3 Payload Decoding, IoT Dashboards, IoT IDE:s and Simple Data Mining

Table of Contents

3.0 Basics of Payload Decoding

In this lab will continue from lab 2 with The Things Network (TTN) and our virtual node/device publisher we created. For settings and installation etc. use the instructions in lab2 which should be finished in order to perform lab3.

We will continue on our C# code MQTT application and improve its functionality to handle real binary payloads and continue writing unit tests. We are going to use both our own payloads and standard payloads as CayenneLPP.

Requirements
C# compiler, VSCode, Arduino, Python, C/C++, a web-browser and some SQL tools.

To get started look thru the links in Learn about the topics below and continue refreshing your Unit Test knowledge from previous courses.

Cayenne LPP:

- https://loranow.com/cayennelpp/
- https://developers.mydevices.com/cayenne/docs/
- https://www.thethingsnetwork.org/docs/devices/arduino/api/cayennelpp.html

3.1 Your own payload decoding in C#

Task

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE
Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

Instead of sending a very inefficient string over LoRa/LoRaWAN we are now going to send only the binary values in IEE-754 format.

Navigate to: https://www.h-schmidt.net/FloatConverter/IEEE754.html and select 3 values for your sensor data.

In the LoRaWAN demo simulator at: **https://labs.mbed.com/simulator** which we used used in lab 2 or your own installation of the simulator.

Comment out the uint8_t tx_buffer code and create your own hard coded message, as for example:

```
// send temperature, humidity and led float numbers as 32 bit IEE-754 in hex
uint8_t tx_buffer[] = { 0x42, 0x02, 0x33, 0x33, 0x42, 0x6A, 0xCC, 0xCD, 0x3E,
0xCC, 0xCC, 0xCD };
```

As you see the message becomes significantly smaller. Only 12 bytes will be used. Remember to adjust the buffer length in the code.

If the simulator does not take your changes press Ctrl-F5. You may have to connect several times since the simulator sometimes does not connect to TTN.

Write the code needed in your C# MQTT program to support decoding of the values and print them out as before (remember the byte order of your format). Example:

```
DotNet-Core 3.1 With The Things Network (TTN) And The C# Library ALoRa

Connected to TTN and listening for MQTT LoRa/LoRaWAN TTN Messages with Virtual Sensor

Press return to exit!

M_client_MqttMsgSubscribed: uPLibrary.Networking.M2Mqtt.MqttClient

Message Timestamp: 2019-10-05T07:31:47, Device: node, Topic:
hajo66-test-one/devices/node/up, Payload: 42-02-33-33-42-6A-CC-CD-3E-CC-CC-CD
Payload Converted To: Temp: 32.55, Hum: 58.7, Led: 0.4
Device location is Latitude: 60.487545, Longitude: 15.409067, Altitude: 160m
```

When writing the code for decoding the hex message extend your help class for this with static methods. You need to converter methods for this. Create unit tests for your methods, for example:

- `FloatIEE754_ShouldConvertValidValues_WhenDecodingInput`
- `ByteSwapX4_ShouldConvertValidValues_WhenDecodingInput`
- `etc.`

So it throws exceptions and can validate a correct hex to float IEE-754 conversion. Get some fun hex examples to test with here: https://en.wikipedia.org/wiki/Hexspeak

Report:

Show that you have a working and correct implementation of the task above with Unit Tests for your converter methods. Hand in your source code and a screen recording over mission accomplished.

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE

Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

3.2 Payload decoding with Cayenne LPP

Now we are going to use the Cayenne LPP standard for our LoRa/LoRaWAN message. There are not many libraries available for this in C#. I used: https://github.com/CSharpFan/cayenne-lpp-parser in my solution which I also attached to the lab.

To understand how Cayenne LPP works examine the bundled Python files: decode_cayennelpp_test.py and encode_cayennelpp_test.py which use the PyCayenne LPP library at: https://github.com/smlng/pycayennelpp

You must look into some Cayenne LPP documentation for how the bytes in the payload is structured for various types. For example: https://github.com/myDevicesIoT/cayenne-docs/blob/master/docs/LORA.md or other resource.

3.2.1 Part 1 with TTN and HTTP Integration

Task
Hard code your uint8_t tx_buffer[] payload in the simulator with a similar content as before, but now in Cayenne LPP format. See the attached Python files for generating suitable hex codes.

