NBIOT field tests

In this blog I want to share the hardware and software used in the field tests for NB-IOT. I invite you to recreate these tests and share your experiences.

We had a lot of fun taking it up in the air and under the ground.

Until now we are very enthusiastic about our network.

Hardware

The goal was to create a device consisting out of standard available components. I wanted to create a device requiring no soldering and no wiring.

So I came up with the theoretical plan to stack 3 boards on top of each other;

- 1 arduino Leonardo base for the mcu

- 1 arduino shield for the modem, GPS and other sensors

- 1 additional shield for the LCD display

Normally for a typical nb-iot solution a display is not used because of the power consumption of the LCD, but to visualise a direct relation between a sensing and transmitting device and an application this would be the way to go.

The build

Anyway that was the theory.

So relying on the Sodaq Kickstarter shield to be delivered, I ordered a 16*2 LCD display

Soon I noticed pin conflict looking at the published designs of the nb-iot shield.

The LCD Display uses a lot of digital out pins to control the display pins in conflict with the nb-iot shied so this wouldn't work.

Looking for a solution I found that the nb-iot shield relies much on the I2c / wire protocol to address the sensors.

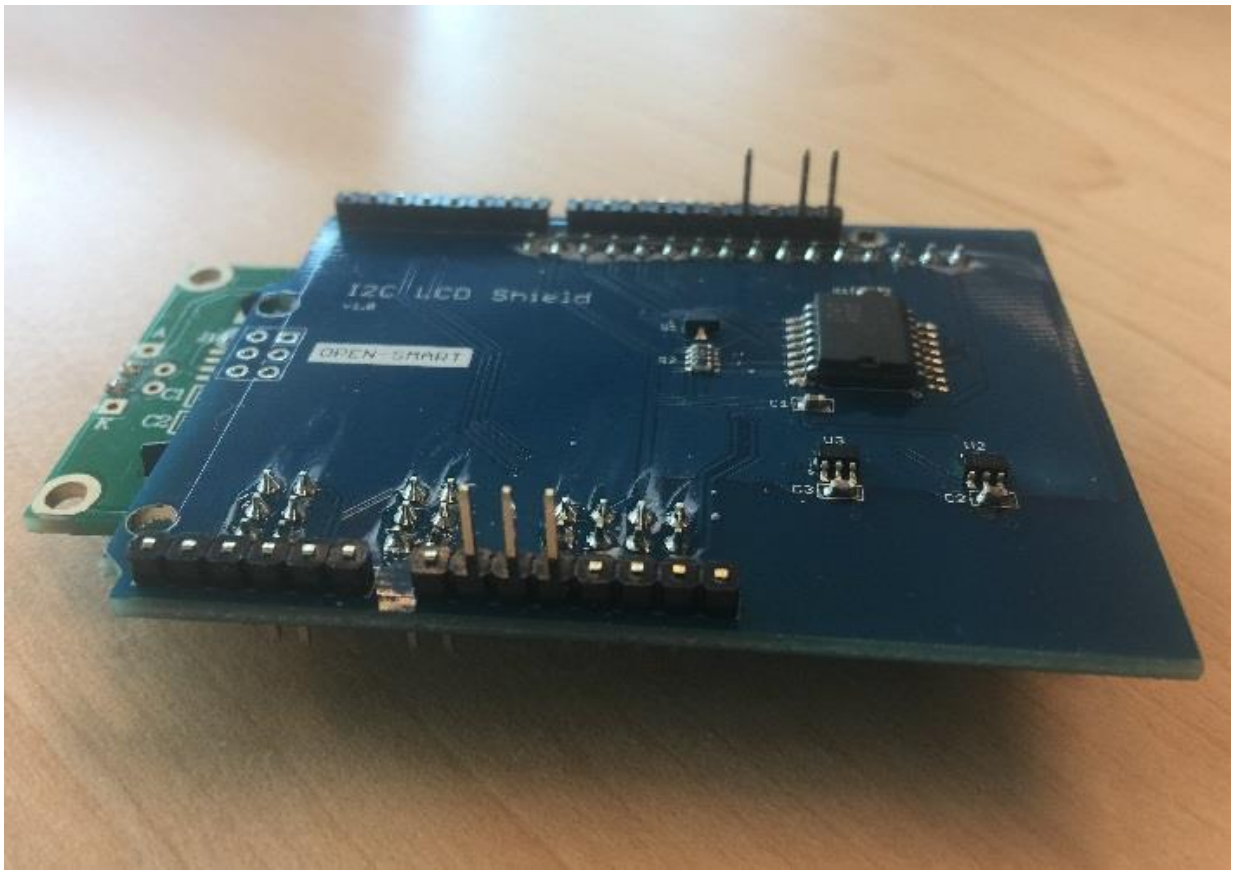
(the nb-iot deluxe shield has a full pack of sensor like temp, humidity, accelerator, magneto etc.)

