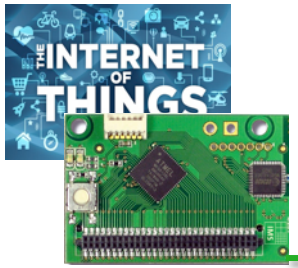
A Low-Cost LoRa Gateway with QoS Features

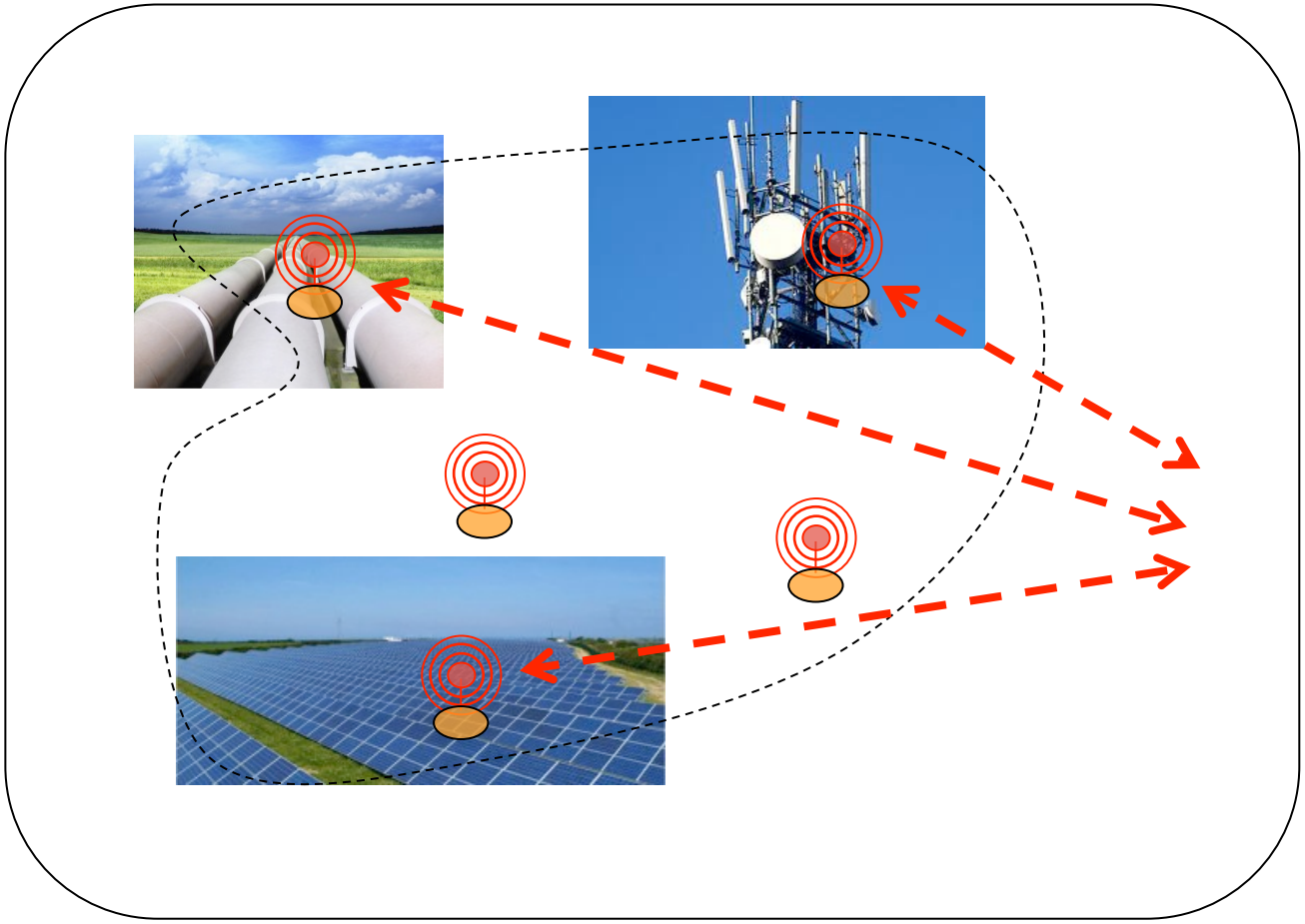
C. Pham (Univ. Pau, France)

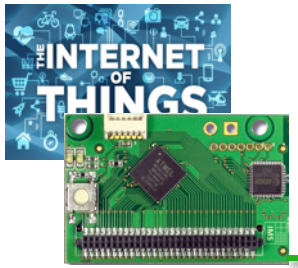
Last update: 23rd May, 2016

Prof. Congduc Pham
http://www.univ-pau.fr/~cpham
Université de Pau, France



Sensors/IoT for surveillance





Connectivity is a challenge

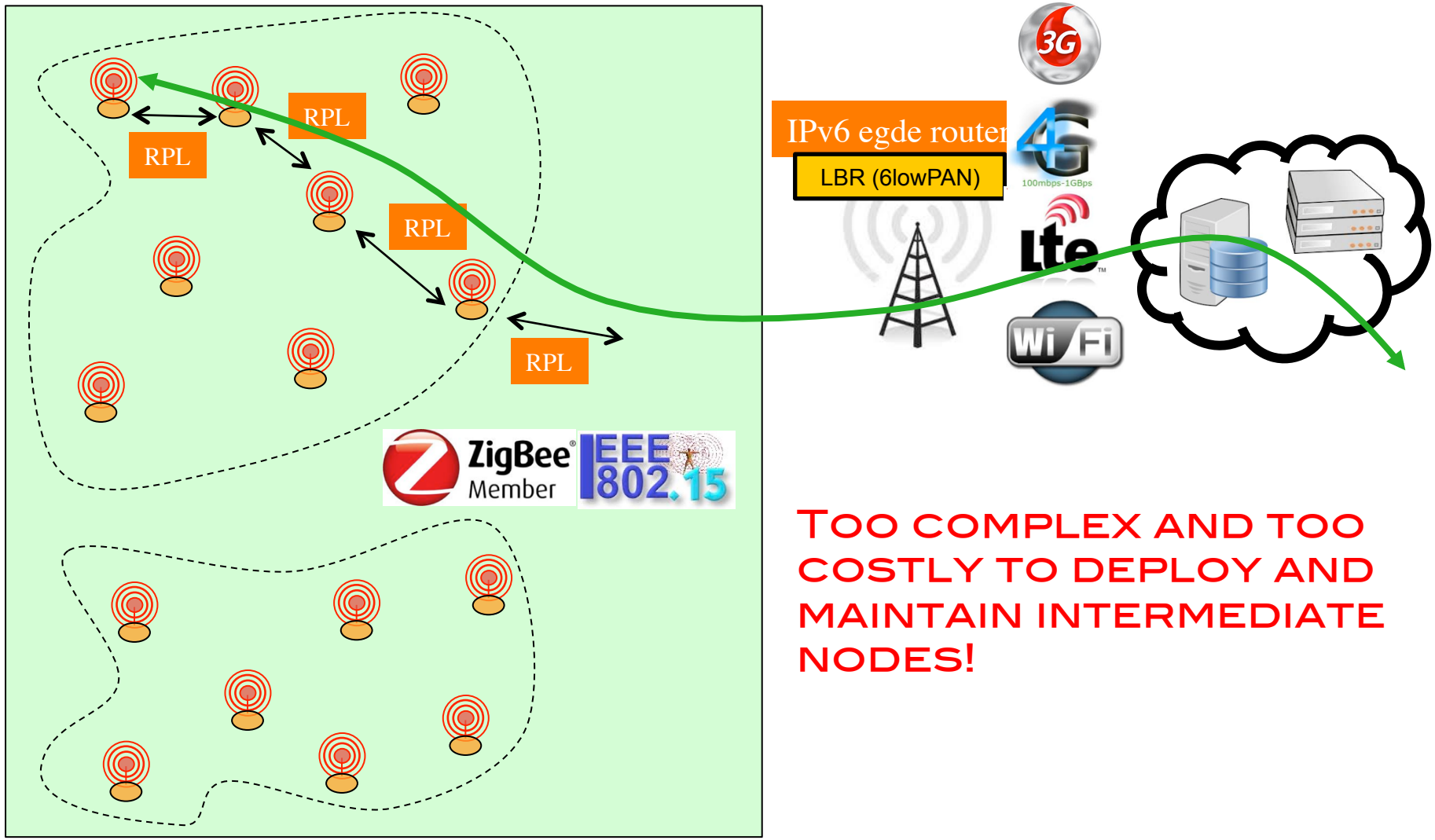
Internet of Objects 80% of volume

Requirements:

- How to connect Low Cost Assets or having no Energy source, non rechargeable?
- Low Cost communication
- Low Cost Infrastructure
- Low Power Technology
- Robust Communication
- Allowing Mobility
- Scalability

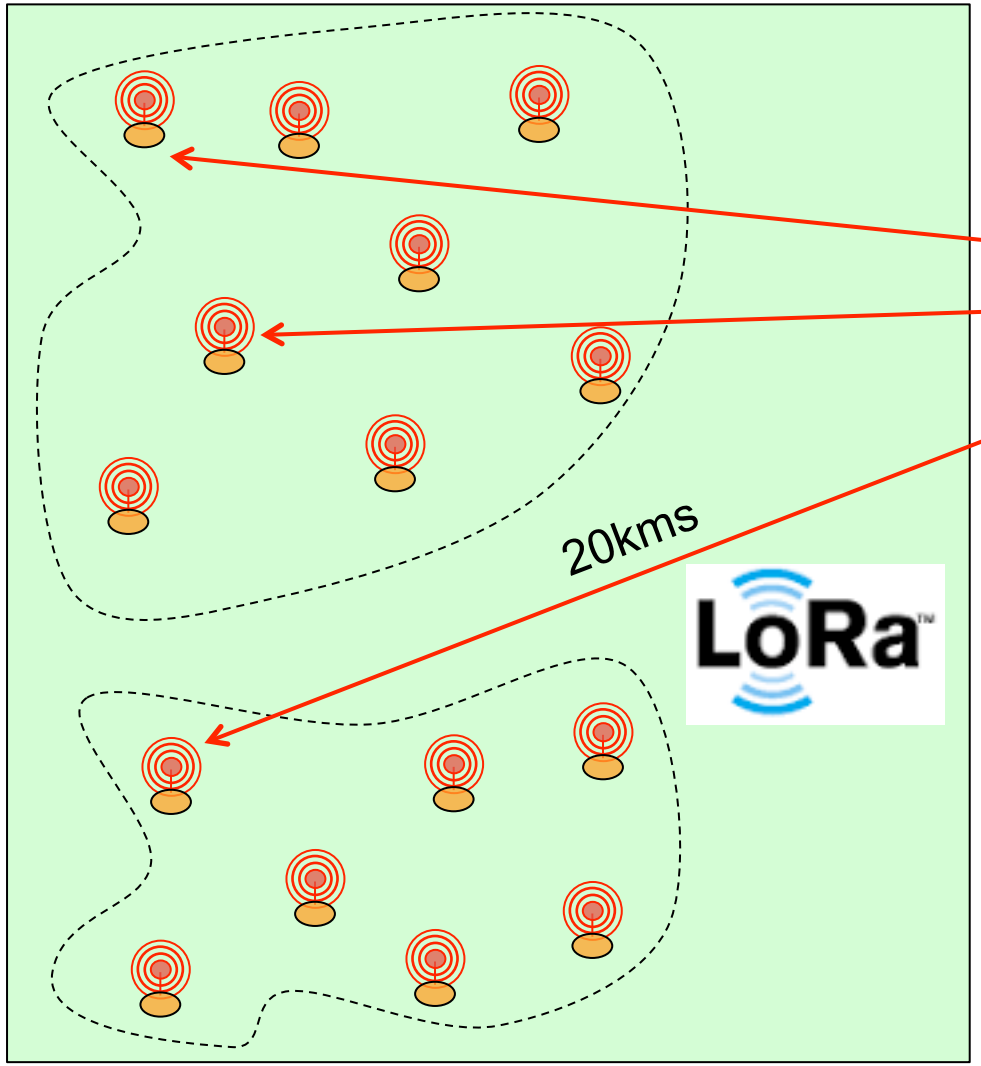
Slide from Semtech

Traditional multi-hop to Sink

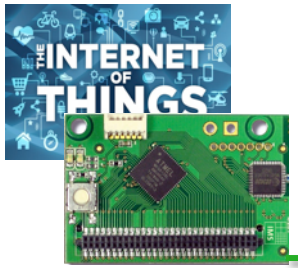


Too complex and too costly to deploy and maintain intermediate nodes!

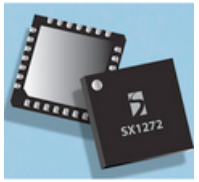
1-hop to Sink with long-range technology



Long-range technologies (Semtech's LoRa™ for instance) reduces the cost of deployment



Semtech's LoRa technology



dBm – power referred to 1 mW,

$P_{dBm}=10\log(P/1mW)$

- Parameters
  - Bandwidth: 125kHz, 250kHz, 500kHz
  - Coding rate: 4/5, 4/6, 4/7, 4/8
  - Spreading factor: 6 to 12

Sensitivity: lowest input power with acceptable link quality, typically 1% PER

SpreadingFactor (RegModemConfig2)	Spreading Factor (Chips / symbol)	LoRa Demodulator SNR
6	64	-5 dB
7	128	-7.5 dB
8	256	-10 dB
9	512	-12.5 dB
10	1024	-15 dB
11	2048	-17.5 dB
12	4096	-20 dB

Bandwidth (kHz)	Spreading Factor	Nominal Rb (bps)	Sensitivity (dBm)
125	6	9380	-122
125	12	293	-137
250	6	18750	-119
250	12	586	-134
500	6	3750	-116
500	12	1172	-131

Rule of thumb

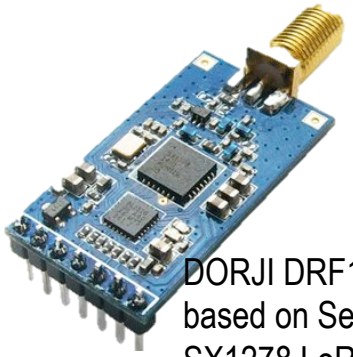
6dB increase = twice the range in LOS

12dB needed for urban areas

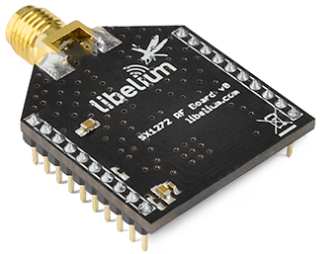
Bandwidth (kHz)	Spreading Factor	Coding rate	Nominal Rb (bps)	Sensitivity (dBm)
125	12	4/5	293	-136
250	12	4/5	586	-133
500	12	4/5	1172	-130

Tables from Semtech

LoRa modules from Semtech's SX127x chips



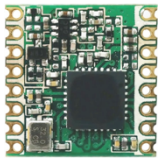
DORJI DRF1278DM is based on Semtech SX1278 LoRa 433MHz



Libelium LoRa is based on Semtech SX1272 LoRa 863-870 MHz for Europe

inAir9 based on SX1276

Froggy Factory LoRa module (Arduino)

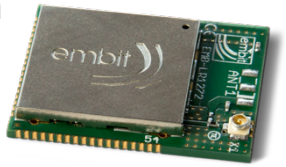


HopeRF RFM series

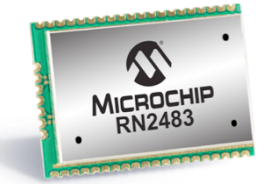
HopeRF HM-TRLR-D

LinkLabs Symphony module

IMST IM880A-L is based on Semtech SX1272 LoRa 863-870 MHz for Europe



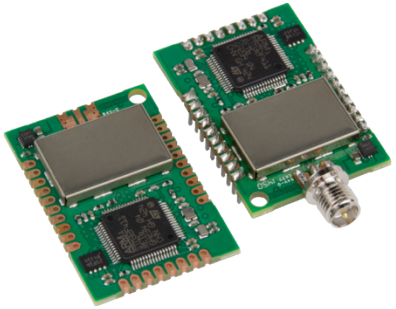
Embit LoRa



Microship RN2483

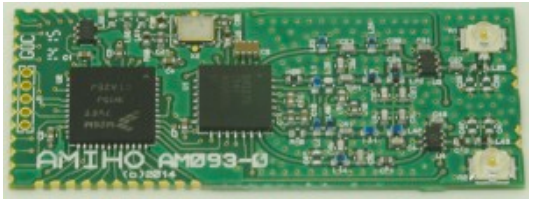
habSupplies

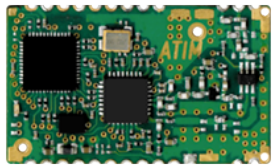
Adeunis ARF8030AA- Lo868



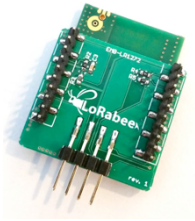
Multi-Tech MultiConnect mDot

AMIHO AM093





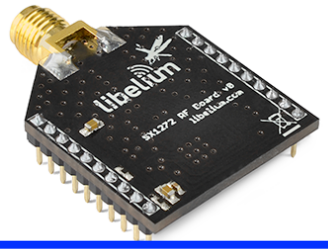
ARM-Nano N8 LoRa module from ATIM



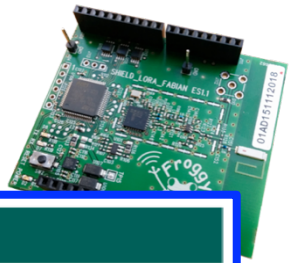
SODAQ LoRaBee Embit

SODAQ LoRaBee RN2483

LoRa modules from Semtech's SX127x chips

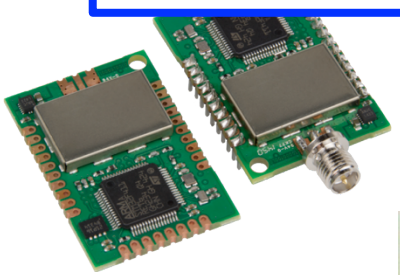


Libelium LoRa is based on Semtech SX1272 LoRa 863-870 MHz for Europe



LoRa® Transceivers

Part Number	Frequency Range (MHz)	Link Budget (dB)	Rx Current (mA)	FSK max DR (kbps)	LoRa DR (kbps)	Max Sensitivity (dBm)	Tx Power (dBm)
SX1272	860 – 1020	158	10	300	0.3 – 37.5	-137	+ 20
SX1273	860 – 1020	150	10	300	1.7 – 37.5	-130	+ 20
SX1276	137 – 1020	168	9.9	300	0.018 – 37.5	-148	+ 20
SX1277	137 – 1020	158	9.9	300	1.7 – 37.5	-139	+ 20
SX1278	137 – 525	168	9.9	300	0.018 – 37.5	-148	+ 20