Now review the payload functions at TTN which can be used to decode the payload at once. For usage and help in TTN see How to: Payload Functions: https://www.youtube.com/watch?v=nT2FnwCoP7w&list=PLM8eOeiKY7JVwrBYRHxsf9p0VM_dVapXl

We are not going to do much. In short, in the TTN console press the Payload Formats tab and select Payload Format => Cayenne LPP instead of Custom and save the setting.

TTN is still going to send the coded Cayenne LPP format to your C# MQTT program, but if you navigate to the Data tab in TTN and expand the last message you will notice that the Fields section now is populated with decoded data in JSON format.

Navigate to your RequestBin.com: https://requestbin.com URL that you set up in lab2 with HTTP Integration: https://www.thethingsnetwork.org/docs/applications/http/.
Review that the latest POST you have done have the additional key "payload_fields" and your decoded data.

Report:
Show that you have a working implementation of the task above. Hand in screen shots or a recording over mission accomplished.

3.2.2 Part 2 with C, (Cayenne Lpp optional) and C# library

Task
You can keep the hard coded uint8_t tx_buffer[] payload in the simulator from task 3.2.1.

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE
Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

Then receive the values in your C# MQTT application from earlier tasks. Begin with creating a Unit Test for your Cayenne LPP payload, so you know it will work in your production code.

My message (output from C#) example:

```
Message Timestamp: 2019-10-05T21:32:10, Device: node, Topic:
hajo66-test-one/devices/node/up, Payload: 0F-67-01-10-0F-68-81-0F-03-00-28
CayenneLPP Payload Converted To: Temp: 27.2, Hum: 0.645, Led: 0.40
Device location is Latitude: 60.487545, Longitude: 15.409067, Altitude: 160m
```

[optional fun part]
If you want to do it without hard coding the payload you can use the lab attached library files in the lorawan_lib.zip file (they got some adjustments by me) with the Mbed simulator. This however means that you must have to installed the simulator on your own computer as described in lab 2.

Now create the code to transform the values from the sensors to Cayenne LPP format and send it. Help: https://www.thethingsnetwork.org/docs/devices/arduino/api/cayennelpp.html
51 bytes is the maximum Cayenne Lpp payload size.

```
#include <CayenneLPP.h>
CayenneLPP lpp(51);

// send with CayenneLPP
lpp.reset();
lpp.addTemperature(1, sht31.readTemperature());
lpp.addRelativeHumidity(2, sht31.readHumidity());
lpp.addAnalogOutput(3, led.read());
lpp.addGPS(4, 60.48726, 15.40954, 150);

printf("Sending %d bytes: \"%s\"\n", lpp.getSize(), lpp.getBuffer());
int16_t retcode = lorawan.send(MBED_CONF_LORA_APP_PORT, lpp.getBuffer(),
lpp.getSize(), MSG_UNCONFIRMED_FLAG);
```

If you do it this way restart the Mbed web app every time after you have compiled a new LoRa binary. See the mbed-simulator-install_v2.docx for more info.

Report:
Show that you have a working implementation of the task above with unit test for your converter method. Hand in your source code and a screen recording over mission accomplished.

3.3 Cayenne myDevices IoT dashboard

Now we are going to create a cloud based dashboard on Cayenne myDevices. Create an account at: https://cayenne.mydevices.com/cayenne/signup.

Once logged in, select the large LoRa icon and then select 'The Things Network' from the lower end of the menu bar on the left.
Then scroll down and click the CayenneLPP option in the settings window that should appear.

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE
Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

You need to give the dashboard/device a name, and then add the Device EUI in the DevEUI box. You can leave the Activation mode set to ‘already registered’ and the tracking box locations as ‘this device moves’ or whatever you want. Save these settings and leave this tab open in your browser.

Finally, we need to go back to The Things Network site, and in our Applications Overview we need to select the ‘Integrations’ tab and click ‘Add integration’. Scroll down and click the myDevices icon.

In the Process ID box, give this a name such as ‘mylabdashboard’ and in the ‘Access Key’ drop-down menu, when you click on the empty box, it should reveal only one option ‘Default key’, next to two buttons that say ‘Devices’ and ‘Messages’.

Click on ‘Default key’ to select this into the ‘Access Key’ box, then click the blue ‘Add integration’ button in the lower right-hand corner. Now switch back to your myDevices page you left open in another tab.

As soon as myDevices receives some data from your Mbed simulator it should automatically make a dashboard for you and display the data as in figure 1 below.

Note! This is the result when you have added all 4 sensordata channels to the Cayenne LPP payload.

