000
01 C0	14:41:00	21	-30	868.30000
01 8C	14:40:43	20	-28	868.10000
01 80	14:40:26	19		
01 80	14:40:11	18		
01 C2	14:39:55	17		
01 95	14:39:39	16		

But now it is hard to understand in HEX

Let's make it easier!

Payload functions to the rescue!



Payload functions are called by TTN for each incoming packet, and their result is part of the MQTT message

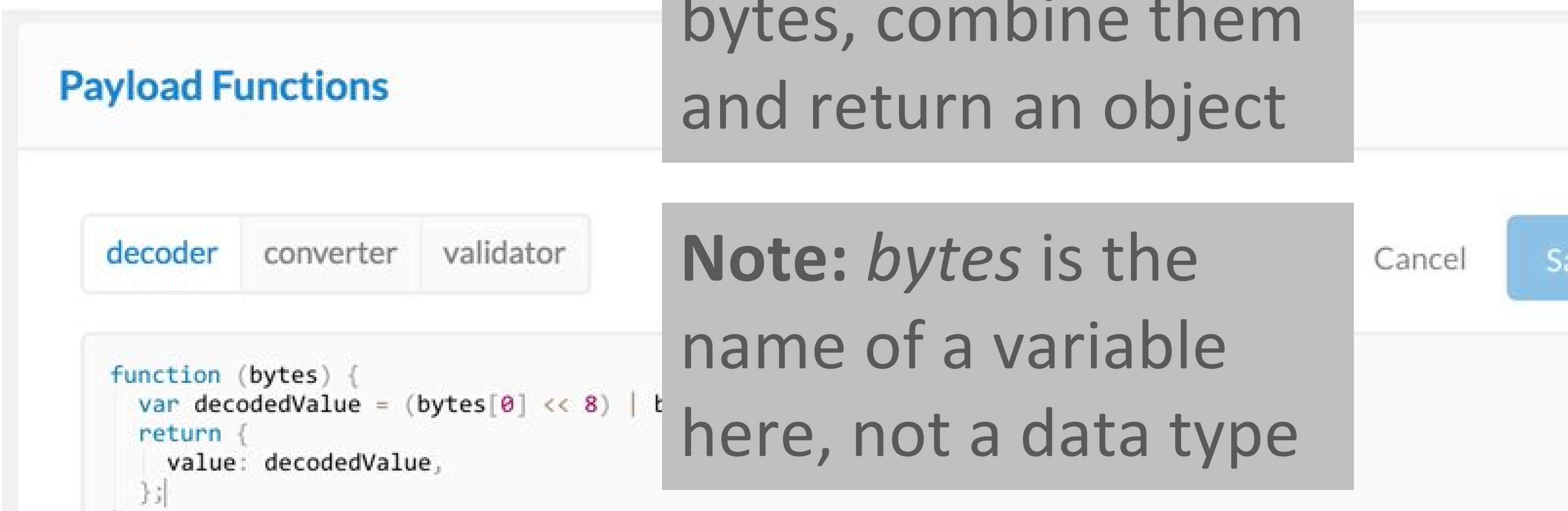
Writing a decoder function

```
function (bytes) {
  var decodedValue = (bytes[0] << 8) | bytes[1];
  return {
    value: decodedValue,
  };
}
```

Payload functions are written in Javascript

Take the first two bytes, combine them and return an object

Note: *bytes* is the name of a variable here, not a data type



Decoded payload in the dashboard

Applications > Demo Network (application) > 27 63 2F 50 (device)

Messages

clear log

payload	time	frame	RSSI	frequency
value 410	14:54:21	71	-33	868.10000
value 445	14:54:04	70	-27	868.50000
value 399	14:53:48	69	-30	868.30000
value 399	14:53:33	68	-33	868.10000
01 91	14:53:16	67	-33	868.50000
01 8F	14:53:00	66	-33	868.30000

Now that's much better!

Types of payload functions

Decoders transform bytes to objects, e.g. an analogRead PIN into a Celsius temperature.

Converters transform the output of decoders even further, e.g. from Celsius to Fahrenheit.

Validators make sure the data is correct, i.e. check the values are not outliers.

5) Forwarding sensor data

Messages sent via LoRaWAN to the network server of TheThingsNetwork become available through the MQTT messaging protocol.

We use a simple Node.js MQTT client to fetch our messages and forward them to ThingSpeak, a storage for sensor data or IFTTT, a rule based platform for mash-ups with 3rd-party services.

Installing Node.js on Mac or PC

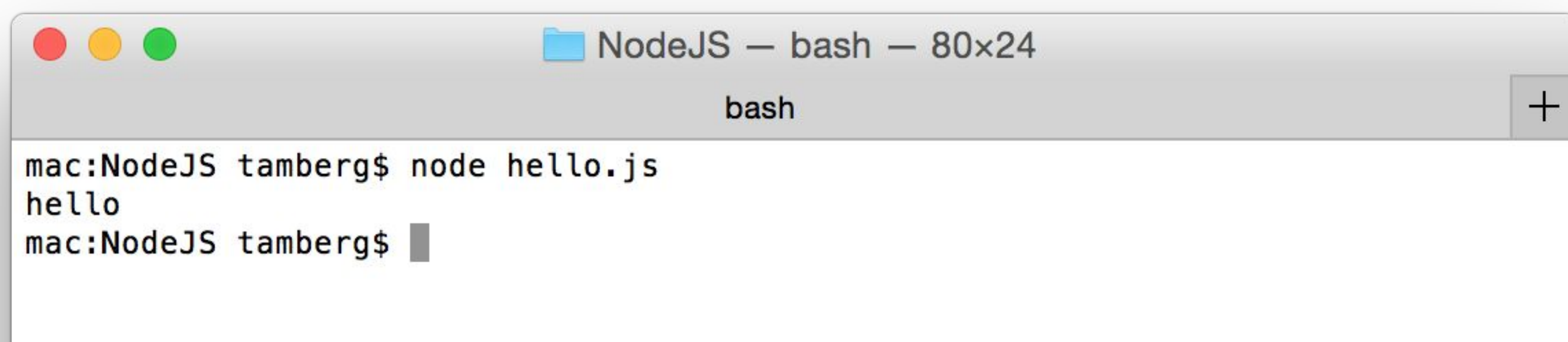
Install Node.js from https://nodejs.org/en/

Create a text file named **hello.js** containing

console.log("hello");

Open a **terminal** at the same location and type

$ *node hello.js*



Installing the ttn Node.js library

Create a new folder for your Node.js project

Open a **terminal** at the same location and type

```
$ npm update
$ npm install ttn --save
```

Getting your data with Node.js

ttn-mqtt-logger.js

In your Node.js project directory, download the *ttn-mqtt-logger* code (use the link or next page)

To install the *ttn* library, type

$ *npm install ttn --save*

To run the Node.js code, type

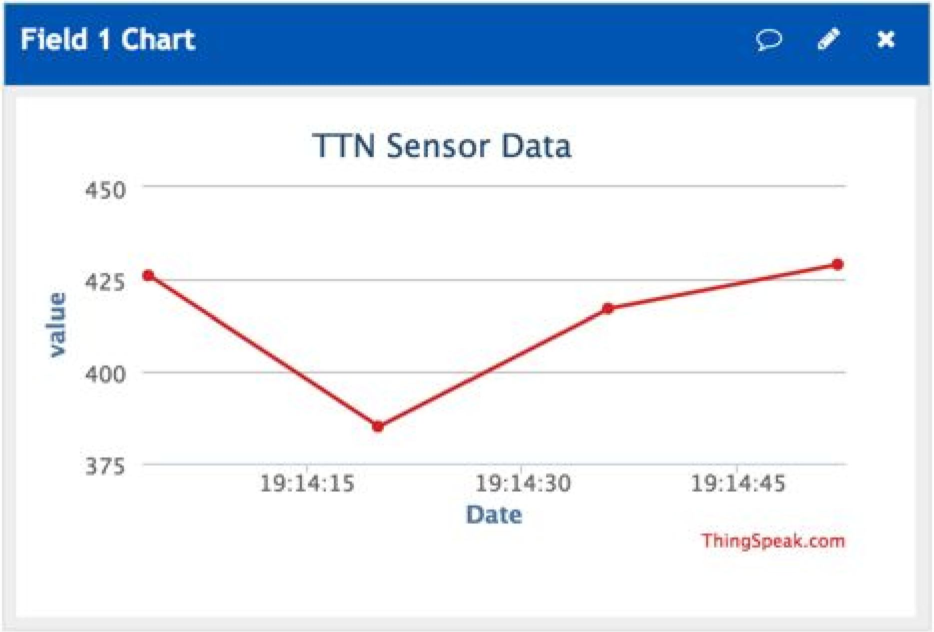
$ *node ttn-mqtt-logger.js*

Incoming LoRaWAN packets should now be logged

Visualising data with ThingSpeak

The ThingSpeak service lets you store, **monitor** and share **sensor data** in open formats. Sign up at https://thingspeak.com/ to create a channel and get API keys.

Let's add it to our Node.js client.