This is an efficient way to reduce the pin utilization. You can think of it like an USB bus, each of your components has an address.

Testing it with the I2c scanner sketch I still didn’t get results the scanner just didn’t work.

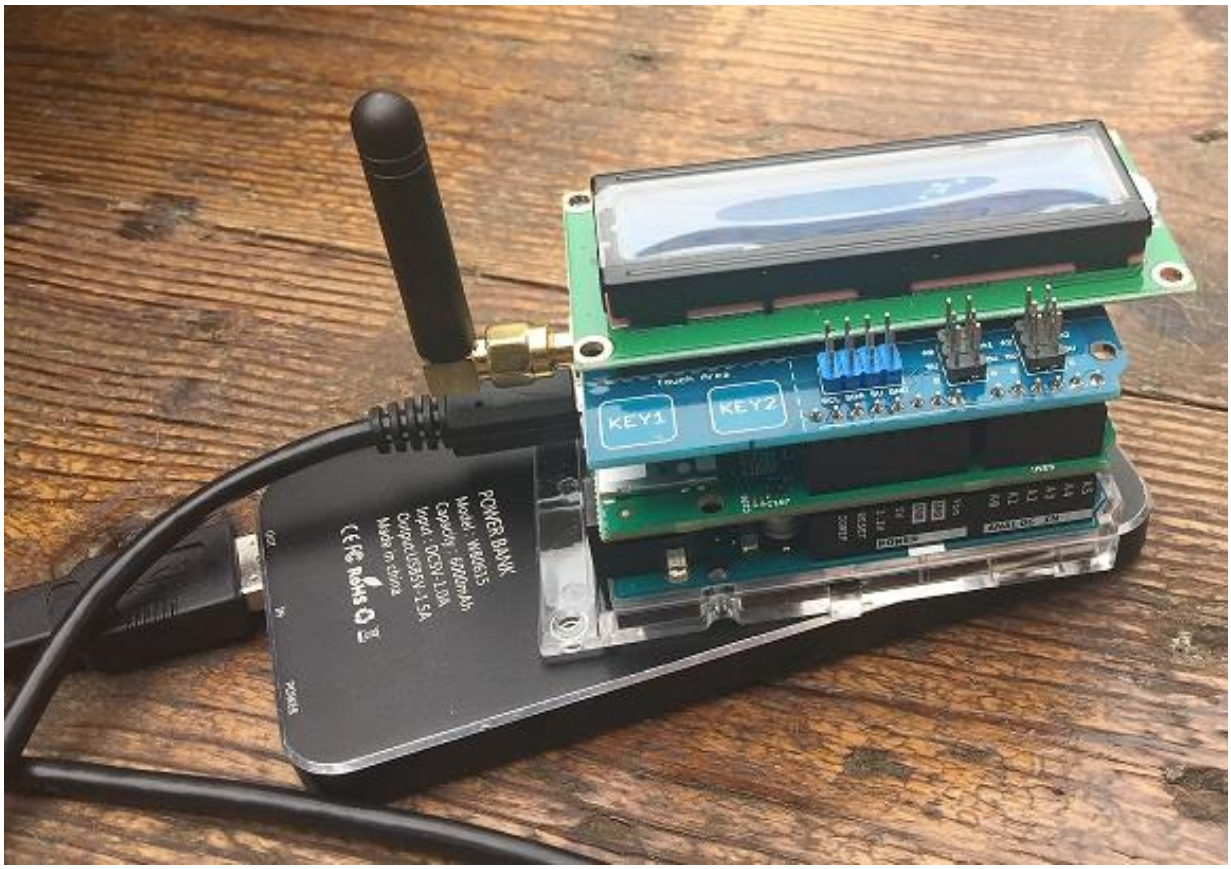
Without the display shield all sensors showed fine, connecting it using wires was working.

