Master Project – Design and Implementation of Cattle Tracking Prototype System

Student: Yunwei Jiang **Email:** yjiang8@albany.edu **Instructor:** Prof. Mariya Zheleva
Session: 2018 Summer

1 Project instruction

This cattle tracking prototype system is wireless network system combined with different wireless protocols to track and estimate the density of cattle in farm, collect their location and exchanged data to build up their social graph. This system will use short range wireless protocol BlueTooth to exchange cattle id, and long range protocol LoRa to transfer the collected data to cloud server. Then the collected data will be used for further research of cattle's individual and herd behavior.

1.1 Project Background

This cattle tracking system is a part of FarmNet project of precision rotational grazing (PRG). PRG involves tracking of moving objects (grazing animals) and state of entities (paddocks), based on multiple sensing modalities, dealing with uncertain, delayed and sparse measurements and offers rich opportunities for optimization, analytics and smart connectivity solutions. Rotational grazing involves subdividing a pasture into paddocks and scheduling grazing for different herds of animals in paddocks, so as to maximize the production of grass and legumes and thus the dairy and meat output of the farming operation within limited pasture land.

The sensing modalities FarmNET will employ include (i) GPS, motion and sound sensors for a subsample of cattle allowing us to track the herd's temporal trajectory, activity (graze/not graze) and health; (ii) low-cost and low-energy Bluetooth beacons for the same subsample to estimate density based on distance to GPS cattle; (iii) AUV-acquired images to estimate and track the grass growth; and (iv) environmental parameters including precipitation and temperature.

1.2 System Structure

There are two kinds of structure for the project. We use the second structure to build up the system.
I also build up a GitHub repository to manage the source code and documents. The link below:
https://github.com/jiangyunwei2015/2018SummerProject_FarmNet
Note: For the second structure there are also two solutions: 1) BLE+LoRaWAN Gateway+TTN cloud broker(Chapter 5.4.1) and 2) BLE+LoRa gateway + AdafruitIO cloud broker(Chapter 5.6.1).
The running code for the system is:

- For BLE+LoRa gateway + AdafruitIO, the source code folders are:

1) For LoRa nano-gateway: Projects/LoRa/LoRaMAC/LoRaMAC_Gateway
2) For LoRa Node & BLE client node devices: Projects/LoRa/LoRaMAC/LoRaMAC_Node2
3) For BLE server device: Projects/Bluetooth/BluetoothNode1

- For BLE+LoRaWAN Gateway+TTN, the source code folders are:

1) For LoRaWAN nano-gateway: Projects/LoRa/LoRaWAN
2) For LoRaWAN Node & BLE client node devices: Projects/ProtypeNodePrj

3) For BLE server device: Projects/Bluetooth/BluetoothNode1

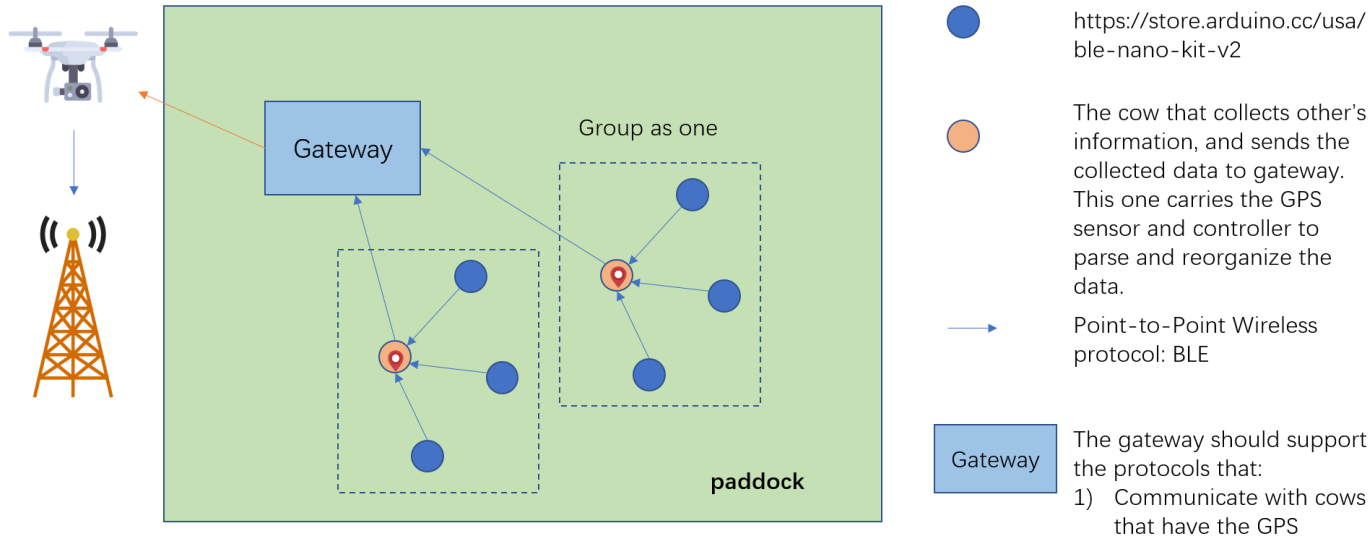


Fig1 Cattle Tracking architecture 1st version

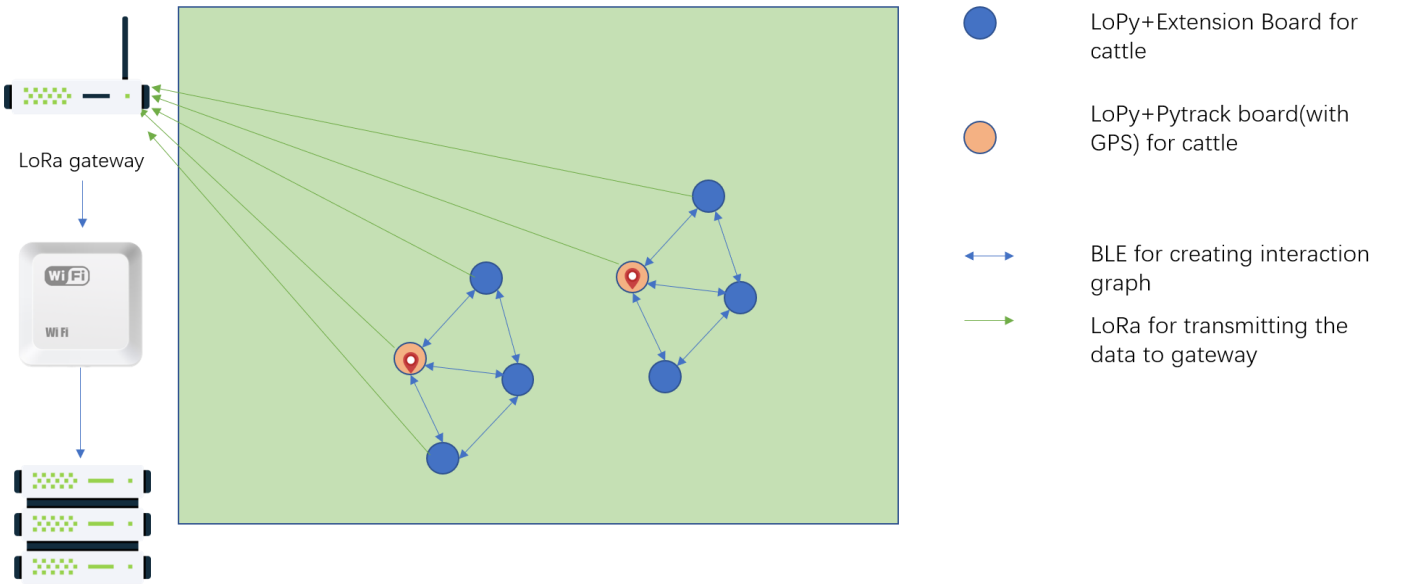


Fig2 Cattle Tracking architecture 2nd version

For the cattle wear GPS module:
1) Controller: Raspberry Pi.
2) LoRa Node device to transmit the data to gateway.
3) BLE module is not necessary because Raspberry Pi can cover that.

For the cattle don't wear GPS module:
1) BLE nano V2 for communication with other cattle.
2) LoRa Node device to transmit the data to gateway.

Fig3 Devices worn by cattle version 1

For the cattle wear GPS module:
1) LoPy module used as BLE+LoRa node.
2) Using Pytrack shield as GPS sensor.

For the cattle don't wear GPS module: LoPy module used as BLE+LoRa node.

Fig4 Devices worn by cattle version 2

Note: We use the second structure: using LoPy+Pytrack/Expansion board to trace the cattle, LoPy as LoRa nano-gateway to forward the data to cloud server.

2 Device List

2.1 Gateway

Works as a bridge between communication devices worn by cattle and the server.
This is actually a LoRaWAN or LoRa gateway that could catch LoRa packet and forward to cloud server as TTN, Loriot, Adafruit IO, AWS,.etc.
Right now use LoPy as the single-channel gateway.

2.2 BLE device

Communicate between cattle.

Device	Parameters	Price	Purchase Link
BLE Nano Kit V2 (Including a companion board, DAPLink v1.5 board, which is used to load firmware into BLE Nano 2 from a PC)	1) nRF52832 BLE SoC 32-bit ARM® Cortex™-M4F CPU with 512kB flash + 64kB RAM 2.4GHz BLE Bluetooth5 2) Mynewt OS	$32.95	Purchase link: https://store.arduino.cc/usa/ble-nano-kit-v2
BLE Nano V2	BLE SoC development board BLE 1xUART 2xSPI 1xI2C On board chip antenna	$17.95	Purchase link: https://store.arduino.cc/usa/ble-nano-v2-with-header-soldered

2.3 GPS sensor

GPS sensors are worn by a certain number of cattle that collect data from other cattle.

Device	Parameters	Price	Purchase Link
Adafruit Ultimate Breakout	Satellites: 22 tracking, 66 searching Patch Antenna Size: 15mm x 15mm x 4mm Update rate: 1 to 10 Hz Position Accuracy: < 3 meters Warm/cold start: 34 seconds Built in antenna	$39.95	https://www.adafruit.com/product/746
GPS Antenna	External Active Antenna - 3-5V 28dB 5 Meter SMA	$14.95	https://www.adafruit.com/product/960
SMA to uFL/u.FL/IPX/IPEX RF Adapter Cable		$3.95	https://www.adafruit.com/product/851

2.4 LoRa device

Device	Parameters	Price	Purchase Link
mDot Long Range LoRa® Modules	The MultiConnect® mDotTM is a secure, CE/FCC/RCM certified, Arm® MbedTM programmable, low-power RF module that provides long-range, low bit rate M2M data connectivity to sensors, industrial equipment and remote appliances.	$31~32(Based on different spectrum)	Introduction: https://www.multitech.com/brands/multiconnect-mdot
MultiConnect® mDotTM Developer Kit	The MultiConnect® mDot™ Developer Kit allows customers to plug in the MultiConnect mDot module and use it for testing, programming and evaluation.	$69.00	Introduction: https://www.multitech.com/brands/micro-mdot-devkit Purchase: https://www.semiconductorstore.com/cart/pc/viewPrd.asp?idproduct=50707
Arduino Shield			

Question: Which Spectrum? mDot supports 868 or 915 MHz.

2.5 Controller

Because the data comes from different sources (GPS, BLE), the controller can reorganize and pack the data, then send to gateway.

Device	Parameters	Price	Purchase Link
BeagleBone Black			
Arduino uno	CPU:ATmega328P 16MHz Flash:32KB RAM:2KB SRAM	$22.00	https://store.arduino.cc/usa/arduino-uno-rev3
Raspberry Pi	ARM Cortex-A53 Bluetooth 4.2/BLE	$35.00	https://www.adafruit.com/product/3775?src=raspberrypi

2.6 Purchase list

Device	Purpose	Price
BLE Nano Kit V2 X1		1* $32.95
BLE Nano V2 X3		3* $17.95
GPS sensor X1		1* $39.95
GPS Antenna X1		1* $14.95
SMA to uFL/u.FL/IPX/IPEX RF Adapter Cable X1		1* $3.95
mDot		1* around $32
mDot Development Kit		1* $69

2.7 Final version device list (2018-06-15)

After searching and comparing for almost two week, now decide to choose LoPy from Pycom as the LoRa node device. LoPy is a BLE+WiFi+LoRa+LoRa nano gateway development board for IoT application.
So now the gateway. To start the work, there are some devices that will used to build up the protype and test:

Device	Number	Usage
LoPy	4	As BLE and LoRa node device, and need to test the performance as a nano gateway.
Antenna	4	For each LoPy, it needs a external antenna to work normally.
Pytrack	1	Development shield for LoPy, also contains a GPS receiver.
Extension Board 2.0	3	Development shield for LoPy.
USB hub	1	Power supply for LoPys.

3 Questions

3.1 What's the gateway range?

The size of the farm is 1100 acres. LoRa gateway can cover the farm without problem.
But still need multiple gateways to resolve data collision problem.

3.2 Is there any connection between the cows with the GPS sensors?

Yes. The goal is to build up the social graph for each cow.

