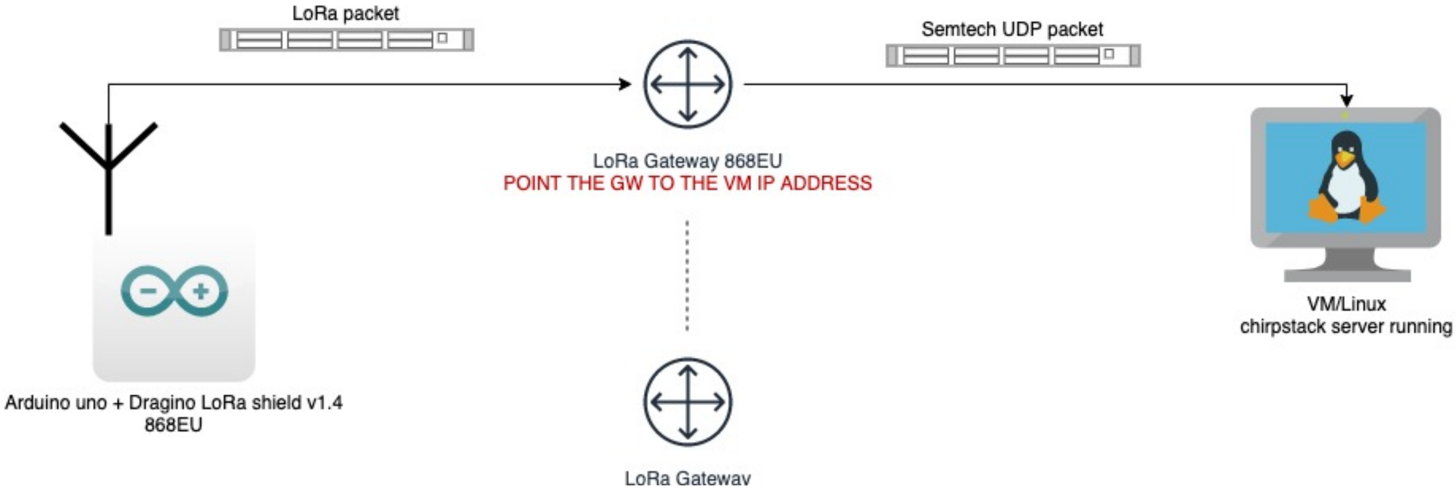
End node setup

Arduino based end-node sending data over LoRa to LoRaWAN network with ABP (Activation by Personalization)

This sketch is based on scenario 1, which assumes that the end nodes are within reach of the LoRa Gateways as the figure below explains:



LoRa packet
Semtech UDP packet
LoRa Gateway 868EU
POINT THE GW TO THE VM IP ADDRESS
VM/Linux
chirpstack server running
Arduino uno + Dragino LoRa shield v1.4
868EU
LoRa Gatewav

SCENARIO.1 ABP

A. General overview

This sketch reads sensor data (Grove Temperature and Humidity sensor[1]) and sends it to chirpstack-application-server installed on a virtual machine (PC). For more info on how to install the chirpstack-server packages on Linux/Ubuntu check the installation document. After you make sure that your server is running without errors. It is time to setup the end nodes

B. Configuration

To start, you need to download this Arduino library[2]. Open Arduino IDE and select Sketch/ include library/Add Zip library to add the downloaded zip library.
Now Open the library folder,
(on Windows it is Documents**Arduino****libraries**\)
(on Mac it is Documents**Arduino****libraries**\ )
(on Linux ubuntu v.20.+ it is Home/snap/arduino/41/Arduino/libraries)

[1] https://wiki.seeedstudio.com/Grove-TemptureAndHumidity_Sensor-High-Accuracy_AndMini-v1.0/

[2] https://github.com/dragino/Arduino-Profile-Examples/tree/master/libraries/Dragino/examples/IoTServer/Cayenne%20and%20TTN/arduino-lmic

Locate the file arduino-lmic-master/src/lmic/config.h and chose your radio type. This example uses EU868.
The pin mapping between the dragino LoRa shield and the UNO is handled in the sketch, so just use the one from this repository. So , go back to Arduino IDE and create a new sketch and paste the code from the example.