So still pin conflicts possibly the push buttons or the voltage levels anyway. I chose to remove the redundant pins what did the trick.

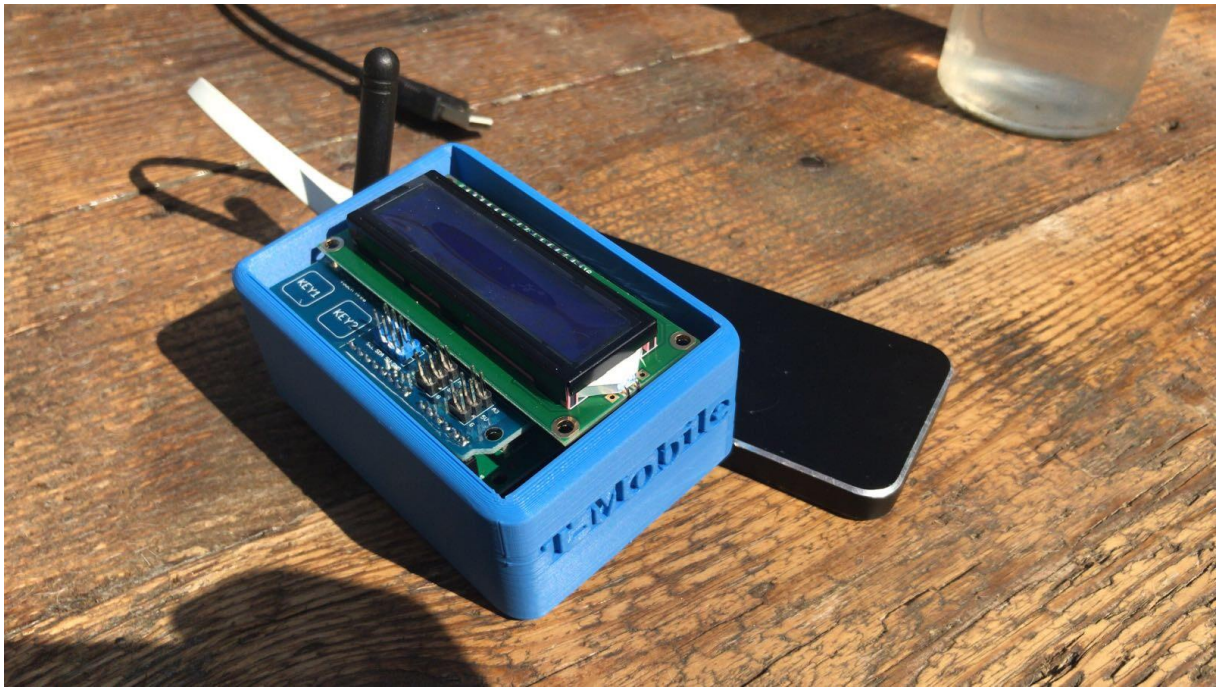


Only 5V, GND, SDA & SCL pins remained

The completed device looks now like this;



And my Colleague Jesse Scholtes printed a nice case for it.



Software

The software sketch loaded into the Arduino Leonardo is based on the standard examples provided by Sodaq here.

I have extended it to publish the Signal quality and the GPS position to the application (AllThingsTalk (ATT) in this case)

And we needed to display certain information on the LCD.

The resulting sketch can be found on our GitHub here.