3.3 What's the data transmittd between BLE devices?

For cows without GPS sensor: device id. The device id will be hard coded in LoPy.

3.4 Does all BLE devices store the data?

Probably not. Only transmit the data when comes close to another device.

3.5 How often will the data be collected?

3.6 Accuracy of the GPS sensor?

Almost all sensor support +/- 3m.

3.7 External Power Supply

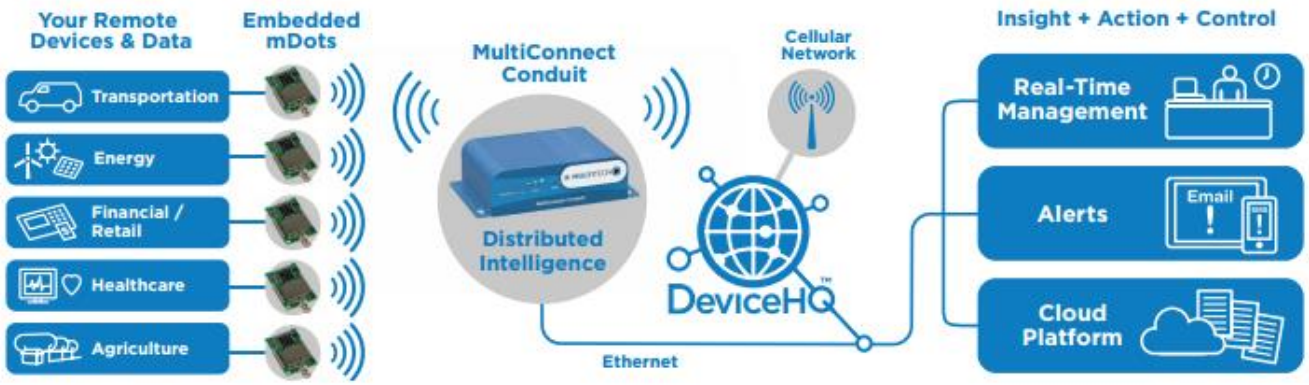
Battery? Power Bank?

3.8 Memory Size

3.9 Micro sd card and card reader for flashing the OS?

3.10 About the gateway: Buy product or self-integrated?

1 There is a gateway also developed by MultiTech: MultiTech® Conduit™ Gateway + mCard around $440
https://www.multitech.com/brands/multiconnect-conduit
And rugged outdoor version MultiConnect® Conduit™ IP67 Base Station.
https://www.multitech.com/brands/multiconnect-conduit-ip67
The figure below shows the structure of mDot with MultiConnect Conduit gateway:



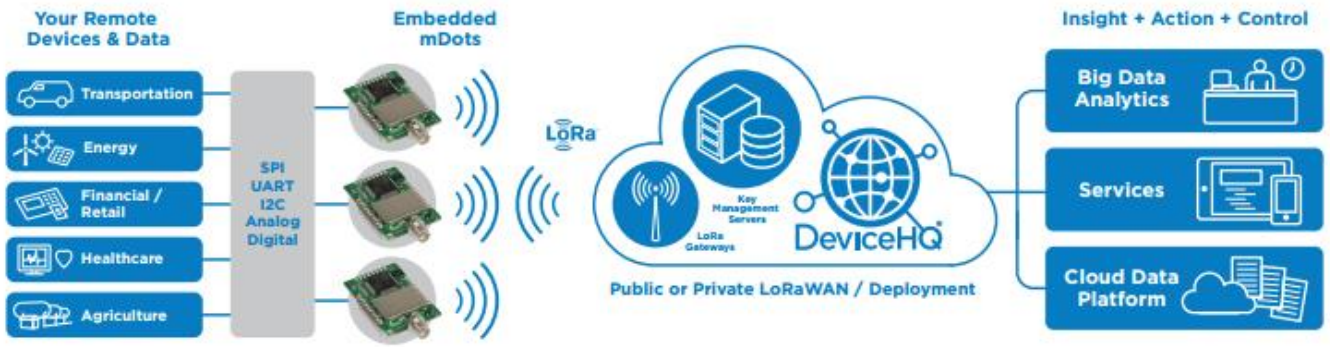
2 Self-integrated
There are some self-integrated gateway solutions.
Still searching.

3.11 ~~How to integrate GPS sensor and mDot to Raspberry pi?~~

1) GPS sensor to Raspberry pi
Input: GPS, output: Rasperry pi
https://learn.adafruit.com/adafruit-ultimate-gps-on-the-raspberry-pi/introduction
2) mDot to Raspberry pi
Input: Raspberry pi, output:mDot

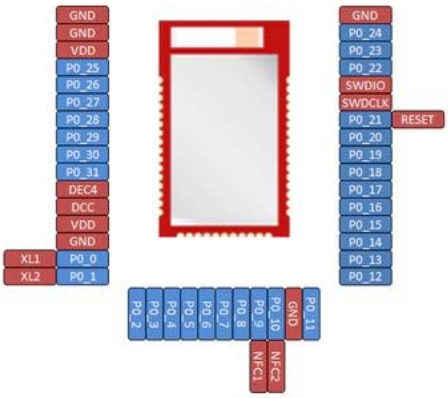
3.12 ~~How to integrate BLE nano V2 to mDot?~~

Using BLE nano V2 as input source, and mDot as the output source. So need to find out if nano V2 can connect to mDot?
Or still need other shield or controller as a bridge?
As the picture shows below:
mDot can accept data from other device by using SPI, UART, I2C.



BLE nano V2 supports these:

- Nordic Semiconductor nRF52832 SoC
  - ARM Cortex-M4F (32-bit, @64MHz)
  - Bluetooth 4.2 certified and 5.0 ready (BLE only)
  - NFC
  - 64 KB SRAM
  - 512 KB Flash
  - FPU
  - DSP
- Onboard Chip Antenna
- Total 32-I/O, includes
  - 1 x UART
  - 2 x I2C
  - 2 x SPI
  - Max. 12 PWM
  - Max. 8 ADC
- Physical Dimensions
  - 10 mm x 18 mm
- Operation Voltage
  - 1.8V to 3.6V



3.13 ~~The communication between BLE nano V2 and Raspberry pi~~

Since

Bluetooth LE on the Raspberry Pi

The Raspberry Pi 3 has built-in support for Bluetooth Low Energy (BLE). BLE is the technology behind many fitness trackers and smartwatches, and allows small, low-power devices to transmit and receive information from a central computer or smartphone.

3.14 How to test the performance of LoPy both as LoRa node and nano gateway?

4 Shopping website

4.1 Aruino

https://www.arduino.cc/

4.2 Sparkfun

https://www.sparkfun.com/

4.3 Adafruit

https://www.adafruit.com/

5 Development Work Progress

The first three weeks is used for project preparation as paper reading, background knowledge learning, information collecting, device selecting.

5.1 Summer Week 4: 06-18~06-22

06-22 Start to work on LoPy

Got all the devices today. First, build up the IDE environment and update firmware for LoPy and Pytrack:

5.1.1 Update Firmware of Pytrack

There is a problem, I can't get any information from the tools as described in the official doc:
https://docs.pycom.io/chapter/pytrackpysense/installation/firmware.html
Except the first time when I followed the instruction and tried to connected Pytrack by using **Zadig** and the information shows as the doc shows, but when I click install driver, it failed and for the rest time Zadig can't recognize Pytrack anymore:

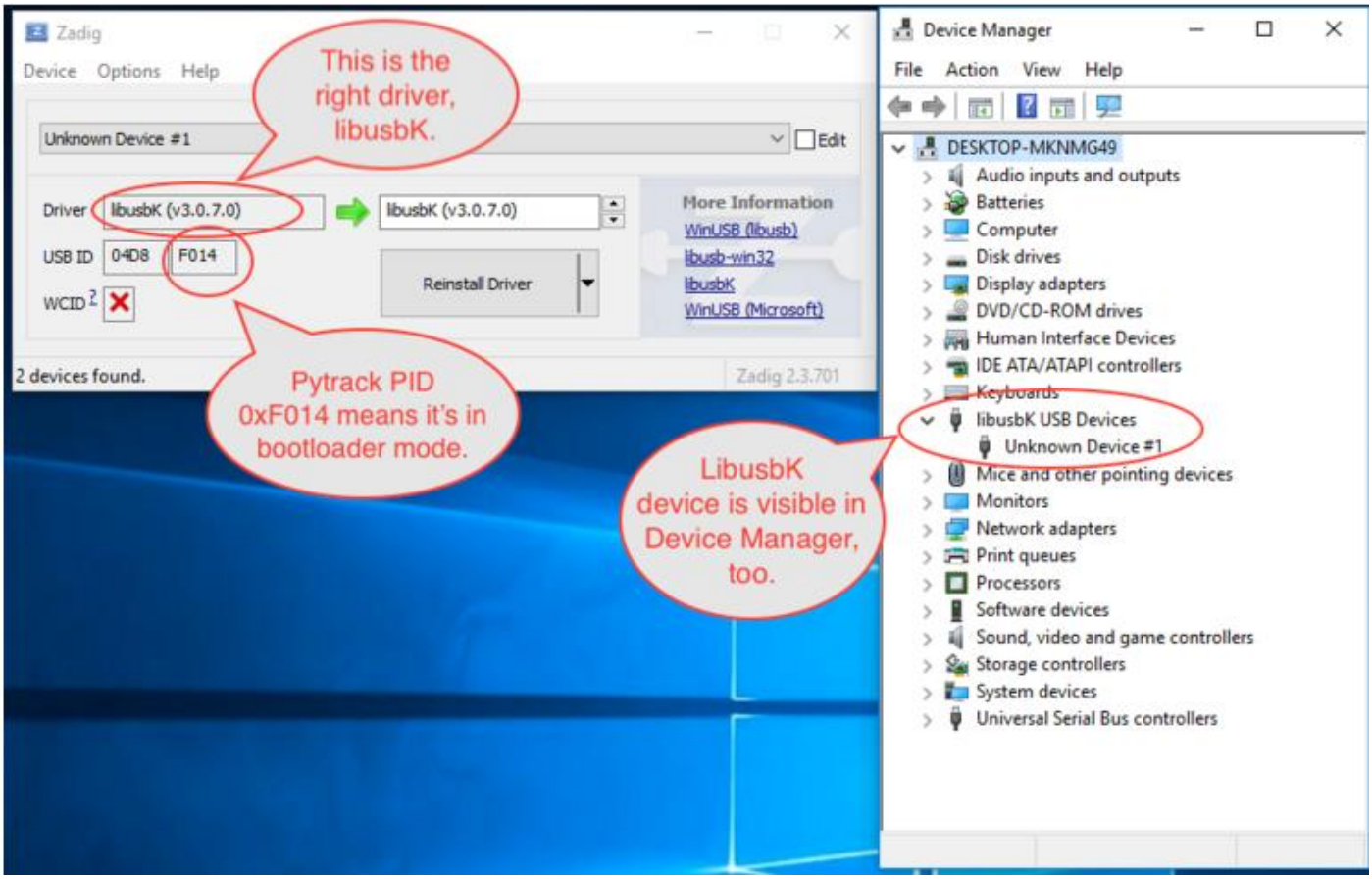


Fig 1 The connection result shown in official doc

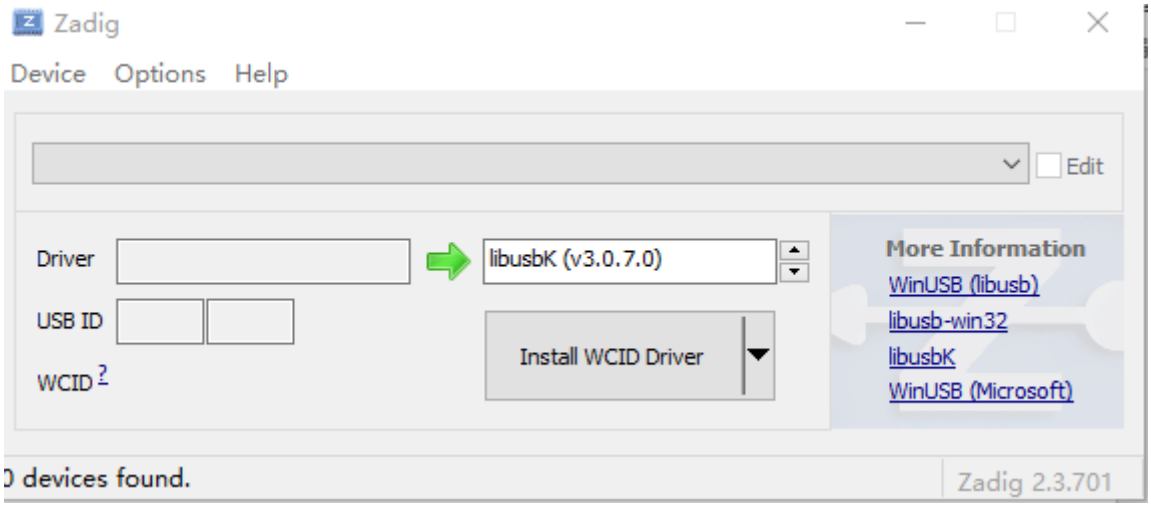


Fig 2

And when using dfu tool to update the firmware, it failed(No DFU capable USB device available).

```
D:\LoRa\PyTrack Frimware\dfu-util-0.9-win64\dfu-util-0.9-win64>dfu-util-static.exe -D pytrack_0.0.8.dfu
dfu-util 0.9

Copyright 2005-2009 Weston Schmidt, Harald Welte and OpenMoko Inc.
Copyright 2010-2016 Tormod Volden and Stefan Schmidt
This program is Free Software and has ABSOLUTELY NO WARRANTY
Please report bugs to http://sourceforge.net/p/dfu-util/tickets/

Match vendor ID from file: 04d8
Match product ID from file: f014
No DFU capable USB device available
```

But! After followed this instruction from forum, I could still update the firmware for Pytrack:

The method I've found works best for me is as follows:

1. Disconnect the board from USB and remove the **Py board
2. prepare the dfu command in your terminal but don't press enter
3. Press and hold the button on the Pysense/Pytrack
4. Insert the USB
5. wait a couple seconds
6. run the dfu command

You may need to try the above process a couple times to get it work, but this always seems to work the best for me.

