Integrating Cayenne on TTN

RM1xx Series

Application Note *v1.1*

INTRODUCTION

Cayenne is an IoT data processing system that can display transmitted data in visualized form with drag-and-drop configuration. A LoRaWAN server (e.g. The Things Network or TTN) can be set up so that it receives data from end-devices and forwards that data to an external application server in the format it can understand. For the application note, we captured the data (such as temperature) on the DVK-RM191 and sent it to Cayenne to show data in icon and graph.

REQUIREMENT

The following are required for this integration:

- DVK-RM816 or DVK-RM191 with the latest firmware (v101.5.0.9 was used in this test with the RM191)
- UwTerminalX – We recommend v1.09a or later
- *smart*BASIC Cayenne application – This is available from the RM1xx application repository on Github. The following is a direct link to the Cayenne application: *cayenne.mydevice.sb*
- LoRaWAN gateway (e.g. Laird Sentrius RG1xx)

Americas: +1-800-492-2320
Europe: +44-1628-858-940
Hong Kong: +852 2923 0610

OVERVIEW

This application note demonstrates that RM1xx captures data on its DVK and sends it over a LoRa network. The gateway is set up as the packet forwarder pointing to TTN as the destination. The TTN is configured to redirect data from end-devices to Cayenne MyDevice so that it can be displayed in widgets on a browser.

TEST SET-UP

To set up the test, follow these steps:

1. Set up the DVK so that the RM1xx can access the temperature sensor as well as Button1, Button2, and LED5 (Figure 1 and Figure 2).

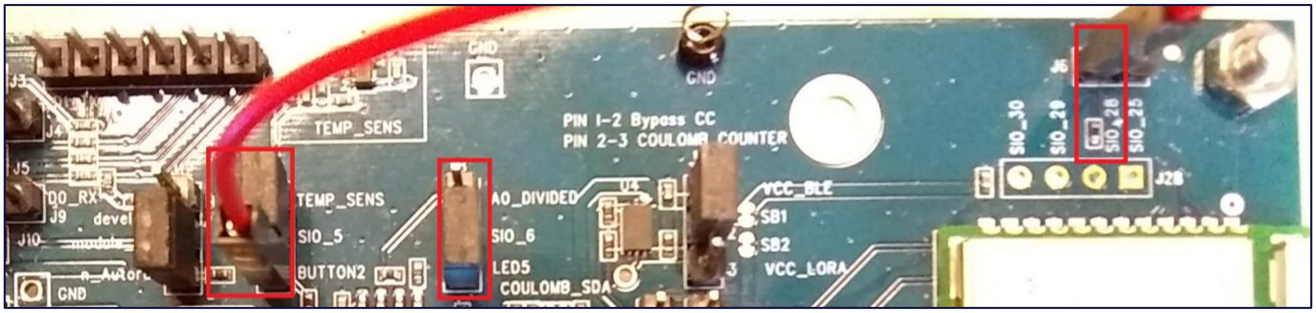

Figure 1: Hardware pin setup (J7pin1-2, J7 pin3 - SIO_28, J8 pin 1-2)