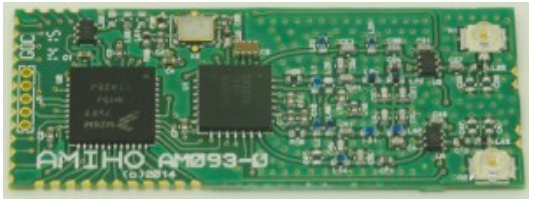
Multi-Tech MultiConnect mDot

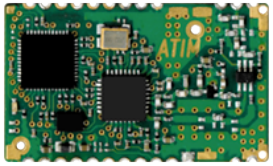
habSupplies

Adeunis ARF8030AA- Lo868

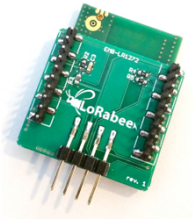
Microship RN2483

AMIHO AM093





ARM-Nano N8 LoRa module from ATIM

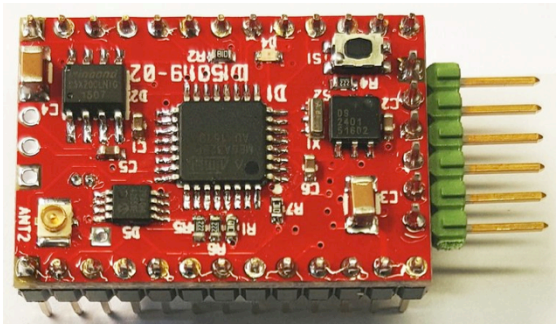


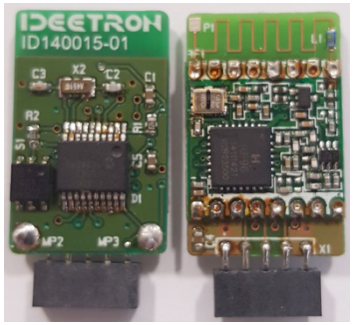
SODAQ LoRaBee Embit

SODAQ LoRaBee RN2483

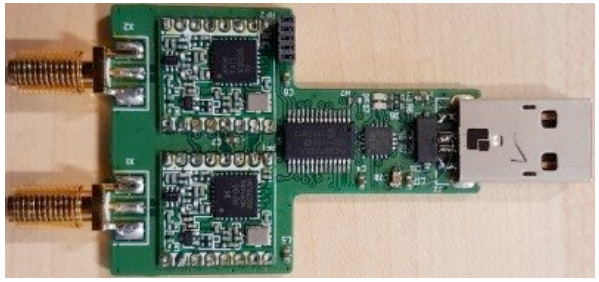
Some ready-to-use LoRa devices

LoRa Mote from Semtech

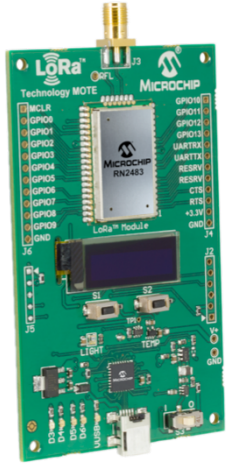




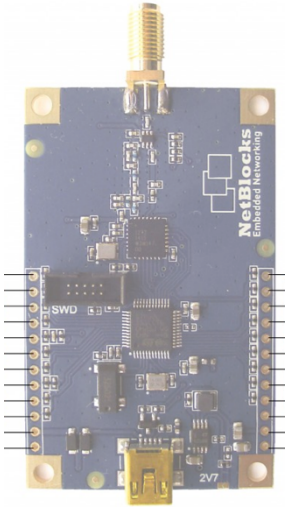
LoRa™ Alliance



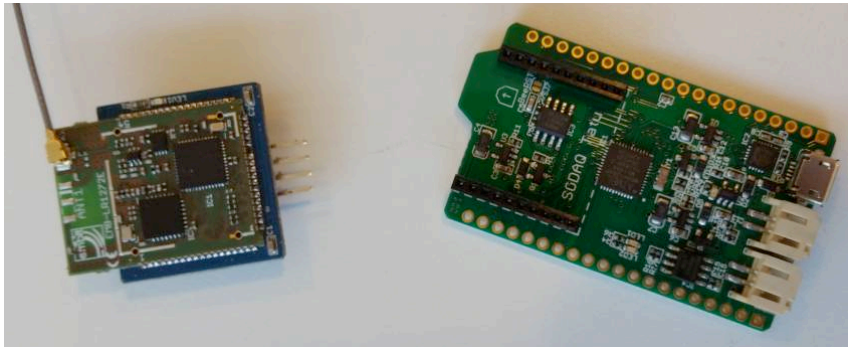
HopeRF/Ideetron motes



Microchip LoRa mote

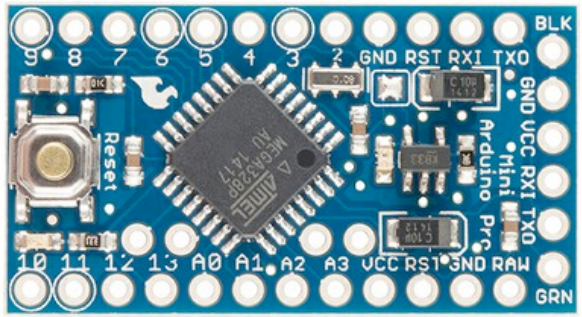


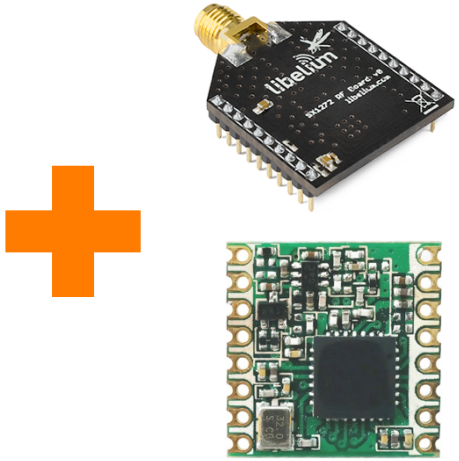
NetBlocks XRange



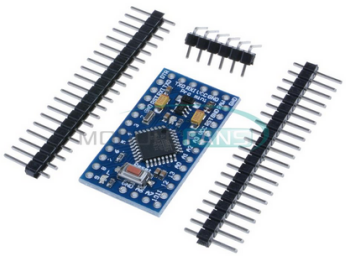
SODAQ Tatu with LoraBee (Embit)

Our low-cost LoRa device





Pro Mini atmega328 3.3V 8M

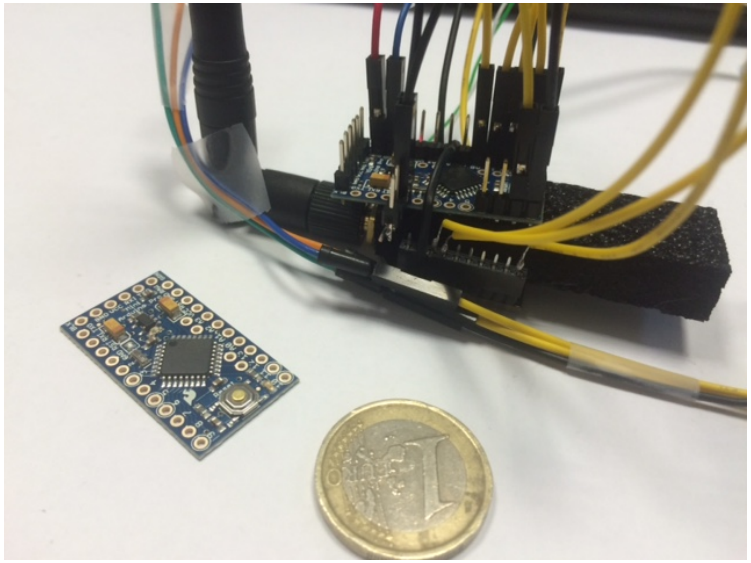


For Arduino Compatible Nano

New Pro Mini atmega328 3.3 V 8 M remplacer ATmega128 pour

US $1.86 / pièce

Expédition Gratuite

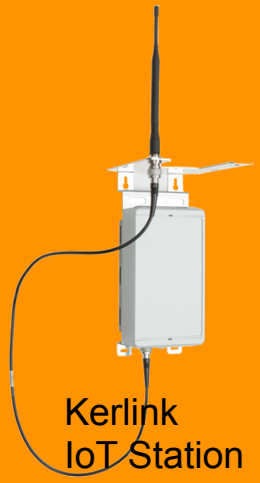


Deploying your LoRa network

Operator-based approach (with subscription)

Privately-based approach

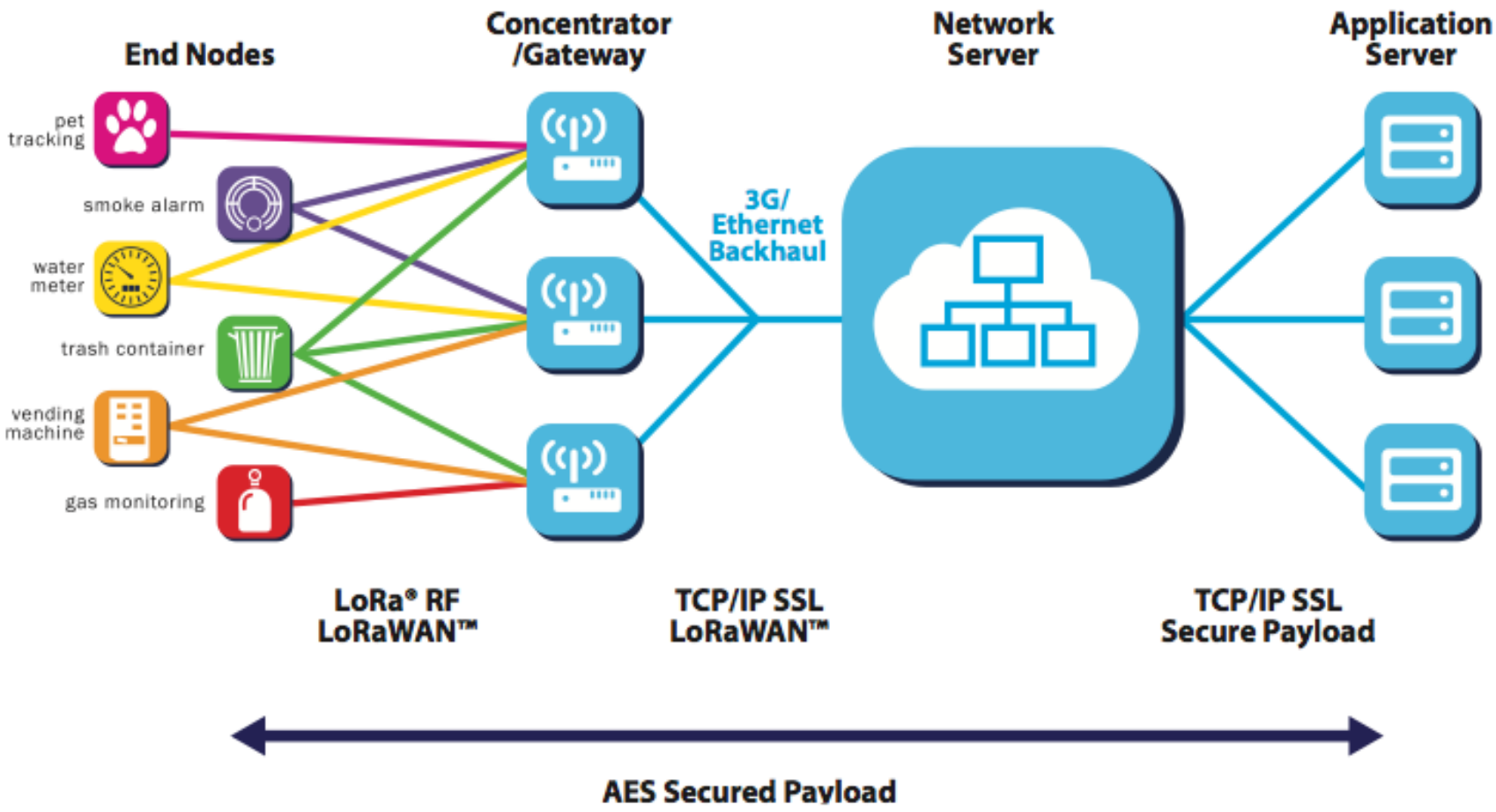
Multi-Tech Conduit



Kerlink
IoT Station

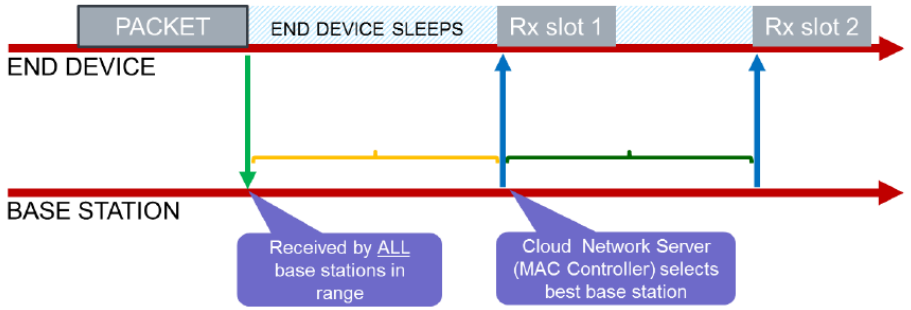
And much more...

LoRaWAN ARCHITECTURE



LoRaWAN Specification

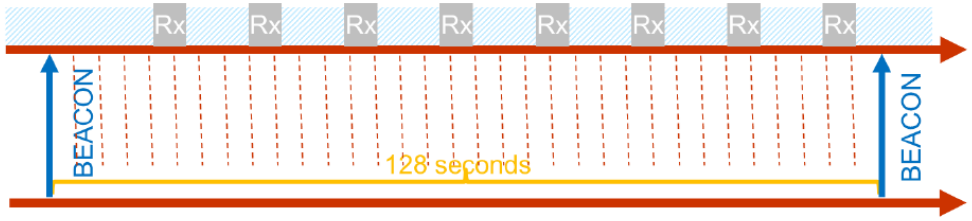
Class A: Receiver Initiated Transmission strategy (RIT)



Class B: Coordinated Sampled Listening (CSL)

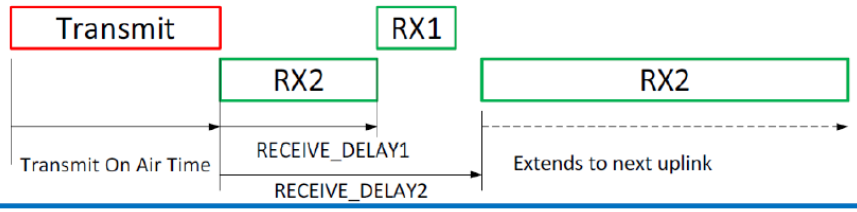
Network may send downlink packet to node at any Rx slot

END DEVICE



BASE STATION

Class C: Continuous Listening



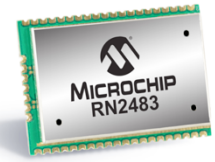
Latency constrained applications

Power Efficiency

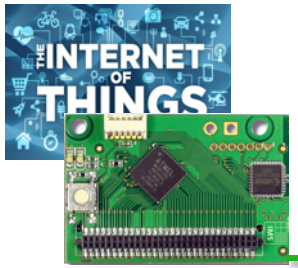
LoRa Alliance

Wide Area Networks for IoT

Application				
LoRa® MAC				
MAC options				
Class A (Baseline)	Class B (Baseline)	Class C (Continuous)		
LoRa® Modulation				
Regional ISM band				
EU 868	EU 433	US 915	AS 430	—



LoRa™ Long-Range Sub-GHz Module (Part # RN2483)



LoRaWAN channels

- EU 863-870MHz ISM Band

DataRate	Configuration	Indicative physical bit rate [bit/s]
0	LoRa: SF12 / 125 kHz	250
1	LoRa: SF11 / 125 kHz	440
2	LoRa: SF10 / 125 kHz	980
3	LoRa: SF9 / 125 kHz	1760
4	LoRa: SF8 / 125 kHz	3125
5	LoRa: SF7 / 125 kHz	5470
6	LoRa: SF7 / 250 kHz	11000
7	FSK: 50 kbps	50000
8..15	RFU	

TXPower	Configuration
0	20 dBm (if supported)
1	14 dBm
2	11 dBm
3	8 dBm
4	5 dBm
5	2 dBm
6..15	RFU

Table 14: Data rate and TX power table

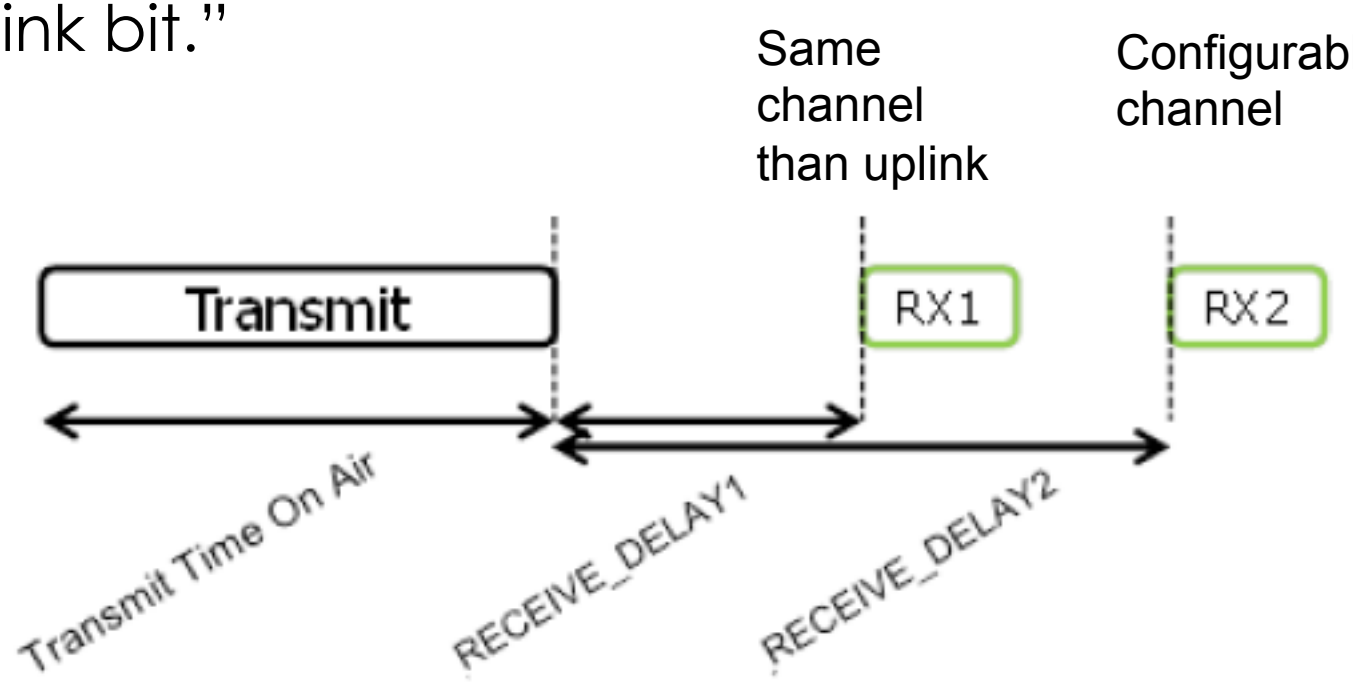
- Minimum set

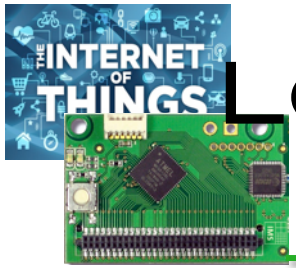
Modulation	Bandwidth [kHz]	Channel Frequency [MHz]	FSK Bitrate or LoRa DR / Bitrate	Nb Channels	Duty cycle
LoRa	125	868.10 868.30 868.50	DR0 to DR5 / 0.3-5 kbps	3	<1%

Table 12: EU863-870 default channels

LoRaWAN MAC protocol main features (1)

- "Following each uplink transmission the end-device opens two short receive windows. The receive window start times is a configured periods are the end of the transmission of the last uplink bit."