After this input the command in console: **dfu-util-static.exe -D pysense_X.X.X.dfu(change X.X.X to the version of dfu file!)**

```
D:\LoRa\PyTrack Frimware\dfu-util-0.9-win64\dfu-util-0.9-win64>dfu-util-static.exe -D pytrack_0.0.8.dfu
dfu-util 0.9

Copyright 2005-2009 Weston Schmidt, Harald Welte and OpenMoko Inc.
Copyright 2010-2016 Tormod Volden and Stefan Schmidt
This program is Free Software and has ABSOLUTELY NO WARRANTY
Please report bugs to http://sourceforge.net/p/dfu-util/tickets/

Match vendor ID from file: 04d8
Match product ID from file: f014
Opening DFU capable USB device...
ID 04d8:f014
Run-time device DFU version 0100
Claiming USB DFU Runtime Interface...
Determining device status: state = dfuIDLE, status = 0
WARNING: Runtime device already in DFU state ?!?
Claiming USB DFU Interface...
Setting Alternate Setting #0 ...
Determining device status: state = dfuIDLE, status = 0
dfuIDLE, continuing
DFU mode device DFU version 0100
Device returned transfer size 64
Copying data from PC to DFU device
Download        [=========================] 100%        16384 bytes
Download done.
state(2) = dfuIDLE, status(0) = No error condition is present
Done!
```

It works!!!

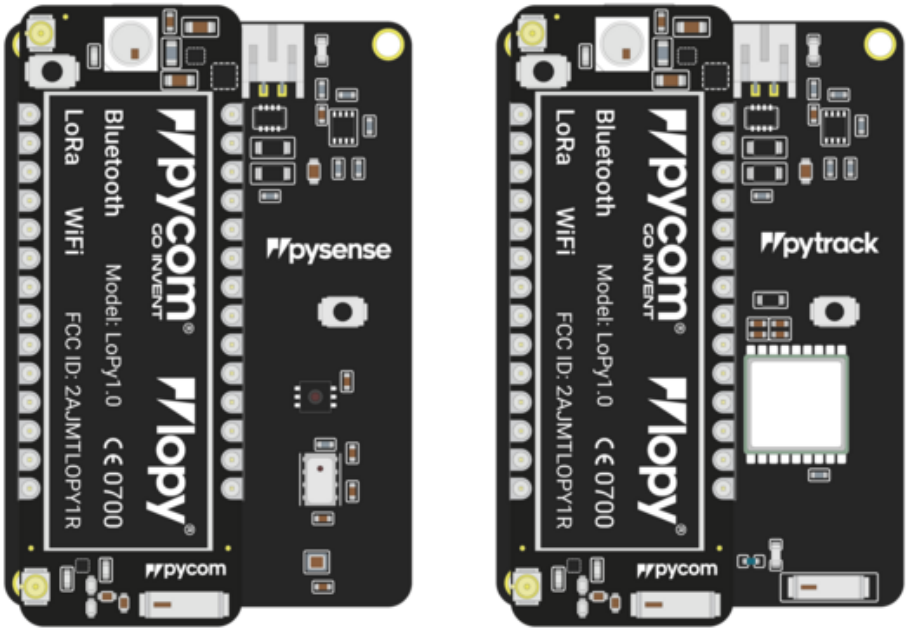
Need to check this with pycom!

I guess maybe for Win10 may just automatically installed the

5.1.2 Update firmware for LoPy by Pytrack

Pytrack is easy compared to expansion board 2.0, since it doesn't need external connection wire:

- Before connecting your module to a Pysense/Pytrack/Pyscan board, you should update the firmware on the Pysense/Pytrack/Pyscan. Instructions on how to do this can be found here.
- Look for the reset button on the LoPy module (located at a corner of the board, next to the LED).
- Locate the USB connector on the Pysense/Pytrack/Pyscan.
- Insert the module on the Pysense/Pytrack/Pyscan with the reset button pointing towards the USB connector. It should firmly click into place and the pins should now no longer be visible.



This is from:
https://docs.pycom.io/chapter/gettingstarted/connection/lopy.html#pic

Then use the Pycom upgrade tool to update device firmware:
https://docs.pycom.io/chapter/gettingstarted/installation/firmwaretool.html#second

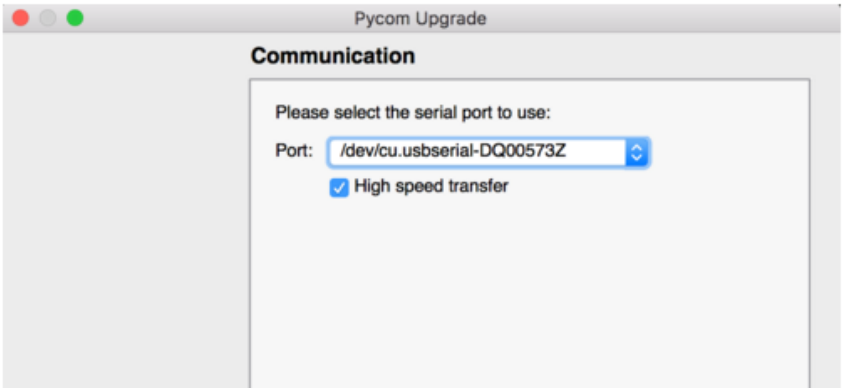
Updating Device Firmware

The basic firmware upgrade procedure can be found below, please follow these steps carefully:

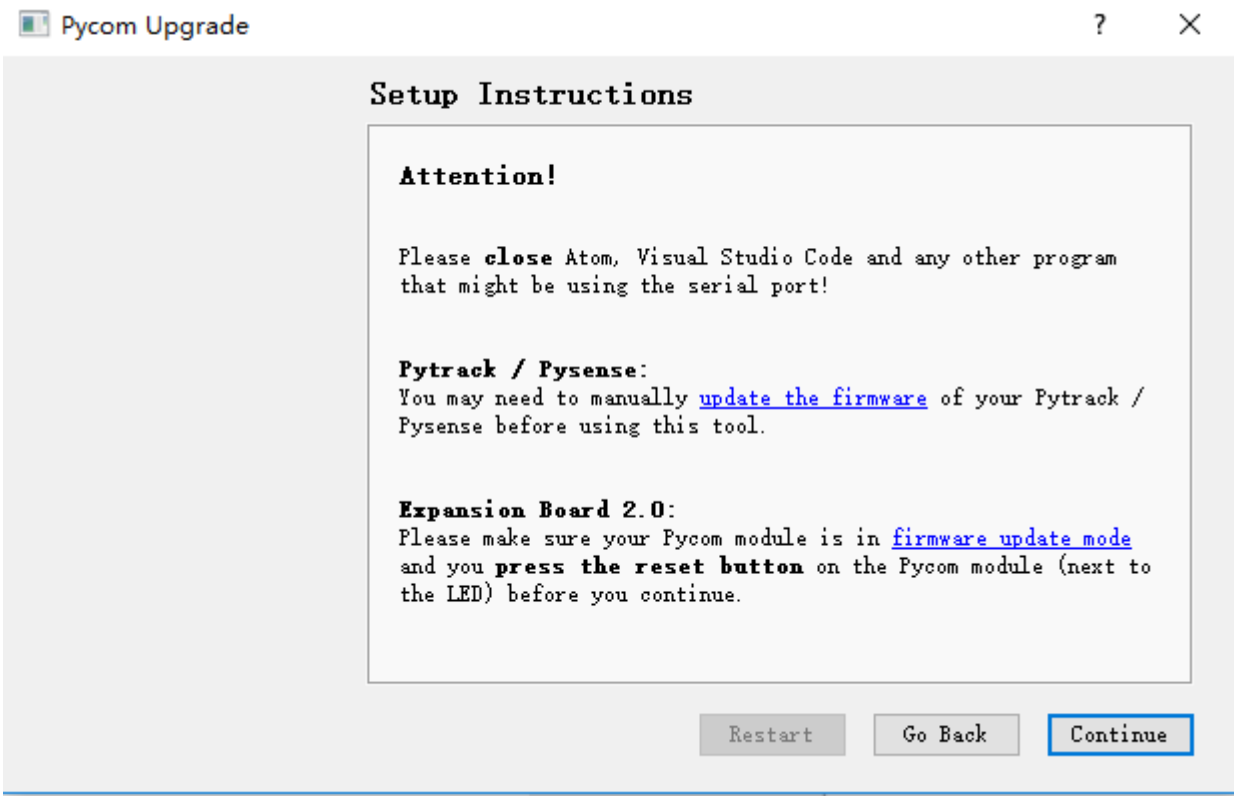
Expansion Board 2.0 | Pysense/Pytrack/Pyscan/Expansion Board 3.0

When using a Pysense/Pytrack/Pyscan/Expansion Board 3.0 to update your module you are not required to make a connection between P2 and GND, the Pysense/Pytrack/Pyscan/Expansion Board 3.0 will do this automatically.

1. Before connecting your module to a Pysense/Pytrack/Pyscan/Expansion Board 3.0 board, you should update the firmware on the Pysense/Pytrack/Pyscan/Expansion Board 3.0. Instructions on how to do this can be found here.
2. Disconnect your device from your computer
3. Insert module into expansion board
4. Reconnect the board via USB to your computer
5. Run the Firmware Upgrade tool



In the Pycom Upgrade Tool:

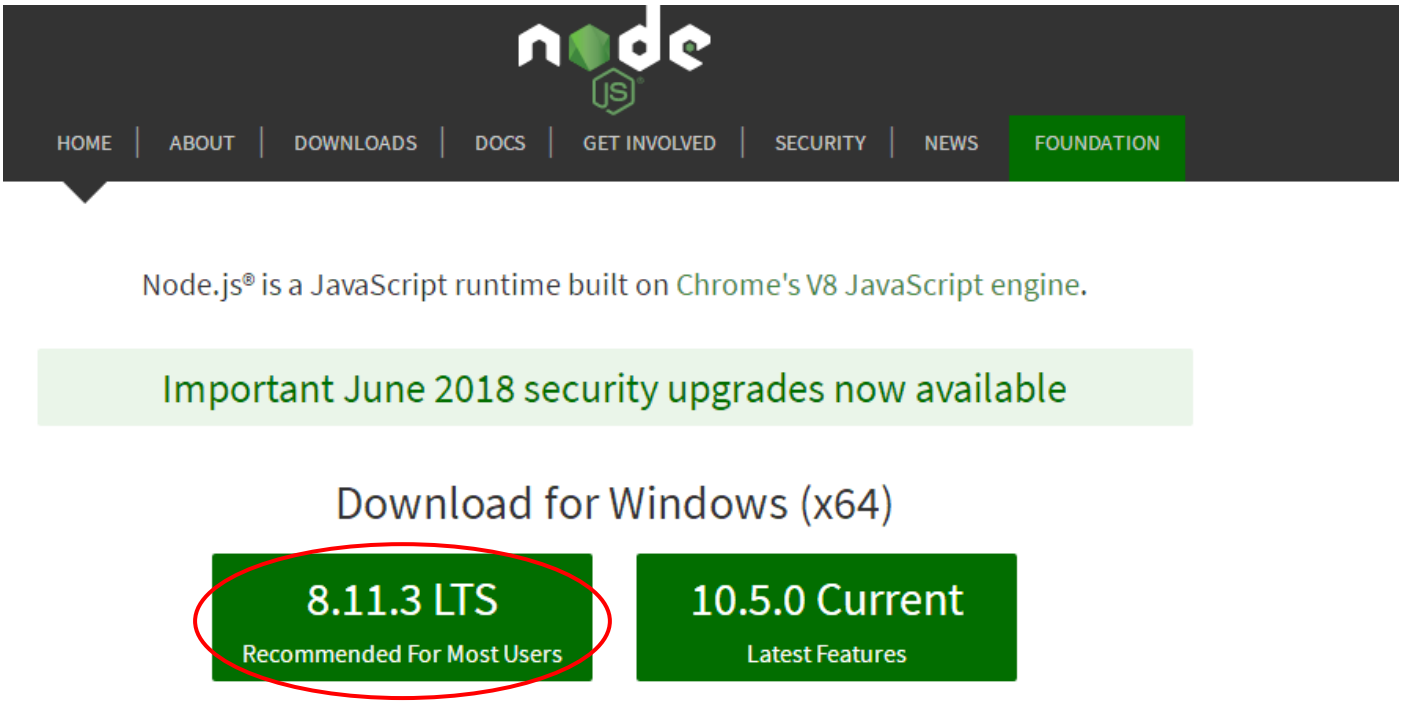


Note: there are some differences from the official doc!