Forwarding data to ThingSpeak

ttn-thing speak-forwarder.js

```
var http = require('http');
client.on('uplink', function (msg) {
 http.get('http://api.thingspeak.com/update?' +
    'api_key=WRITE_API_KEY' +
    '&field1=' + msg.fields.value);
});
```

Use your key

Open a terminal at your Node.js project location, type

$ *node ttn-thingspeak-forwarder.js*

Send some numbers with *DraginoTtnAbpTxInt.ino* then check https://thingspeak.com/channels/CHANNEL_ID

Mash-up cloud services with IFTTT

If This Then That (IFTTT) is a **mash-up** platform

An IFTTT Recipe connects two Web services (or a service and a device) using their Web **API**s

The IFTTT **Maker Channel** uses **Webhooks** (outgoing HTTP requests) to call your device, and you can use **Web requests** to trigger IFTTT

Forwarding data to IFTTT

ttn-ifttt-forwarder.js

Download the *ttn-ifttt-forwarder.js*, and set the keys.

```
…
var appEUI = 'TTN_APP_EUI';
var accessKey = 'TTN_ACCESS_KEY=';
var makerChannelKey = 'IFTTT_MAKER_CHANNEL_KEY';
…
```

Comments => how to get key

Open a terminal at your Node.js project location, type

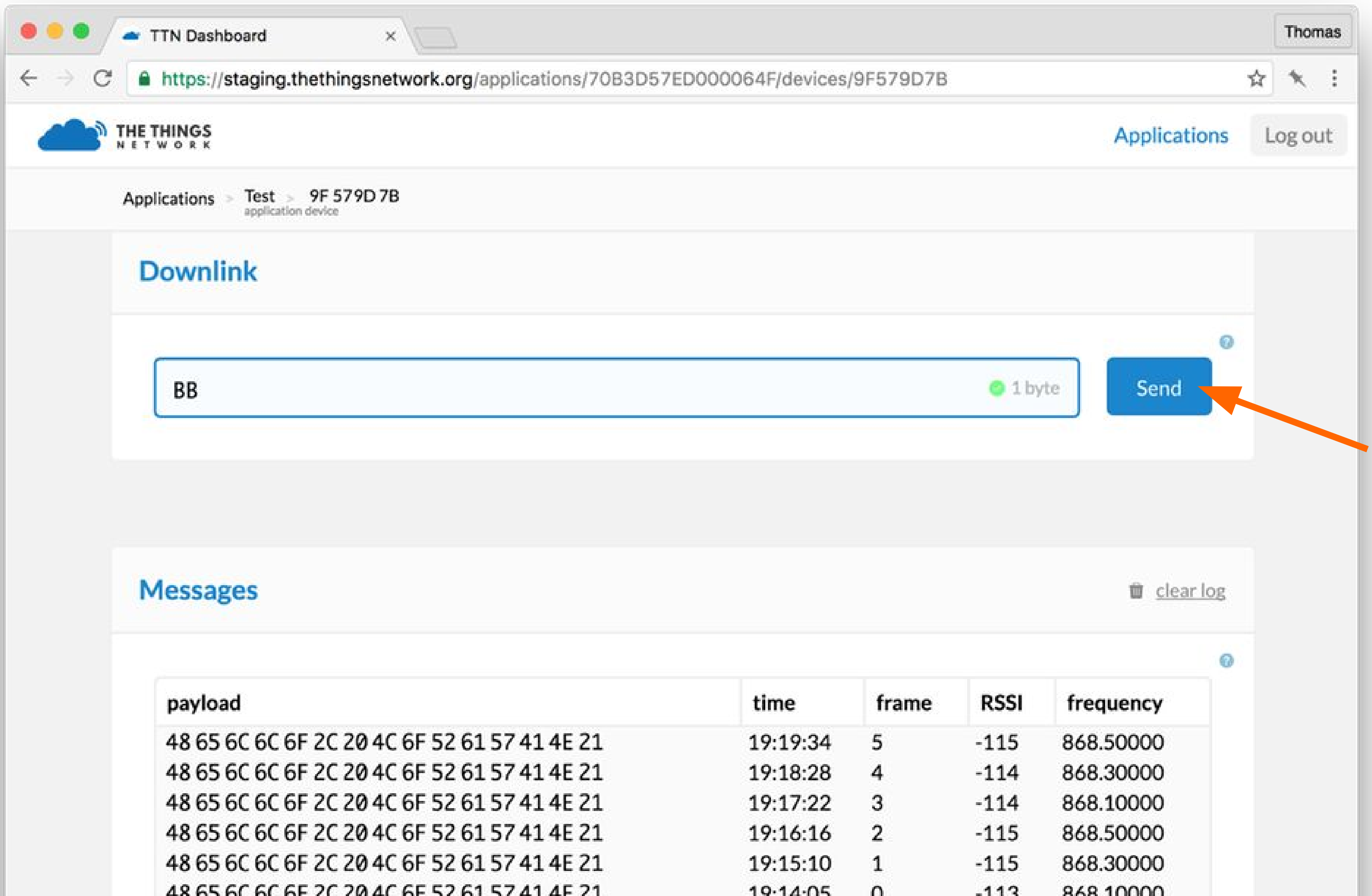
$ *node ttn-ifttt-forwarder.js*

Create a recipe on IFTTT using the Maker channel, e.g.
https://ifttt.com/recipes/335901-lorawan-log

6) Sending downlink messages

How to send data from the TheThingsNetwork backend to the LoRaWAN node

Sending data to your Arduino



Receiving data on the Arduino

DraginoTtn
AbpRxText.ino

```
#include <lmic.h> ...
void onEvent (ev_t ev) {
  if (ev == EV_TXCOMPLETE) {
    if (LMIC.dataLen) {
      uint8_t data[LMIC.dataLen];
      memcpy(&data, &(LMIC.frame + LMIC.dataBeg)[0],
        LMIC.dataLen);
      for (int i = 0; i < LMIC.dataLen; i++) {
        Serial.println(data[i]);
      }
    }
    // Schedule next transmission
    os_setTimedCallback(&sendjob, os_getTime()+sec2osticks(TX_INTERVAL), do_send);
}}
```

Controlling an LED remotely

DraginoTtn
ControllingArduino.ino

```
#include <lmic.h> ...
void setup () { … pinMode(5, OUTPUT); … }
void onEvent (ev_t ev) {
  if (ev == EV_TXCOMPLETE) {
    if (LMIC.dataLen) {
      uint8_t downlink[LMIC.dataLen];
      memcpy(&downlink,
        &(LMIC.frame+LMIC.dataBeg)[0],LMIC.dataLen);
      digitalWrite(5, downlink[0] == 42);
    }
    // Schedule next transmission
    os_setTimedCallback(&sendjob, os_getTime()+sec2osticks(TX_INTERVAL), do_send);
  }
}
```