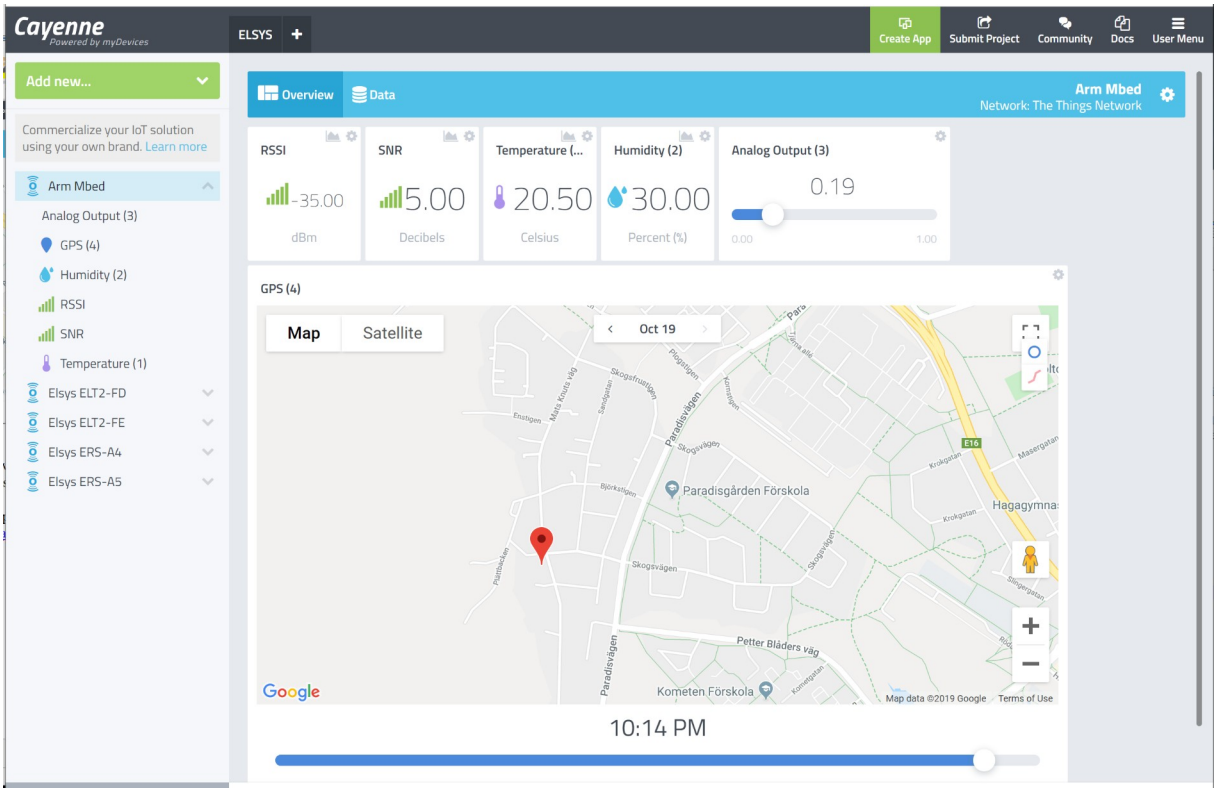

Fig 1. Cayenne myDevices.

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE
Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

If you wish to add more sensors later on press Add new… > Device/Widget > Select LoRa > The Things Network > CayenneLPP, ELSYS, Custom or whatever sensor widget you want to add.

When everything works you can try the Cayenne myDevices app: https://play.google.com/store/apps/details?id=com.mydevices.cayenne to get real-time data to your handheld device.

3.4 Get a sketch with sensors and LoRa/TTN support to work in the Arduino IDE and get familiar with the PlatformIO IDE for VSCode

The guy with the Swiss accent have an IDE comparison between Arduino and PlatformIO in YouTube clip #264 PlatformIO for Arduino, ESP8266, and ESP32 Tutorial: https://www.youtube.com/watch?v=0poh_2rBq7E where he favors PlatformIO.

3.4.1 With Arduino IDE

Get the Arduino IDE at: https://www.arduino.cc/en/Main/Software
Follow the TTN Arduino example at:
https://www.thethingsnetwork.org/docs/devices/arduino/

Install the Arduino MKR WAN 1300 board:
https://www.digikey.com/en/maker/blogs/2019/how-to-setup-the-arduino-mkr-1300
or other board with LoRa support.

In your code use at least the following libraries:

- #include <TheThingsNetwork.h>
- #include <DHT.h>
- #include <CayenneLPP.h>

Install library: https://www.arduino.cc/en/guide/libraries in this case it may be an external library (Add .ZIP library…) as well.