The ATT device profile can be found in the first part of the sketch.

(Note it is only possible to use ATT as a target whit the devices and SIM cards provided with the Kickstarter, if you want to work with another platform (and hardware) of your choice you should register for the TMNL Early Access program here).

Measurements

What we measure using the default libraries is the perceived signal quality of the Device. This is retrieved using the AT+CSQ command.

AT+CSQ Get Signal Strength Indicator

The terminal will provide a current signal strength indicator of 0 to 255 where larger is generally better.

This information is based on a single measurement so can be expected to change greatly over short periods of time and may never use all possible (or even the majority) of the entire possible range or codes.

Execution command returns received signal strength indication <rssi> and channel bit error rate <ber> from the MT.

Parameter

<rssi> Integer type

0	-113dBm or less
1	-111dBm
2...30	-109... -53dBm
31	-51dBm or greater
99	Not known or not detectable

So this is an indexed scale not showing dBm values

For the values returned between 2 and 30 in the library are calculated back to a dBm value using this formula; RSSI = -113 + 2 * CSQ

A more accurate result can be obtained from the AT +NUESTATS command which we plan to incorporate in the next version of our coverage mapping device.

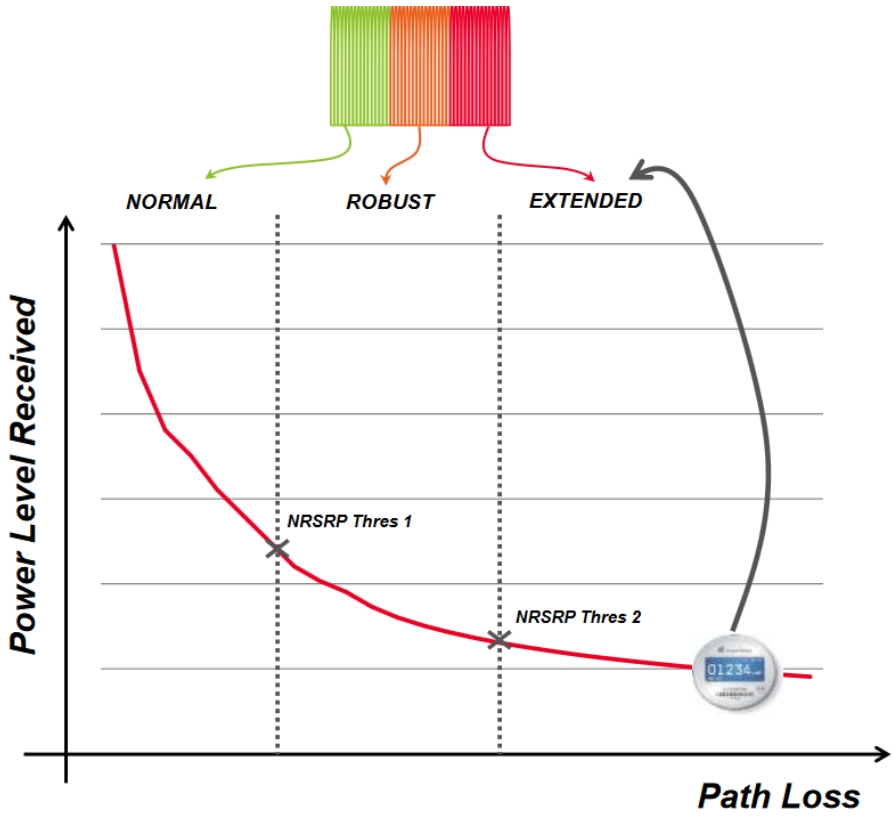
AT+NUESTATS Radio specific information

Read AT+NUESTATS[=RADIO]