Add a LED on D5

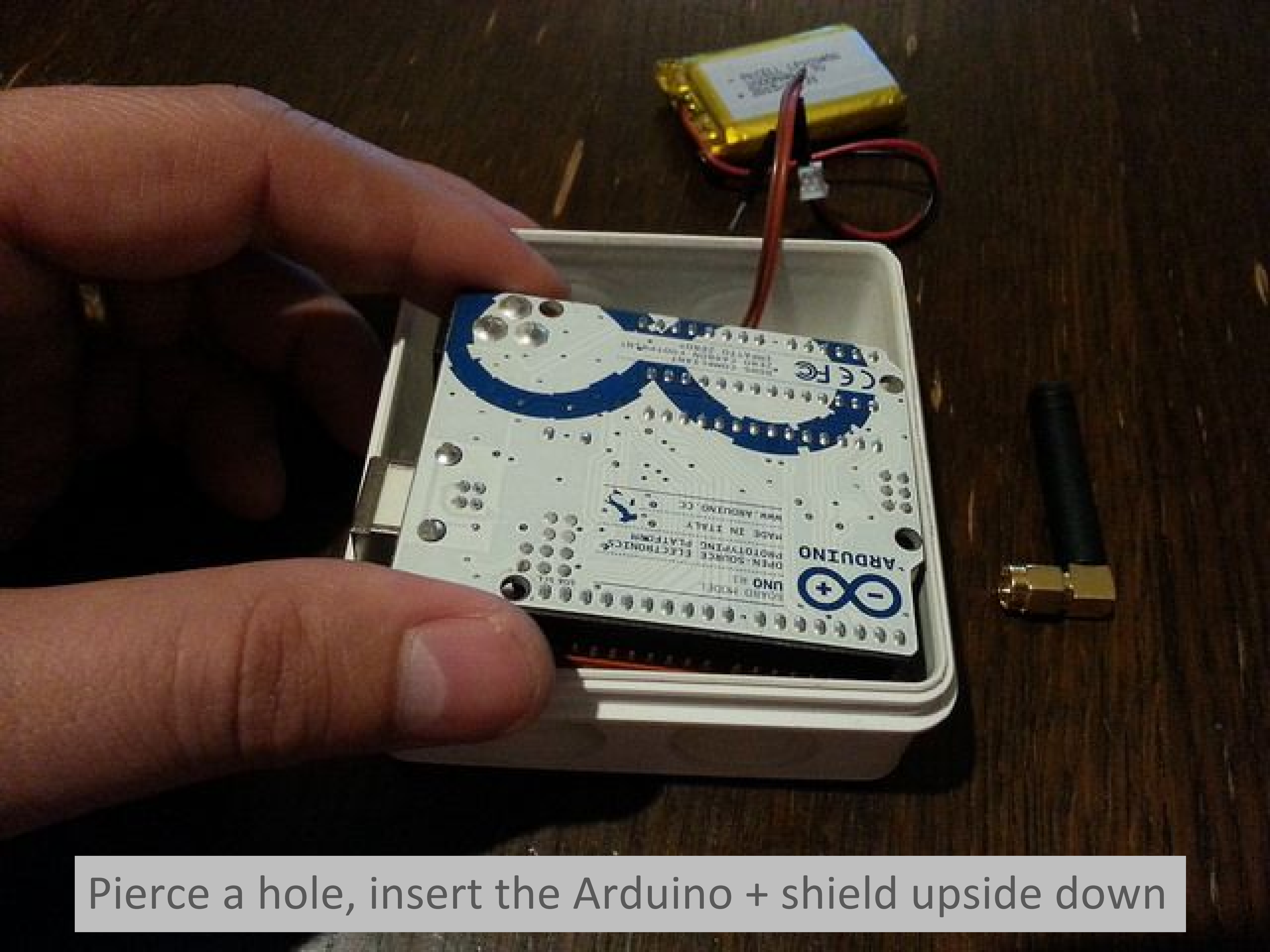
Parse downlink

Use its value to turn LED on/off

7) Deploying a node

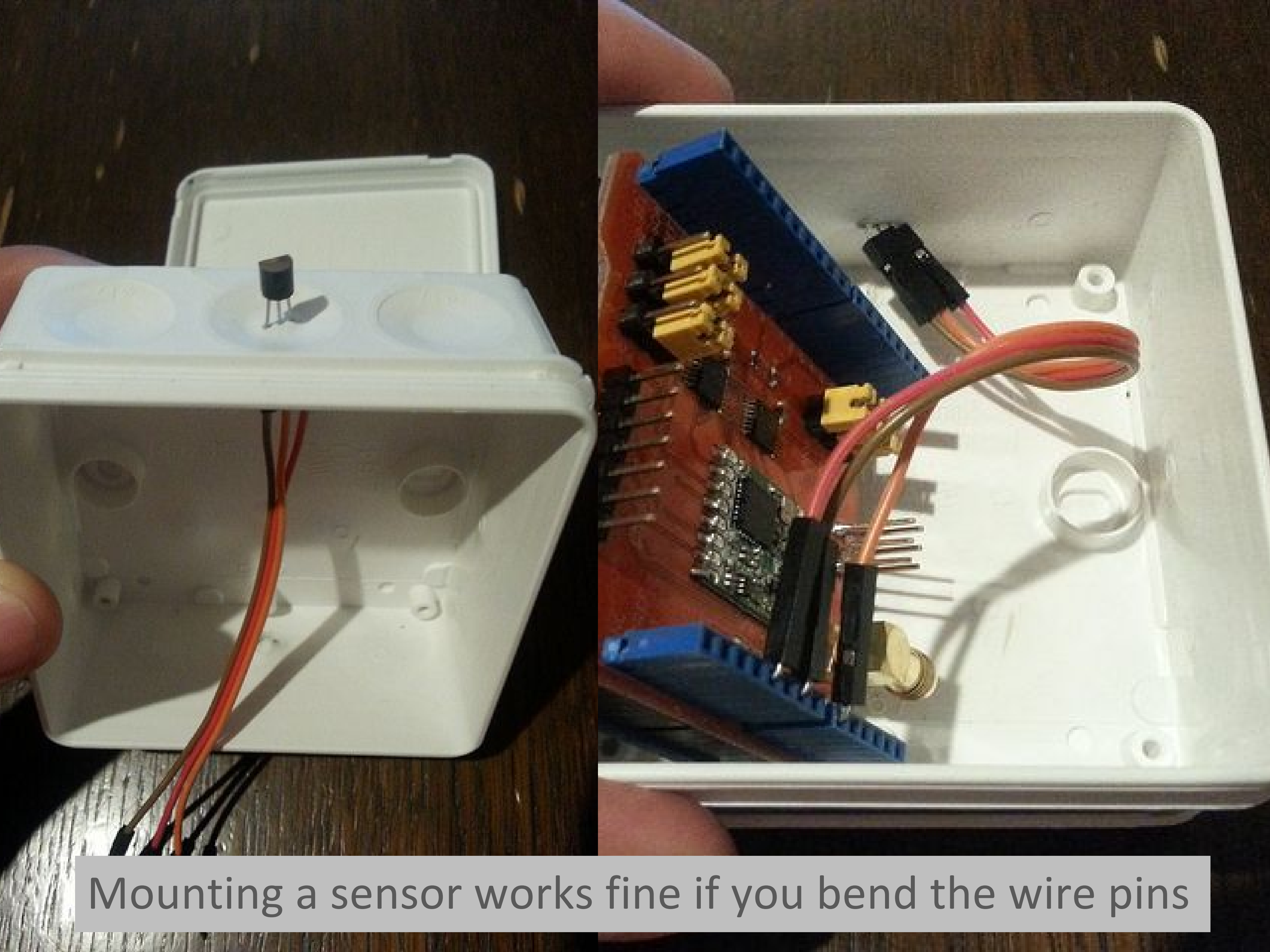
How to add a watertight case around the LoRa node and power it either by battery or a used USB cable cut and prepared for our needs.

Add GND and VIN wires and bend the pins as shown



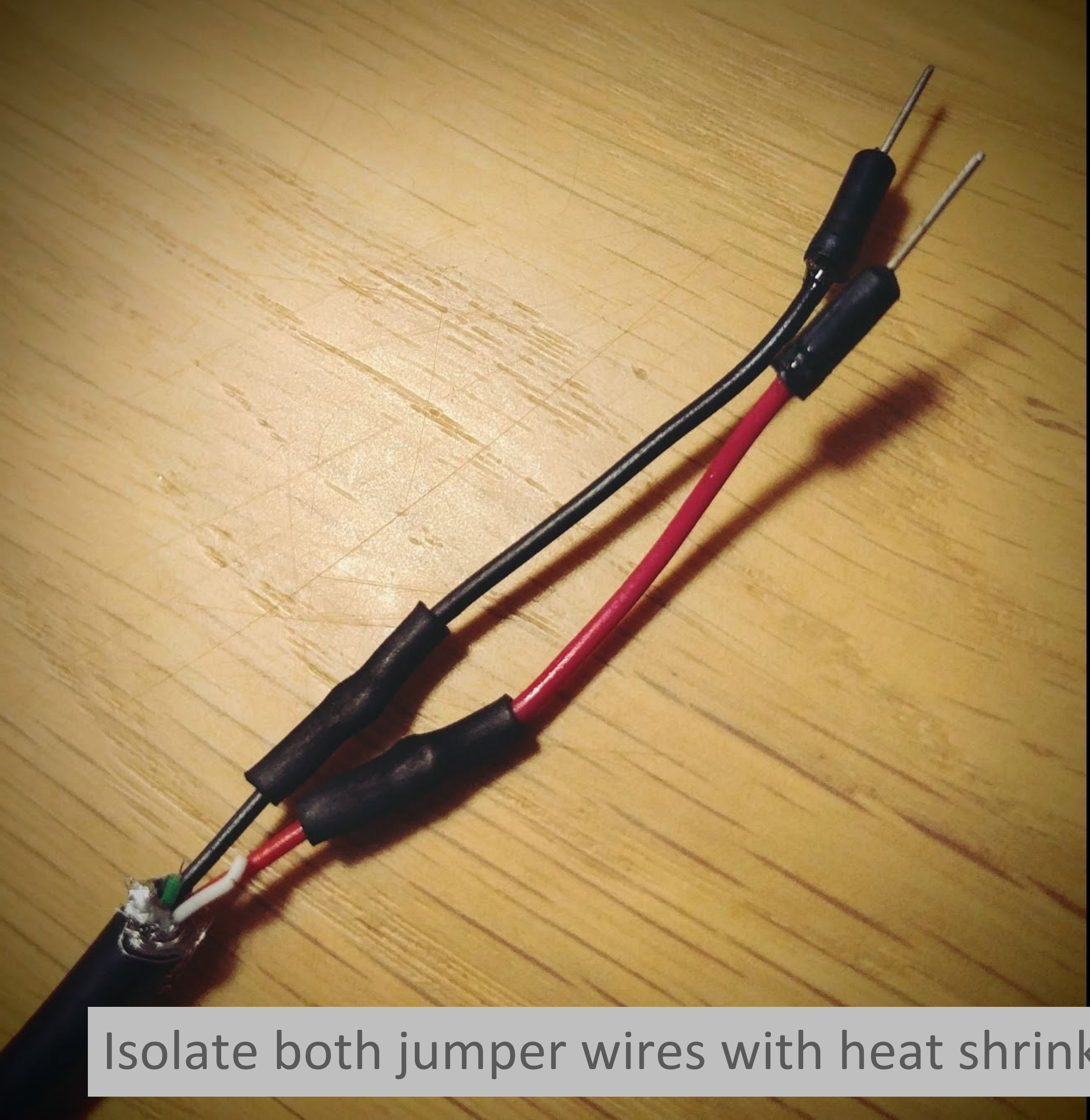
Pierce a hole, insert the Arduino + shield upside down

Connect antenna, then connect (battery or USB) power



Mounting a sensor works fine if you bend the wire pins

Cut an old USB cable, solder jumper wire to red, black



Isolate both jumper wires with heat shrink tubing

Thread the USB cable through, fix it with a zip tie

8) Deploying a TTN gateway

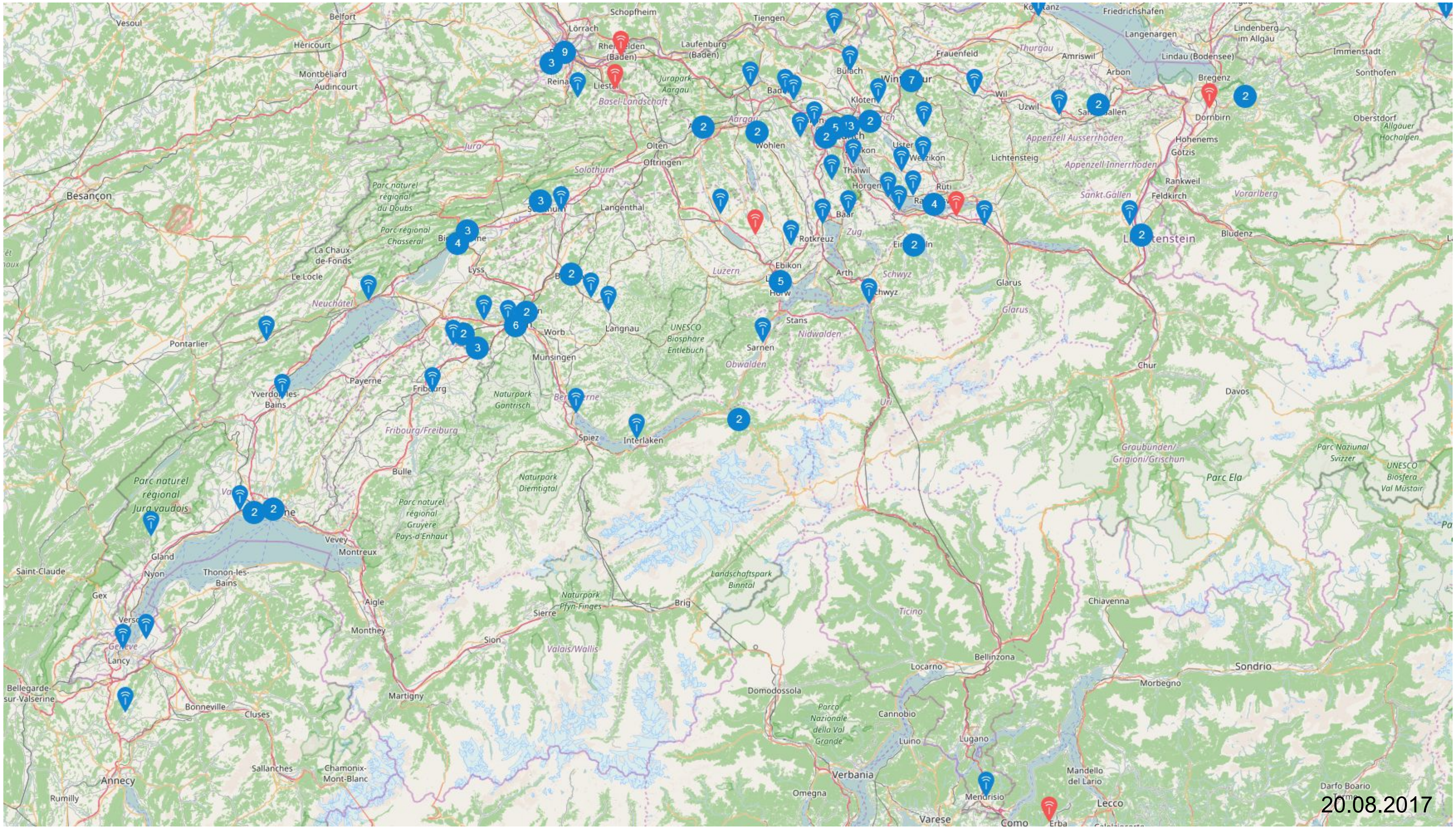
Do you want to extend the TTN coverage in your area? Build a gateway, it's easy and fun. And also pretty cheap (~300 CHF) compared to commercial alternatives (between 500 CHF and 1600 CHF).