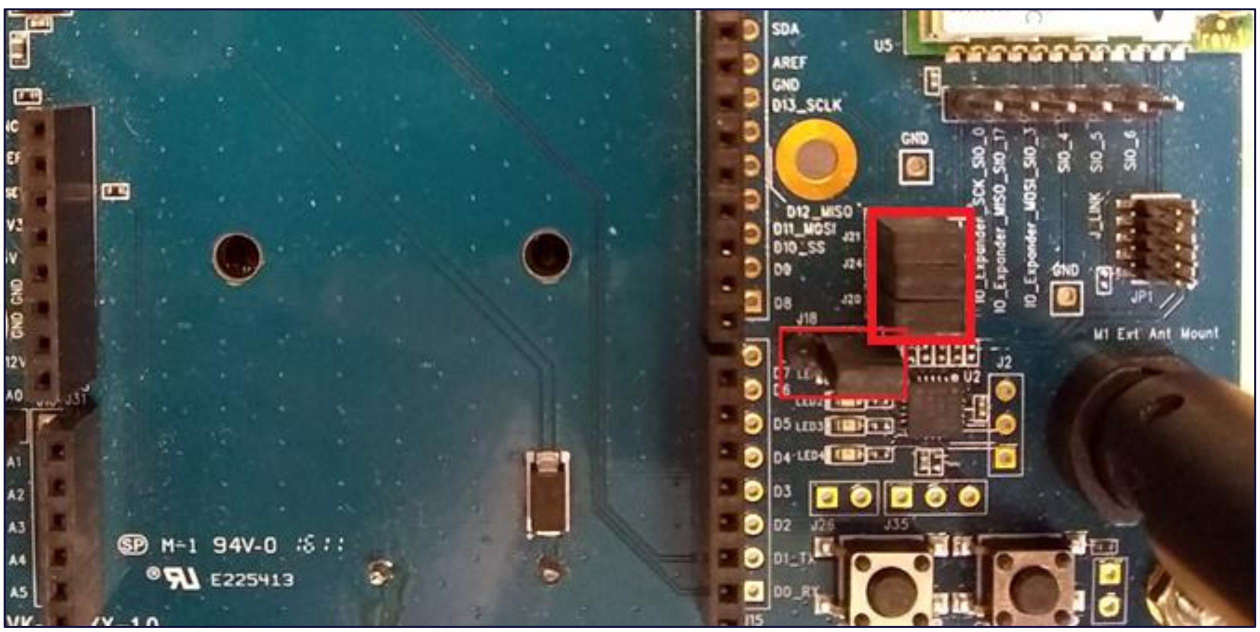

Figure 2: Hardware pin setup (J18 pin 2-3, J20, J21, J24)

2. Set up TTN and Cayenne by following these steps:
   a. Register for an account on cayenne https://cayenne.mydevices.com/cayenne/dashboard/start and verify your account.
   b. Sign up for a TTN account and login on https://console.thethingsnetwork.org/
   c. Set up your gateway, application, end-devices on TTN
      - https://www.thethingsnetwork.org/docs/gateways/registration.html

- https://www.thethingsnetwork.org/docs/applications/add.html
- https://www.thethingsnetwork.org/docs/devices/registration.html

d. Add a new device on Cayenne at the following location: **LoRa > The Things Network > Cayenne LPP**. Paste the dev EUI here from TTN and add it.
e. On TTN, return to the application > Integrations tab and click **Add Integration**.
f. Select *Cayenne*, then enter the process ID you get from the Cayenne page URL after the /lora/ part of the URL (for example, the highlighted part in https://cayenne.mydevices.com/cayenne/dashboard/lora/3e795080-xxxx-xxxx-xxxx-51a105d3afc2)
g. Go to TTN's application payload formats tab and change it from custom to *Cayenne LPP*.

3. Set up the RM1xx by following these steps:
   a. Open UwTerminalX and configure the AppEUI, DevEUI, and AppKey with the values from TTN by using the following command respectively for OTAA. (For using ABP, refer to LoraWAN Keys/ID document)
      - at+cfgex 1010 "<AppEUI>"
      - at+cfgex 1011 "<DevEUI>"
      - at+cfgex 1012 "<AppKey>"

   b. (RM191-US and RM191-AU only) Set up a sub-band to be used with at+cfg 1001 and at+cfg 1002. For example, use the following to use sub-band 2:
      - at+cfg 1001 2 – This sets the sub-band 2 among available sub-band options ranging from 1 to 8.
      - at+cfg 1002 1 – This decides what key a sub-band can be set up with, as shown in Table 1. Alternatively, at+cfgex 1009 can be used to set a channel mask as shown in Table 2. In this case, at+cfg 1002 2 should be used to go along with it.

Table 1 Channel Map select type

at+cfg 1002	Action
0 (default)	Stack default – All channels enabled
1	Use at+cfg 1001
2	Use at+cfgex 1009

Table 2 ChannelMask commands

Sub-Band	Frequency Range (MHz)		Channels	Command
	US	AU		
1	902.3–903.7	915.2-916.6	0-7	at+cfgex 1009 "000100000000000000ff"
2	903.9–905.3	916.8-918.2	8-15	at+cfgex 1009 "0002000000000000ff00"
3	905.5–906.9	918.4-919.8	16-23	at+cfgex 1009 "00040000000000ff0000"
4	907.1–908.5	920.0-921.4	24-31	at+cfgex 1009 "000800000000ff000000"
5	908.7–910.1	921.6-923.0	32-39	at+cfgex 1009 "0010000000ff00000000"
6	910.3–911.7	923.2-924.6	40-47	at+cfgex 1009 "00200000ff0000000000"
7	911.9–913.3	924.8-926.2	48-55	at+cfgex 1009 "004000ff000000000000"
8	915.5–914.9	926.4-927.8	56-63	at+cfgex 1009 "0080ff00000000000000"
All bands	902.3–914.9	915.9-917.1	0-63	at+cfgex 1009 "00ffffffffffffffffff"

   c. Reset via *atz*.

d. If ABP was chosen, *cayenne.mydevice.sb* must be modified as described in comment below. If running with OTAA, skip this step.

```
        //OTAA. To use ABP, set with LORAMAC_JOIN_BY_PERSONALIZATION or 0
#define JOIN_TYPE         LORAMAC_JOIN_BY_REQUEST
```

e. Right-click on UwTerminalX and click **XCompile + Load + Run**.
f. Choose the *smart*BASIC Cayenne application (cayenne.mydevice.sb can be downloaded from https://github.com/LairdCP/RM1xx-Applications).
g. If successfully downloaded, the application starts immediately. RM1xx joins a LoRaWAN network and then transmits data shown on Cayenne periodically.