Signal power: <power>	Signal power: -508	This corresponds with the RSSI in centibels
Total power: <tot_power>	Total power: -500	
TX power: <tx_power>	TX power: -30	
TX time: <tx_time>	TX time: 2393	
RX time: <rx_time>	RX time: 28903	
Cell ID: <cell_ID>	Cell ID: 25	The Cell ID currently in use by the terminal
DL MCS: <dl_mcs>	DL MCS: 5	Message coding scheme used
UL MCS: <ul_mcs>	UL MCS: 5	Message coding scheme used
DCI MCS: <dci_mcs>	DCI MCS: 5	Message coding scheme used
ECL: <ECL>	ECL:1	This indicates whether Coverage Extension is applied
SNR: <snr>	SNR:20	Last Signal to Noise Ratio value
EARFCN: <earfcn>	EARFCN:30	Search frequencies
PCI: <pci>	PCI:11	

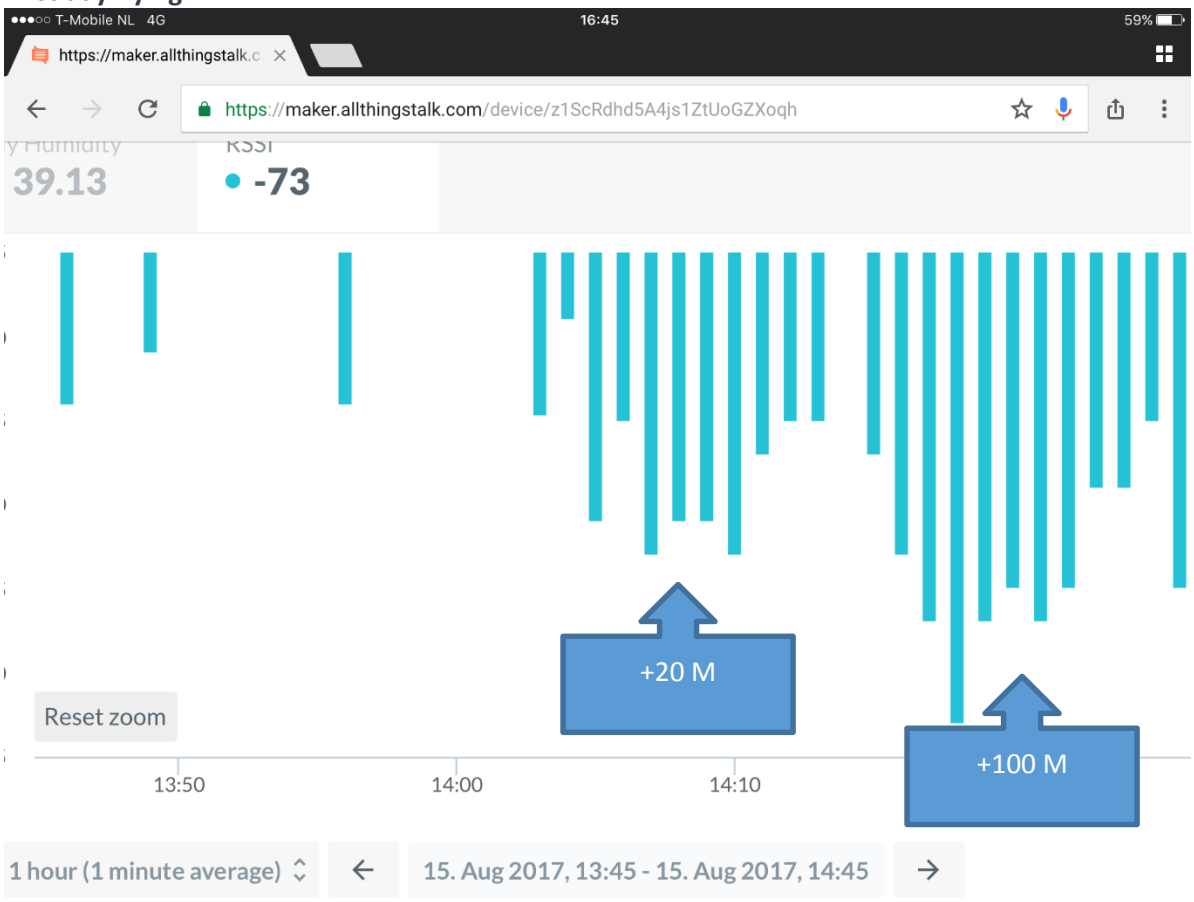
Particularly interesting is to know the Coverage Extension Level (ECL) being used during the tests