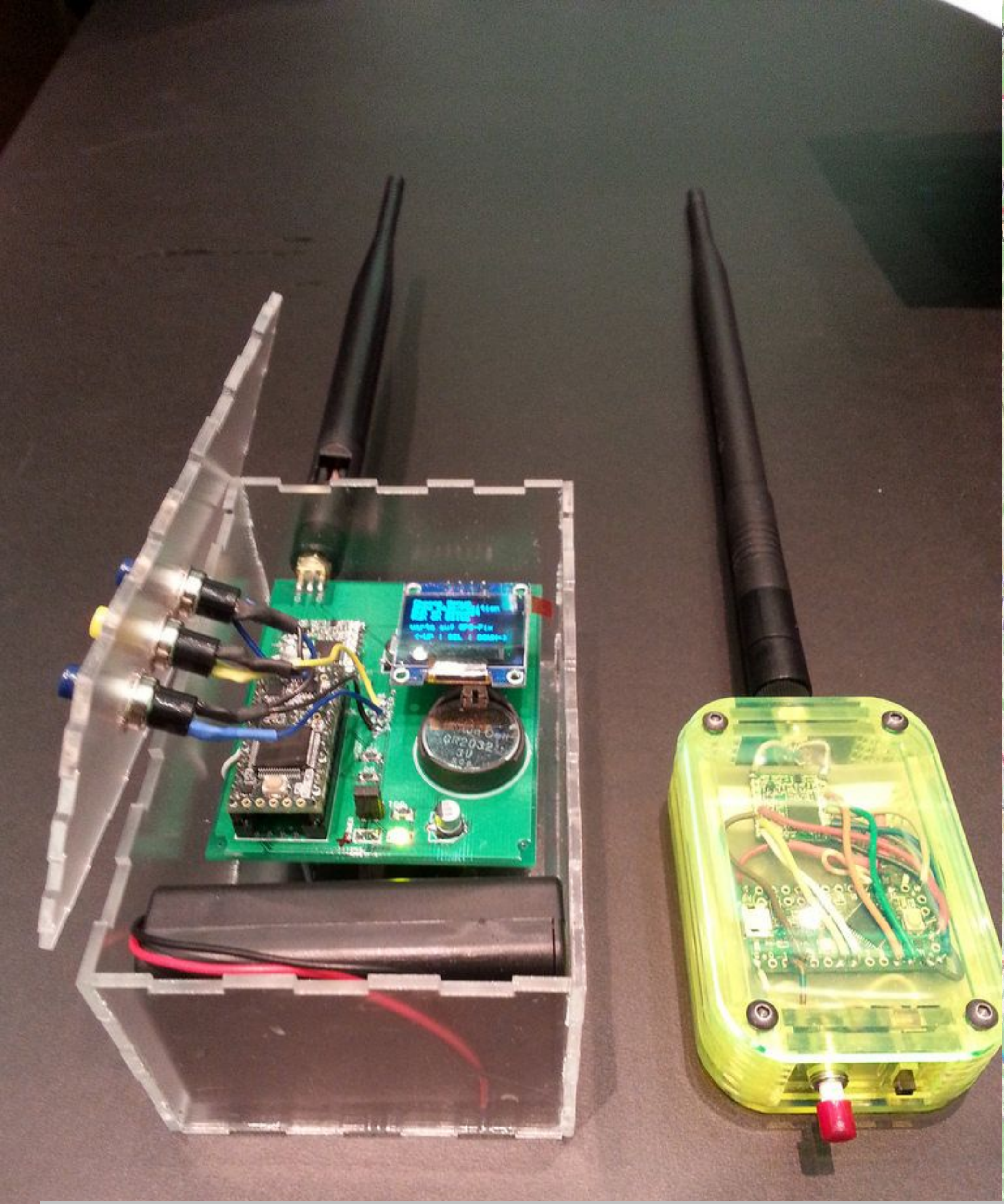
Once the gateway is up, you can measure the network coverage with simple means. Add a GPS to your node, or use your phone's GPS to record the time and place a packet was sent.

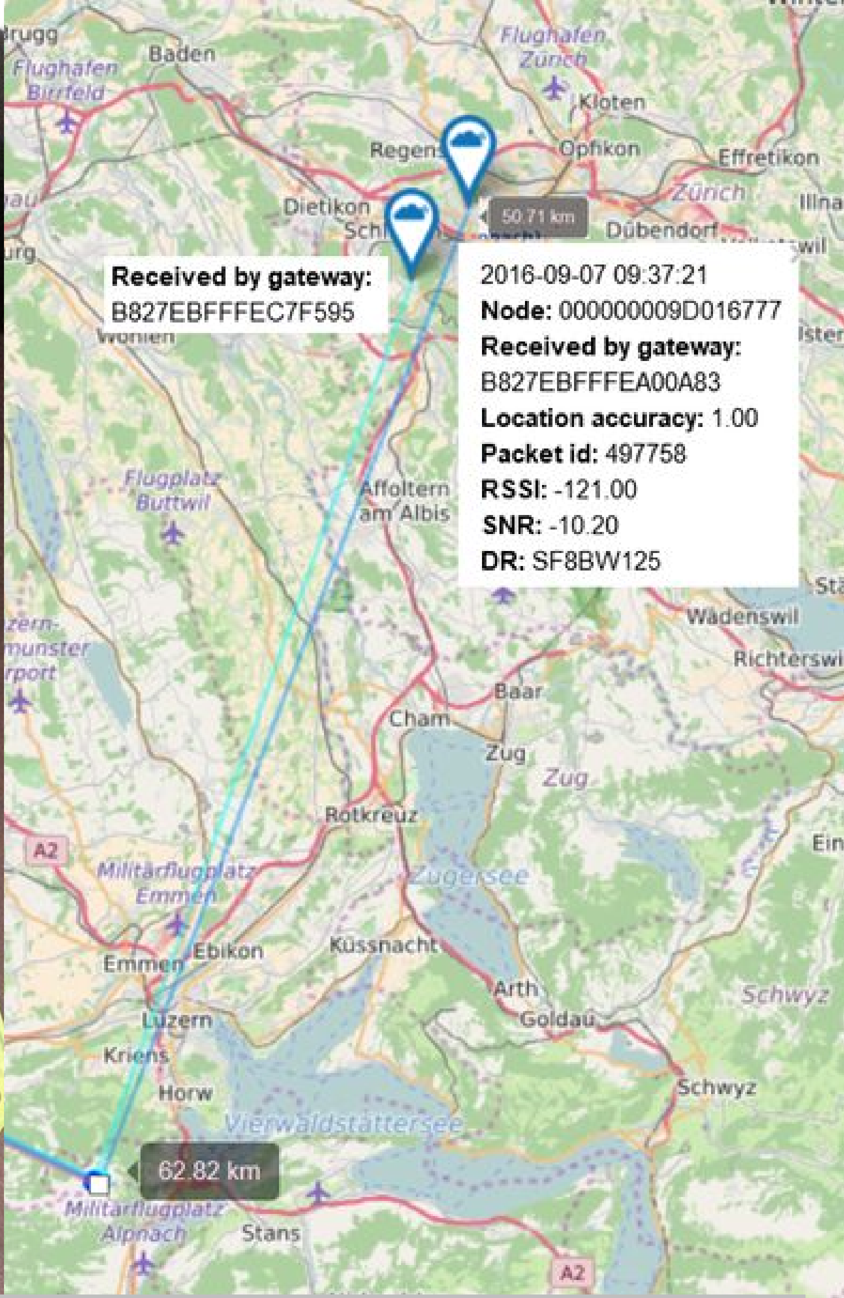
https://github.com/ttn-zh/ic880a-gateway/wiki