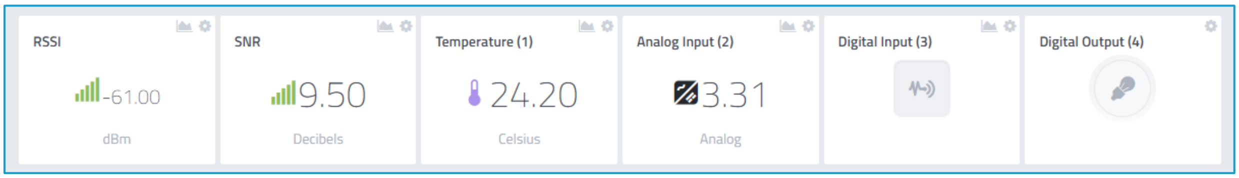

Figure 3 Displayed data on Cayenne

In Figure 3…

- *Analog Input (2)* is the power supply/battery in voltage.
- *Digital input (3)* is Button1 on the DVK.
- *Digital Output (4*) is the LED5 status on the board.

These names can be changed by accessing their applicable settings located at the right upper corner of each widget. They can be changed to look like those in Figure 4, for example.

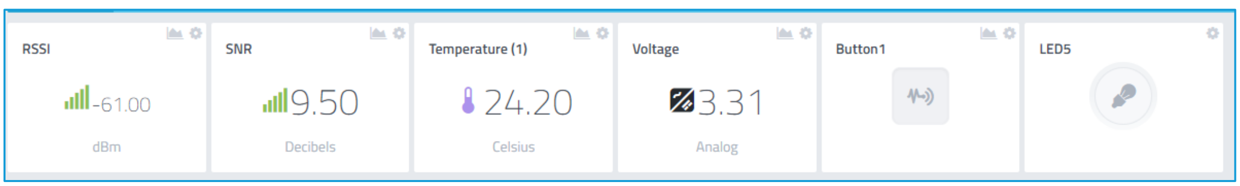

Figure 4: Renamed widgets on Cayenne

If Button2 on DVK is clicked, it toggles the LED5 status and includes it in the transmitted data to Cayenne (Figure 5).

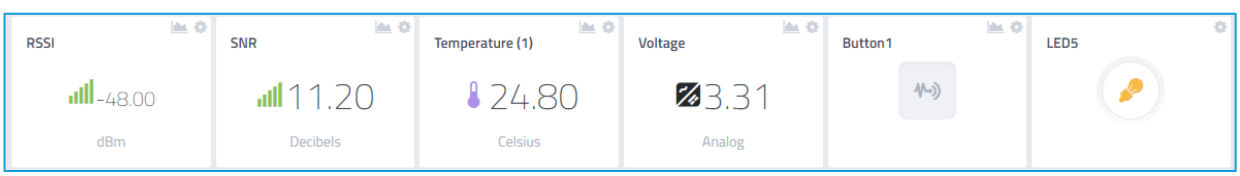

Figure 5: LED5 turned on

If Button1 is being pressed while RM1xx is transmitting data, Button1 status is updated on Cayenne.

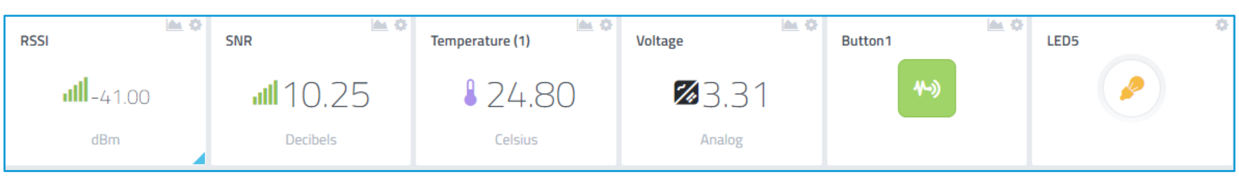

Figure 6: Button1 pressed

You can set notification on *trigger* which is located on right corner of each widget. The allows you to receive an email or text message when certain condition is met. Figure 7 shows a setup for notification when temperature is above 20°C. Figure 8 shows a sample text message notification.