LoRaWAN MAC protocol main features (2)

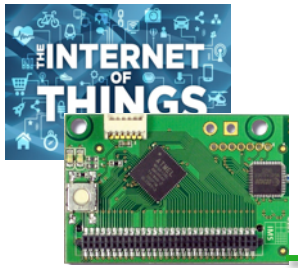
- Adaptive Data Rate, ACKnowlegements, Retransmission procedure, Encryption
- Duty-cycle and channel management
  - The maximum end-device transmit duty cycle is:

$$aggregated\ duty\ cycle = \frac{1}{2^{\mathbf{MaxDCycle}}}$$

  - The valid range for **MaxDCycle** is [0 : 15]. A value of 0 corresponds to —no duty cycle limitation except the one set by the regional regulation (e.g. 1%)

- Channel off-time after transmission

$$\mathrm{Toff}_{\mathbf{subband}} = \frac{\mathrm{TimeOnAir}}{\mathrm{DutyCycle}_{\mathbf{subband}}} - \mathrm{TimeOnAir}$$

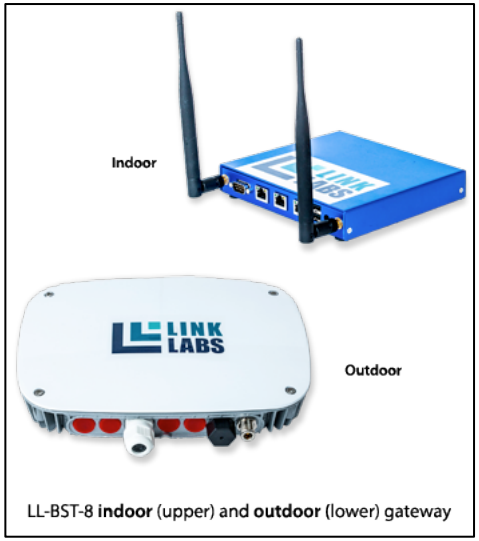


LoRa Gateways
(non exhaustive list)

Multi-Tech Conduit

Embedded Planet
EP-M2M-LORA

Ideetron Lorank 8



LinkLabs Symphony

Kerlink IoT Station

PicoWAN from
Archos

TheThingNetwork

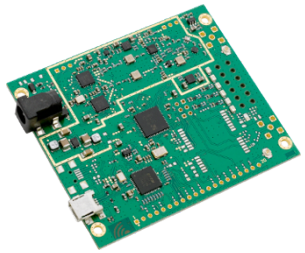
Or build your own one:
Arduino, Rasperry PI, …

Commercial LoRa gateways

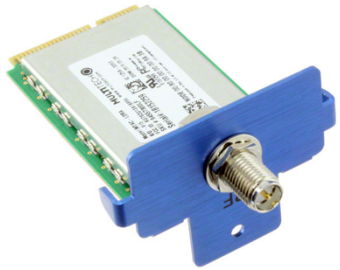
- Most of them use the SX1301 which is a « Base Band Processor for Data Concentrator for Long Range Communication Network"
- Main feature is simultaneous reception on several channels

From Nestor Ayuso github

Semtech IoT starter



IMST iC880A



Multitech mCard-LoRa

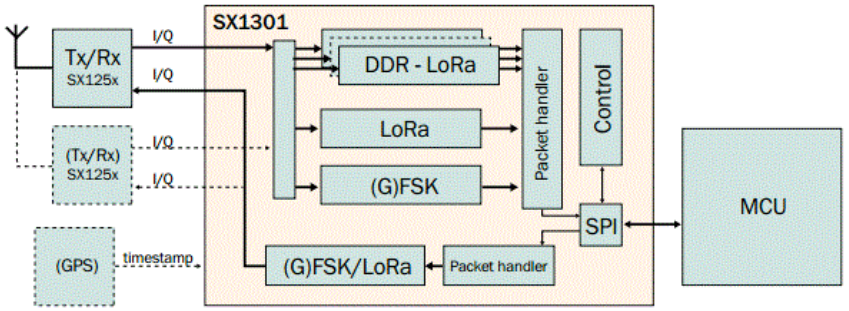
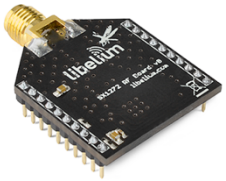
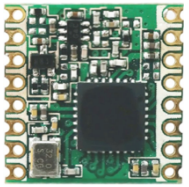


Figure from Semtech

Low-cost LoRa gateways

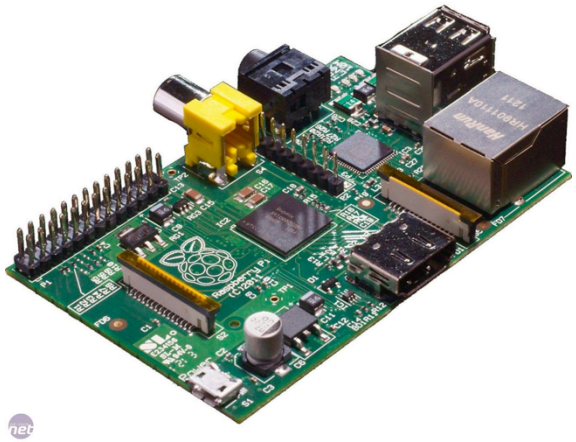
- Commercial gateways
  - cost hundredth of euros or dollars
  - are sometime non-open source or difficult to customize
- Low-cost gateways can be built for medium-scale ad-hoc deployment scenarios

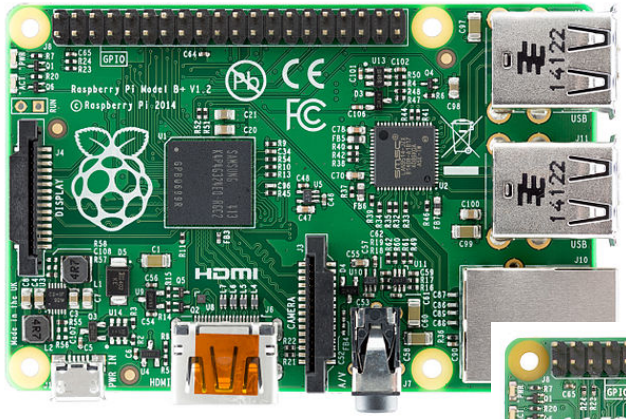


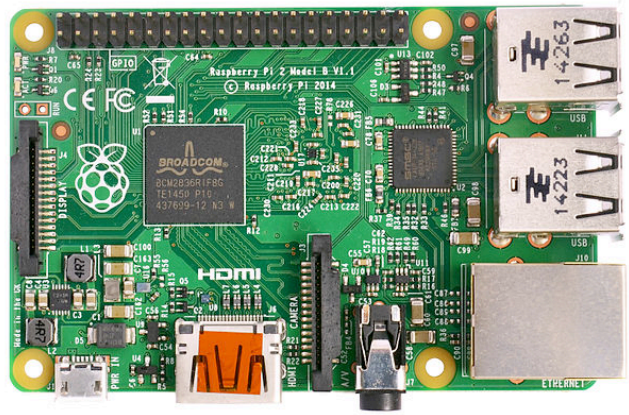


  - Use regular SX1272/76 chip for « single connection » from end-devices to the gateway
  - Use off-the-shelves embedded UNIX platforms for much lower-cost and easy enhancement
  - Use standard UNIX tools and high-level language for maximum flexibility and evolution
  - Use freely available cloud-based data management tools and middleware

Raspberry-based LoRa gateway

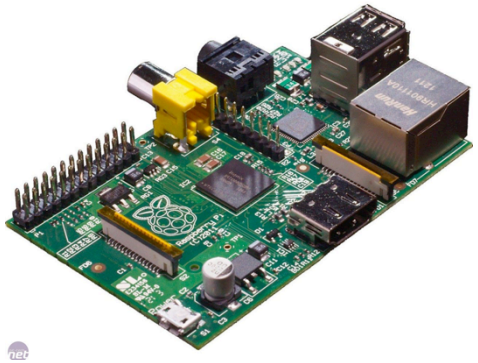


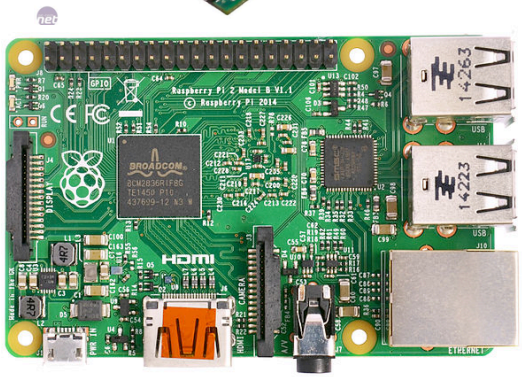


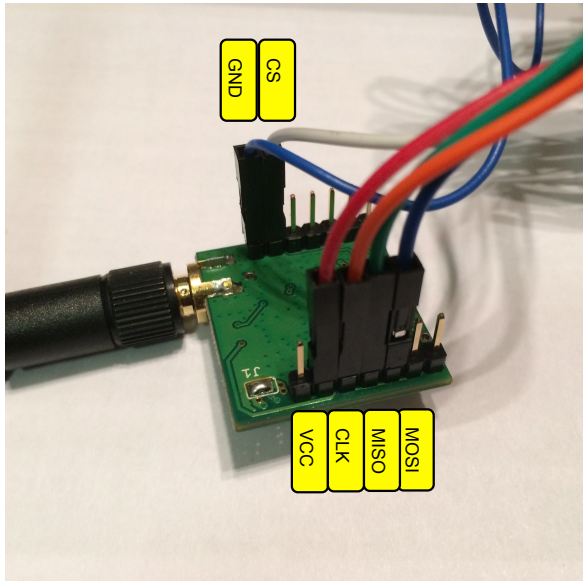


You can use Raspberry 1 model B or B+ or Raspberry 2 model B. The most important usefull feature is the Ethernet interface for easy Internet connection. You can use WiFi to get Internet connection by adding a WiFi USB dongle.

Connecting the radio module







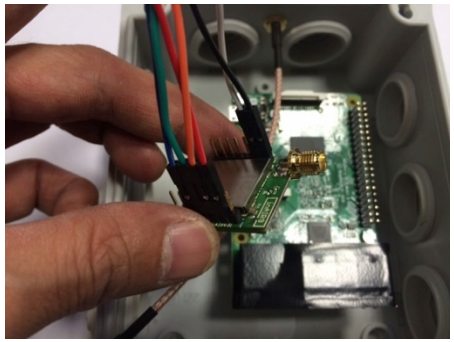
GPIO#	2nd func.	Pin#	Pin#	2nd func.	GPIO#
	+3.3 V	1	2	+5 V	
2	SDA1 (I2C)	3	4	+5 V	
3	SCL1 (I2C)	5	6	GND	
4	GCLK	7	8	TXD0 (UART)	14
	GND	9	10	RXD0 (UART)	15
17	GEN0	11	12	GEN1	18
27	GEN2	13	14	GND	
22	GEN3	15	16	GEN4	23
	+3.3 V	17	18	GEN5	24
10	MOSI (SPI)	19	20	GND	
9	MISO (SPI)	21	22	GEN6	25
11	SCLK (SPI)	23	24	CE0_N (SPI)	8
	GND	25	26	CE1_N (SPI)	7
(RPi 1 Models A and B stop here)					
EEPROM	ID_SD	27	28	ID_SC	EEPROM
5	N/A	29	30	GND	
6	N/A	31	32		12
13	N/A	33	34	GND	
19	N/A	35	36	N/A	16
26	N/A	37	38	Digital IN	20
	GND	39	40	Digital OUT	21

Depending on the model, you can have the « short » or the « long » GPIO interface. However, the SPI pins are at the same location therefore it does not change the way you connect the radio module if you take pin 1 as the reference. Connect the SPI pins (MOSI, MISO, CLK, CS) of the radio to the corresponding pins on the RPI. Note that CS goes to CE0_N on the RPI.

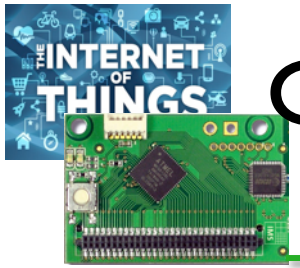
Put it in a box








You can have a more integrated version, with a box for outdoor usage and PoE splitter to power the Raspberry with the Ethernet cable. See how we also use a DC-DC converter to get the 5V for the RPI.



Our Low-Cost gateway architecture

Kept as simple as possible

high-level lang. e.g. python

Most of user or application specific logics is done here!
We provide some basic features, up to you to enhance them

radio bridge program

stdout

stdin

post processing

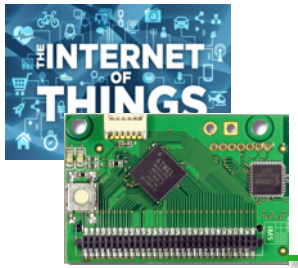
stdout

lora_gateway program

Modified SX1272 lib

ArduPi lib

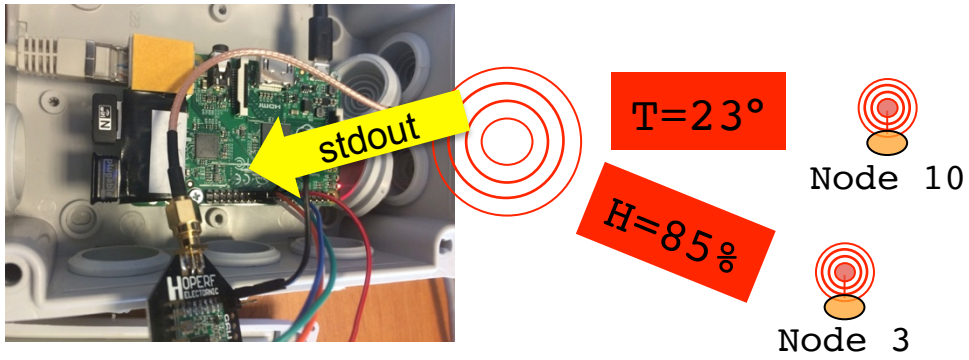
Raspbian



Other DIY initiatives

- piinthesky
  - LoRa gateway by Dave Ackerman for High Altitude Ballooning
  - Nice work with ncurses and 'curl'-based upload
  - Too « monolithic », difficult to modify or adapt to new applications
- The Thing Network
  - « The Things » gateway with LoRaWAN
  - Detailed not very available yet
- Nestor Ayuso's Raspberry LoRa gateway
  - Raspberry PI+mCard (SX1301 concentrator) with LoRaWAN using LoRaMAC
- Our gateway
  - Raspberry PI+SX1272/76 for lower cost and medium-size deployment
  - No LoRaWAN to ease ad-hoc development & deployment
  - More elaborated channel access method and QoS proposals
  - Strong separation between gateway and post-processing
  - Should be seen as a starting point for user/application specific needs

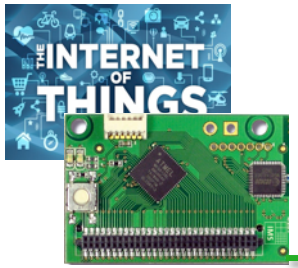
Transparent LoRa bridge




```
> sudo ./lora_gateway
Power ON: state 0
LoRa mode: 4
Setting mode: state 0
Channel CH_10_868: state 0
Power M: state 0
Get Preamble Length: state 0
Preamble Length: 8
LoRa addr 1 : state 0
SX1272/76 configured as LR-BS. Waiting RF input for transparent RF-serial bridge

--- rxlora. dst=1 type=0x10 src=10 seq=0 len=5 SNR=9 RSSIpkt=-54
^p1,16,10,0,5,9,-54
T=23°
--- rxlora. dst=1 type=0x10 src=3 seq=0 len=5 SNR=8 RSSIpkt=-54
^p1,16,3,0,5,8,-54
H=85%
```

Accepts remote commands to:

- Change LoRa mode
- Change channel
- Change transmission power
- Enable/Disable ACK
- …



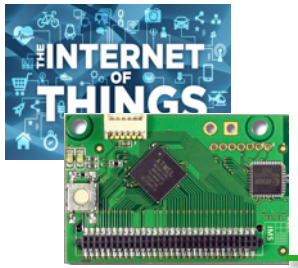
Post-processing received data

- Pre-defined sequences inserted by the gateway or the end-device allow for information exchanged between the gateway and the post-processing program

```
> sudo ./lora_gateway | python ./parseLoRaStdin.py
Power ON: state 0
LoRa mode: 4
Setting mode: state 0
Channel CH_10_868: state 0
Power M: state 0
Get Preamble Length: state 0
Preamble Length: 8
LoRa addr 1 : state 0
SX1272/76 configured as LR-BS. Waiting RF input for transparent RF-serial bridge

--- rxlora. dst=1 type=0x10 src=10 seq=0 len=5 SNR=9 RSSIpkt=-54
^p1,16,10,0,5,9,-54
Rcv ctrl packet info 1,16,10,0,5,9,-54
(dst=1 type=0x10 src=10 seq=0 len=5 SNR=9 RSSI=-5
T=23°
--- rxlora. dst=1 type=0x10 src=3 seq=0 len=5 SNR=8 RSSIpkt=-54
^p1,16,3,0,5,8,-54
Rcv ctrl packet info 1,16,3,0,5,8,-54
(dst=1 type=0x10 src=3 seq=0 len=5 SNR=8 RSSI=-54)
H=85%
```

All lines that are not prefixed by specific character sequence are displayed unchanged

^p provides information on the last received packet: dst, type, src, seq, len, SNR & RSSI



IoT cloud?

Dropbox

Firebase

FIWARE

Axēda®

ioBridge®
Connect things.