The Things Network CH

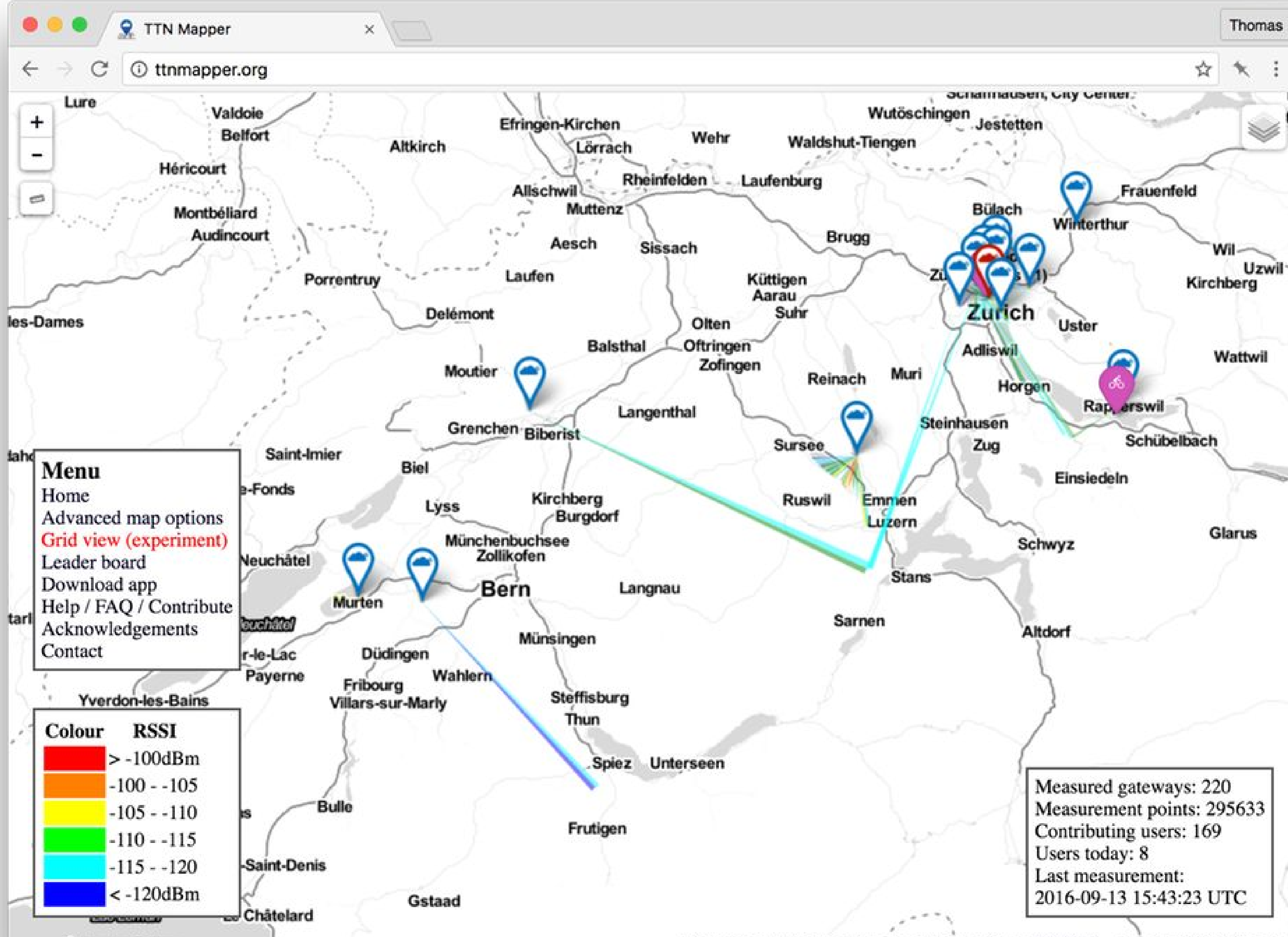
Gateways:







https://github.com/urs8000/LoRa-Gateway-Locator



Try http://ttnmapper.org/ and the ttnmapper app

Appendix A - Limitations

Why not just send ASCII?

Duty Cycle limitations:
Only 1% air time allowed on these frequencies.

TTN's Fair Usage policy:
30 seconds of airtime/day for uplink
10 downlink/day

smaller payload = **more** messages/day

How much data can be sent?

LoRaWAN protocol **adds 13 bytes** (at least)

Spreading Factor (SF) affects airtime required
SF7 = Most efficient, SF12 = Least efficient

Pack payload in binary as efficiently as possible

Use **built-in features** to **reduce data** (ports for data types, built-in counter, etc).

How much data can be sent?

	Payload sample	Payload/Total size	Msg/day
Simple payload	{ "Count": 1234, "Temperature": 20.635 }	40/53	292
Remove counter	{"t":20.63}	11/24	486
No JSON	20.63	5/18	582
16-bits integer	0x080F	2/15	**648**

Here's an air time calculator spreadsheet

Why is downlink so limited?

Gateways are **half-duplex**, if they are sending, they are not listening.

Duty cycle limitations of 1% apply **per device**.

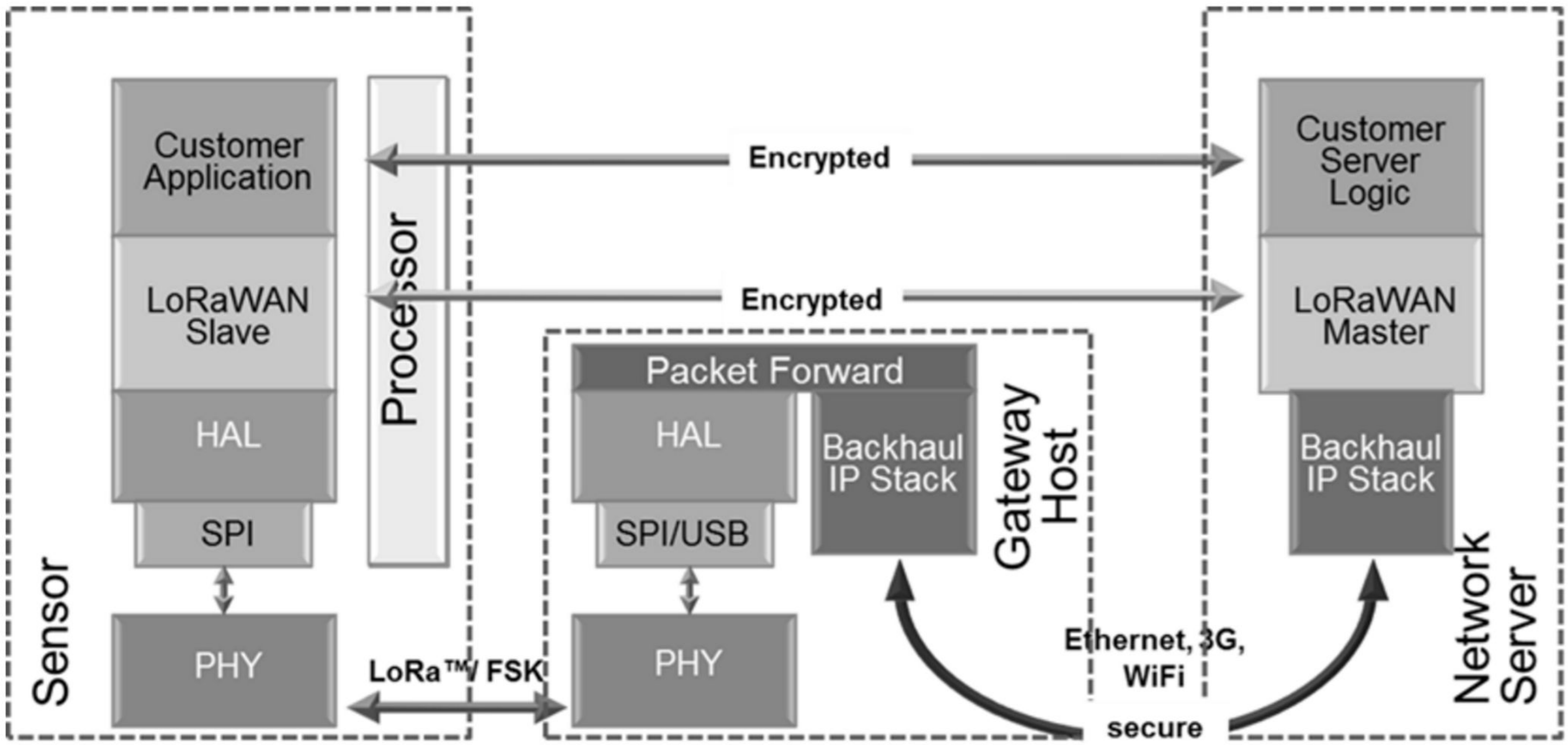
A **gateway** is also subject to the same limitation, but it has to **share its airtime** between all nodes around it.

Which use-cases could still work given this restriction?