Use the bundled **\ttn_dht\ttn_dht.ino** file and make it compile without errors (some warnings are ok).

Note that Arduino supports ESP32 boards: https://randomnerdtutorials.com/ttgo-lora32-sx1276-arduino-ide/

Report:
Show that you have a working compilation of the task above with no major errors. Hand in a screen recording over mission accomplished

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE
Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

3.4.2 With PlatformIO IDE for VSCode

Follow the guy with the Swiss accent YouTube previous clip above or other tutorial and guides here: https://docs.platformio.org/en/latest/tutorials/index.html

Save your present workspace and create a new workspace for your PlatformIO code. Otherwise it will not work or you will get major problems sooner or later with VSCode. You must have the PlatformIO toolbar (bottom left) visible and working in VSCode for it to work.
See figure 2 below:

Fig 2. VSCode bottom left toolbar.

If the toolbar is not visible you must go to extensions and disable the workspace and then enable the workspace again.

Install platform support for one of the very popular ESP32 LoRa32 / LoRa 32 boards, Heltec WiFi LoRa 32 and TTGO LoRa32-OLED.

Note that PlatformIO also support Arduino Uno and MKR WAN 13xx board etc.

When you created an ESP32 project check the platform.ini file so it supports your selected ESP32 board. This ini config for example works with my own TTGO LoRa 32 v1 boards:

```
[env:ttgo-lora32-v1]
platform = espressif32
board = ttgo-lora32-v1
framework = arduino
monitor_speed = 115200
```

Paste in the Hello World sample or other example code from the PlatformIO examples at: https://github.com/platformio/platform-espressif32

Espressif Systems Github page have even more examples: https://github.com/espressif

I have also bundled sample code that works for me and can send data to TTN via CayenneLpp. You only have to insert your own OTAA values from the TTN console in **LSB** mode. My ESP32 example code is located in the **\platformio\main.cpp** file.

Here is a good walk-thru explanation of the code in main.cpp: https://learn.adafruit.com/the-things-network-for-feather?view=all > Code Overview.

My ESP32 example code compiler output log

```
Processing ttgo-lora32-v1 (platform: espressif32; board: ttgo-lora32-v1; framework: arduino)
------------------------------------------------------------------------------------------------------------------------
Verbose mode can be enabled via `-v, --verbose` option
CONFIGURATION: https://docs.platformio.org/page/boards/espressif32/ttgo-lora32-v1.html
```


Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE
Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

```
PLATFORM: Espressif 32 1.11.1 > TTGO LoRa32-OLED V1
HARDWARE: ESP32 240MHz, 320KB RAM, 4MB Flash
DEBUG: Current (esp-prog) External (esp-prog, iot-bus-jtag, jlink, minimodule, olimex-arm-usb-ocd, olimex-arm-usb-ocd-h, olimex-arm-usb-tiny-h, olimex-jtag-tiny, tumpa)
PACKAGES: toolchain-xtensa32 2.50200.80 (5.2.0), framework-arduinoespressif32 2.10004.191002 (1.0.4), tool-esptoolpy 1.20600.0 (2.6.0), tool-mkspiffs 2.230.0 (2.30)
LDF: Library Dependency Finder -> http://bit.ly/configure-pio-ldf
LDF Modes: Finder ~ chain, Compatibility ~ soft
Found 33 compatible libraries
Scanning dependencies...
Dependency Graph
|-- <LMIC-Arduino> 1.5.0+arduino-2
|   |-- <SPI> 1.0
|-- <CayenneLPP> 1.0.3
|   |-- <ArduinoJson> 6.13.0
|-- <SPI> 1.0
|-- <ESP8266_SSD1306> 4.1.0
|   |-- <Wire> 1.0.1
|   |-- <SPI> 1.0
Building in release mode
Compiling .pio\build\ttgo-lora32-v1\src\main.cpp.o
Linking .pio\build\ttgo-lora32-v1\firmware.elf
Building .pio\build\ttgo-lora32-v1\firmware.bin
Retrieving maximum program size .pio\build\ttgo-lora32-v1\firmware.elf
Checking size .pio\build\ttgo-lora32-v1\firmware.elf
Advanced Memory Usage is available via "PlatformIO Home > Project Inspect"
DATA:    [=         ]   5.2% (used 16892 bytes from 327680 bytes)
PROGRAM: [==        ]  19.7% (used 257690 bytes from 1310720 bytes)
========================================================================
[SUCCESS] Took 5.34 seconds
========================================================================
```