ThingSpeak

GroveStreams

SensorCloud™

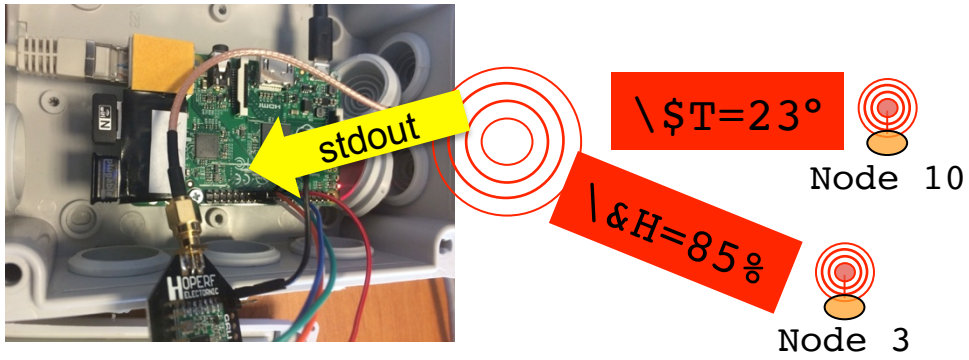
freeboard

dweet.io

OpenRemote

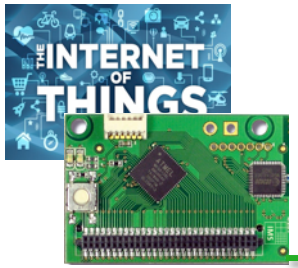
TempoIQ

LOG RECEIVED MESSAGES USING CLOUD SERVICES




```
> sudo ./lora_gateway | python ./parseLoRaStdin.py
Power ON: state 0
LoRa mode: 4
Setting mode: state 0
Channel CH_10_868: state 0
Power M: state 0
Get Preamble Length: state 0
Preamble Length: 8
LoRa addr 1 : state 0
SX1272/76 configured as LR-BS. Waiting RF input for transparent RF-serial bridge

--- rxlora. dst=1 type=0x10 src=10 seq=0 len=5 SNR=9 RSSIpkt=-54
Rcv ctrl packet info 1,16,10,0,5,9,-54
(dst=1 type=0x10 src=10 seq=0 len=5 SNR=9 RSSI=-54)
rcv msg to log (\$) on dropbox : T=23°
--- rxlora. dst=1 type=0x10 src=3 seq=0 len=5 SNR=8 RSSIpkt=-54
Rcv ctrl packet info 1,16,3,0,5,8,-54
(dst=1 type=0x10 src=3 seq=0 len=5 SNR=8 RSSI=-54)
rcv msg to log (\&) on firebase : H=85%
```

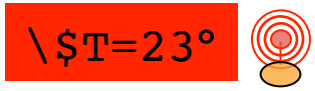
\$ or \& before the data indicates that the data should be logged on a file or server. It is up to the end-device to decide which option

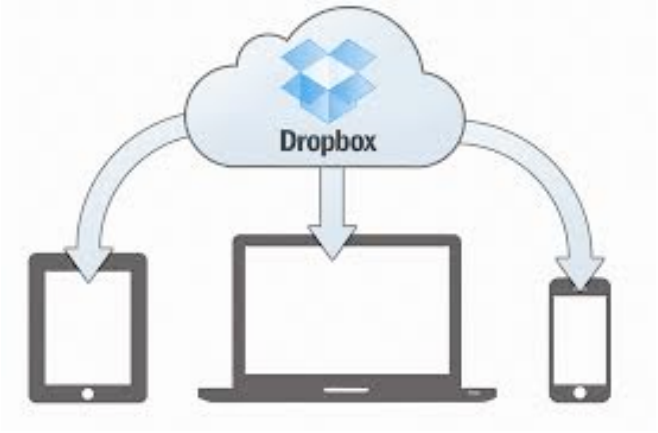


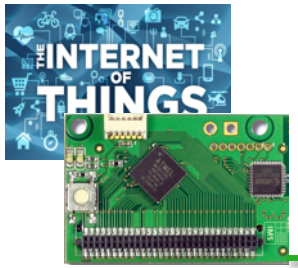
Using Dropbox

- A message starting with '\$' is logged in a file 'telemetry.log' in a folder shared through Dropbox

```
(src=10 seq=0 len=5 SNR=9 RSSI=-54) 2015-11-04T10:14:30.328413> T=23°
(src=10 seq=1 len=7 SNR=8 RSSI=-54) 2015-11-04T10:14:37.443350> T=23.2°
(src=10 seq=2 len=5 SNR=8 RSSI=-53) 2015-11-04T10:16:23.343657> T=24°
...
```

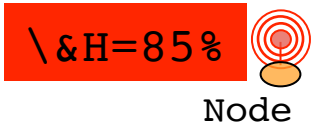


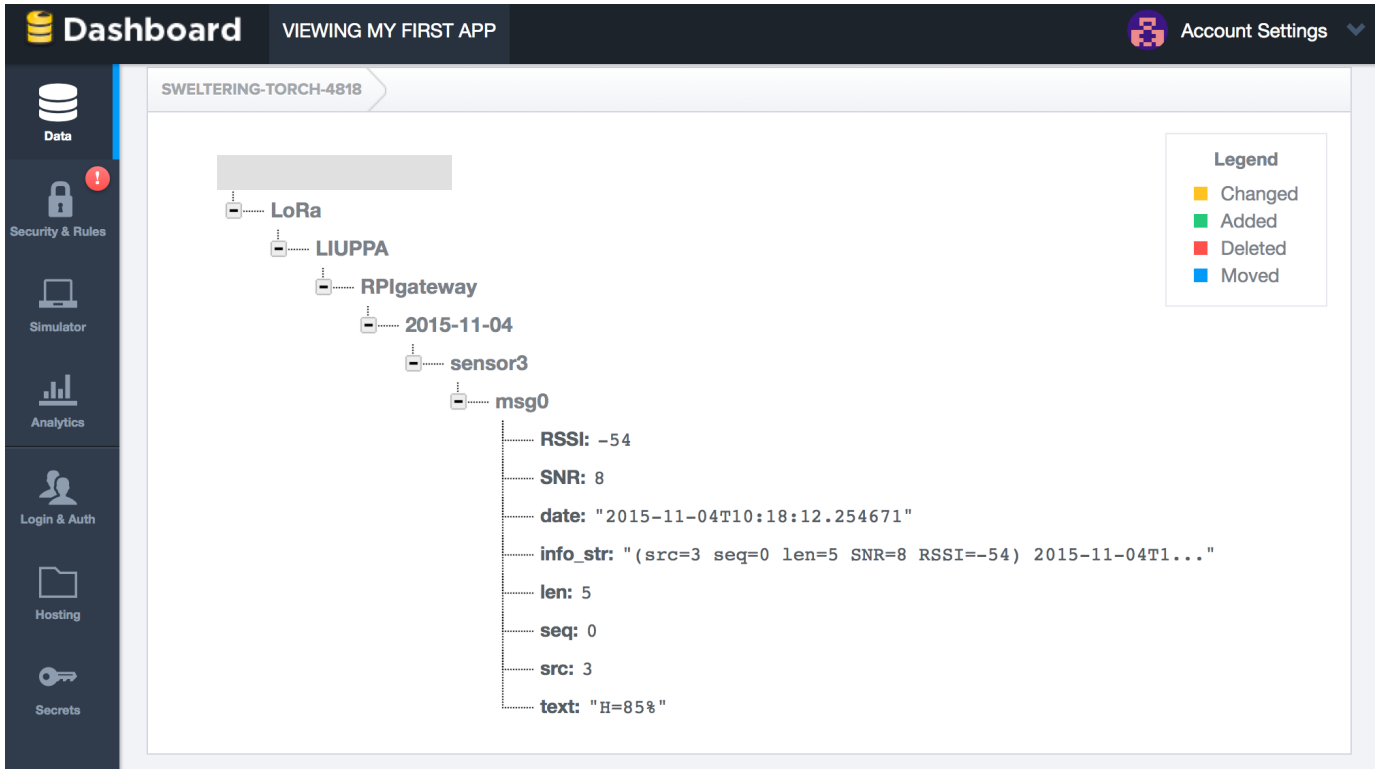


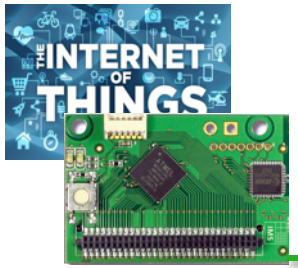


Using Firebase

- A message starting with '\&' is logged in a Firebase database

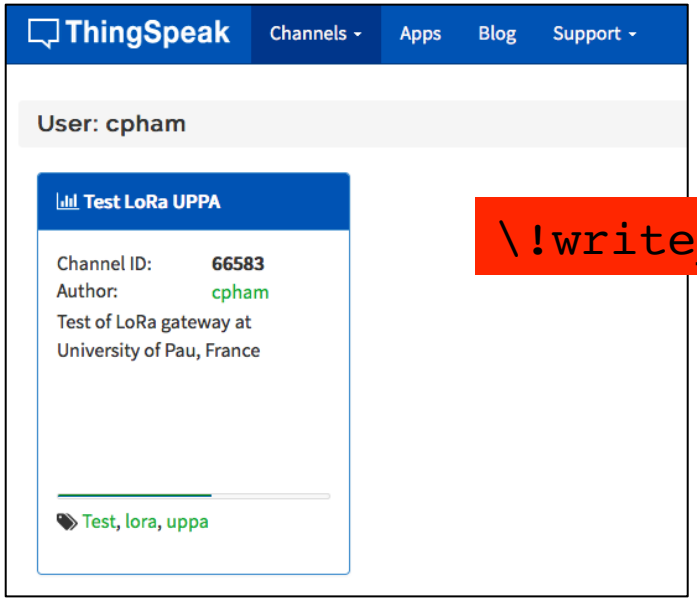






USING ThingSpeak

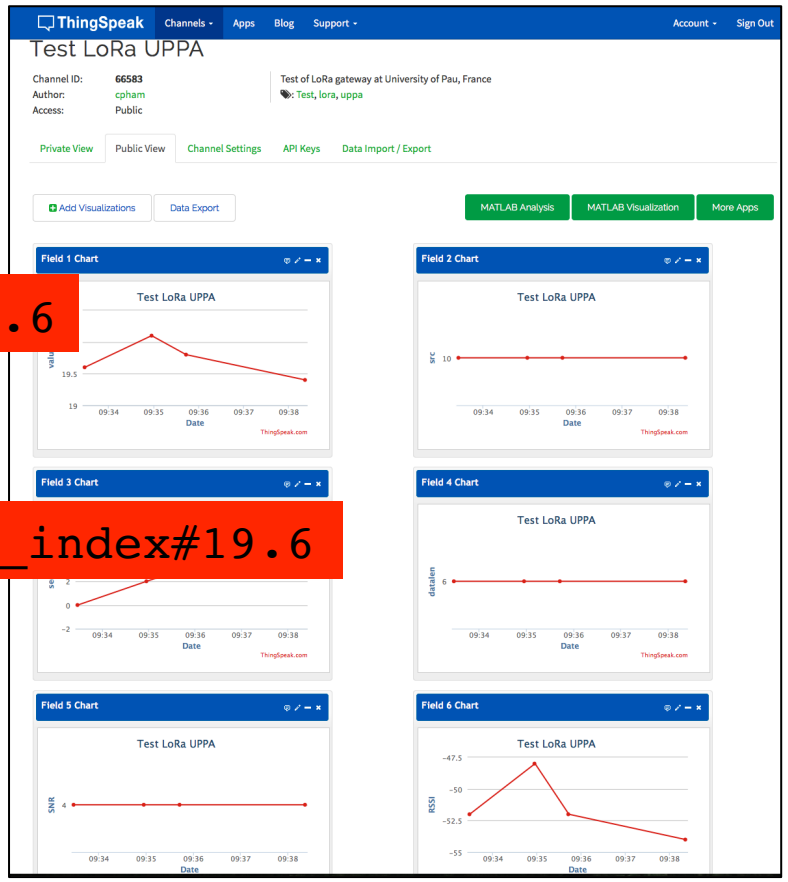
- A message starting with '\!' is logged in a ThingSpeak channel

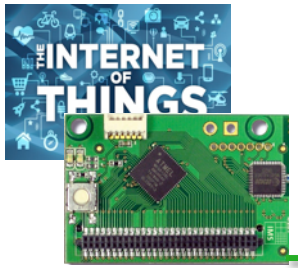


\!##19.6

Node 10

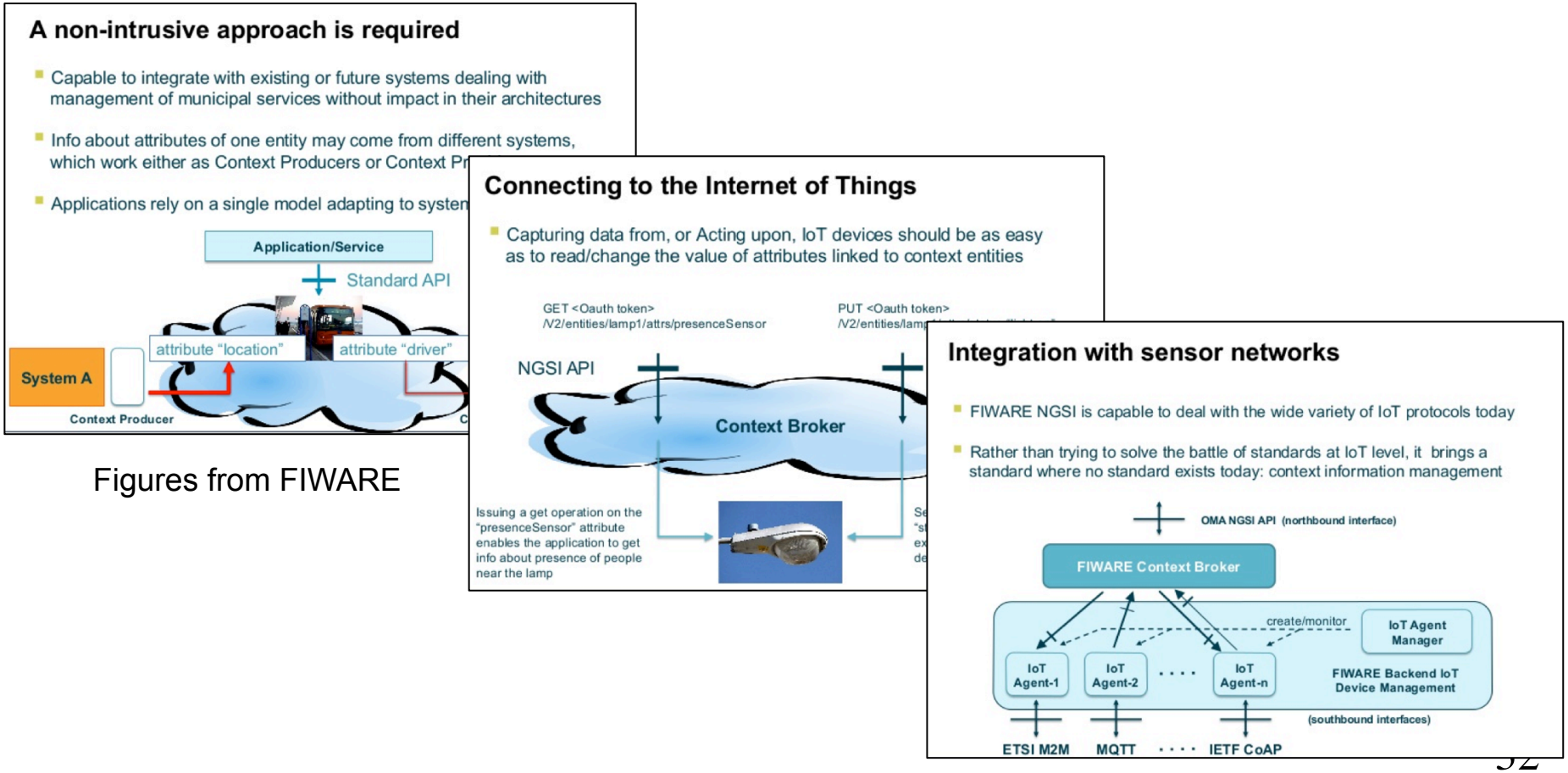
\!write_key#field_index#19.6



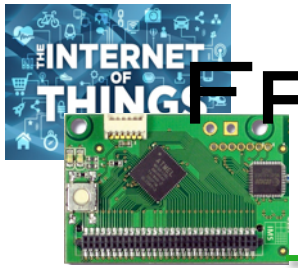


USING FIWARE CLOUD

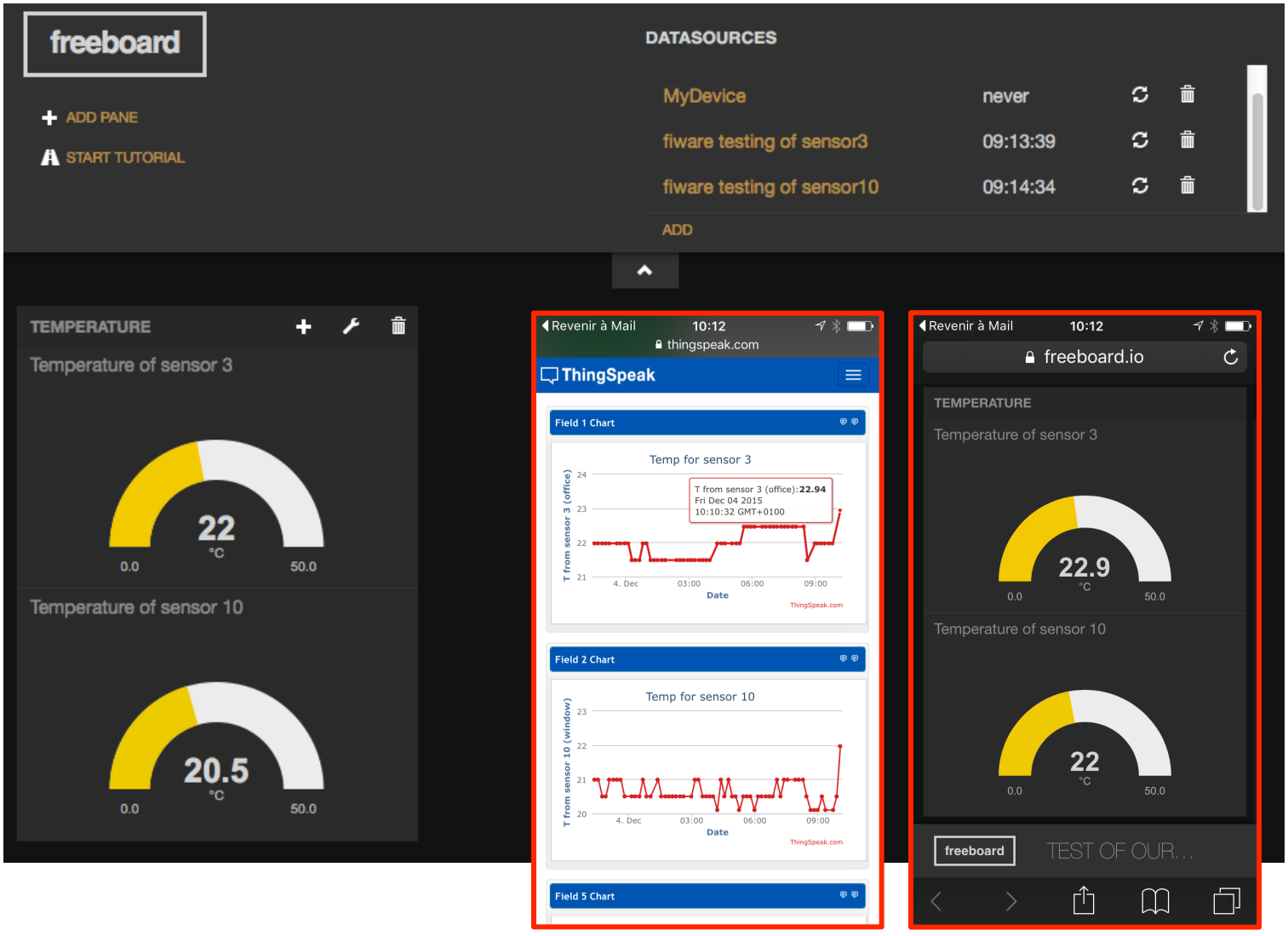
- FIWARE support has been added with EGM script