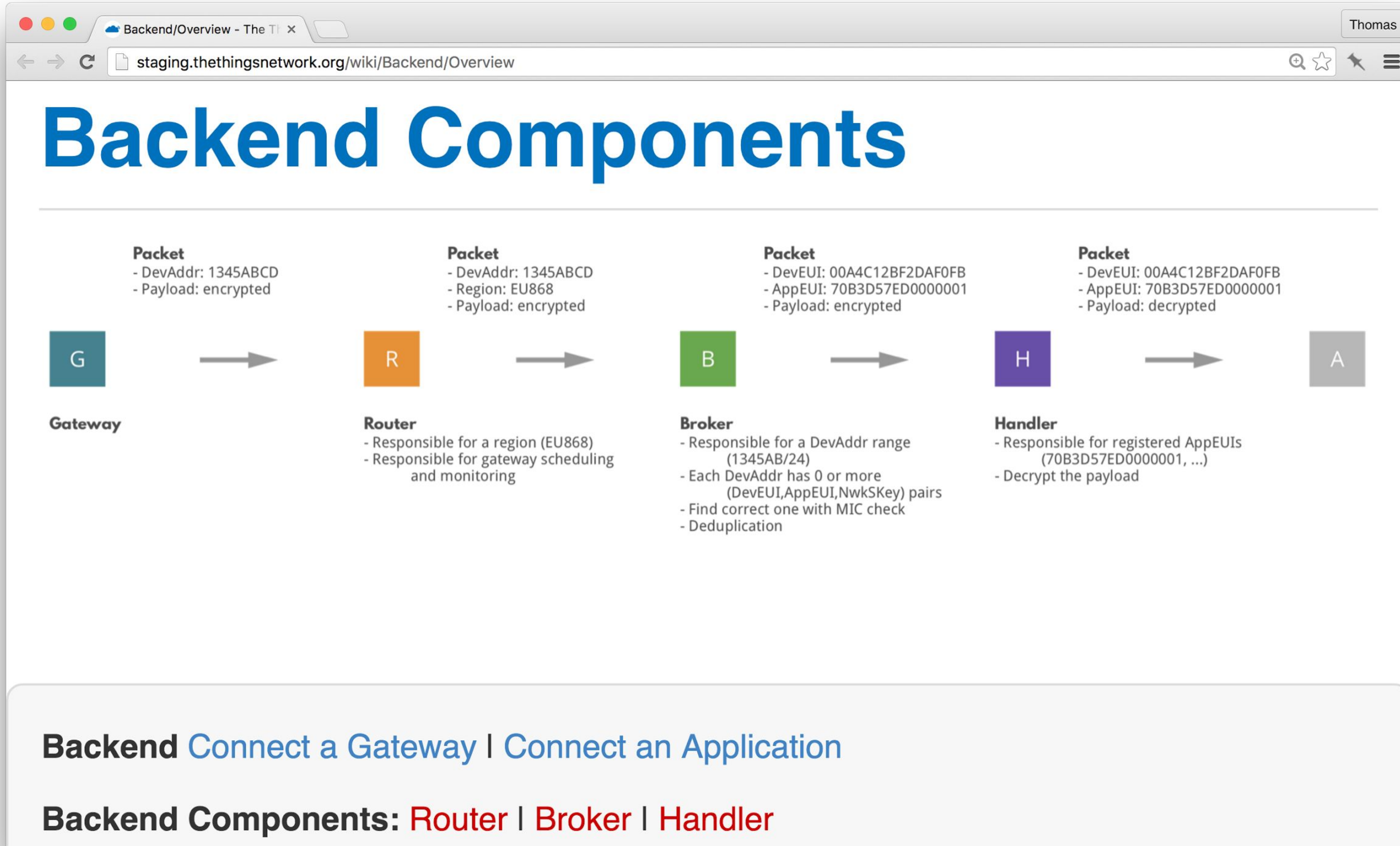
Appendix B - Architecture

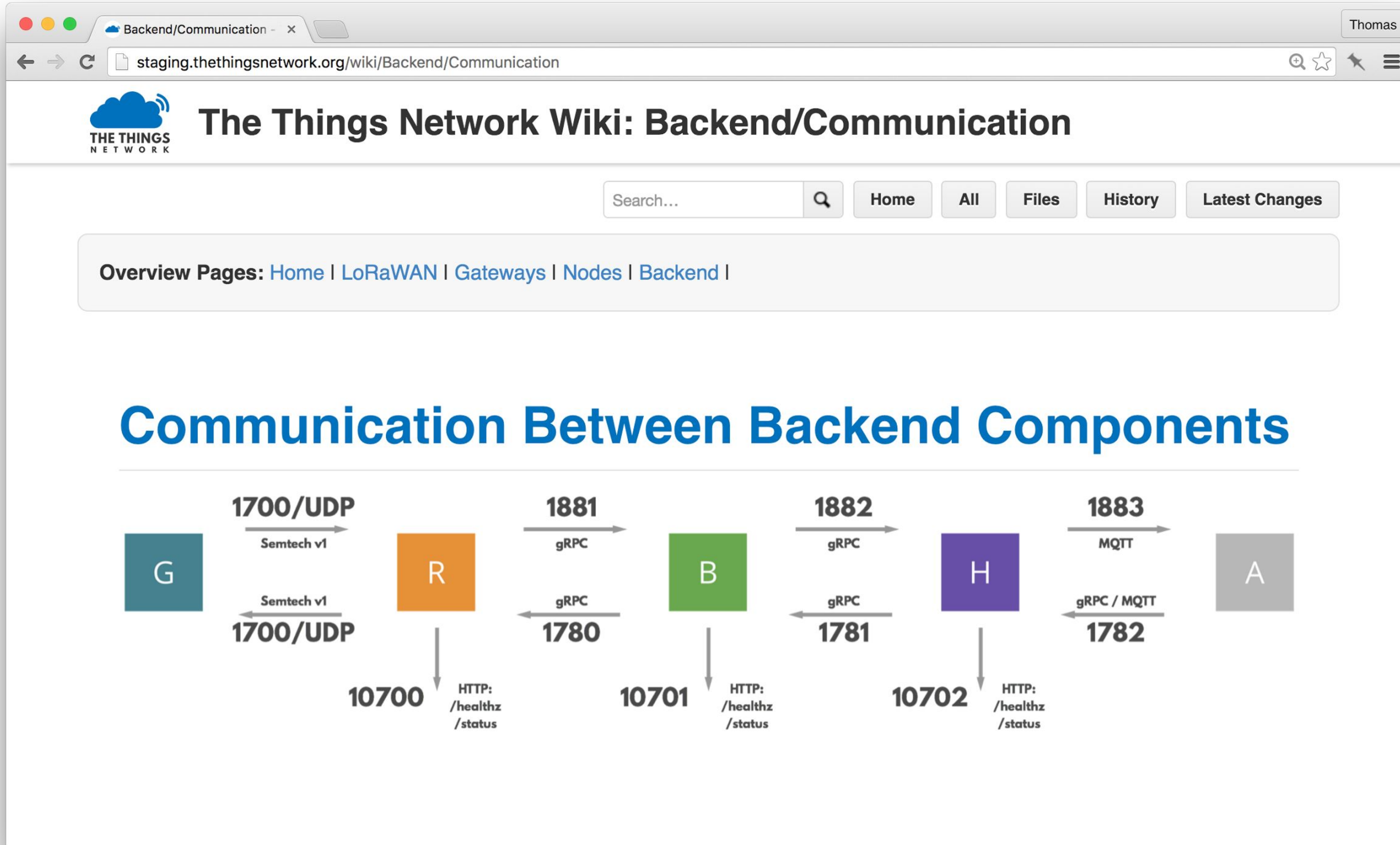
LoRaWAN R1.0 Open Stand

https://www.lora-alliance.org/kbdetail/Contenttype/ArticleDet/moduleId/583/Aid/23/PR/PR

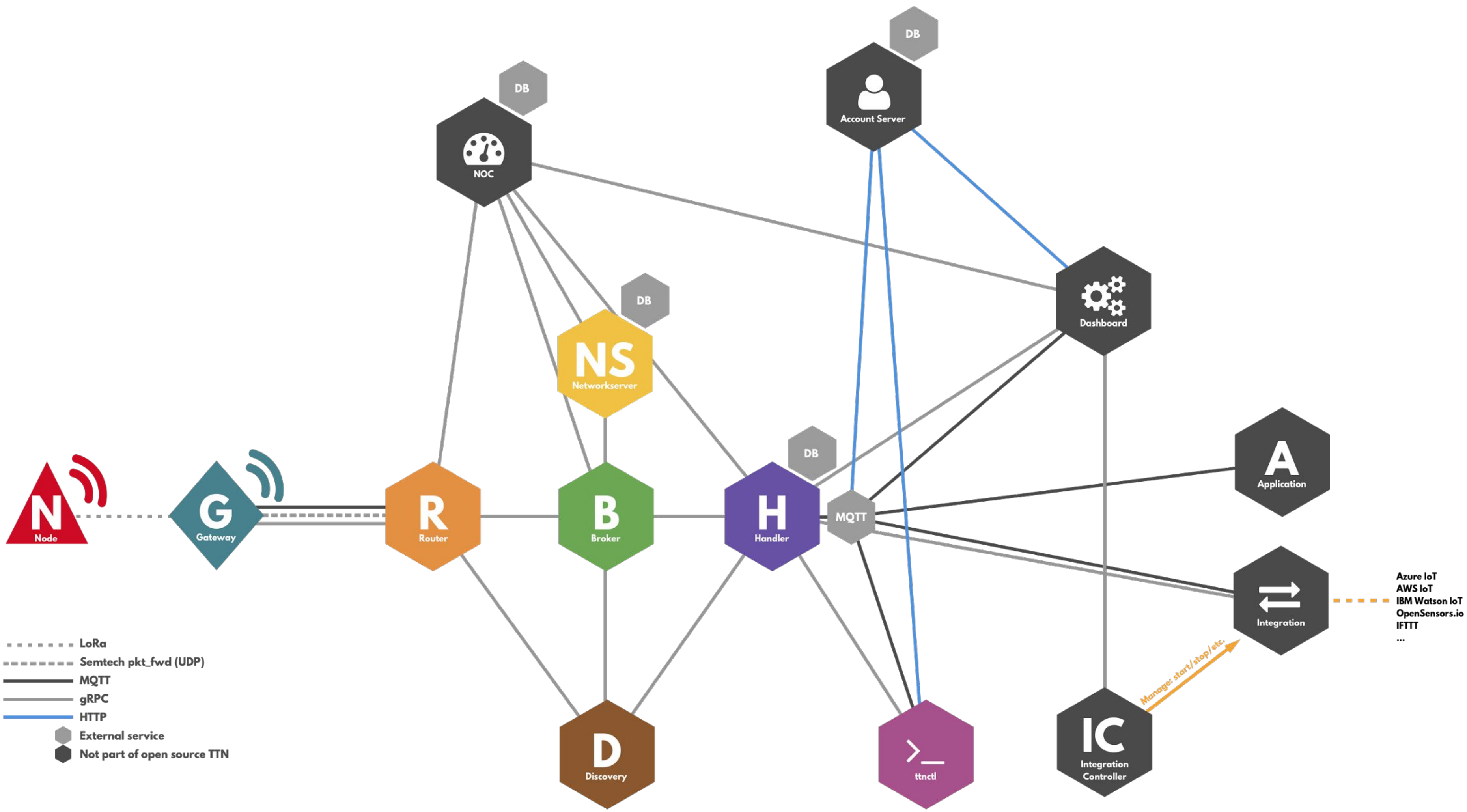


LoRaWAN network architecture is a typical star-of-stars topology in which the **gateways** are a transparent bridge relaying messages between **end-devices** and a central **network server**. Gateways are connected to the network server via standard IP connections, while end-devices use single-hop wireless communication to one or many


Backend/Overview - The Tl
Thomas
staging.thethingsnetwork.org/wiki/Backend/Overview
Backend Components
Packet
- DevAddr: 1345ABCD
- Payload: encrypted
Packet
- DevAddr: 1345ABCD
- Region: EU868
- Payload: encrypted
Packet
- DevEUI: 00A4C12BF2DAF0FB
- AppEUI: 70B3D57ED0000001
- Payload: encrypted
Packet
- DevEUI: 00A4C12BF2DAF0FB
- AppEUI: 70B3D57ED0000001
- Payload: decrypted
G
R
B
H
A
Gateway
Router
- Responsible for a region (EU868)
- Responsible for gateway scheduling and monitoring
Broker
- Responsible for a DevAddr range (1345AB/24)
- Each DevAddr has 0 or more (DevEUI,AppEUI,NwkSKey) pairs
- Find correct one with MIC check
- Deduplication
Handler
- Responsible for registered AppEUIs (70B3D57ED0000001, ...)
- Decrypt the payload
Backend Connect a Gateway | Connect an Application
Backend Components: Router | Broker | Handler


Backend/Communication -
Thomas
staging.thethingsnetwork.org/wiki/Backend/Communication
THE THINGS NETWORK
The Things Network Wiki: Backend/Communication
Search...
Home
All
Files
History
Latest Changes
Overview Pages: Home | LoRaWAN | Gateways | Nodes | Backend |
Communication Between Backend Components
G
R
B
H
A
1700/UDP
Semtech v1
Semtech v1
1700/UDP
1881
gRPC
gRPC
1780
1882
gRPC
gRPC
1781
1883
MQTT
gRPC / MQTT
1782
10700
HTTP:
/healthz
/status
10701
HTTP:
/healthz
/status
10702
HTTP:
/healthz
/status

Backend components in detail



Appendix C - Community

The Things Network

OUR VISION

The Internet was created by people that connected their networks to allow traffic from, to and over their servers and cables to pass for free. As a result, there was abundant data communication and exponential innovation.