5.1.3 Update firmware for LoPy by Expansion Board 2.0

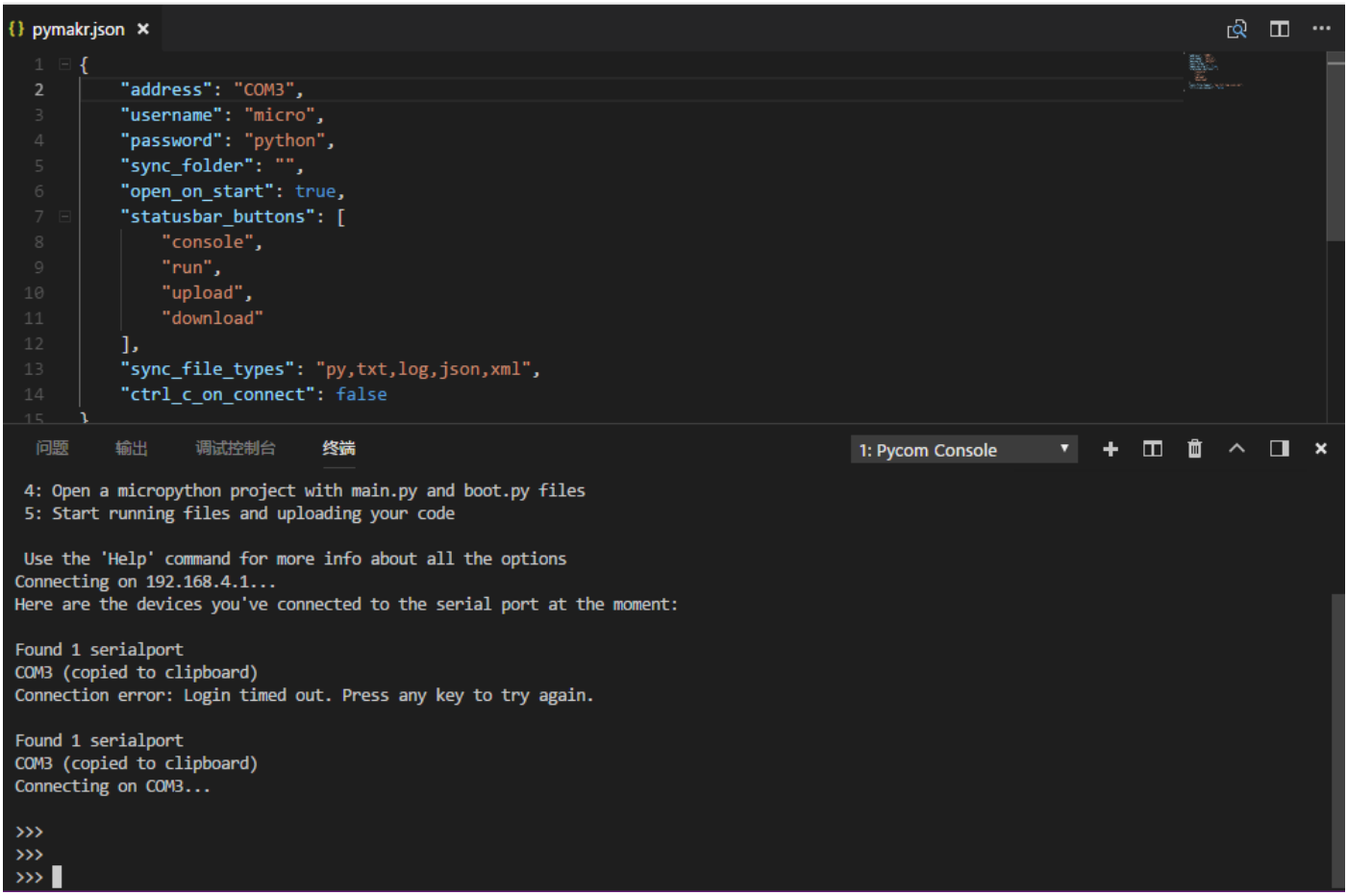
5.1.4 Setup the IDE environment

Just follow the instruction: https://docs.pycom.io/chapter/pymakr/installation/vscode.html
LoPy use the MicroPython as the programming language.
The recommended IDE is: Visual Studio Code(1.24.1) + NodeJS Latest LTS Version (I installed 8.11.3) + Pymakr Plugin for VS code.



Pymakr is the plugin provided from pycom that could used in vs code.
https://marketplace.visualstudio.com/items?itemName=pycom.Pymakr

Just follow the instruction correctly, then will connect LoPy successfully:
This is connected by serial ports:



5.1.5 Connect to LoPy by Telnet

5.1.6 Connect to LoPy by FTP

`url`: ftp://192.168.4.1
`username`: micro
`password`: python

5.1.7 First MicroPython example

LoPy has a lot API:
https://docs.pycom.io/chapter/firmwareapi/

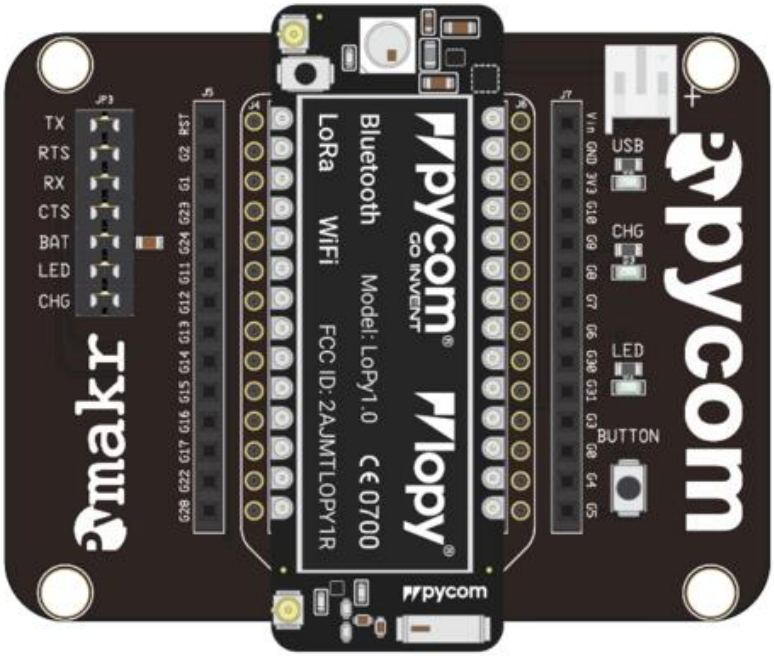
5.2 Summer Week 5: 06-25~06-29

06-25

5.2.1 Update firmware for LoPy by Expansion Board 2.0

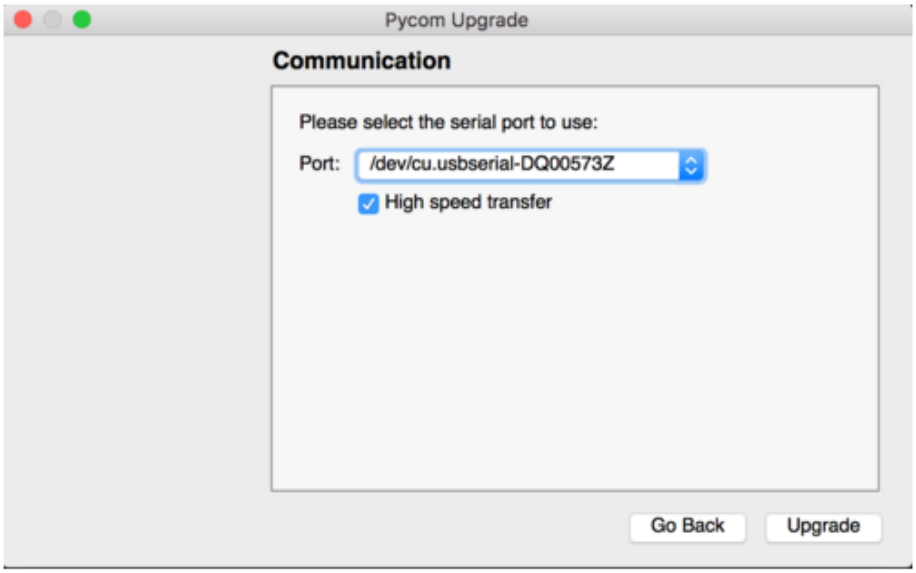
Finish firmware update for all LoPy boards today.
About how to insert LoPy to expansion board 2.0:
https://docs.pycom.io/chapter/gettingstarted/connection/lopy.html

- Look for the reset button on the module (located at a corner of the board, next to the LED).
- Locate the USB connector on the expansion board.
- Insert the LoPy module on the the expansion board with the reset button pointing towards the USB connector. It should firmly click into place and the pins should now no longer be visible.



Since don't need to update the firmware of expansion board 2.0, so just insert Lopy to it. But when update firmware for LoPy, need to use a wire to connect G23 and GND in expansion board.
Follow the link below:
https://docs.pycom.io/chapter/gettingstarted/installation/firmwaretool.html

1. Disconnect your device from your computer
2. Insert module into the expansion board
3. Connect a jumper cable or wire between G23 and GND
4. Reconnect the board via USB to your computer, this puts the device in 'firmware update mode'.
5. Run the Firmware Upgrade tool

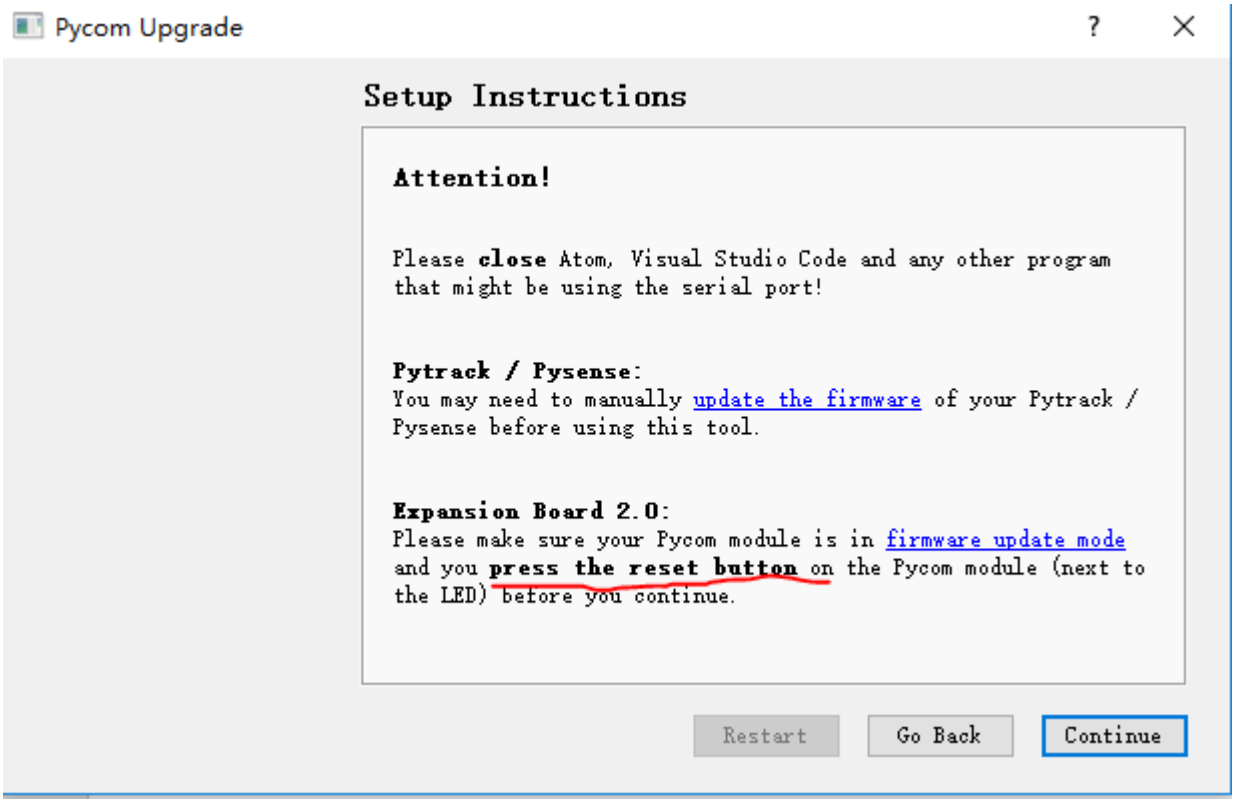


6. Remove the G23 to GND jumper cable/wire
7. Reboot the device (button or power off then on), your device is now ready to use!

If you are having any issues, make sure the **TX and RX jumpers** are present on your expansion board, as the jumpers sometimes come loose in the box during transport. Without these jumpers, the updater will fail.