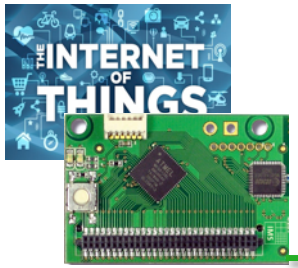


Figures from FIWARE



FREEBOARD IOT CLOUD WITH FIWARE





USING SensorCloud™

SensorCloud DEVICES ACCOUNT CSV UPLOADER DATA SETTINGS MATHENGINE® USE JS VIEWER congduc.pham@univ-pau.fr LOG OUT

liuppa_sensors Clear

Sensor	Channel	Graph
sensor3	temp	☐
sensor6	temp	☑

Metadata Focus

Download Share All Data Last Upload: 3 hours 36 minutes

Zoom: 10s 1m 5m 10m 1h 1d 1w 1M max Timezone: L: Central European

Hide Legend

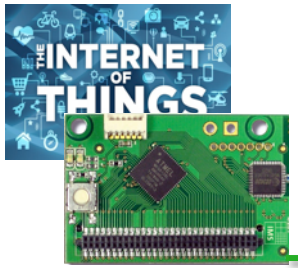
sensor6 - temp

25.4 25.2 25 24.8 24.6 24.4 24.2 24 23.8 23.6

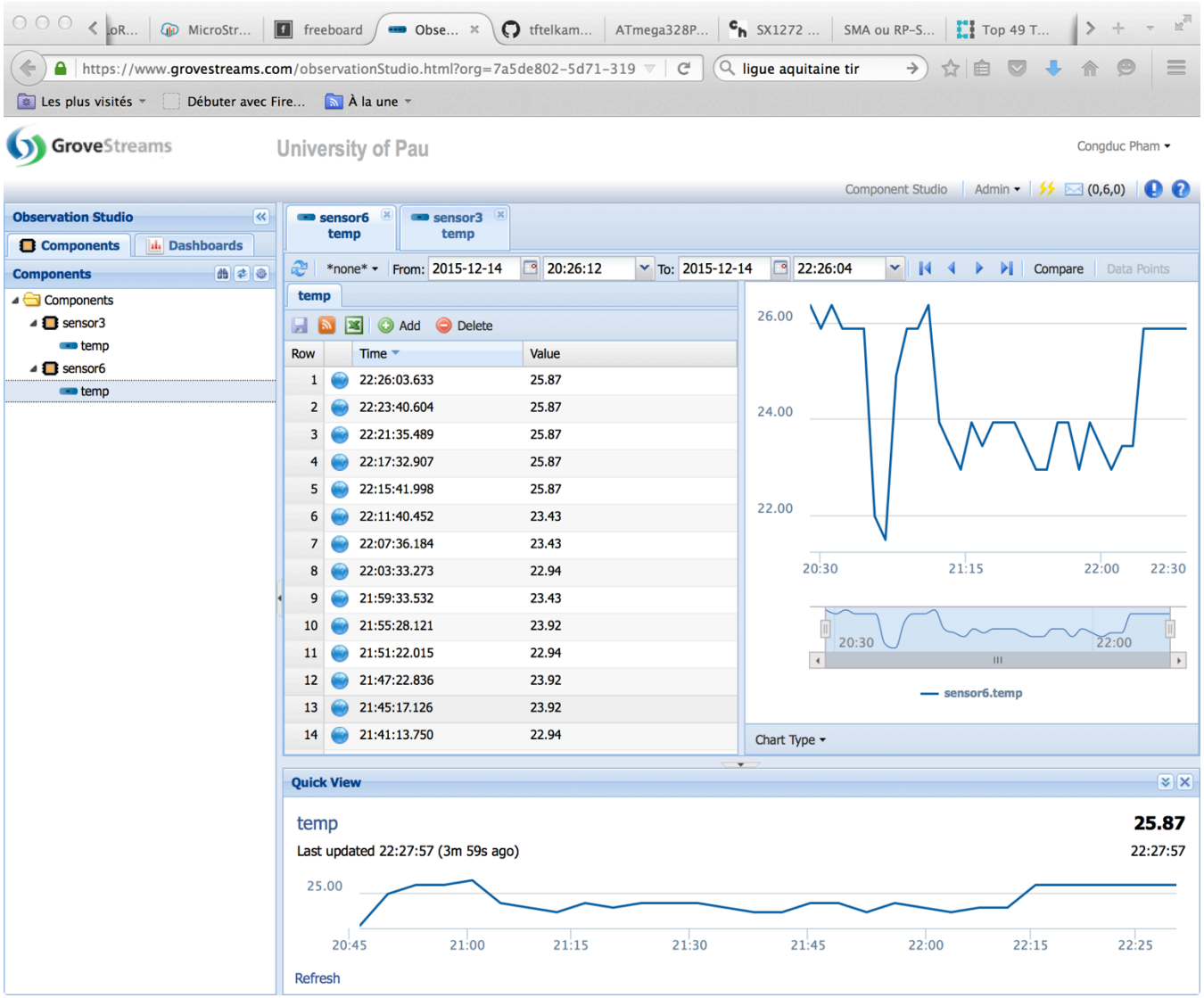
14:30 14:33 14:36 14:40 14:43 14:46 14:50 14:53 14:56 15:00 15:03 15:06 15:10 15:13 15:16 15:20 15:23 15:26

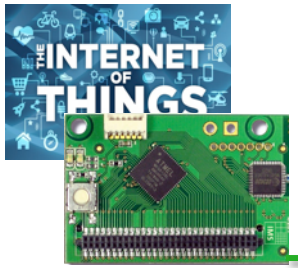
12/13/15 14:28 zoom 12/13/15 15:29

12/11/15 8:33 12/13/15 15:29



Using GroveStreams

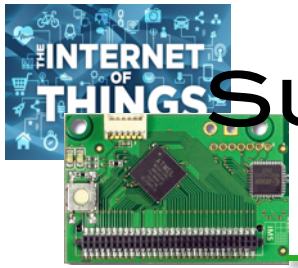




Gateway logs

- Gateway message can also be logged in a local file system or on a Dropbox-shared folder
- The gateway simply prefix important output with '^$' then post-processing can implement the logging service if desired

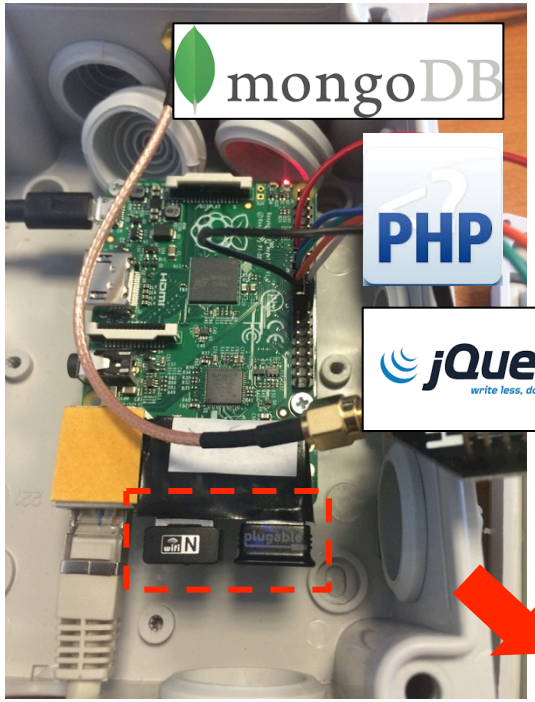
```
2015-11-06T17:32:17.133425> Power ON: state 0
2015-11-06T17:32:17.133425> LoRa mode: 4
2015-11-06T17:32:17.153508> Setting mode: state 0
2015-11-06T17:32:17.168722> Channel CH_10_868: state 0
2015-11-06T17:32:17.193738> Power M: state 0
2015-11-06T17:32:17.204246> Get Preamble Length: state 0
2015-11-06T17:32:17.219356> Preamble Length: 8
2015-11-06T17:32:17.243266> LoRa addr 1 : state 0
2015-11-06T17:32:17.259278> SX1272/76 configured as LR-BS. Waiting RF input for transparent RF-serial bridge
2015-11-06T17:32:28.954388> Parsing command
2015-11-06T17:32:28.969103> /@M1#
2015-11-06T17:32:38.912402> Set LoRa mode to 1
2015-11-06T17:32:38.929712> LoRa mode: state 0
```

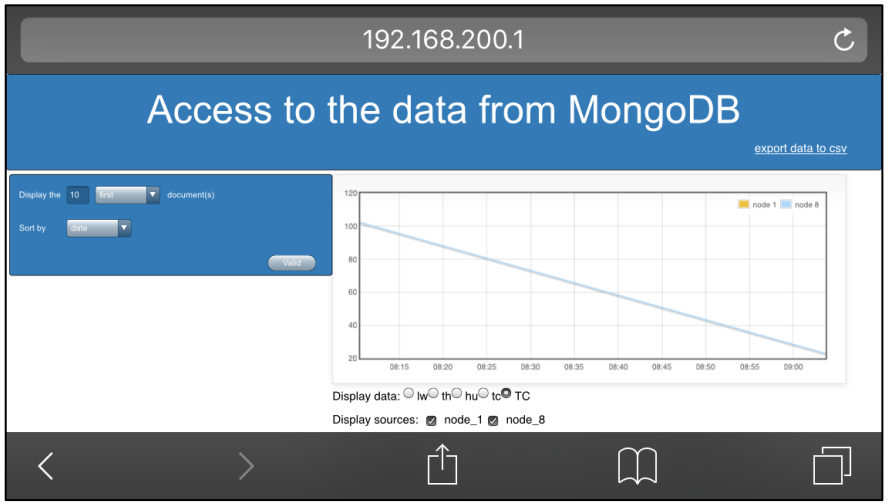


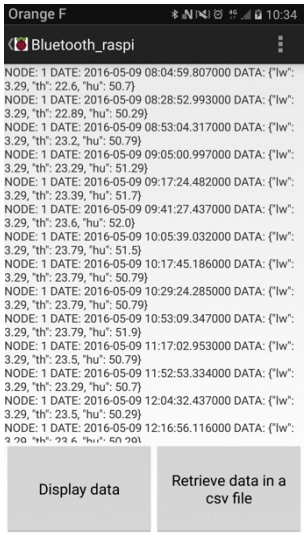
Summary of output prefixes for post-processing

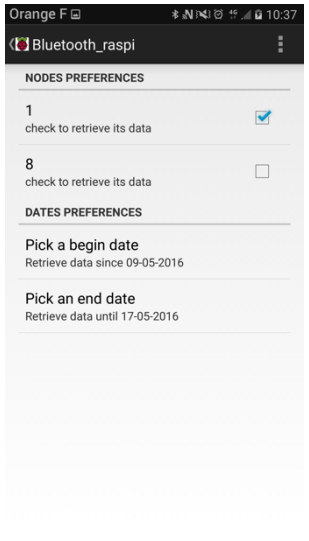
- Control information
  - '^p': Give information on the last received packet
  - '^$': Gateway output to be logged in a file
  - '^&': Gateway output to be logged on Firebase
- End-device messages
  - '\$': end-device msg to be logged in a file
  - '\&': end-device msg to be logged on Firebase
  - '\!': end-device msg for a ThingSpeak channel
- …just implement your own prefixes and cloud services…
- …or change the behavior of currently defined prefixes

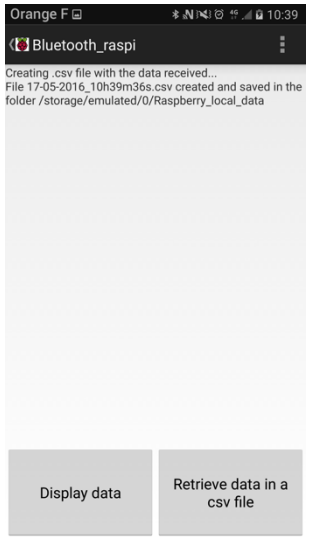
RUNNING WITHOUT INTERNET ACCESS





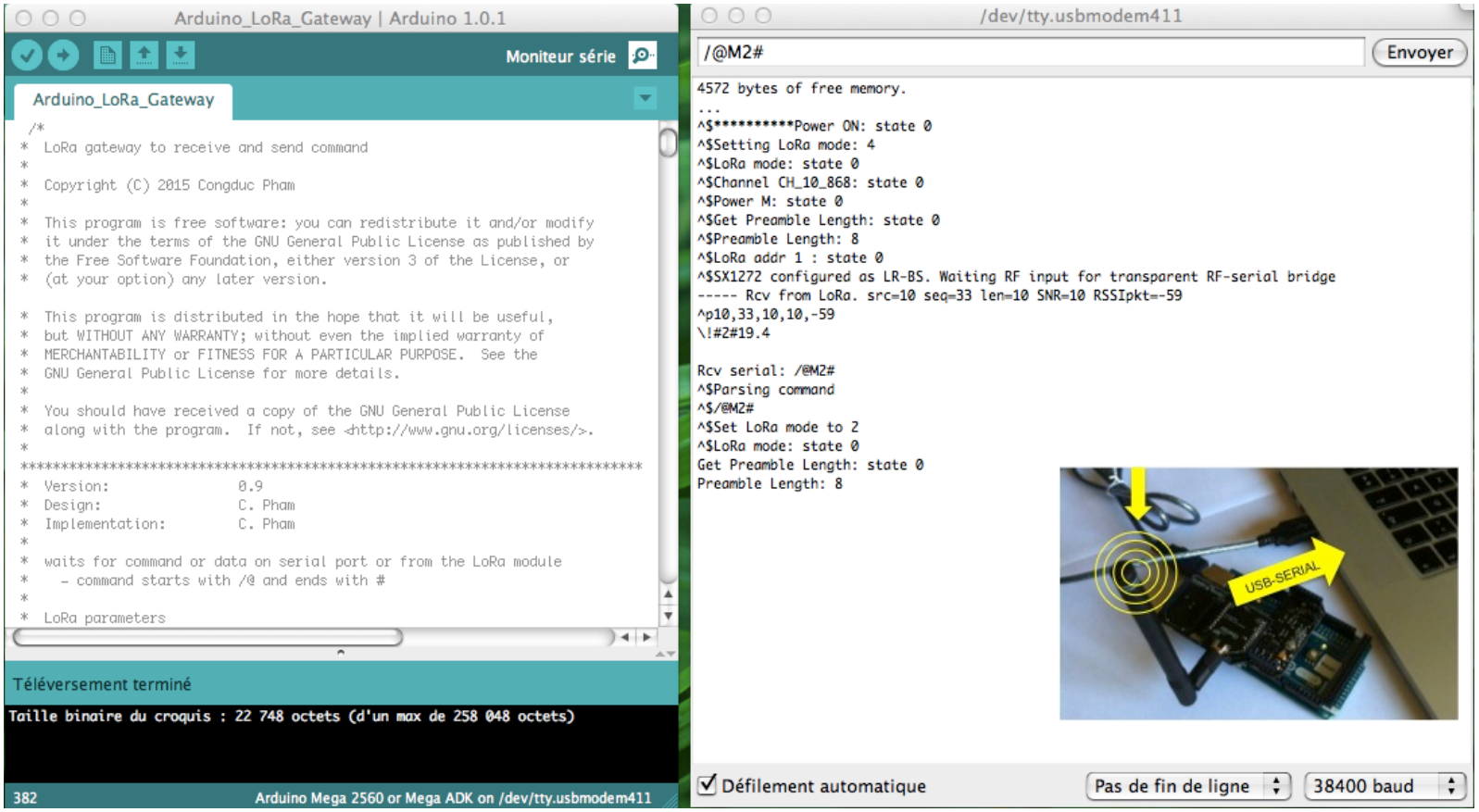


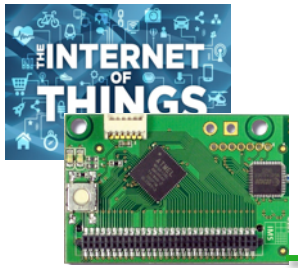




Gateway for on-the-go tests

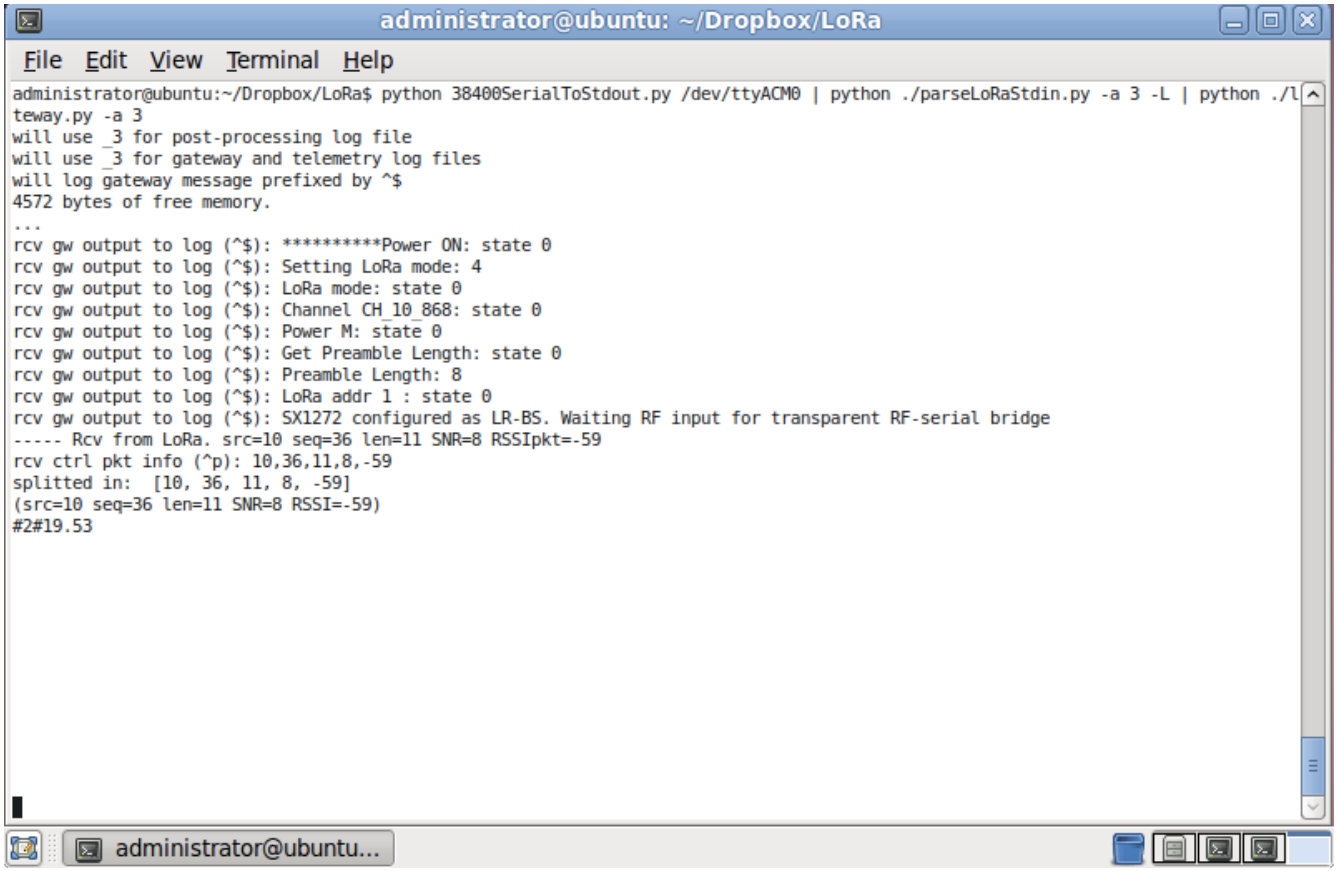
- Use an Arduino board w/laptop as a transparent radio bridge