My ESP32 example upload to board output log

```
esptool.py v2.6
Configuring upload protocol...
AVAILABLE: esp-prog, espota, esptool, iot-bus-jtag, jlink, minimodule, olimex-arm-usb-ocd, olimex-arm-usb-ocd-h, olimex-arm-usb-tiny-h, olimex-jtag-tiny, tumpa
CURRENT: upload_protocol = esptool
Looking for upload port...
Auto-detected: COM7
Uploading .pio\build\ttgo-lora32-v1\firmware.bin
esptool.py v2.6
Serial port COM7
Connecting....
Chip is ESP32D0WDQ6 (revision 1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
MAC: 24:6f:28:9e:6e:40
Uploading stub...
Running stub...
Stub running...
```


Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE
Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

```
Changing baud rate to 460800
Changed.
Configuring flash size...
Auto-detected Flash size: 4MB
Compressed 15872 bytes to 10319...

Writing at 0x00001000... (100 %)
Wrote 15872 bytes (10319 compressed) at 0x00001000 in 0.2 seconds (effective 522.4
kbit/s)...
Hash of data verified.
Compressed 3072 bytes to 128...

Writing at 0x00008000... (100 %)
Wrote 3072 bytes (128 compressed) at 0x00008000 in 0.0 seconds (effective 1759.9 kbit/s)...
Hash of data verified.
Compressed 8192 bytes to 47...

Writing at 0x0000e000... (100 %)
Wrote 8192 bytes (47 compressed) at 0x0000e000 in 0.0 seconds (effective 6474.9 kbit/s)...
Hash of data verified.
Compressed 257808 bytes to 135034...

Writing at 0x00010000... (11 %)
Writing at 0x00014000... (22 %)
Writing at 0x00018000... (33 %)
Writing at 0x0001c000... (44 %)
Writing at 0x00020000... (55 %)
Writing at 0x00024000... (66 %)
Writing at 0x00028000... (77 %)
Writing at 0x0002c000... (88 %)
Writing at 0x00030000... (100 %)
Wrote 257808 bytes (135034 compressed) at 0x00010000 in 3.4 seconds (effective 610.5
kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
========================================================================
[SUCCESS] Took 13.74 seconds
========================================================================
```

A fun project to try in PlatformIO is: https://github.com/cyberman54/ESP32-Paxcounter
A fix for Bluetooth: https://github.com/cyberman54/ESP32-Paxcounter/issues/272

Report:
Show that you have a working compilation of the task above. Hand in a screen recording over mission accomplished

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE
Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

3.5 Bring home data for simple data mining with SQL

From Cayenne myDevices you can download your accumulated data in CSV-file format. I included my own very small file Cayenne myDevices.csv.
Refer to the small SQLite manual below for how-to put millions of rows in a SQLite database in no time and get turbo charged SQL queries working.

1. Get the latest sqlite shell executable from: http://www.sqlite.org/download.html
2. Start a cmd shell and run:
   ```
   sqlite3.exe mydbfile.db3
   ```
3. Create a table for the sensordata in the SQLite shell as:
   ```
   create table mysensor (timestamp text, deviceid text, channel
   integer, sensorname text, sensorid text, datatype text, unit text,
   value real);
   ```
4. Give the type of separator between the values for the import file to your table, .csv or tab or other delimiter in the SQLite shell as:
   ```
   .separator ","
   .separator "\t"
   ```
   Sometimes you may have to change a character for a value or other text in your import file. It is best done with Notepad++: http://notepad-plus-plus.org/.
5. Remove the first row in the CSV-file if it contains column descriptions.
6. Import all rows in the SQLite shell with .command file table as:
   ```
   .import sample.csv mysensor
   ```
   A 1 GB large CSV-file usually only takes some seconds to SQLite-ify!
7. Quit the SQLite shell
   ```
   .quit
   ```
   and make sure you got a resulting file named mydbfile.db3.
8. Get the free portable Database .NET application from: http://fishcodelib.com/ and connect to the database file mydbfile .db3. Now can SQL be used with the measurement data.
9. Create a query as:
   ```
   SELECT * FROM [mysensor] where sensorname = 'Temperature' order by
   timestamp DESC LIMIT 500
   ```

Report:
Show that you have a working implementation of the task above. Hand in a screen recording over mission accomplished.

3.6 Lab feedback

a) Were the labs relevant and appropriate and what about length etc?
b) What corrections and/or improvements do you suggest for these labs?

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE
Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/