After you're done with upgrading, you can use the Pymakr Plugins to upload and run programs in your device.

Attention! When using the pycom upgrade tool with expansion board 2.0, when this dialog shows, press the reset button before continue!



5.2.2 LoRa Communication between two LoPys

5.2.3 BLE

5.2.4 GPS example of Pytrack

06-27

Finish Bluetooth communication between two LoPys tests. One as server(which sending the ID all time) and the other one as client(receiving ID and disconnect after get ID).
TO DO list:
1) Still need to know the background knowledge of BLE,GATT
2) There are some other tests need to do:
One server to multiple clients
One client to multiple servers
Multiple servers to multiple clients

06-28

1 I tried to figure out how does BLE work in LoPy:

1) As a server/Central:
Set up an advertisement and send it indefinitely
Add callback function to check the status

bluetooth.callback(trigger=None, handler=None, arg=None)

Creates a callback that will be executed when any of the triggers occurs. The arguments are:

- `trigger` can be either **Bluetooth.NEW_ADV_EVENT**, **Bluetooth.CLIENT_CONNECTED** or **Bluetooth.CLIENT_DISCONNECTED**
- `handler` is the function that will be executed when the callback is triggered.
- `arg` is the argument that gets passed to the callback. If nothing is given the bluetooth object itself is used.

An example of how this may be used can be seen in the **bluetooth.events() method.**

bluetooth.events()

Returns a value with bit flags identifying the events that have occurred since the last call. Calling this function clears the events.

The basic idea of using the callback in server is straight forward:

- Bluetooth.events will return the big flags identifying those three events {Bluetooth.CLIENT_CONNECTED, Bluetooth.CLIENT_DISCONNECTED, Bluetooth.NEW_ADV_EVENT}

To DO list:
1) Check if a LoPy could act as Bluebooth server and client at the same time
2) Multiple thread application test
3) Test BLE and LoRa working at same time

5.3 Summer Week 6: 07-02~07-06

07-02

5.3.1 Multiple threads test

1) Test multiple thread running in the LoPy
Exception catch and exit thread
2) BLE and LoRa running in two different threads
These could work!

To do list:
1) When running BLE and LoRa at the same time, need to deal with a shared buffer that for these two protocol.
BLE has both W/R permission.
LoRa only can read.
2) Still need to deal with the client connect to one

07-05

1) Test GPS sensor in Pytrack, it works but something need to be considered first:

- It need to be used outdoor, I tested in front of UAB, the result shows below:
  The first two may failed? Need to consider the timeout before it could return the right coordinate

```
>>> (None, None)
(None, None)
(42.67696, -73.82353)
(42.67695, -73.82352)
(42.67695, -73.82352)
(42.67695, -73.82353)
(42.67696, -73.82357)
(42.67695, -73.82358)
(42.67696, -73.82359)
(42.67696, -73.82358)
(42.67696, -73.82358)
(42.67696, -73.82357)
(42.67696, -73.82358)
(42.67695, -73.82356)
```

- If it works correctly, it will return a tuple of (lantitude,longitude) and this tuple can be used to locate cattle.
- For the cattle that don't wear pytrack so just add (None,None) in the transferred data
- **Make a strategy that deals with the situation when GPS doesn't work and it could still work as a normal node, and when GPS data recovers just add the data to LoRa package again.**

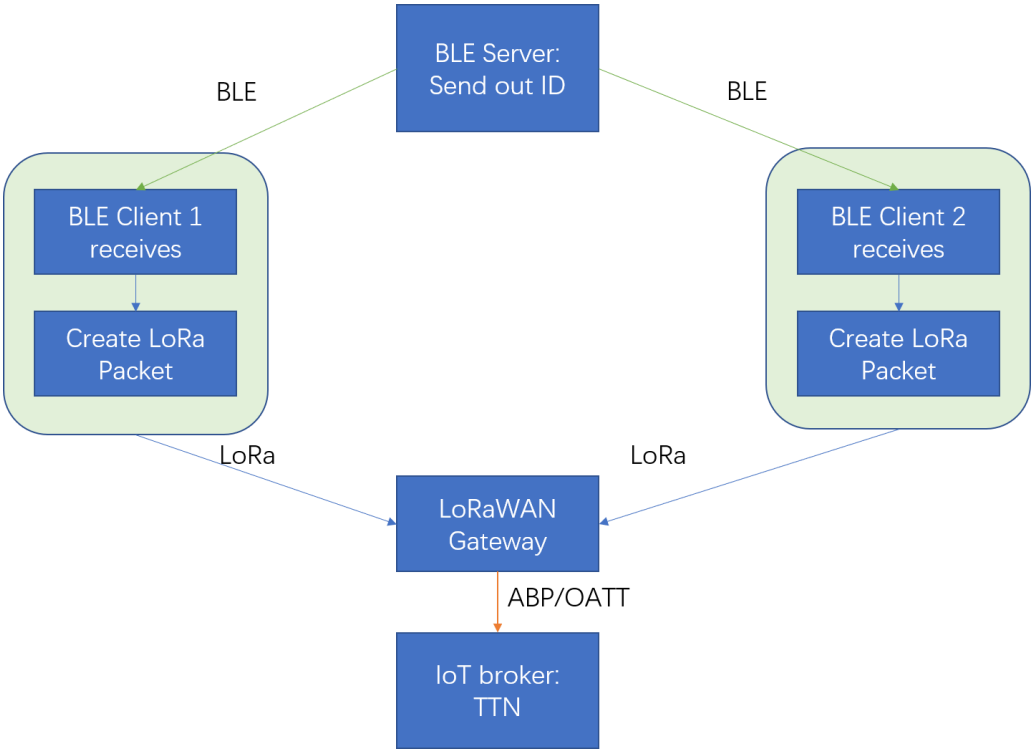
TO DO list:
1) BLE communication still has problem, I have mailed pycom and hope could get their response soon.
2) LoRa gateway testing which includes three parts:
    - Test LoPy wifi connecting to a WiFi network
    - Test LoPy to TTN cloud
    - Test LoPy as LoRa node send data to LoPy gateway

5.4 Summer Week 7: 07-09~07-13

07-09

5.4.1 LoRaWAN start

Today start to test LoRaWAN to TTN server.
Follow the instruction from pycom documentation. Link below:
https://docs.pycom.io/chapter/tutorials/lora/lorawan-nano-gateway.html
The whole structure is:



1 When set up the gateway in TTN, there is something need to notice:

1) Gateway ID:

Use the code bellow to get LoPy nano-gateway id:

```
from network import WLAN
import ubinascii
wl = WLAN()
ubinascii.hexlify(wl.mac())[:6] + 'FFFE' + ubinascii.hexlify(wl.mac())[6:]
```

2) Set the frequency to US 915 in config.py

3) The configuration could be done in TTN web page.

Right now the LoPy gateway could connected to TTN server:

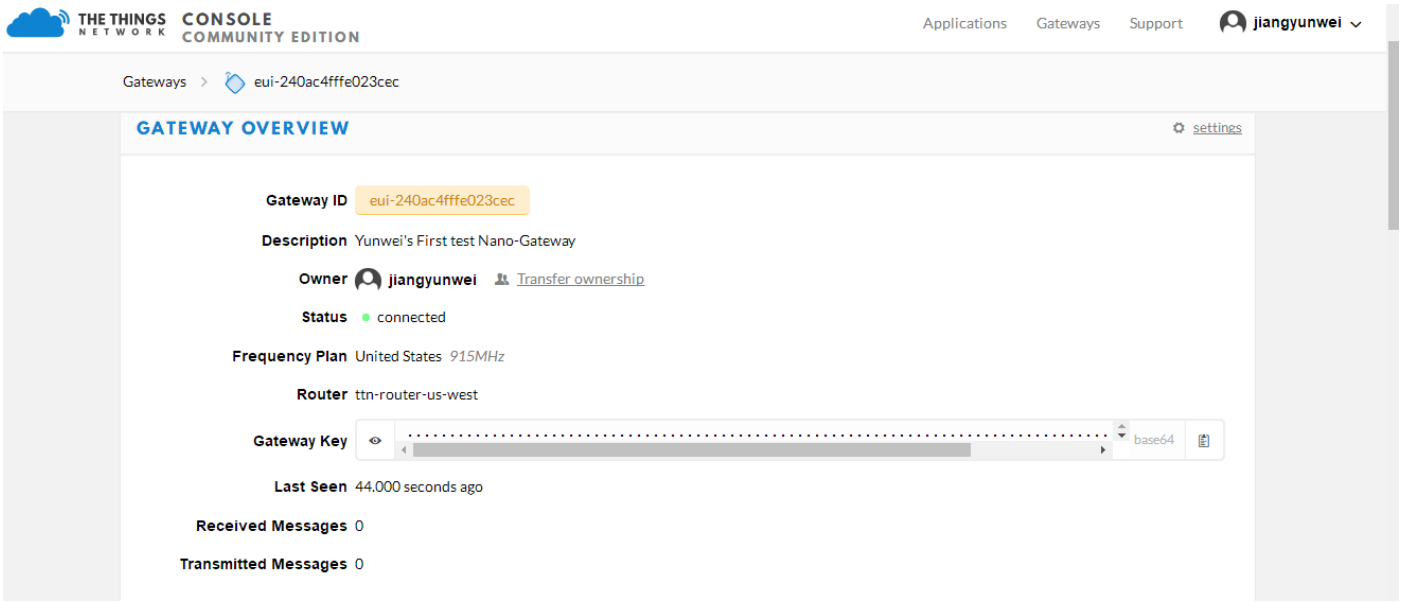


Figure 5.4.1.1 LoPy Gateway connected to TTN server

And the gateway also can push and pull acknowledge from server:

```
[  1026.415] Starting LoRaWAN nano gateway with id: b'240AC4FFFE023CEC'
[  1034.587] WiFi connected to: LAPTOP-BHHF3UMN 4817
[  1034.593] Syncing time with pool.ntp.org ...
[  1034.950] RTC NTP sync complete
[  1035.234] Opening UDP socket to router.us.thethings.network (13.66.213.36) port 1700...
[  1035.248] Setting up the LoRa radio at 903.9001 Mhz using SF7BW125
[  1035.264] LoRaWAN nano gateway online
[  1035.267] You may now press ENTER to enter the REPL
[  1035.446] Push ack
[  1060.430] Pull ack
[  1085.422] Pull ack
[  1095.445] Push ack
[  1110.612] Pull ack
[  1135.489] Pull ack
[  1155.968] Push ack
[  1160.472] Pull ack
```

Figure 5.4.1.2 LoPy Gateway connected to TTN server

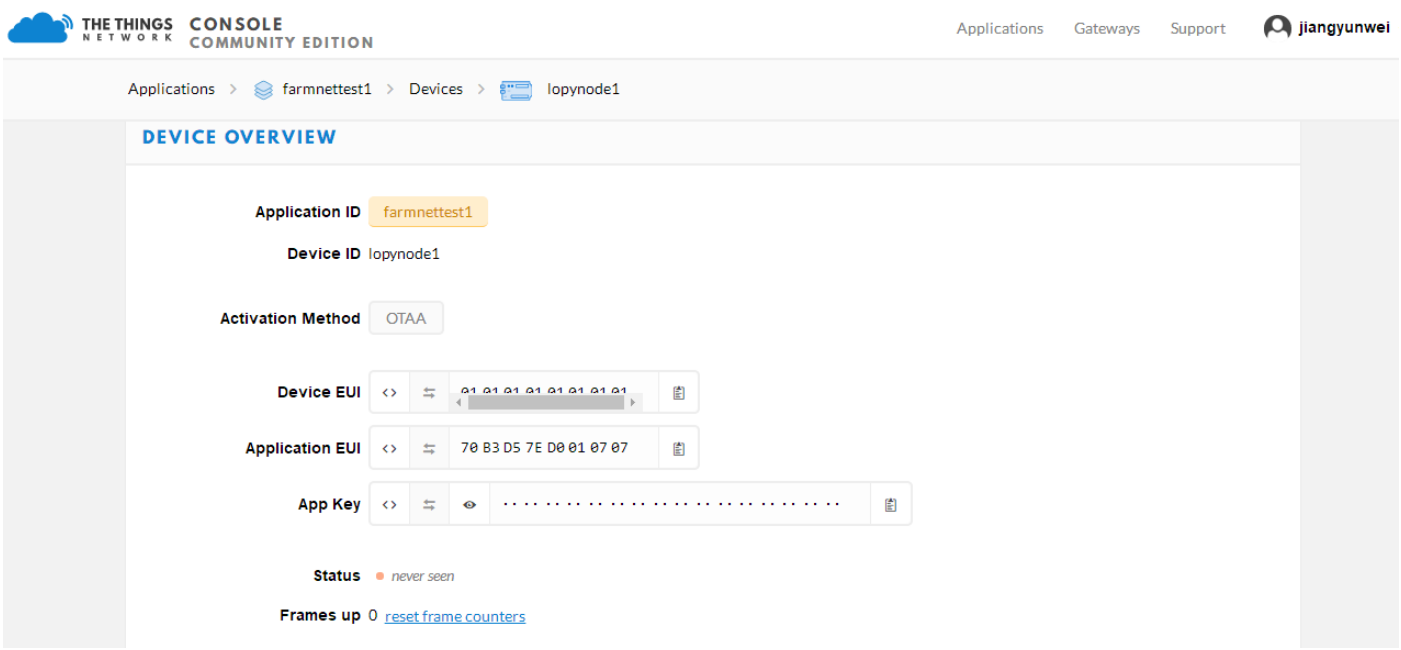
Please notice that: when modify the config.py file in gateway application, the TTN router for US need is **router.us.thethings.network** !!! This part took me hours to solve!!!