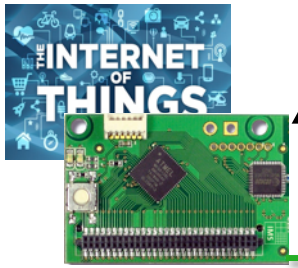




Adding full post-processing

- Forward serial data to stdout, then use post-processing python scripts unchanged

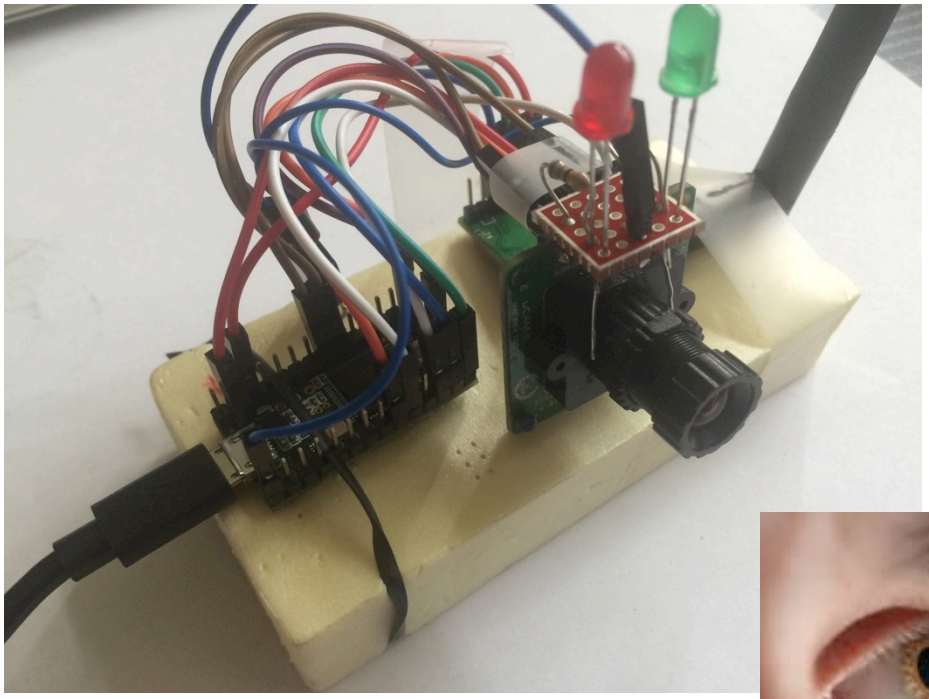


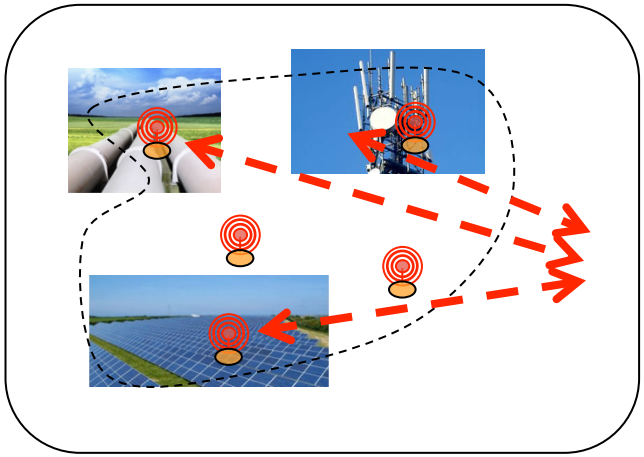


ADVANCED FEATURES (1): ENCRYPTION

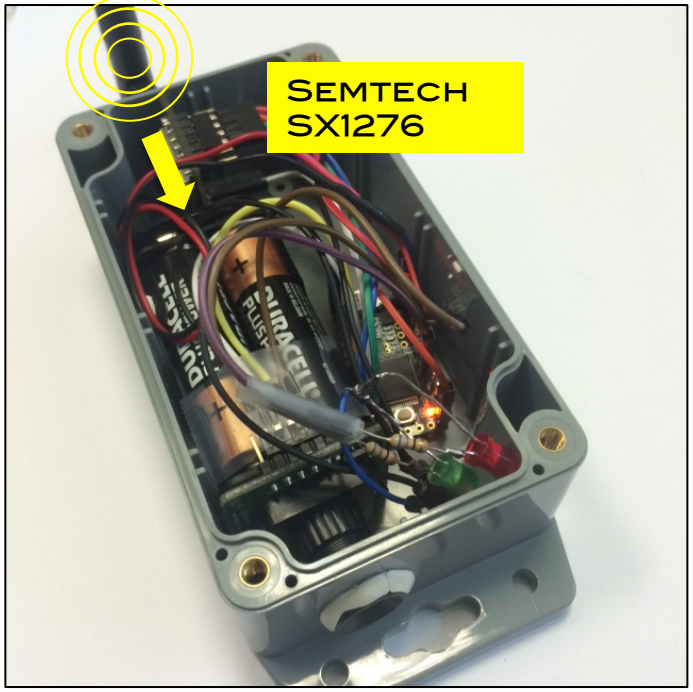
- The end-device encrypts the payload with AES-128
- The gateway forwards the encrypted payload to the post-processing stage (python)
- The post-processing stage can either be configured to
  - Decrypt the data and process decrypted data as usual. Need to store the AES key in the post-processing script
  - Push encrypted data to a client-managed server
- We demonstrate the flexibility of our architecture by implementing AES encrypt/decrypt just by adding a few lines of python code

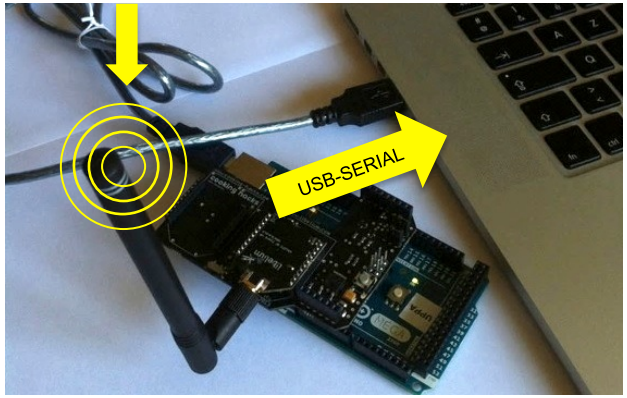
Advanced feature (2): image transmission

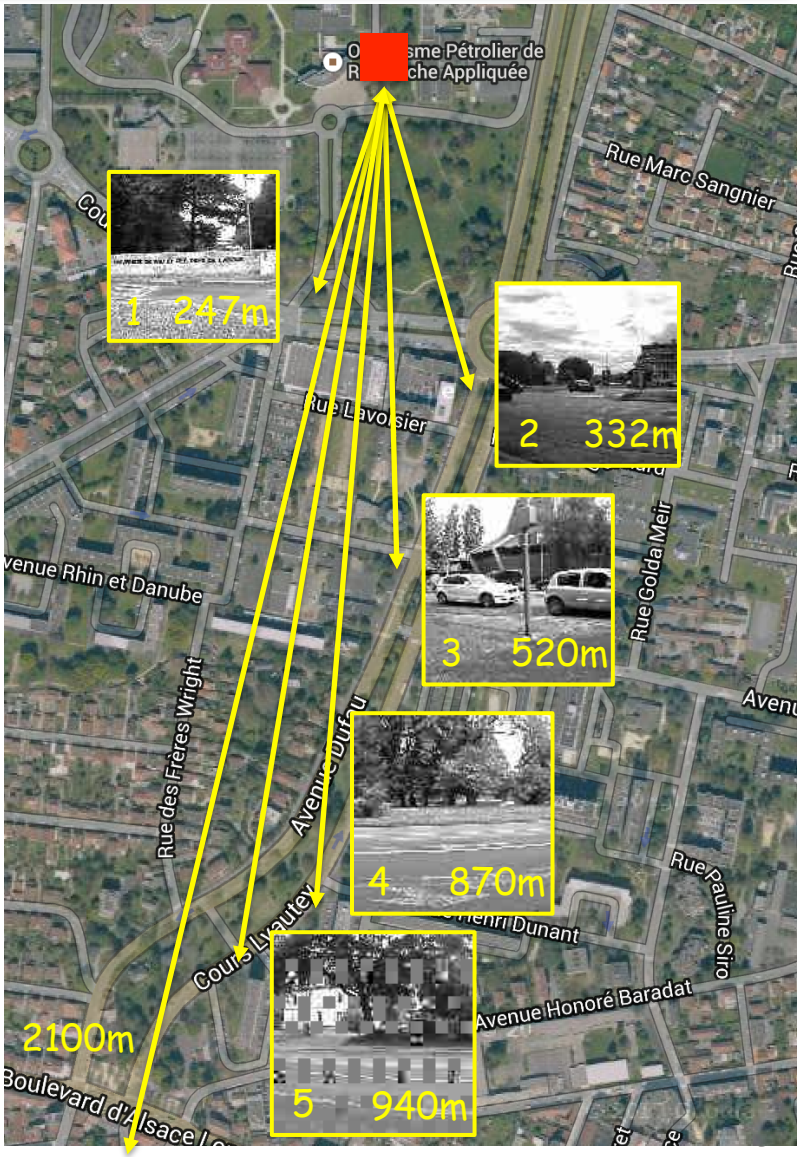




Long-range version of our image sensor







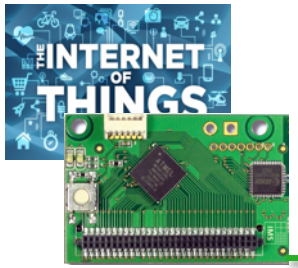


IMAGE ENCODING

FF 55 0003 00 14 0059

Original 128x128

Collaboration with CRAN laboratory, Nancy, France. Very robust image encoding techniques against packet losses

```
FF 55 0003 00 14 0059 00 00 CD E5 43 C2 3A C8 1B D0 A0 E7 2A C7 D1 6C EE 60 FD 5A A8
AB 53 87 C0 29 D4 85 01 43 B0 95 C5 93 7C A8 65 9C 1C 13 F4 16 CA 02 27 08 EB 4A 38
AB 06 06 22 8E 74 61 EA 18 4A BD EB 77 0D 46 D9 8F 8B 05 92 E1 EE D9 9A 98 D4 A3 3F
83 16 E0 D0 6B 87 14 48 58 4E 13 7F FF 55 0003 01 14 0056 00 20 D5 BE 0B 70 DF 3C 0F
1B 48 1                                                  3E 88 36 B6 02 77 CE 9F F8 32
42 18 A                                                  47 87 EC 69 61 60 7D AB 96 B9
AC CD B                                                  46 36 15 FF 55 0003 02 14 0057
00 32 E                                                  E4 9E A9 1E CD EB 45 36 B6 12
18 FA 6                                                  CB DC 5D 06 E4 29 A3 B6 70 C5
E0 F9 C                                                  FB 54 6A 08 A8 1A 70 9E A2 DA
50 AC 2                                                  2 33 69 5E 96 2E 7E 7F EB 8D 59
88 9F 2                                                  73 FC CF BF 36 DC B1 86 22 98
02 93 C                                                  87 7B 43 80 42 5B E9 4B 78 58
CF 59 F                                                  0058 00 5B D9 2E 88 09 71 6D 35
C4 7A E                                                  F5 A7 E6 2B 39 20 E6 07 67 34
CB 1F B                                                  CE 95 73 9A 75 B5 8E A1 F0 1C
2E DF 1                                                  12 DB 20 E2 C7 FF 55 0003 05 14
0058 00                                                  D7 CD 4A 0D A2 B3 9B D2 DC 91 EA
63 9A 0                                                  9E 84 ED 54 4A ED 62 CC 8D 6D
71 66 6                                                  88 EC EA 01 1C 73 E0 4A 3D 94
BE FC 14 C6 3F FF 55 0003 06 14 0056 00 84 D3 27 D6 C1 A4 52 F8 F2 4C 83 6F 05 6E F7
91 32 F2 3F D7 9B 09 3E D6 F6 72 AA 63 0C C1 DC 48 E2 8F DD 55 53 80 F0 94 C3 D4 5D
8D 27 F2 EE C2 80 F3 3F 55 8A 7F 47 BA 67 FA CB 17 16 EE 24 A2 72 FA E0 BA CA C7 0F
C6 07 0F 11 36 69 FA 7A FC 82 66 33 49 43 0058 .. ..
```

FF 55: framing bytes for image packet

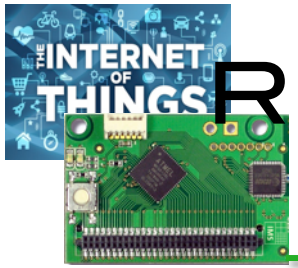
0003: source node address

00: image packet sequence number

14: quality factor (between 0 and 100)

0059: data length

00 00 CD E5 … : encoded image data

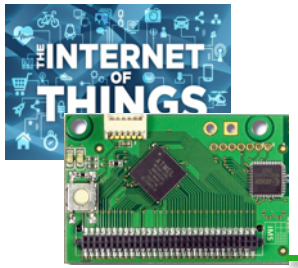


Receiving images at LoRa gateway

- Current post-processing tasks are
  - look for framing bytes 0xFF0x55
  - get information on source node
  - create new entry for image source node if needed
  - save all image packets into a file that can be shared/ uploaded and displayed at client side

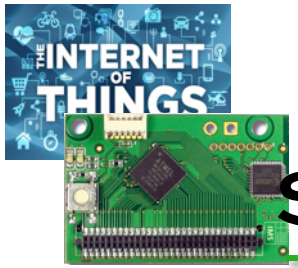
```
FF 55 0003 00 14 0059 00 00 CD E5 43 C2 3A C8 1B D0 A0 E7 2A C7 D1 6C EE 60 FD 5A A8
AB 53 87 C0 29 D4 85 01 43 B0 95 C5 93 7C A8 65 9C 1C 13 F4 16 CA 02 27 08 EB 4A 38
AB 06 06 22 8E 74 61 EA 18 4A BD EB 77 0D 46 D9 8F 8B 05 92 E1 EE D9 9A 98 D4 A3 3F
83 16 E0 D0 6B 87 14 48 58 4E 13 7F FF 55 0003 01 14 0056 00 20 D5 BE 0B 70 DF 3C 0F
1B 48 13 93 9E 73 45 03 94 1E 9B A5 0B BB CF 52 04 4D 3E 88 36 B6 02 77 CE 9F F8 32
42 18 A8 F8 AA 79 9E 35 20 A0 A5 98 FE 32 16 1E 97 90 47 87 EC 69 61 60 7D AB 96 B9
AC CD B4 B8 5D B9 88 23 60 20 32 44 30 8A DA .. ..
```

`tmp_0-node0003-cam#0-Q20.dat`



Python code

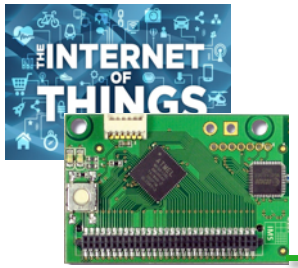
```
if (ch == '\xFF'):
                ch=sys.stdin.read(1)
                if (ch >= '\x50' and ch <= '\x54'):
                                cam_id=ord(ch)-0x50;
                                src_adr_msb = ord(sys.stdin.read(1))
                                src_adr_lsb = ord(sys.stdin.read(1))
                                src_adr = src_adr_msb*256+src_adr_lsb
                                seq_num = ord(sys.stdin.read(1))
                                Q = ord(sys.stdin.read(1))
                                data_len = ord(sys.stdin.read(1))

                                if (src_adr in nodeL):
                                                #already in list
                                                #get the file handler
                                                theFile=fileH[src_adr]
                                else:
                                                #new image packet from this node
                                                nodeL.append(src_adr)
                                                filename = "tmp_%d-node#%.4d-cam#%d-Q%d.dat" % (rcvImg,src_adr,cam_id,Q)
                                                print "first pkt from node %d" % src_adr
                                                print "creating file %s" % filename
                                                theFile=open(filename,"w")
                                                # associates the file handler to this node
                                                fileH.update({src_adr:theFile})
                                                rcvImg=rcvImg+1
                                                t = Timer(35, image_timeout)
                                                t.start()
                                                #log only the first packet and the filename
                                                f=open(os.path.expanduser("~/Dropbox/LoRa/image.log"),"a")
                                                f.write(info_str+' ')
                                                now = datetime.datetime.now()
                                                f.write(now.isoformat()+'> ')
                                                f.write(filename)
                                                f.close()

                                print "pkt %d from node %d data size is %d" % (seq_num,src_adr,data_len)
                                print "write to file"

                                theFile.write(format(data_len, '04X')+' ')

                                for i in range(1, data_len):
                                                ch=sys.stdin.read(1)
                                                # sys.stdout.write(hex(ord(ch)))
                                                print (hex(ord(ch))),
                                                theFile.write(format(ord(ch), '02X')+' ')

                                print "End"
                                sys.stdout.flush()
                                theFile.flush()
                                continue
```