The Things Network is doing the same for the Internet of Things by creating abundant data connectivity. So applications and businesses can flourish.

New data connectivity technologies

New data network technologies allow for things to connect to the internet without using 3G or WiFi.
This technology is called LoRaWAN and it is perfect for the Internet of Things as it is low battery, long range, and low bandwidth. Imagine a network that can be used without cumbersome WiFi passwords. mobile subscriptions. and zero setup costs.

Low costs

Because the reach is very high and the cost of the equipment is low, covering an entire city can be done with a small investment. The city of Amsterdam was covered with only 10 gateways at the cost of 1200 dollars each.

Crowd sourced

Because the costs are very low, we do not have to rely on large telco corporations to build such a network. Instead, we can crowdsource the network and make it public without any form of subscription. Our mission is to enable a network by the users for the users.

The Things Network

HOW TO CONTRIBUTE

Take a look at the most booming communities! We are highlighting the communities that are putting in work everyday. Want to find out why these communities are moving so fast? Take a look at their community pages.

Get coverage in your area by placing a Gateway

Place Gateway

Contribute to our open source code.

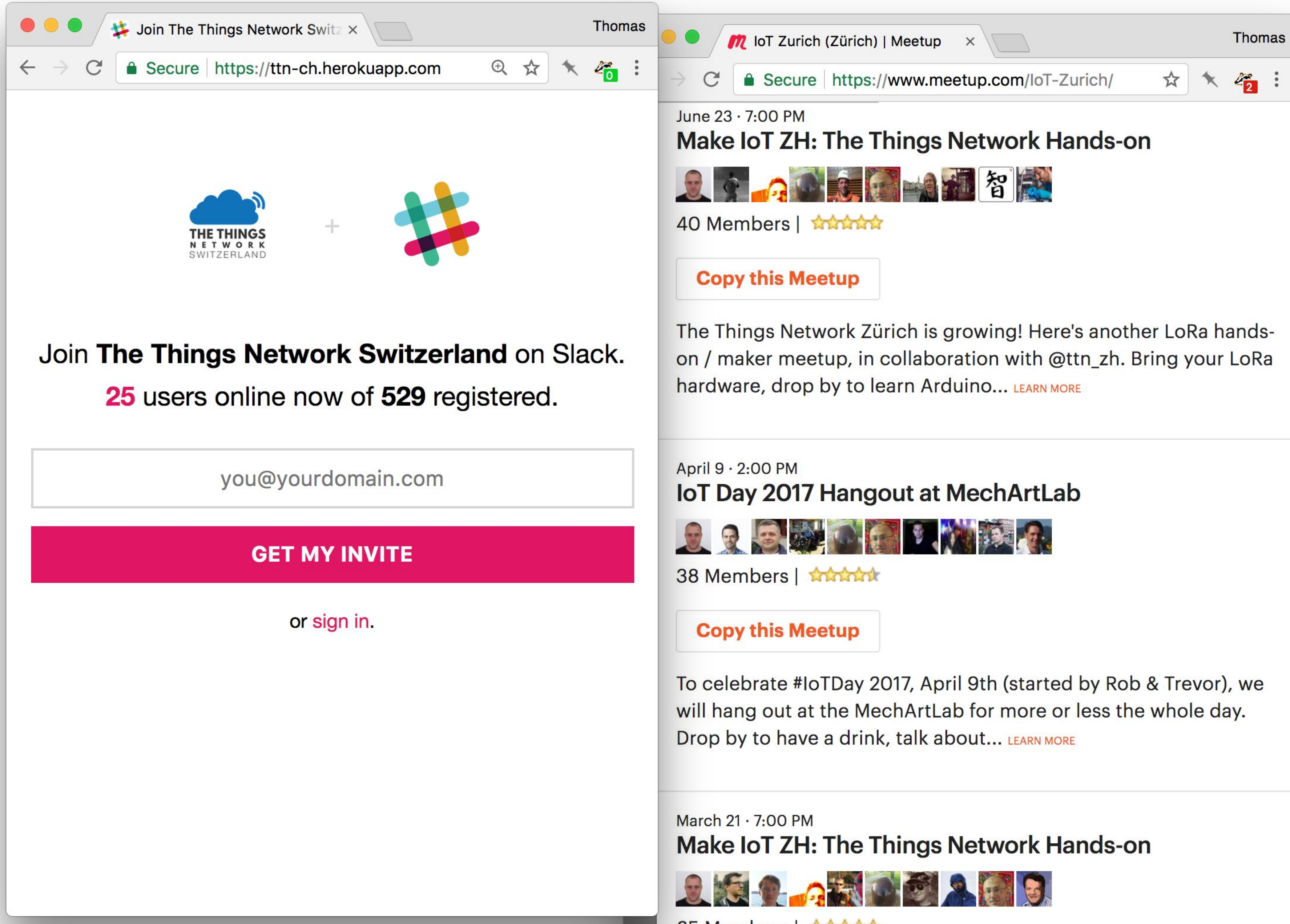
See our code

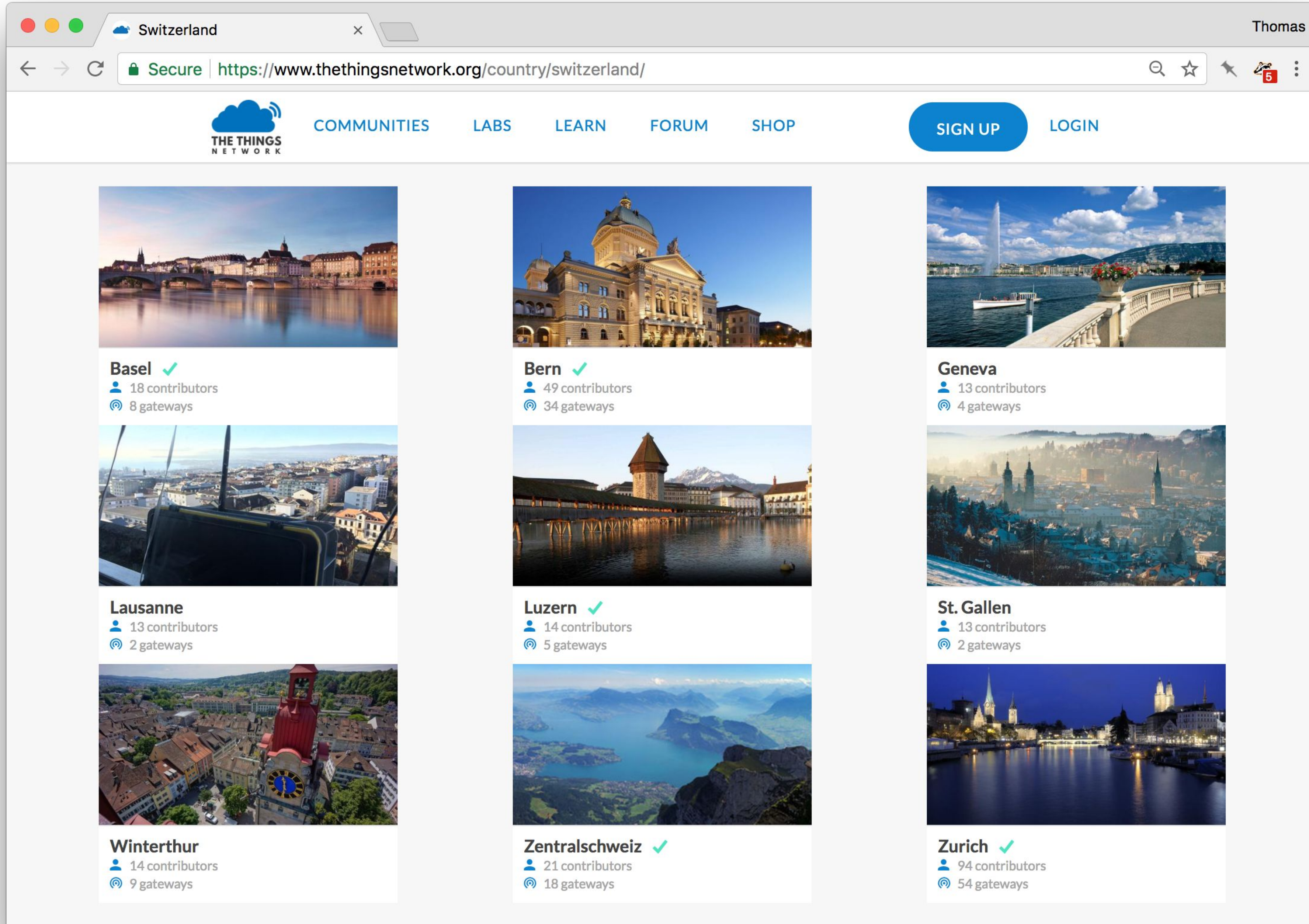
Help building our network through a community.