The original question in forum:

https://forum.pycom.io/topic/2787/solved-lopy-connection-to-us-ttn/2

> The correct gateway router address for TTN US is `router.us.thethings.network` and not `us-west.thethings.network`
>
> Hope this helps.

2 When set up LoRa node device in TTN, need to first set up an Application:



Notice: I have tried whole afternoon to connect one Lopy node to TTN server through a LoPy nano-gateway, OTAA

can't work, **but when change to ABP, the gateway can finally receive the data and push to server!!!** Still don't know why, but this means the data could sent to server!

TO DO list:
1) Configure the server to analyze the package in application data. Right now just know the server could received data.
2) Find out why it could work in ABP mode
3) Set up three LoPys to send data to gateway.
4) Try to build up a protype network that receiving data by BLE and send to server by LoRa

07-11

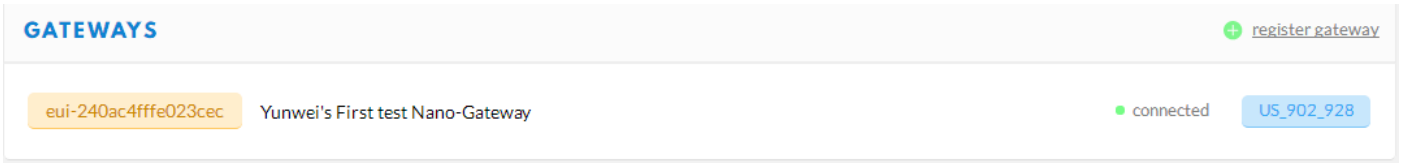
5.4.2 LoRaWAN continue: TTN gateway and LoRa node test

1 Latest Work progress

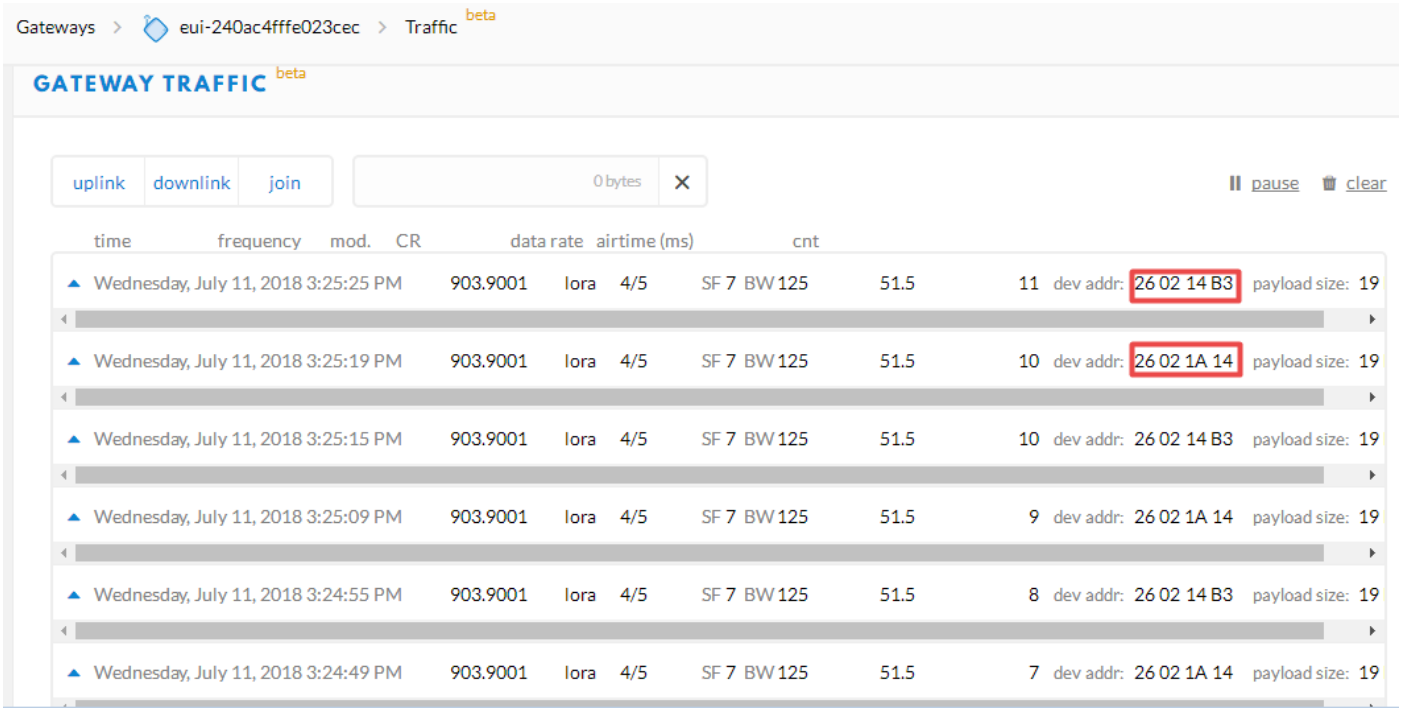
Yesterday I still continued to test the TTN and LoRa node. There are some problems:
1) Although in the "traffic" tag of Gateway in TTN, I can see all the data that pushed by the nano-gateway, but in my application data tag, I can't see any data from the node device··· Have sent the email to Pycom, wait for their reply.
2) There are some details need to figure out:

- The range of default Blue tooth communication
  Since Blue tooth need to be used as short range communication protocol for cattle social graph, the range can't be very wide. But based on the previous reply from Pycom, the range can't be changed right now by user. I have sent email asking about the default range today.
- The range of LoPy work as a LoRaWAN singl channel gateway
  Also need to find out this, since LoRa is used as long range protocol for transferring the collected data from LoRa nodes to server.
- Data frequency
  Need to ask professor.

2 LoRaWAN test: two nodes
I have added a gateway in TTN:

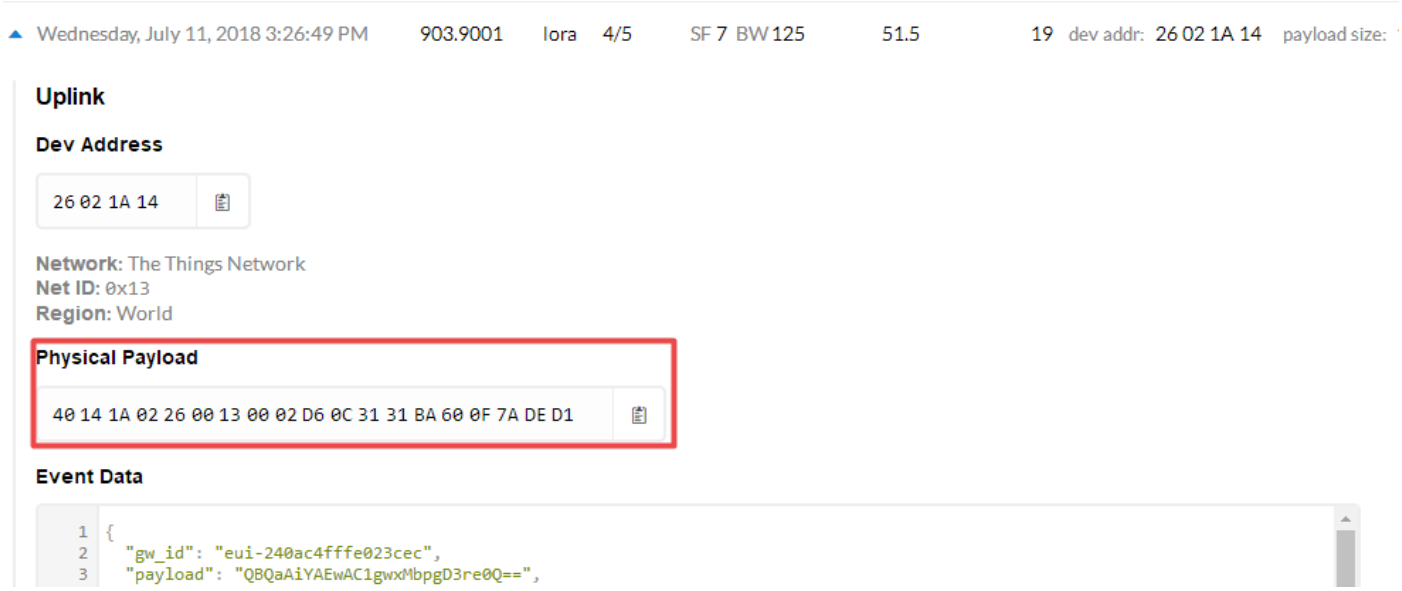


Then I use two Lopy as LoRa nodes that send packet, and the nano gateway pushes all of these uplink request to TTN, result shows below:

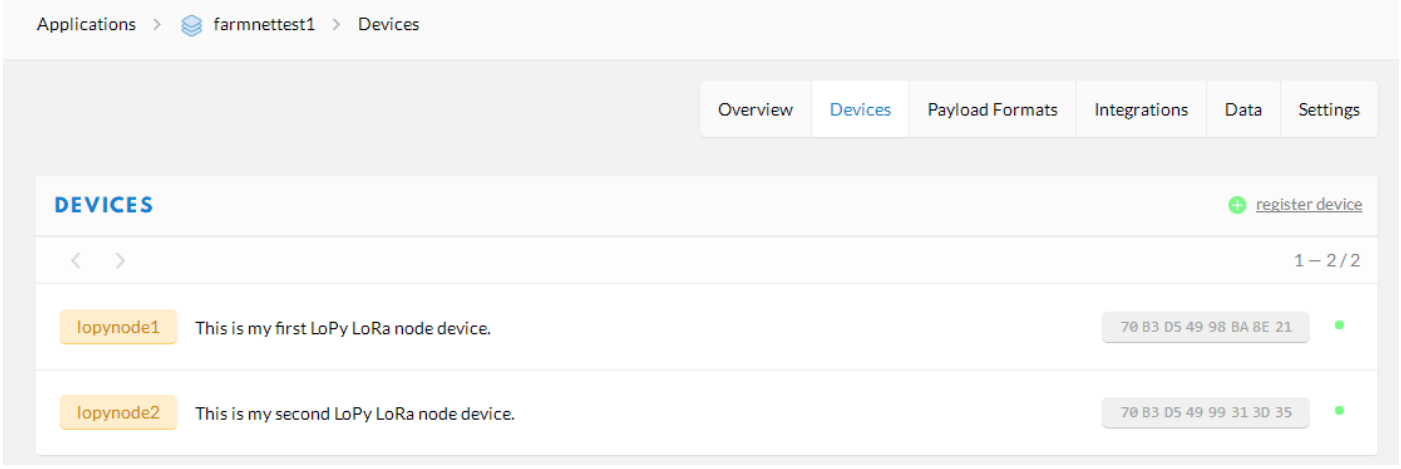


And the details of each data packet:

Notice that it only can show payload as raw data, need to write the decode function in the "Application" console to analyze the raw data in right format. The data could be decoded in json format and then forwarded to the real server that need to store and manage these data.



