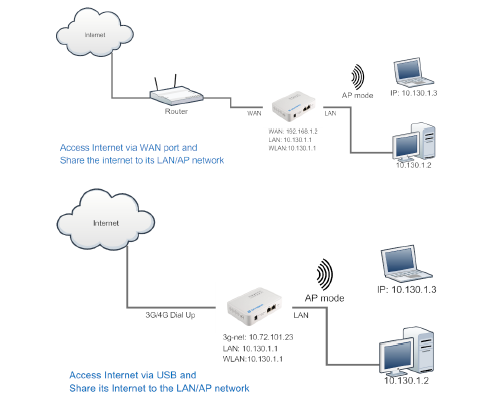
Edit and transfer files:



The LG01 support SCP protocol and has a built SFTP server.

**Question: What is an SCP protocol and what is SFTP server ?**

***Answer: Secure copy protocol****or****SCP****is a means of securely transferring computer files between a local host and a remote host or between two remote hosts. It is based on the Secure Shell (****SSH****) protocol. "SCP" commonly refers to both the Secure Copy Protocol and the program itself (wiki:* [*https://en.wikipedia.org/wiki/Secure\_copy*](https://en.wikipedia.org/wiki/Secure_copy)*)*

***Secure File Transfer Protocol or SFTP****, which stands for SSH File Transfer Protocol, or Secure File Transfer Protocol, is a separate protocol packaged with SSH that works in a similar way over a secure connection. The advantage is the ability to leverage a secure connection to transfer files and traverse the filesystem on both the local and remote system.In almost all cases, SFTP is preferable to FTP because of its underlying security features and ability to piggy-back on an SSH connection. FTP is an insecure protocol that should only be used in limited cases or on networks you trust. Although SFTP is integrated into many graphical tools, this guide will demonstrate how to use it through its interactive command line interface.*

There are many ways to edit and transfer files using these two protocols. One of the easiest is through WinSCP utility.

**Question: What is WinSCP utility ?**

***Answer:WinSCP****(****Win****dows****S****ecure****C****o****p****y) is a free and open-sourceSFTP, FTP, WebDAV, Amazon S3 and SCP client for Microsoft Windows. Its main function is secure file transfer between a local and a*[*remote computer*](https://en.wikipedia.org/wiki/Remote_computer)*. Beyond this, WinSCP offers basic file manager and*[*file synchronization*](https://en.wikipedia.org/wiki/File_synchronization)*functionality. For secure transfers, it uses Secure Shell (*[*SSH*](https://en.wikipedia.org/wiki/Secure_shell)*) and supports the SCP protocol in addition to SFTP.*[*[3]*](https://en.wikipedia.org/wiki/WinSCP#cite_note-3)

*Development of WinSCP started around March 2000 and continues. Originally it was hosted by the*[*University of Economics in Prague*](https://en.wikipedia.org/wiki/University_of_Economics,_Prague)*, where its author worked at the time. Since July 16, 2003, it is licensed under the*[*GNU GPL*](https://en.wikipedia.org/wiki/GNU_General_Public_License)*and hosted on*[*SourceForge.net*](https://en.wikipedia.org/wiki/SourceForge.net)*.*[*[4]*](https://en.wikipedia.org/wiki/WinSCP#cite_note-4)

*WinSCP is based on the implementation of the SSH protocol from [PuTTY](https://en.wikipedia.org/wiki/PuTTY" \o "PuTTY) and FTP protocol from*[*FileZilla*](https://en.wikipedia.org/wiki/FileZilla)*.*[*[5]*](https://en.wikipedia.org/wiki/WinSCP#cite_note-5)*It is also available as a*[*plugin*](https://en.wikipedia.org/wiki/Plug-in_(computing))*for [Altap Salamander](https://en.wikipedia.org/wiki/Altap_Salamander" \o "Altap Salamander)*[*file manager*](https://en.wikipedia.org/wiki/File_manager)*,*[*[6]*](https://en.wikipedia.org/wiki/WinSCP#cite_note-6)*and there exists a third-party plugin for the*[*FAR*](https://en.wikipedia.org/wiki/FAR_Manager)*file manager.*

*(wiki:* [*https://en.wikipedia.org/wiki/WinSCP*](https://en.wikipedia.org/wiki/WinSCP)*)*

After access via WinSCP to the device, user can use a FTP alike window to drag/drop files to the LG01 or edit the files directly in the windows.

\*\*\*The LG01 has a 16MB flash and a 64MB RAM. /tmp /var changes continuously.

The Linux system use around 8MB-10MB flash size which means there is not much room for user to store data in the LG01 flash. \*\*\*User can use an external USB flash to extend the size for storage.