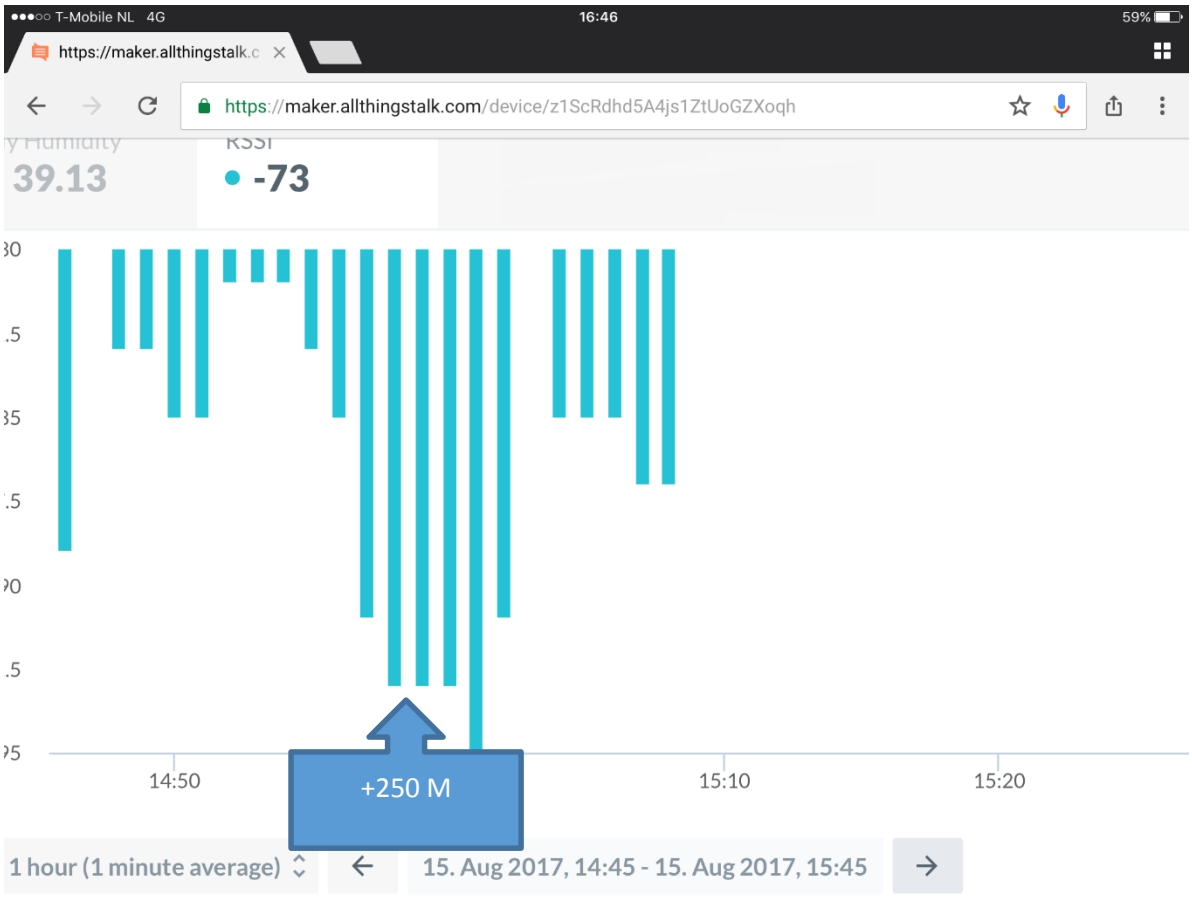
NB-IOT Coverage Extension Levels (Normal, Robust, Extended);