In my application, I add two LoPy as LoRa node devices:



And the information of each device (the status of device is forwarded to this application and TTN can also check their connection status):

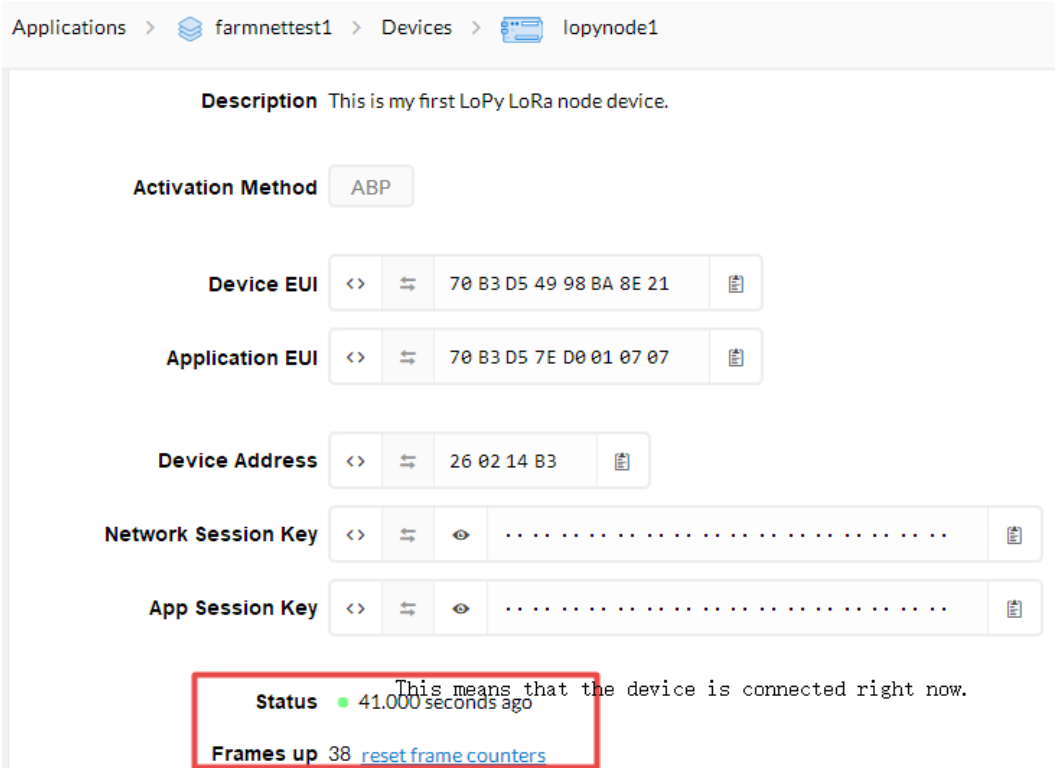
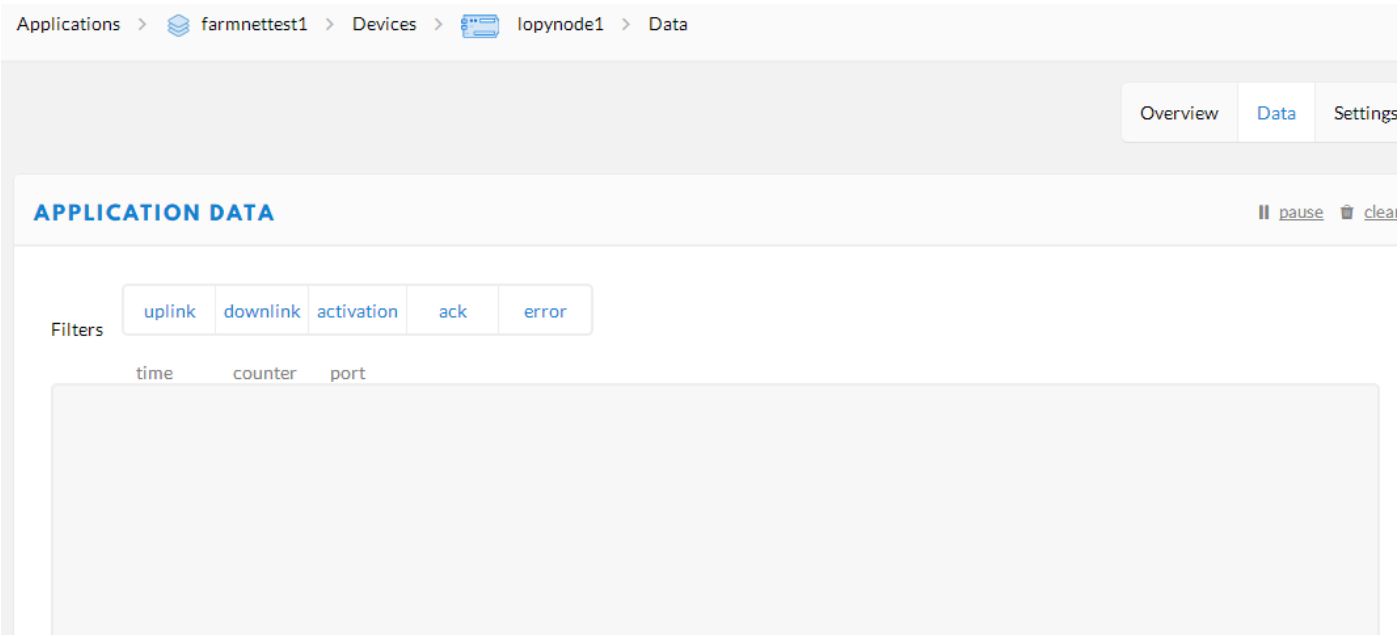


Fig Node1 is connected in

But! OTAA can't work for LoPy nodes. ABP could work but the data won't show in "application data" tag in TTN··· Waiting for Pycom reply.



07-12

5.4.3 BLE test: one server and two clients

Today I finished another Blue tooth communication test.

One server to two clients.
The clients will connect and disconnect to the server alternately, and print out the data received from server.
The function and logic is ok, but one of the client(may be any one) will have an exception after repeating this action for around 125 times…
Just sent email to pycom and wait for reply.

07-13

5.4.4 BLE and LoRa node test

Use one LoPy as Blue tooth server, one as LoRaWAN gateway, one as blue tooth client and LoRa node.

This could work. The one as blue tooth client and LoRa node could received the data from server, and also send this data to LoRaWAN gateway.
So this means that the basic structure of the system:
1) BLE client connects and receives data(ID of other cattle) from server and create data packet

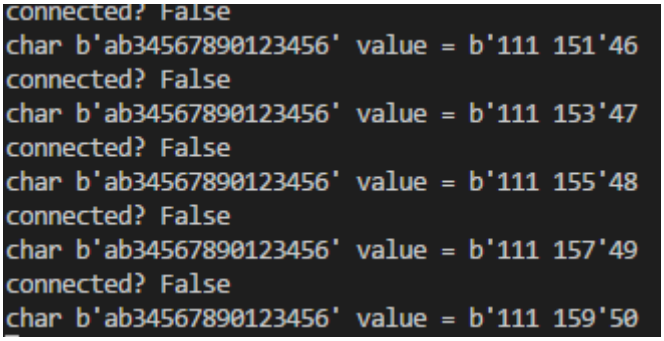


Fig BLE client print message

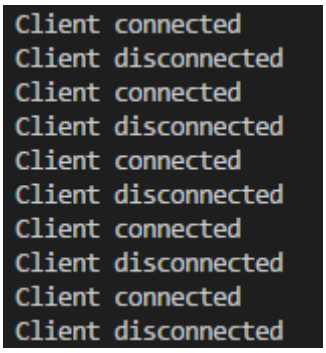


Fig BLE server print message

2) Send the packet by LoRa to LoRaWAN gateway, disconnect from this server, then scan for other servers
3) LoRaWAN gateway push packet to TTN

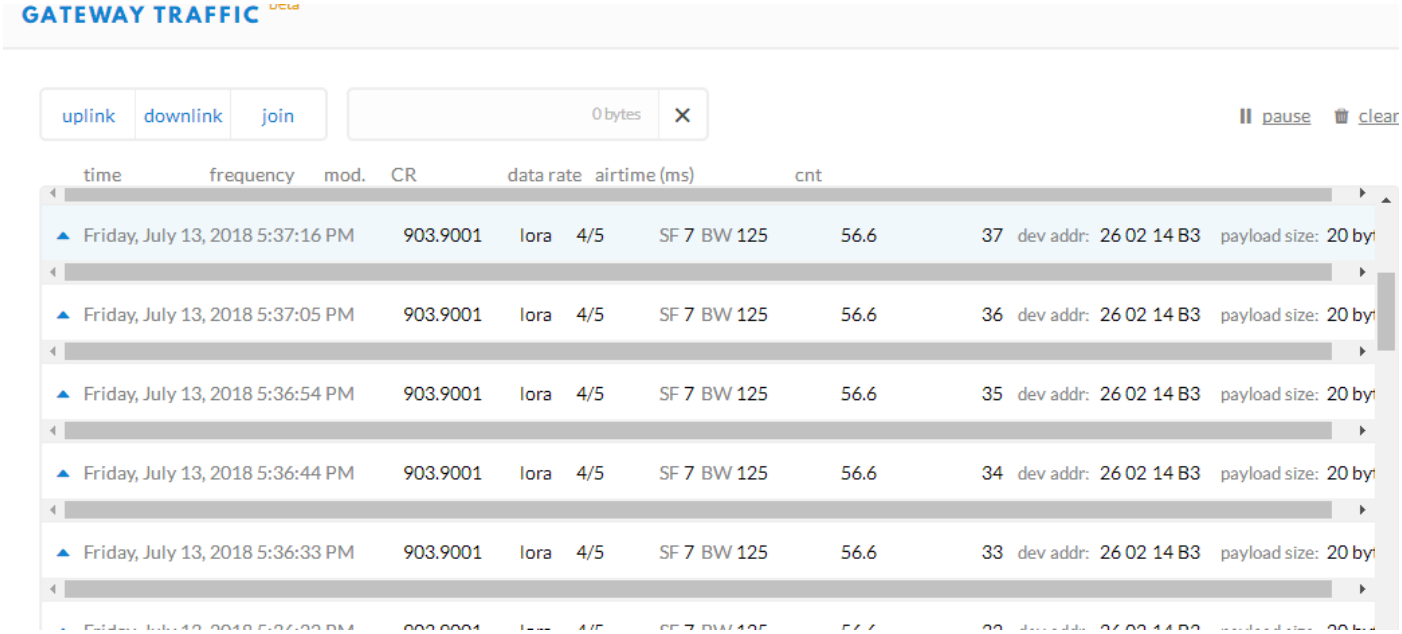


Fig TTN Gateway traffic

This structure could work as expected.

But there are still some problems:

1) BLE client crush after repeating connecting & disconnecting after around 120 times
2) ABP data from LoRa Node can't show in TTN…
3) Data frequency for research

I have send email to pycom asking the first two questions.

The last question need to ask professor.

5.5 Summer Week 8: 07-16~07-20

07-19

5.5.1 Three LoPy BLE communication test

This test will be the protype test for BLE node communication for all of these node devices.

1) Each node works as BLE server and client
2) As server: they set up an advertisement and a service

07-20

5.5.2 LoRaWAN problems

Still has the problems:

1) LoPy maybe can't work as GATT server and client at same time, even start both functions, but only one role could be set at one time?

Or find a way to end the procedure when the client disconnected from server

2) LoRaWAN device

Still can't see the device data in TTN

But when I posted the question in TTN forum, someone answered that maybe the frequency of gateway and device is not the same? Don't know yet… Since for node device, there is only one frequency set, and it's the same to gateway.

TODO list:

1) LoRaWAN device

Maybe the channel need to be set?

- Reset the frame counter for ABP device
- Try to use channel 1(Start from 0) next week.
- Or in the ABP code sample:

Read the documentation again!!!

https://docs.pycom.io/chapter/firmwareapi/pycom/network/lora.html

Set block and then unblock

```
# make the socket blocking
# (waits for the data to be sent and for the 2 receive windows to expire)
s.setblocking(True)

# send some data
s.send(bytes([0x01, 0x02, 0x03]))

# make the socket non-blocking
# (because if there's no data received it will block forever...)
s.setblocking(False)
```

Adding the code below after create lora object:

```
# initialize in LoRaWAN mode
lora.init(mode=LoRa.LORAWAN)
```

Some other similar questions and solutions in TTN forum:

https://www.thethingsnetwork.org/forum/t/datta-node-is-not-shown-in-console-but-meta-data-is-shown/6954/14