Unlicensed sub-GHz spectrum constraints

- It is shared medium so long-range transmission in dense environments can create lots of interference!
- Activity time is constrained from 0.1% to 1% duty-cycle depending on frequency: 3.6s to 36s/hour

Band	Edge Frequencies		Field / Power	Spectrum Access	Band Width
	Fc-	Fc+			
g(Note 7)	865 MHz	868 MHz	+6.2 dBm /100 kHz	1 % or LBT AFA	3 MHz
g(Note 7)	865 MHz	870 MHz	-0.8 dBm / 100 kHz	0.1% or LBT AFA	5 MHz
g1	868 MHz	868.6	14 dBm	1 % or LBT AFA	600 kHz
g2	868.7 MHz	869.2 MHz	14 dBm	0.1% or LBT AFA	500 kHz
g3	869.4 MHz	869.65 MHz	27 dBm	10 % or LBT AFA	250 kHz
g4	869.7 MHz	870 MHz	7 dBm	No requirement	300 kHz
g4	869.7 MHz	870 MHz	14 dBm	1 % or LBT AFA	300 kHz



Time on Air of LoRa Transmissions

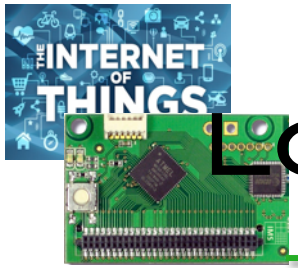
With 1% radio activity time…

				time on air in second for payload size of					
LoRa mode	BW	CR	SF	5 bytes	55 bytes	105 bytes	155 Bytes	205 Bytes	255 Bytes
1	125	4/5	12	0.95846	2.59686	4.23526	5.87366	7.51206	9.15046
2	250	4/5	12	0.47923	1.21651	1.87187	2.52723	3.26451	3.91987
3	125	4/5	10	0.28058	0.69018	1.09978	1.50938	1.91898	2.32858
4	500	4/5	12	0.23962	0.60826	0.93594	1.26362	1.63226	1.95994
5	250	4/5	10	0.14029	0.34509	0.54989	0.75469	0.95949	1.16429
6	500	4/5	11	0.11981	0.30413	0.50893	0.69325	0.87757	1.06189
7	250	4/5	9	0.07014	0.18278	0.29542	0.40806	0.5207	0.63334
8	500	4/5	9	0.03507	0.09139	0.14771	0.20403	0.26035	0.31667
9	500	4/5	8	0.01754	0.05082	0.08154	0.11482	0.14554	0.17882
10	500	4/5	7	0.00877	0.02797	0.04589	0.06381	0.08301	0.10093

Range

Throughput

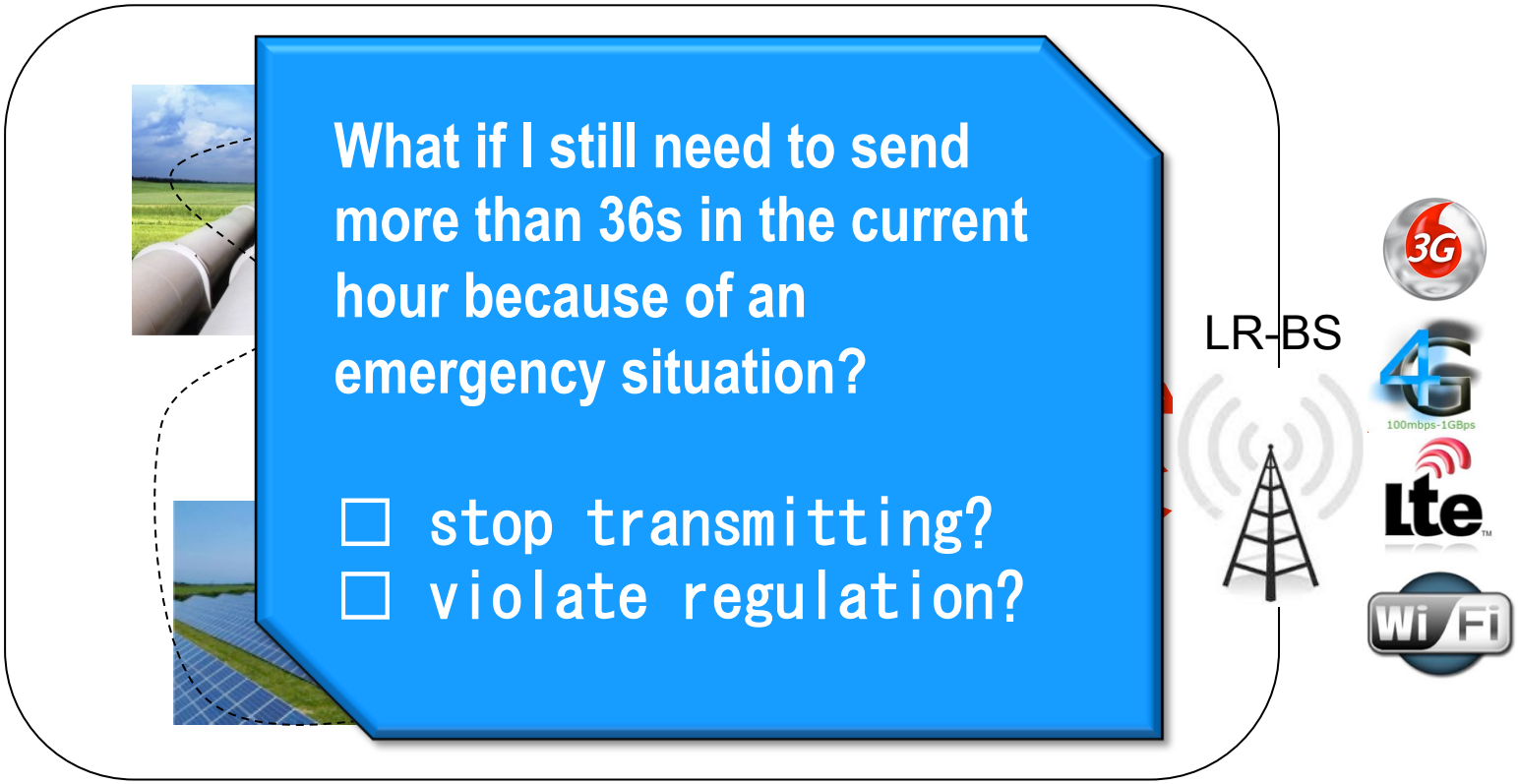
…can send 13 50-byte msg/h or 59 50-byte msg/h

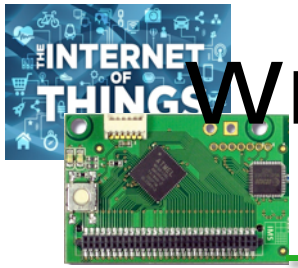


LoRa Devices & gateways

Regulations stipulate that radio activity duty-cycle should be enforced at devices and that end-users should not be able to modify it « easily ».

LoRaWAN specification from LoRa Alliance is a first attempt to standardize LoRa networks but no issues on quality of service.





WHAT IF I WANT TO TRANSMIT IMAGES?

				time on air in second for payload size of					
LoRa mode	BW	CR	SF	5 bytes	55 bytes	105 bytes	155 Bytes	205 Bytes	255 Bytes
1	125	4/5	12	0.95846	2.59686	4.23526	5.87366	7.51206	9.15046
2	250	4/5	12	0.47923	1.21651	1.87187	2.52723	3.26451	3.91987
3	125	4/5	10	0.28058	0.69018	1.09978	1.50938	1.91898	2.32858
4	500	4/5	12	0.23962	0.60826	0.93594	1.26362	1.63226	1.95994
5	250	4/5	10	0.14029	0.34509	0.54989	0.75469	0.95949	1.16429
6	500	4/5	11	0.11981	0.30413	0.50893	0.69325	0.87757	1.06189
7	250	4/5	9	0.07014	0.18278	0.29542	0.40806	0.5207	0.63334
8	500	4/5	9	0.03507	0.09139	0.14771	0.20403	0.26035	0.31667
9	500	4/5	8	0.01754	0.05082	0.08154	0.11482	0.14554	0.17882
10	500	4/5	7	0.00877	0.02797	0.04589	0.06381	0.08301	0.10093

Optimized image encoding at medium quality: 16384b down to 1366b (ratio 12).

Will generate 7 pkts using 250 max payload

7*9.15= 64.05s

7*1.96 = 13.72s

Deploying your LoRa network

Operator-based approach (

subscriptio

Whatever the deployment approach, the gateway knows how many devices are deployed by a given organization

Our proposition is to view all device' activity time in a global manner, with the gateway taking care of radio time usage consistency

Privately-based

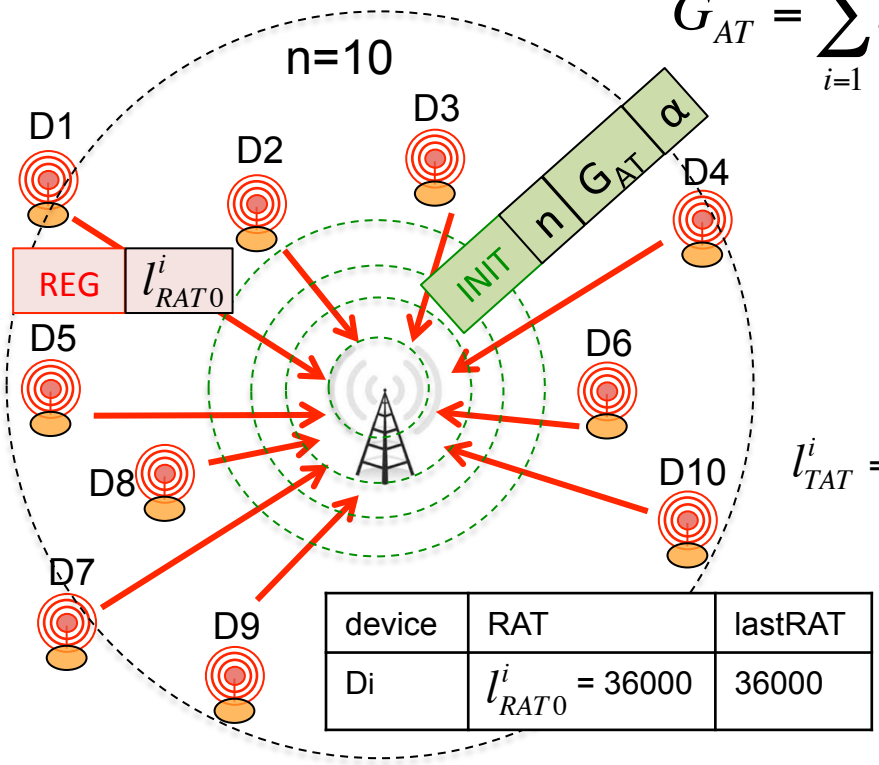
oach

-Tech Conduit

Kerlink
loT Station

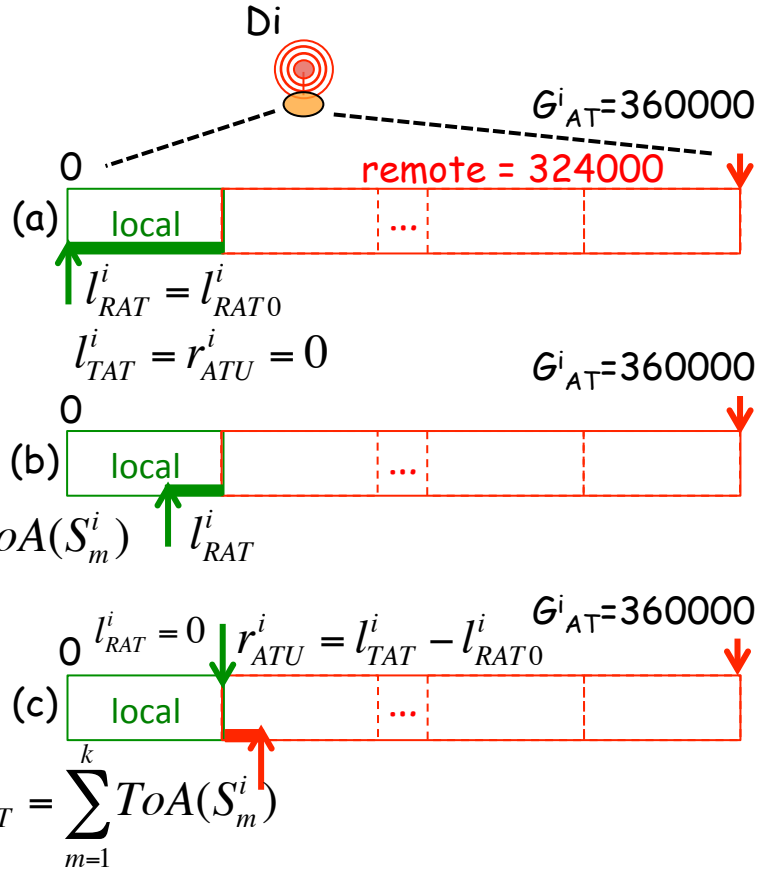
And much more...

Long-range Activity Sharing (LAS)



device	RAT	lastRAT
Di	l^i_{RAT0} = 36000	36000

$$G_{AT} = \sum_{i=1}^{n} l^i_{RAT0}$$

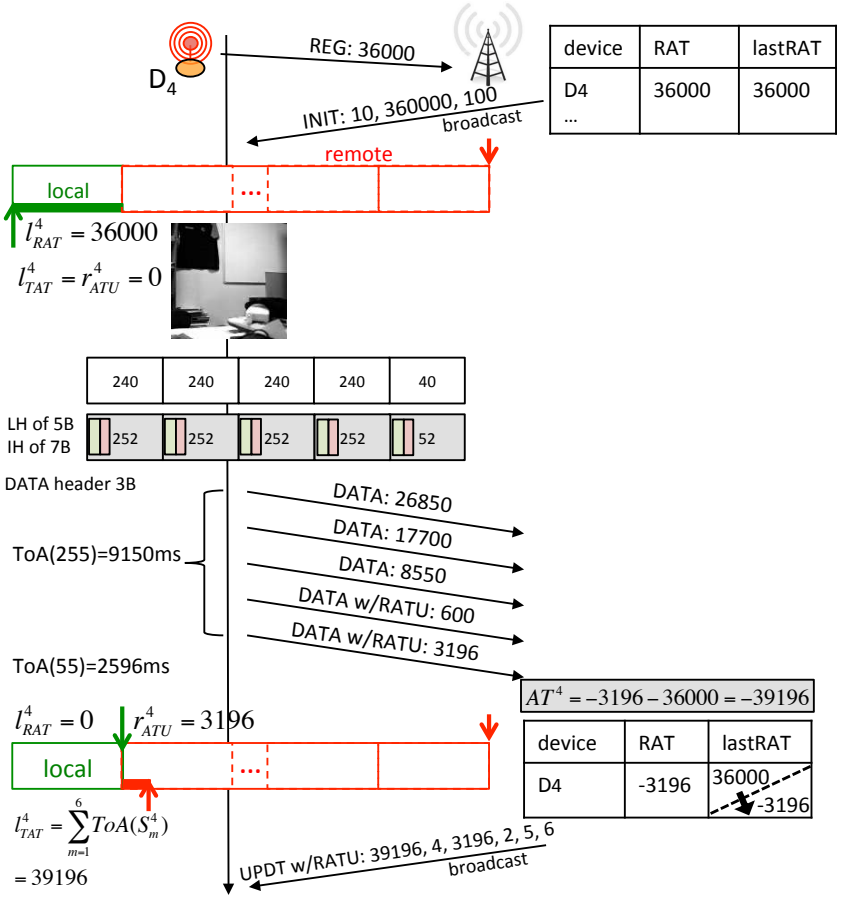


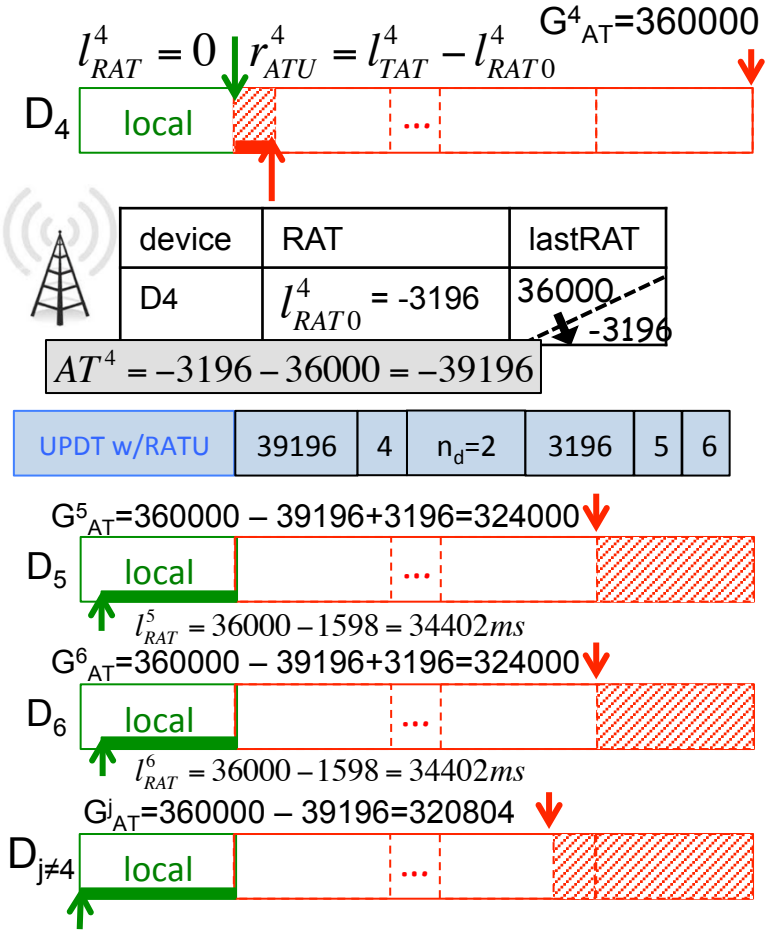
$$l^i_{TAT} = \sum_{m=1}^{k} ToA(S^i_m)$$

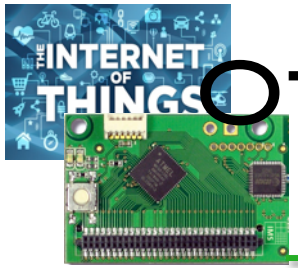
$$l^i_{TAT} = \sum_{m=1}^{k} ToA(S^i_m)$$

A device can transmit more if needed, provided that other devices will decrease their radio activity time accordingly.