What is Firmware: <https://wiki.ubuntu.com/Kernel/Firmware>

The bridge library is the most important part of LG01-P.

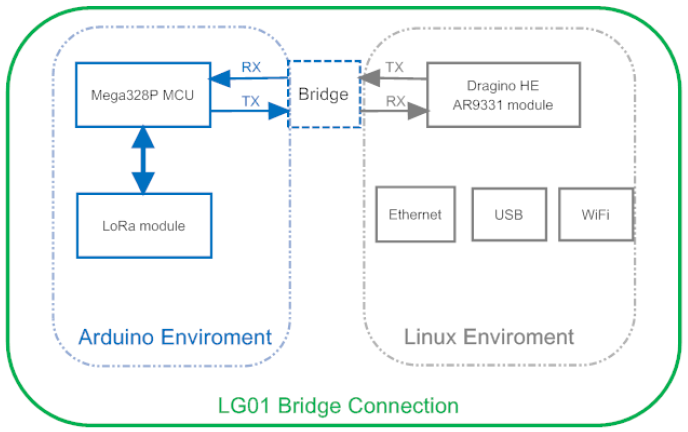


Figure 1: The bridge connection between the Mega328P MCU and Linux

Bridges library defines a mechanism how the MCU talk to the CPU (ar9331). With the bridge library, the MCU can send data to CPU, get commands result from CPU or call commands in CPU. The bridge library use UART port to communicate between MCU and ar9331.

**Question: What is UART port ? what is the difference between MCU and CPU ?**

**Answer:** *check this website:* [*https://www.microcontrollertips.com/microcontrollers-vs-microprocessors-whats-difference/*](https://www.microcontrollertips.com/microcontrollers-vs-microprocessors-whats-difference/)

*and this one:* [*https://stackoverflow.com/questions/973982/what-is-the-difference-between-uart-port-and-serial-port-in-a-computer*](https://stackoverflow.com/questions/973982/what-is-the-difference-between-uart-port-and-serial-port-in-a-computer)

[*https://en.wikipedia.org/wiki/Universal\_asynchronous\_receiver-transmitter*](https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter)

**Example**: Integrate LoRa with RESTFul API

We need to know first what is RESTFul API for that go check this website:

[https://www.quora.com/What-is-a-REST-API#](https://www.quora.com/What-is-a-REST-API)

And this: <https://www.mulesoft.com/resources/api/restful-api>

and this: <https://stackoverflow.com/questions/671118/what-exactly-is-restful-programming>

And you need to know what are the different APIs that cloud servers usually support ? for example the server ThingSpeak supports indeed a RESTful API but what are the other possibilities ? also you need to ask what OPtech’s server supports ?

The point of using a server is to know how to use LG01 to communicate with IoT server via Restful API, so to achieve the goal to upload sensor data to IoT server or download commands from IoT server.

We need to have an account or access to a server and then create a channel and type the channel info. The channel ID is the unique ID to store our data in the server.

We then need to let the LG01 communicate with the channel, we also need the API keys and API method.

1. Uplink Test

How to test the uplink of the data?

To do an uplink test we will try to program LG01 to uplink data to the server (in data sheet the server is ThingSpeak). The data flow in this example is as below:

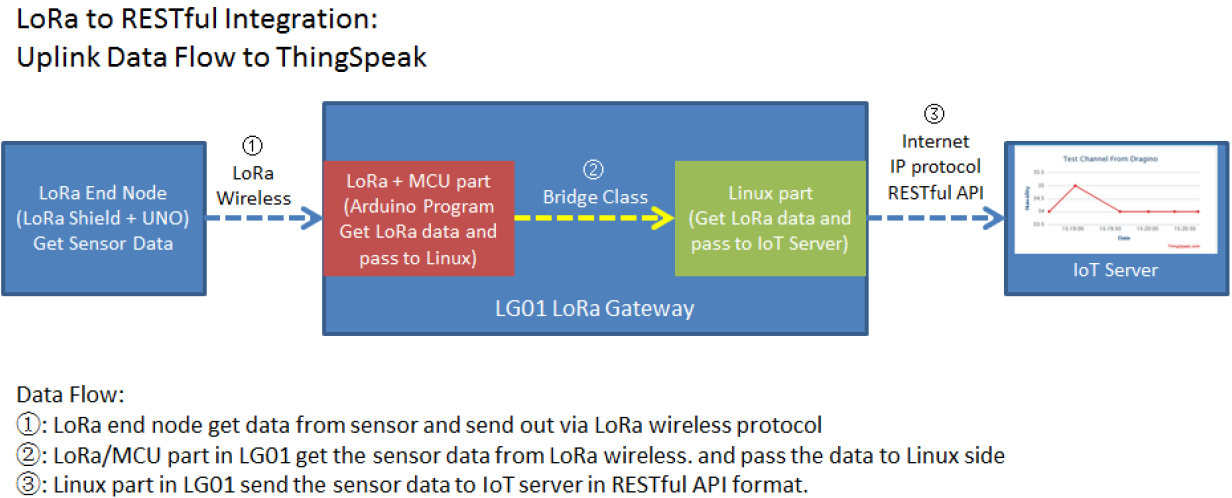


Figure 2: The data flow from the object to the server

To try the 1 we need to upload a simple sketch to the MCU of the gateway using the Arduino IDE check the datasheet page: . To try the 2 go check the datasheet page:

In the following paragraph we need to integrate the three steps together for a complete uplink example. (page: 42)

* **Try Restful API call in web**

look for the API in the server that we will use for the application. We will have next the API call to update the channel feed. In API call we have three variables:

api\_key: Define on which channel you will upload the data

Field: Each channel have max 8 fields(Define which field to be updated)

Value: Field1=0 means update the field1 with value 0

Refer to page: 45

* **Try RESTful API call with LG01 Linux command**

We need to make sure the LG01 has access to internet. We can log in the SSH and ping an internet address and see if it get through. (page 46)

LG01 has built-in Linux tool curl. It is very powerful tool for http communication. We can use this tool to handle RESTful API call in LG01.

The command to update a feed: curl -k “<https://api.thingspeak.com/update?api_key=B9Z0R25QNVEBKIFY&field1=40>” (“”” must be included) page(46)

We should success to use LG01 to uplink data to the server. By following the data sheet we will succeed at doing that but by using the ThingSpeak server.

The curl command is executed in the Linux side. Then we will have to call curl command with sensor data variable in Arduino side. This is through the process class in Arduino: <https://www.arduino.cc/en/Tutorial/Process>

* **Integrate LoRa, Bridge and Curl**

- LoRa End Node: LoRa Shield+UNO+DHT11. The LoRa End node keeps getting temperature and humidity from the sensor and sends out via LoRa periodically.

- LoRa Gateway LG01: Listening on the LoRa wireless channel, while there is new LoRa packet arrives, parse it and send out to IoT Server.

1. Downlink Test

We will try to program LG01 to fetch download data from ThingSpeak, then broadcast this data to local LoRa network. The end node will gets this message and check if they need to do something.

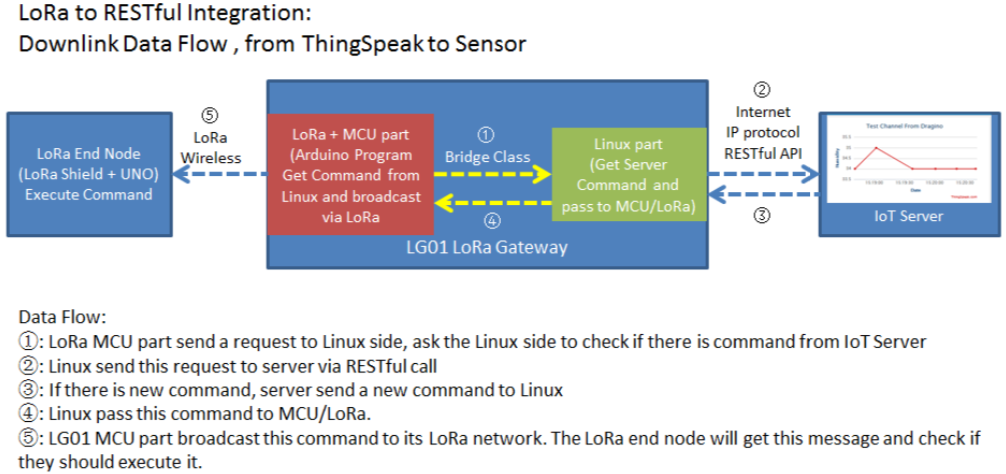


Figure 3:The data flow from server to the object (sensor)

We will try first to do it on PC, and then do it in Linux, and finally integrate it with LoRa.

* **Create Talkback command and try RESTful API call in web**

To do downlink test we need to first create a talkback command in ThingSpeak. As below, From this page, we can get the talkback API key and set the command to be sent to the LoRa End Device.