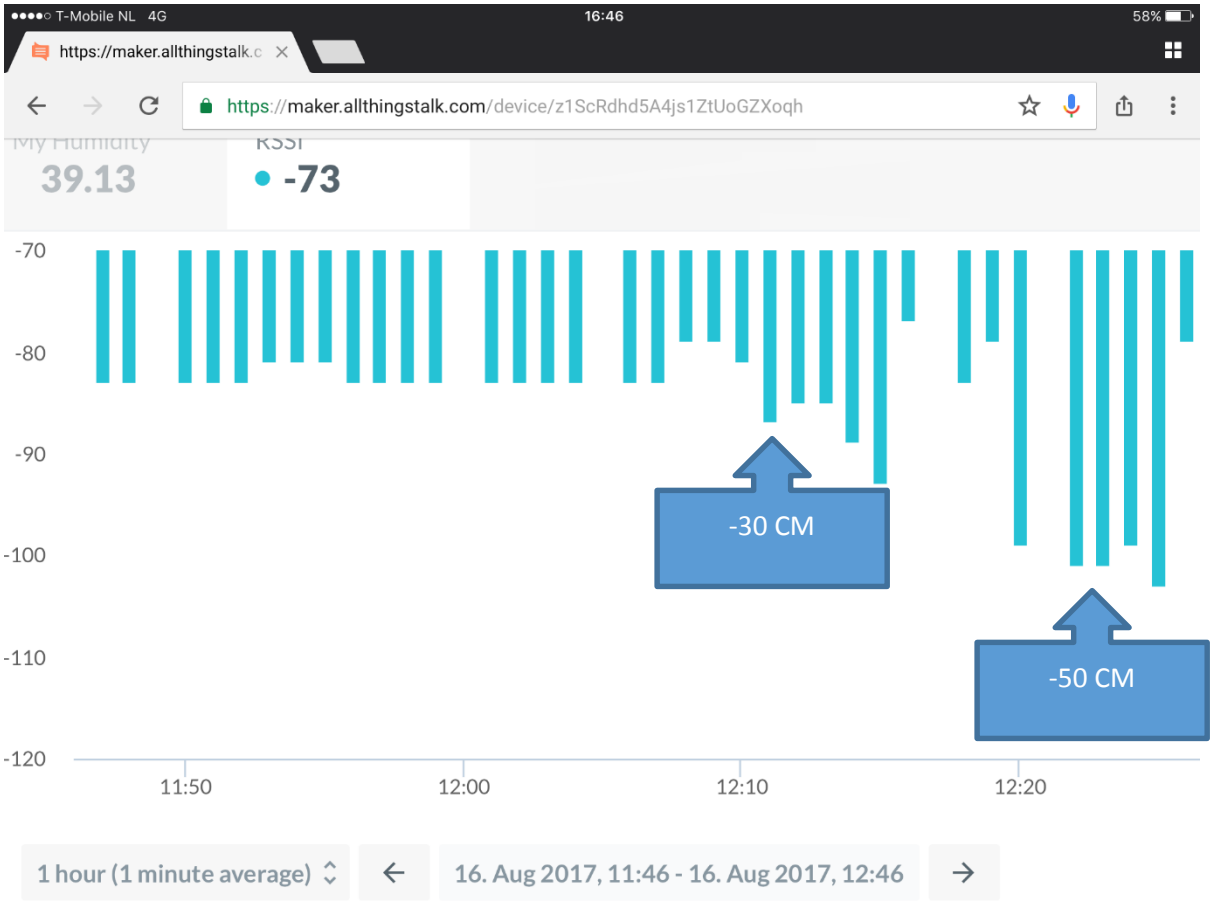
Results

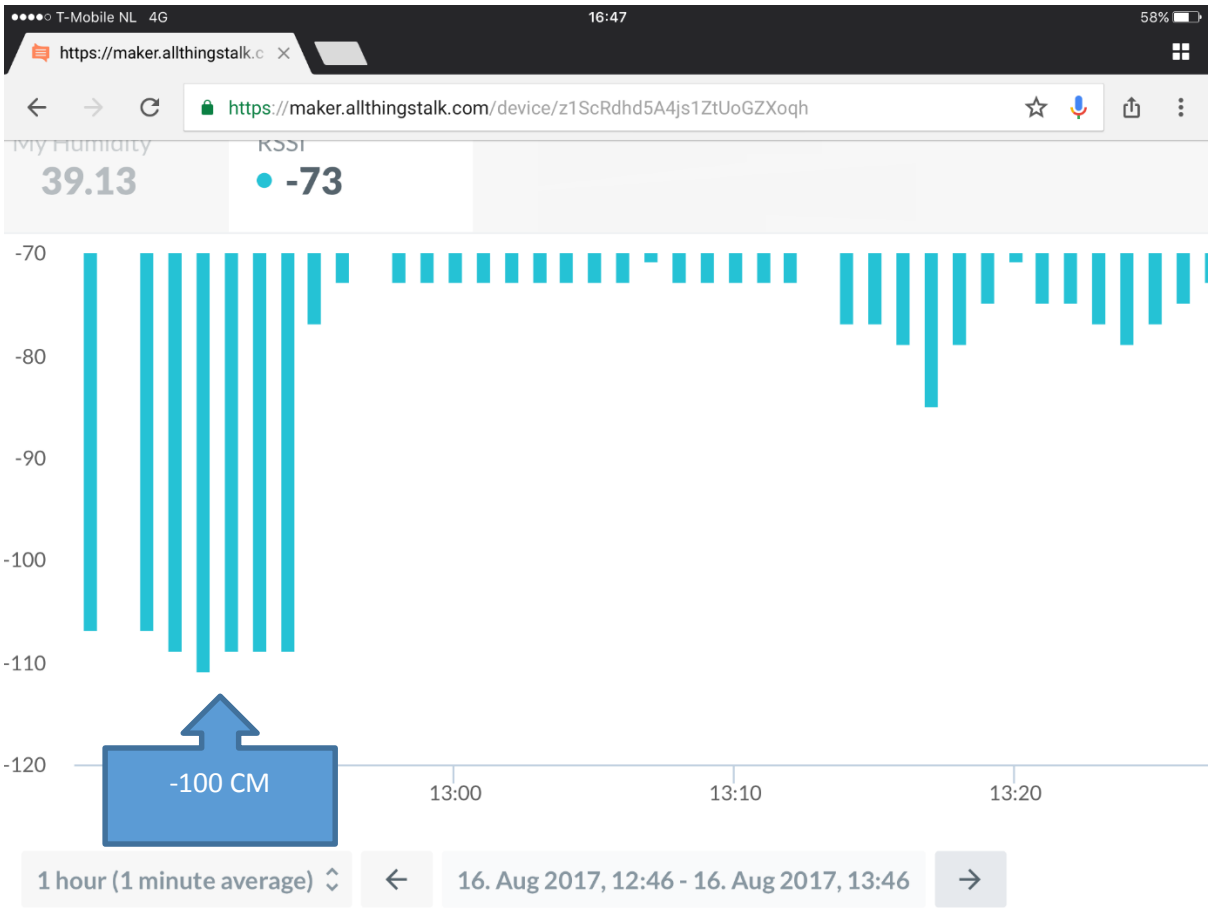
First day flying




T-Mobile NL 4G
16:46
59%
https://maker.allthingstalk.c
https://maker.allthingstalk.com/device/z1ScRdhd5A4js1ZtUoGZXoqh
39.13
-73
14:50
15:10
15:20
+250 M
1 hour (1 minute average)
15. Aug 2017, 14:45 - 15. Aug 2017, 15:45

Second day digging




T-Mobile NL 4G
16:47
58%
https://maker.allthingstalk.c
https://maker.allthingstalk.com/device/z1ScRdhd5A4js1ZtUoGZXoqh
39.13
RSSI
-73
-70
-80
-90
-100
-110
-120
13:00
13:10
13:20
-100 CM
1 hour (1 minute average)
16. Aug 2017, 12:46 - 16. Aug 2017, 13:46