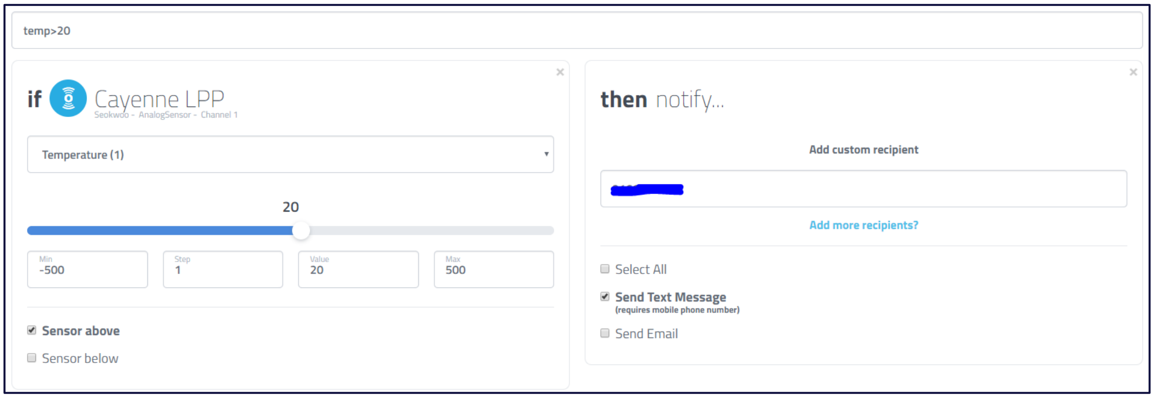

Figure 7: Notification setup for temperature above 20°C

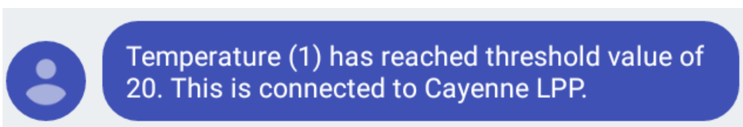

Figure 8: Arrived text message on mobile phone

Cayenne Data Format

In *cayenne.mydevice.sb*, the LoraPost function explains how data should be formatted before it is sent to Cayenne server. Data for each entity (e.g. for sensor) consists of three parts like the following:

- First byte – Data channel. Individually classifies each sensor in a frame
- Second byte – Data type (e.g. temperature)
- N bytes after second byte – Data value where N (the size of data) varies across different data types

For example, in the application, the temperature sensor data is comprised of tempType (\01\67) and tempVal (in hex). In this example:

- *0x01* means that 01 of data channel is used
- *0x67* means that the temperature sensor is the type of data to be used

The following data is the value for tempVal as formatted in the required Cayenne data-type, as seen in Table 3.

```
function LoraPost()
    if (okToTx == 1) then
        //There is available duty cycle
        dim tempType$, tempVal$,txData$, powerVal$, bttn1State$, led5State$
        dim atm, theoreticalMax, actualMax

        //Data format for Cayenne - first byte is data channel and second byte is data type
        tempType$="\01\67"    // "\67" : Temperature
        atm = ReadTemp()
        EndianSwap(atm)
        rc=bleencode16(tempVal$,atm,0)

        powerVal$ = "\02\02"    // "\02" : Analog input
        atm = ReadPwrSupplyMv()/10
        EndianSwap(atm)
        rc=bleencode16(powerVal$,atm,2)

        atm = ReadBtnState()
        bttn1State$ = "\03\00"    // "\00" : Digital input
        rc=bleencode8(bttn1State$,atm,2)

        led5State$ = "\04\01"    // "\01" : Digital output

        if (led == 0) then
            rc=strsetchr(led5State$,0,2)
        else
            rc=strsetchr(led5State$,1,2)
        endif

        txData$ = tempType$ + tempVal$ + powerVal$ + bttn1State$ + led5State$
```

Table 3 Cayenne Data Type

Type	Data type (Hex)	Data size(Bytes)
Digital Input	0	1
Digital Output	1	1
Analog Input	2	2
Analog Output	3	2
Illuminance Sensor	65	2
Presence Sensor	66	1
Temperature Sensor	67	2
Humidity Sensor	68	1
Accelerometer	71	6
Barometer	73	2
Gyrometer	86	6
GPS Location	88	9

Cayenne Payload Structure reveals a full table for data format of each type.

Resources

- Cayenne Payload Structure – https://mydevices.com/cayenne/docs/lora/#lora-cayenne-low-power-payload
- RM1xx Setup Guides – http://www.lairdtech.com/products/rm1xx-lora-modules#documentation-tab
- RM1xx Sample Applications – https://github.com/LairdCP/RM1xx-Applications

Revision History

Version	Date	Notes	Approver
1.0	24 Oct 2017	Initial Release	Jonathan Kaye
1.1	25 Oct 2017	Updated Cayenne links. Minor grammatical edits.	Jonathan Kaye