…

(page: 49)

* **Try RESTful API call with LG01 Linux command**

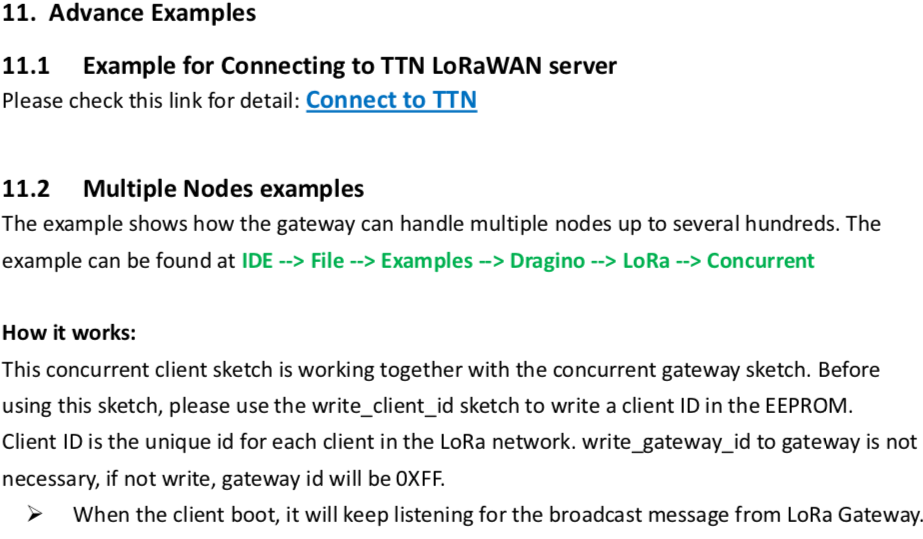
The command to be used is Execute the Next TalkBack Command, with curl, it is  
curl -k "https://api.thingspeak.com/update?api\_key=B9Z0R25QNVEBKIFY&field1=40"

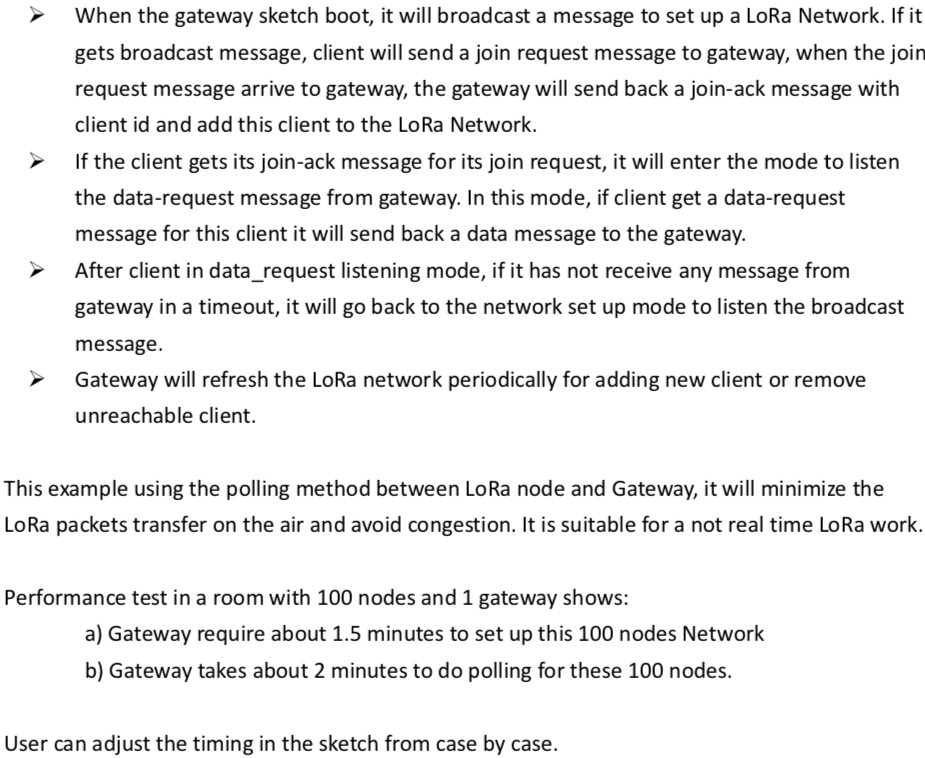
Integrate LoRa, Bridge and Curl

1. **Advance Examples:**

* Example for connecting to TTN LoRaWAN server

<http://wiki.dragino.com/index.php?title=Connect_to_TTN>





**What kind of server the LG01 can support ?**

The Linux side of LG01 is **OpenWrt**, it is open source and users can develop application over it. Basically it can support most IoT servers if use the right API. We have examples for how to connect some servers via typical protocol (MQTT, RESTful) for IoT, MQTT or RESTful. From this link: <https://github.com/dragino/Arduino-Profile-Examples/tree/master/libraries/Dragino/examples/IoTServer> (IoT Server Examples).