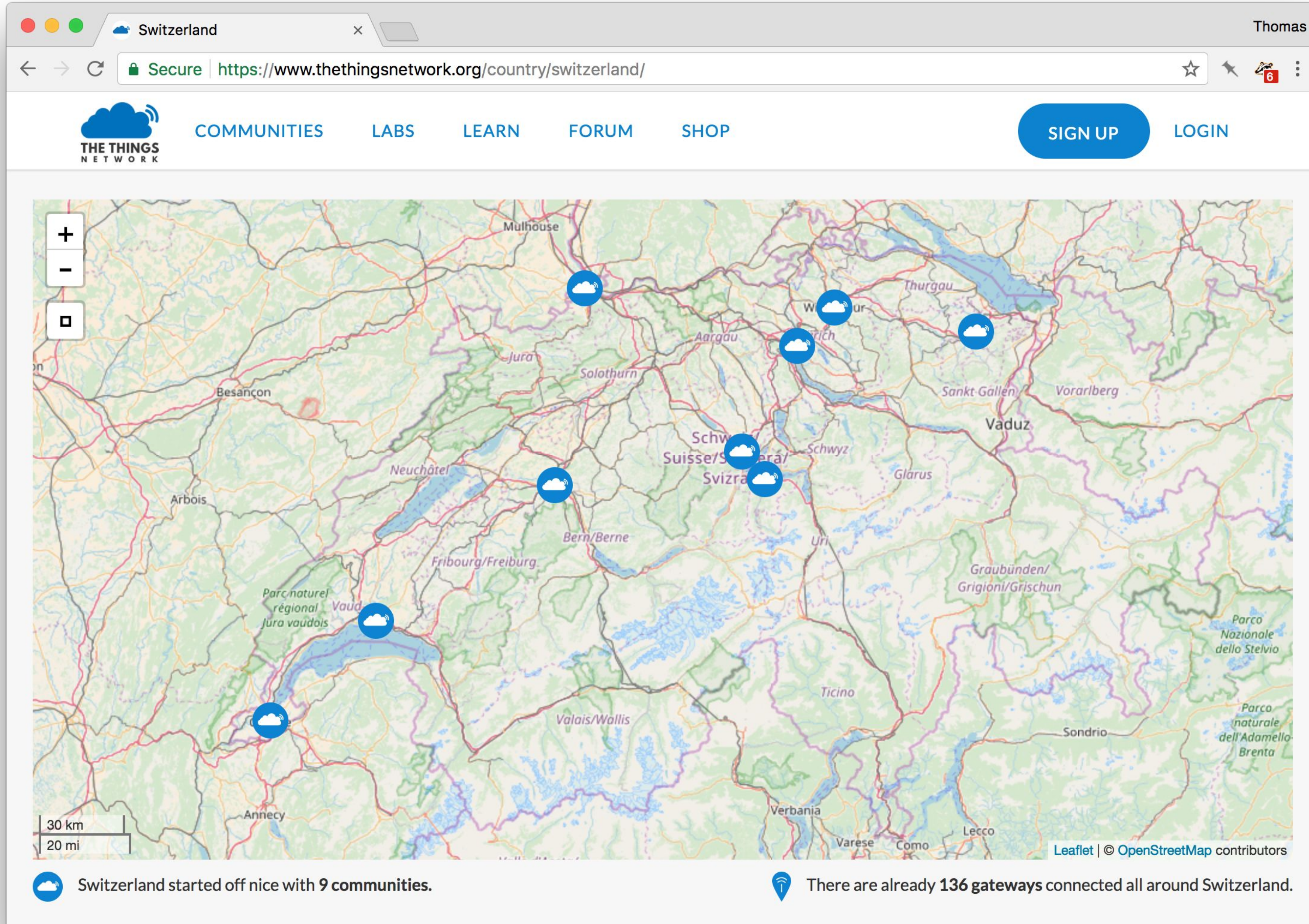
Communities

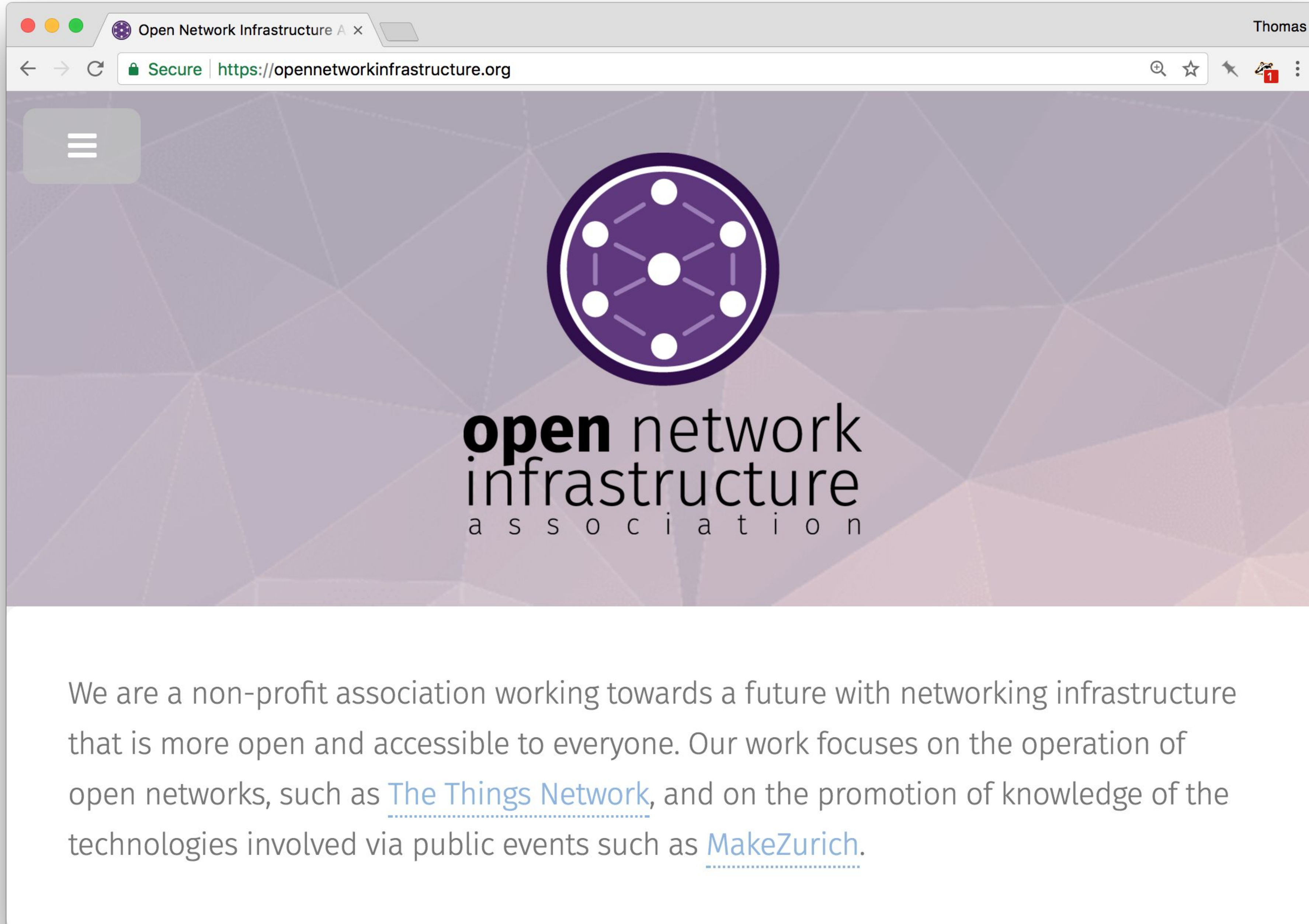
Work on use cases & connect with like-minded people on Labs.

Labs


Join The Things Network Switz
Thomas
Secure
https://ttn-ch.herokuapp.com
THE THINGS
NETWORK
SWITZERLAND
+
Join The Things Network Switzerland on Slack.
25 users online now of 529 registered.
you@yourdomain.com
GET MY INVITE
or sign in.
IoT Zurich (Zürich) | Meetup
Thomas
Secure
https://www.meetup.com/IoT-Zurich/
June 23 · 7:00 PM
Make IoT ZH: The Things Network Hands-on
智
40 Members |
Copy this Meetup
The Things Network Zürich is growing! Here's another LoRa hands-on / maker meetup, in collaboration with @ttn_zh. Bring your LoRa hardware, drop by to learn Arduino... LEARN MORE
April 9 · 2:00 PM
IoT Day 2017 Hangout at MechArtLab
38 Members |
Copy this Meetup
To celebrate #IoTDay 2017, April 9th (started by Rob & Trevor), we will hang out at the MechArtLab for more or less the whole day. Drop by to have a drink, talk about... LEARN MORE
March 21 · 7:00 PM
Make IoT ZH: The Things Network Hands-on


Switzerland
Thomas
Secure
https://www.thethingsnetwork.org/country/switzerland/
THE THINGS NETWORK
COMMUNITIES
LABS
LEARN
FORUM
SHOP
SIGN UP
LOGIN
Basel
18 contributors
8 gateways
Bern
49 contributors
34 gateways
Geneva
13 contributors
4 gateways
Lausanne
13 contributors
2 gateways
Luzern
14 contributors
5 gateways
St. Gallen
13 contributors
2 gateways
Winterthur
14 contributors
9 gateways
Zentralschweiz
21 contributors
18 gateways
Zurich
94 contributors
54 gateways


Switzerland
Thomas
Secure
https://www.thethingsnetwork.org/country/switzerland/
THE THINGS NETWORK
COMMUNITIES
LABS
LEARN
FORUM
SHOP
SIGN UP
LOGIN
Mulhouse
Thurgau
Aargau
Jura
Solothurn
Besançon
Sankt Gallen
Vorarlberg
Vaduz
Schwyz
Neuchâtel
Glarus
Arbois
Bern/Berne
Uri
Fribourg/Freiburg
Graubünden/
Grigioni/Grischun
Parc naturel
régional
Jura vaudois
Vaud
Parco
Nazionale
dello Stelvio
Ticino
Valais/Wallis
Parco
naturale
dell'Adamello
Brenta
Sondrio
Annecy
Verbania
Lecco
Varese
Como
30 km
20 mi
Leaflet | © OpenStreetMap contributors
Switzerland started off nice with **9 communities.**
There are already **136 gateways** connected all around Switzerland.


Open Network Infrastructure A
Thomas
Secure
https://opennetworkinfrastructure.org
open network
infrastructure
association
We are a non-profit association working towards a future with networking infrastructure that is more open and accessible to everyone. Our work focuses on the operation of open networks, such as The Things Network, and on the promotion of knowledge of the technologies involved via public events such as MakeZurich.

Thanks

Questions?

Contact @gnz, @ttn_zs or @tamberg

And join https://thethingsnetwork.org/c/zurich, https://thethingsnetwork.org/c/zentralschweiz/

Reducing E-waste

Tired of hacking?

Donate your hardware…

e.g. MechArtLab
Hohlstrasse 52
8004 Zürich