Distributing Remote Activity Time Usage

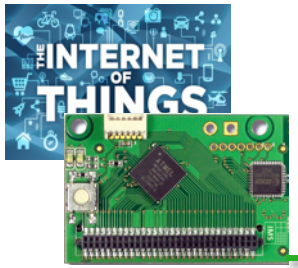






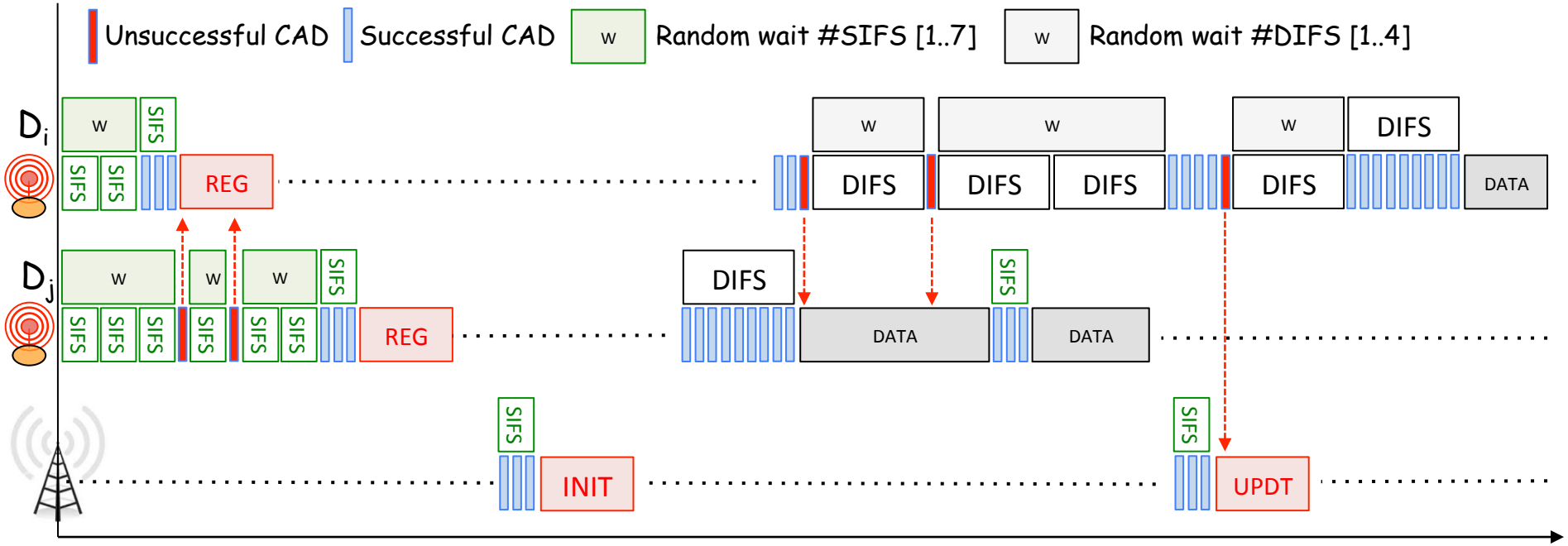
OTHER ISSUES TO TAKE INTO ACCOUNT

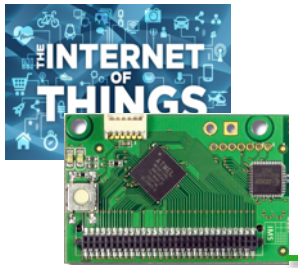
- Minimise the number of UPDT messages sent by the gateway because the gateway's radio time is also limited
  - UPDT can have cumulative behavior if no remote activity time has been used
- Support sleep periods of end-devices
  - The network is synchronized for control messages (REG, INIT, UPDT). UPDT msg that can not use cumulative behavior are queued for transmission at next transmission slot. At rcv, UPDT have to be applied sequentially.
- Maintain (loose) synchronization
  - If no UDPT are scheduled, the gateway periodically sends a BEACON. Clock drift is limited to a BEACON period
- Dynamic insertion of new end-devices
  - New devices can either stay out of the managed pool (then only 36s of activity time/h is allowed), or join by waiting for the next UPDT/BEACON msg
  - Every hour, end-devices decide if they want to join the pool or not
- Give priority to control msg
  - SIFS/DIFS mechanism are implemented using LoRa Channel Activity Detection
- Avoid interleaving of several image transmissions
  - Use DIFS for first image packet, then SIFS
- Improve LoRa network efficiency
  - Move from pure ALOHA to CSMA mechanism with CAD+RSSI tests prior to any transmission



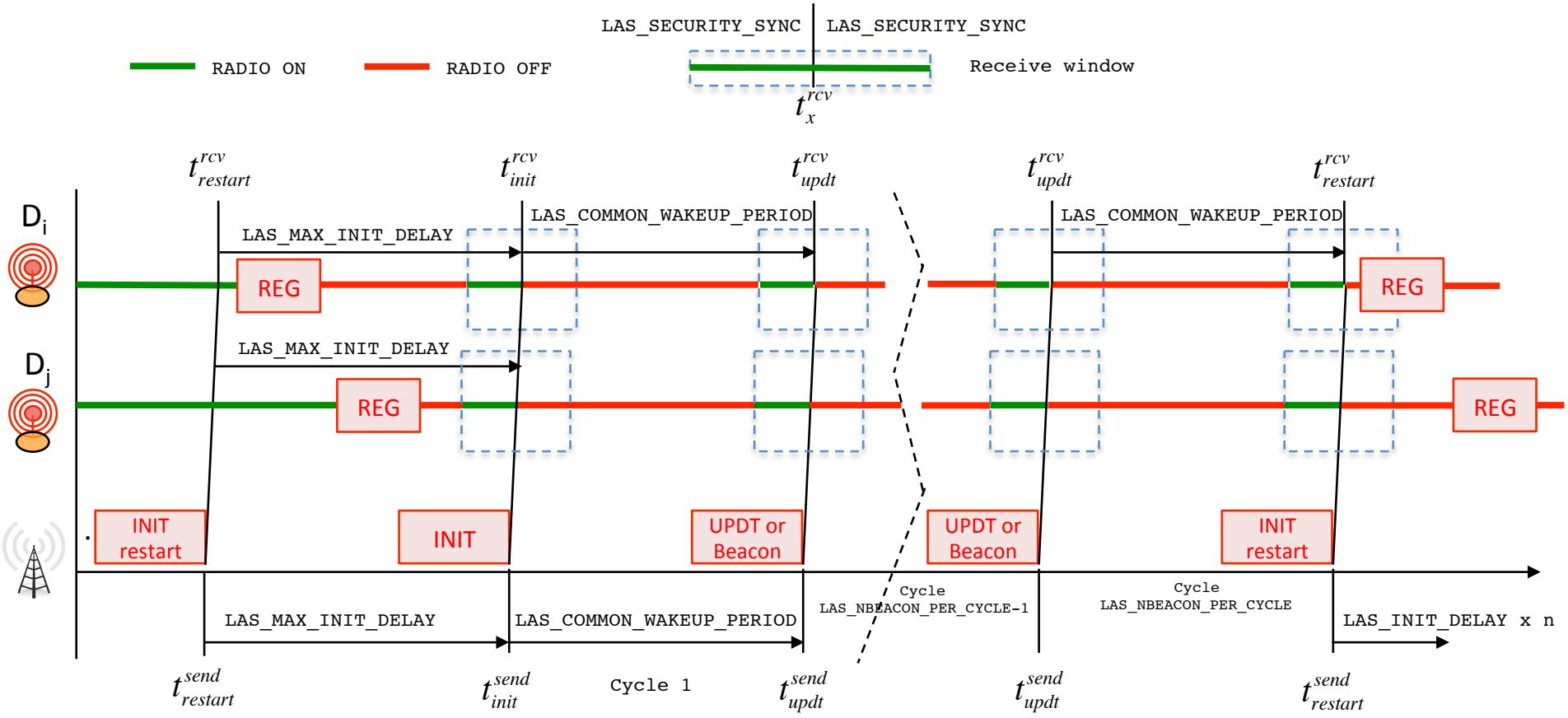
Enhanced channel access

- Current LoRa networks are ALOHA system
- Our SX1272 modified library adds CSMA-like channel access and RSSI check before transmission

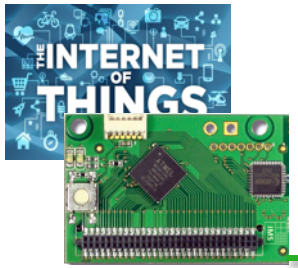




Summary of gateway-device scheduling

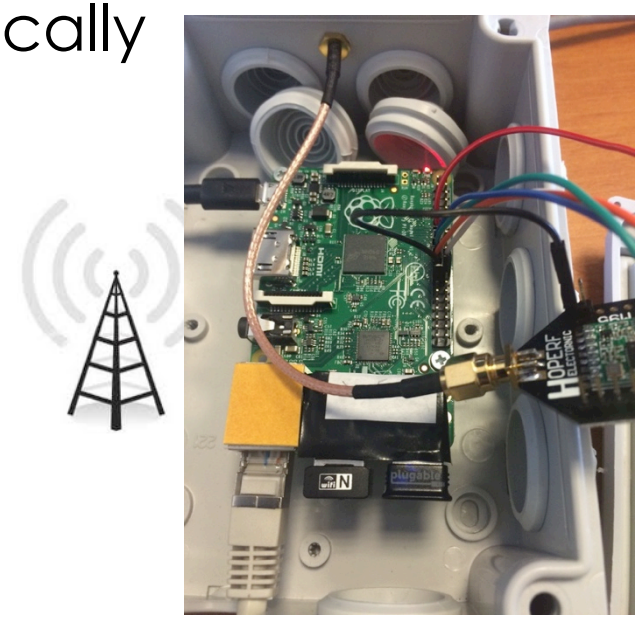


1 hour

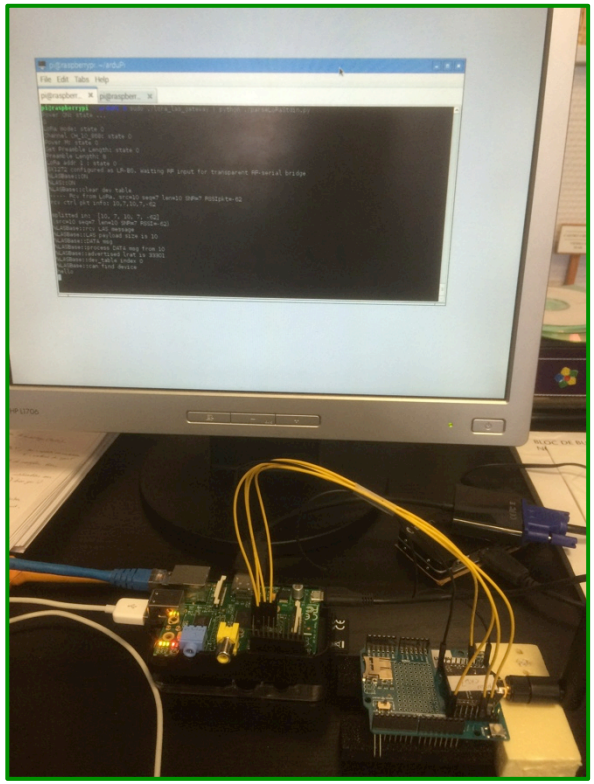


Implementations & libraries

- Own gateway based either Raspberry PI or Arduino MEGA/Due with gateway LAS services
- C/C++ Library for end-devices to send data packets under LAS services. Radio duty-cycling and interactions with LAS gateway are performed automatically

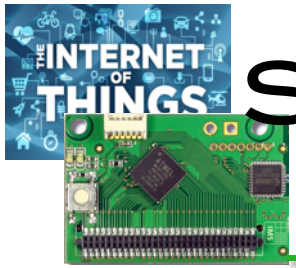


Gateway startup

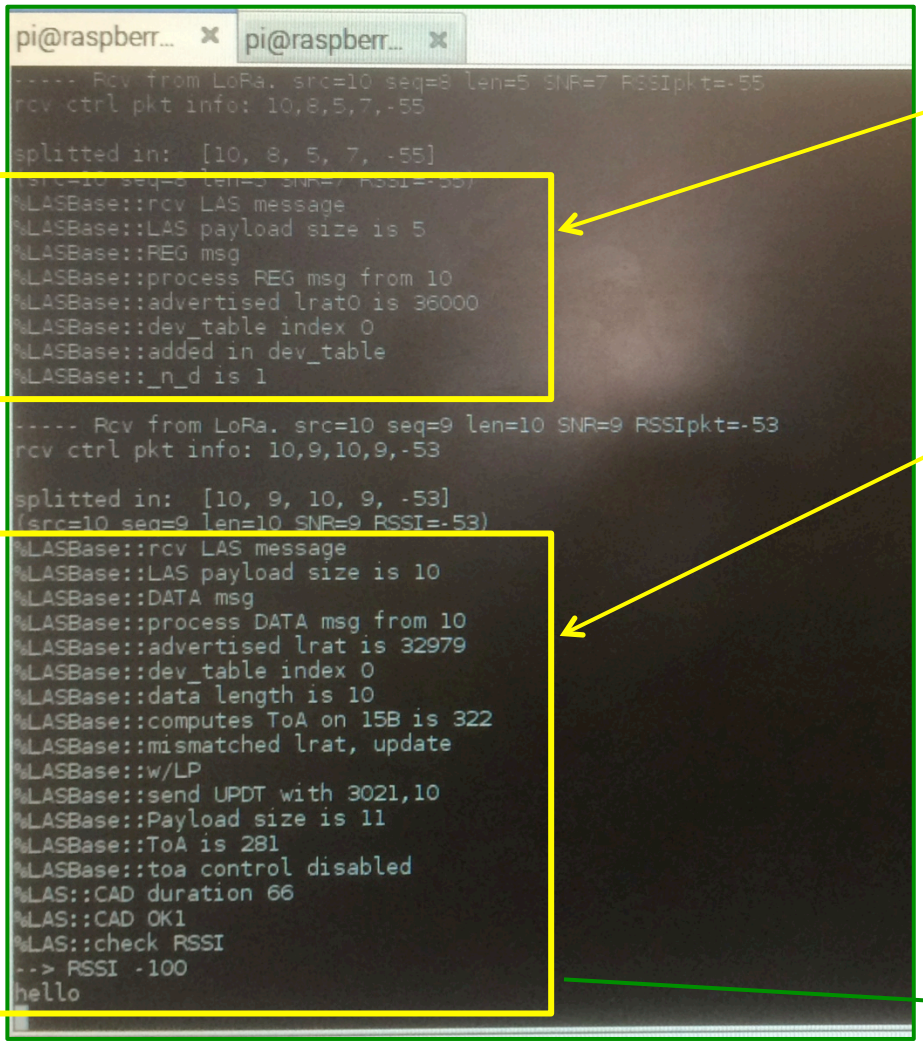


```
pi@raspberr...   pi@raspberr...
Power ON: state ...
0
LoRa mode: state 0
Channel CH_10_868: state 0
Power M: state 0
Get Preamble Length: state 0
Preamble Length: 8
LoRa addr 1 : state 0
SX1272 configured as LR-BS. Waiting RF input for transparent RF-serial bridge
%LASBase::ON
%LAS::ON
%LASBase::clear dev table
----- Rcv from LoRa. src=10 seq=7 len=10 SNR=7 RSSIpkt=-62
rcv ctrl pkt info: 10,7,10,7,-62

splitted in:  [10, 7, 10, 7, -62]
(src=10 seq=7 len=10 SNR=7 RSSI=-62)
%LASBase::rcv LAS message
%LASBase::LAS payload size is 10
%LASBase::DATA msg
%LASBase::process DATA msg from 10
%LASBase::advertised lrat is 33301
%LASBase::dev_table index 0
%LASBase::can find device
hello
```

```
Rcv serial: hello
Sending. Length is 5
hello
LASDevice::Payload size is 15
LASDevice::ToA is 322
LASDevice::alpha*gat is 36000
LASDevice::_ltat is 2699
LASDevice::_lrat is 33301
LASDevice::sending w/LP
LAS::CAD duration 138
LAS::CAD OK1
--> waiting for 6 CAD = 96
--> CAD duration 138
LAS::CAD OK2
LAS::check RSSI
--> RSSI -114
LASDevice::LoRa Sent in 541
LASDevice::LoRa Sent w/CAD in 916
Packet sent, state 0
```

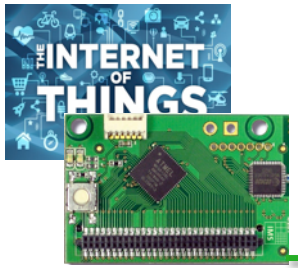


Sending message under LAS services



```
Rcv serial: /@REG#
Parsing command
Send LAS REG msg
LASDevice::REG with 36000
LASDevice::Payload size is 10
LASDevice::ToA is 281
LASDevice::disabled
LAS::CAD duration 46
LAS::CAD OK1
LAS::check RSSI
--> RSSI -115
LASDevice::LoRa Sent in 499
LASDevice::LoRa Sent w/CAD in 546
```

```
hello
LASDevice::Payload size is 15
LASDevice::ToA is 322
LASDevice::alpha*gat is 36000
LASDevice::_ltat is 3021
LASDevice::_lrat is 32979
LASDevice::sending w/LP
LAS::CAD duration 138
LAS::CAD OK1
--> waiting for 6 CAD = 96
--> CAD duration 138
LAS::CAD OK2
LAS::check RSSI
--> RSSI -115
LASDevice::LoRa Sent in 541
LASDevice::LoRa Sent w/CAD in 915
Packet sent, state 0
Rcv from LoRa. src=1 seq=0 len=6 SNR=8
^1,0,6,8,-55

LASDevice::rcv LAS message

LASDevice::UPDT msg
LASDevice::process UPDT msg 4426617
LASDevice::AT is 3021
LASDevice::Di is 10
LASDevice::nothing to be done
```



Conclusions

- Long-range radios appear as a very promising technology to deploy sensors and so-called IoT devices without the complexity and cost of multi-hop networking
- A low-cost gateway for custom development initiatives
- More robust channel access method with CS/LBT features
- However, as the radio activity time is limited by regulations, it is difficult to guarantee a level of service of the deployed infrastructure, especially for some data-intensive applications
- We propose a Long-range Activity Sharing mechanism to manage a pool of deployed devices in a flexible manner
- Quality of Service can be ensured as devices can transmit critical information when needed

ANNEX.1: PARTS LIST

- Raspberry 1 B+
- Micro SD card 8GB
- HopeRF RFM92W
- RFM92W adapter kit (including the straight break away headers and the antenna plug)
- Male SMA<> Female RP-SMA cable
- 868MHz antenna Male RP-SMA
- Some jumper wires F/F
- Some standoffs and screws
- Plastic box, waterproof
- Power
  - 2A USB AC/DC Power Charger Adapter + Micro USB Cable …
  - Or PoE kit with a 5V regulator at the RPI side (shown in picture)
- A WiFi dongle (optional)