2) If still can't solve the problem, then change to LoRaMAC gateway to AWS

https://forum.pycom.io/topic/236/lopy-nano-gateway/38

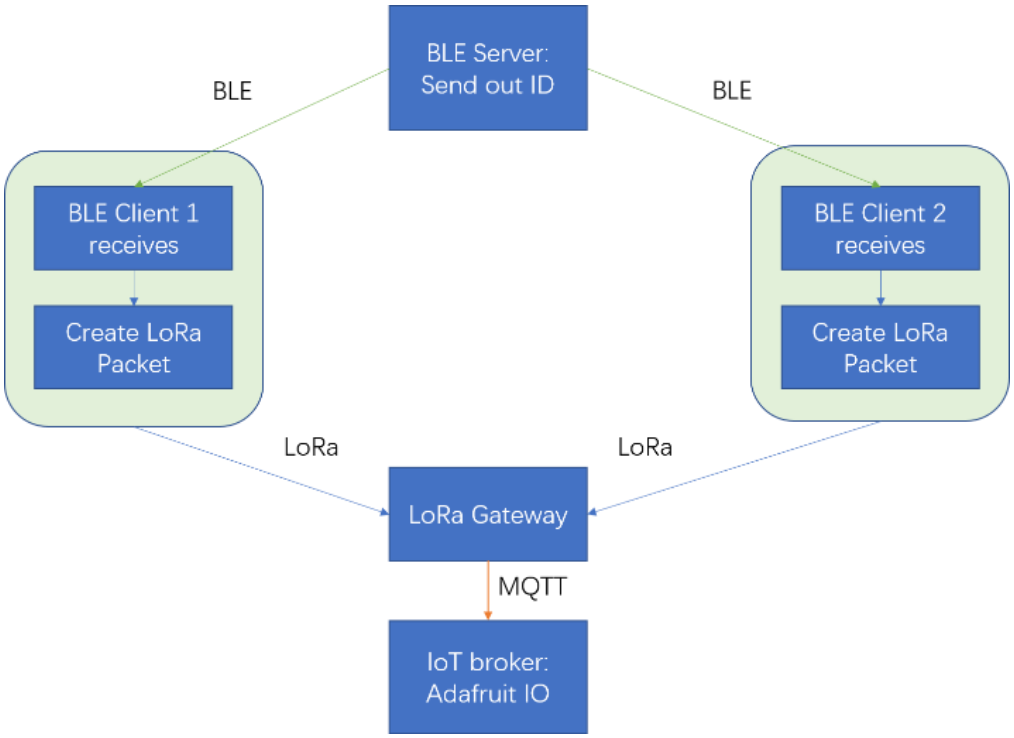
5.6 Summer Week 9: 07-23~07-27

07-24

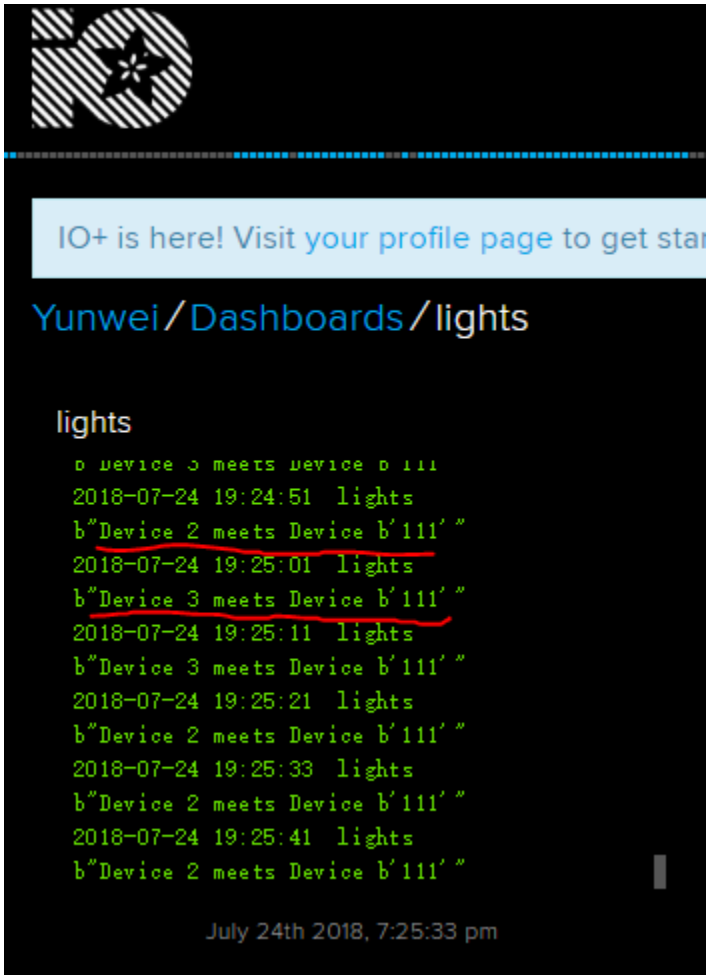
5.6.1 Another prototype structure: LoRa gateway+Adafruit IO

Since I still didn't get the reply from pycom company about 1) device data can't show in TTN and 2) LoPy can't run as BLE server and client at same time problems. So I change the structure to another way:
1) Use LoRa gateway(plain lora) configuration instead of LoRaWAN gateway
LoRaWAN is designed to share and support different LoRa nodes device connecting to the open gateway around, and forward the data to IoT broker as TTN,Loriot. Since we only need to build up a private LoRa network, so a simple LoRa nano-gateway could do the work.
2) Using MQTT to connect to another IoT platform as AWS, AdafruitIO, etc
I have tried to use the AdafruitIO platform to obtain and manage the data forwarded by gateway

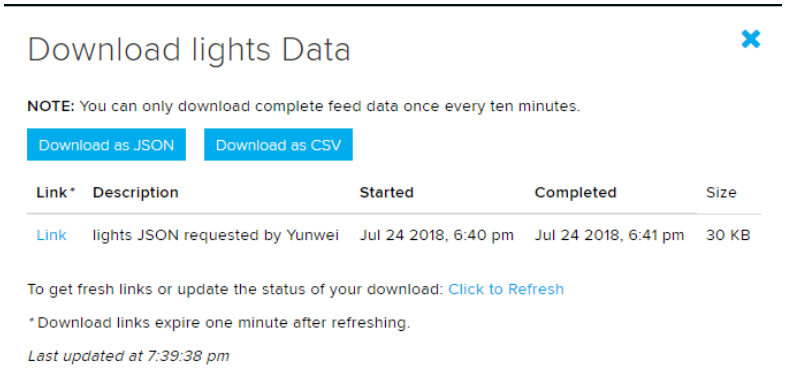
So right now the structure is like below:



There are some screen shots below:
1 The result from Adafruit IO:
Ignore the name lights...
The message stream show the data that sent by gateway:
Means one LoPy get the ID from other server and send to gateway.



The data can be download as json or csv format:



539	0DY09182	b"Device 2 meets Device b'111'"	846871	2018-07-24 23:24:21 UTC
540	0DY091B4	b"Device 3 meets Device b'111'"	846871	2018-07-24 23:24:31 UTC
541	0DY091E6	b"Device 3 meets Device b'111'"	846871	2018-07-24 23:24:41 UTC
542	0DY091H8	b"Device 2 meets Device b'111'"	846871	2018-07-24 23:24:51 UTC
543	0DY091MA	b"Device 3 meets Device b'111'"	846871	2018-07-24 23:25:01 UTC
544	0DY091QC	b"Device 3 meets Device b'111'"	846871	2018-07-24 23:25:11 UTC

2 Gateway print out message:

```
Device: 2 - Pkg:  b'Device 2 meets Device '
b"b'Device 3 meets Device '"
Device: 3 - Pkg:  b"Device 3 meets Device b'111'"
Sending b"Device 3 meets Device b'111'"
b'b"Device 3 meets Device b\'111\'"'
Device: 2 - Pkg:  b"Device 2 meets Device b'111'"
Sending b"Device 2 meets Device b'111'"
Device: 3 - Pkg:  b"Device 3 meets Device b'111'"
b'b"Device 2 meets Device b\'111\'"'
Sending b"Device 3 meets Device b'111'"
Device: 2 - Pkg:  b"Device 2 meets Device b'111'"
b'b"Device 3 meets Device b\'111\'"'
Device: 3 - Pkg:  b"Device 3 meets Device b'111'"
```

3 BLE server:

```
Client disconnected
Client connected
Client disconnected
Client connected
Client disconnected
Client connected
Client disconnected
Client connected
Client disconnected
Client connected
Client disconnected
Client connected
```

4 BLE client + LoRa node:

```
char b'ab34567890123456' value = b'111'
connected? False
char b'ab34567890123456' value = b'111'
connected? False
```

So in this way, the problem can also be solved.

6 Helpful information

6.1 About LoRaWAN gateway configuration

https://docs.pycom.io/chapter/tutorials/lora/lorawan-nano-gateway.html

6.1.1 How to get gateway id?

Using the code below in console:

```
from network import WLAN
import ubinascii
wl = WLAN()
ubinascii.hexlify(wl.mac())[:6] + 'FFFE' + ubinascii.hexlify(wl.mac())[6:]
```

6.1.2 How to get device EUI?

```
from network import LoRa
import ubinascii

lora = LoRa()
print("DevEUI: %s" % (ubinascii.hexlify(lora.mac()).decode('ascii')))
```

6.1.3 Pycom GitHub link

https://github.com/pycom/pycom-libraries

6.2 LoPy helpful instructions

6.2.1 OS and firmware information

```
>>> os.uname()
(sysname='LoPy', nodename='LoPy', release='1.17.5.b6', version='v1.8.6-849-56d00234 on 2018-05-18', machine='LoPy with ESP32', lorawan='1.0.2')
```

6.3 About LoRaWAN

6.3.1 LoRaWAN overview

The contents are from https://www.thethingsnetwork.org/docs/lorawan/

LoRaWAN is a media access control (MAC) protocol for wide area networks. It is designed to allow low-powered devices to communicate with Internet-connected applications over long range wireless connections. LoRaWAN can be mapped to the second and third layer of the OSI model. It is implemented on top of LoRa or FSK modulation in industrial, scientific and medical (ISM) radio bands. The LoRaWAN protocols are defined by the LoRa Alliance and formalized in the LoRaWAN Specification which can be requested on the LoRa Alliance website.

6.3.1.1 Terminology

- **End Device, Node, Mote** - an object with an embedded low-power communication device.
- **Gateway** - antennas that receive broadcasts from End Devices and send data back to End Devices.
- **Network Server** - servers that route messages from End Devices to the right Application, and back.
- **Application** - a piece of software, running on a server.
- **Uplink Message** - a message from a Device to an Application.
- **Downlink Message** - a message from an Application to a Device.

Note: In this project, use Lopy as either node or gateway.

6.3.1.2 End Devices

The LoRaWAN specification defines three device types. All LoRaWAN devices must implement Class A, whereas Class B and Class C are extensions to the specification of Class A devices.

Class A devices support bi-directional communication between a device and a gateway. Uplink messages (from the device to the server) can be sent at any time (randomly). The device then opens two receive windows at specified times (1s and 2s) after an uplink transmission. If the server does not respond in either of these receive windows (situation 1 in the figure), the next opportunity will be after the next uplink transmission from the device. The server can respond either in the first receive window, or in the second receive window, but should not use both windows.

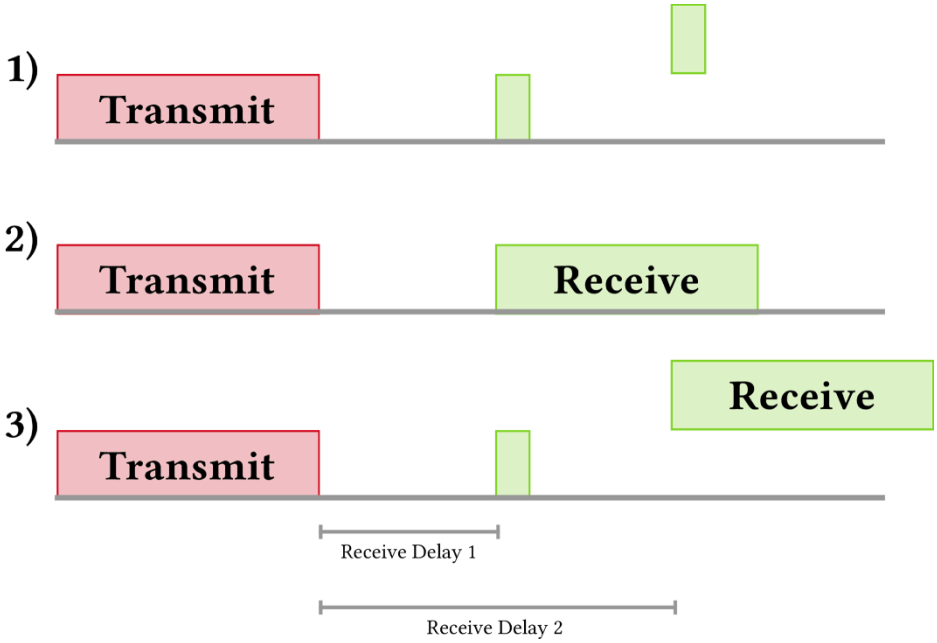


Fig LoRaWAN Class A device

Note: When LoPy as device, it's Class A device.

Class B devices extend Class A by adding scheduled receive windows for downlink messages from the server. Using time-synchronized beacons transmitted by the gateway, the devices periodically open receive windows.

Class C devices extend Class A by keeping the receive windows open unless they are transmitting, as shown in the figure below. This allows for low-latency communication but is many times more energy consuming than Class A devices.

6.3.1.3 Frequency and Bands

US 902-928 MHz

In the United States, LoRaWAN operates in the 902-928 MHz frequency band. Unlike the European band, the US band has dedicated uplink and downlink channels. The band is divided into 8 sub-bands that each have 8x125 kHz uplink channels, 1x500 kHz uplink channel and 1x500 kHz downlink channel. **The Things Network uses the second sub-band (number 1 if you start counting at 0).**

In the US902-928 bands the data rates are as follows:

Data Rate	Configuration	bits/s	Max payload
DR0	SF10/125kHz	980	19
DR1	SF9/125kHz	1 760	61
DR2	SF8/125kHz	3 125	133
DR3	SF7/125kHz	5 470	250
DR4	SF8/500kHz	12 500	250
DR8	SF12/500kHz	980	41
DR9	SF11/500kHz	1 760	117
DR10	SF10/500kHz	3 900	230
DR11	SF9/500kHz	7 000	230
DR12	SF8/500kHz	12 500	230
DR13	SF7/500kHz	21 900	230

6.4 LoRaWAN Data Rate Limitations

The data message for each node device has the limitations in LoRaWAN, there is the topic that in TTN forum:

https://www.thethingsnetwork.org/forum/t/limitations-data-rate-packet-size-30-seconds-uplink-and-10-messages-downlink-per-day-fair-access-policy/1300/42

6.4.1 Uplink limitation policy

https://www.thethingsnetwork.org/forum/t/universal-lora-wan-gateway-limitations-because-physics/1749/8