The grove sensor pins are very simple, it uses i2c interface. So simply connect pin SDA and SCL to the Arduino SCL and SDA pins(last 2 digital pins). For better accuracy use 5v. If not you can use 3.3v. both work fine.

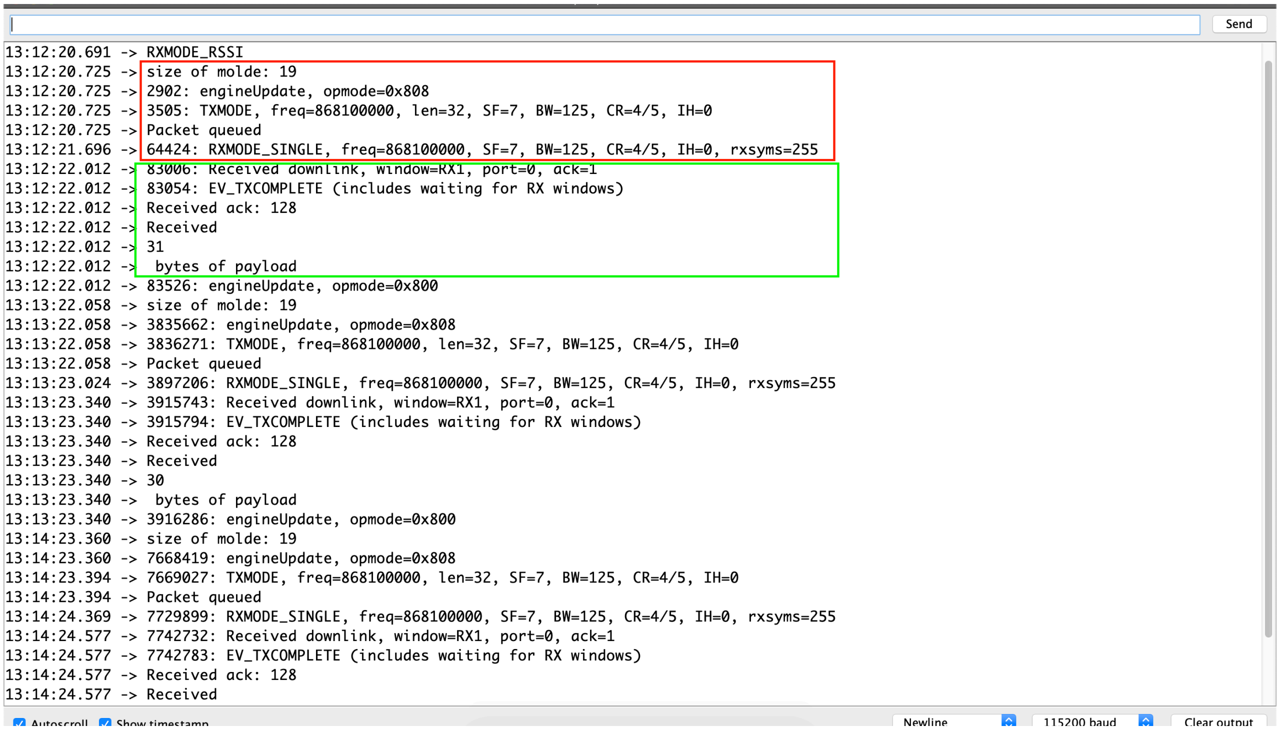
This sketch uses ABP method to connect to the chirpstack server. To do so , we need 3 keys from the chirpstack-application-server:

- NWSKEY Network Session Key. In MSB <Most Significant Bit>
- APPSKEY APPLICATION KEY. In MSB <Most Significant Bit>
- DEVADDR Device Address. In MSB <Most Significant Bit>

These values can be found when you create an application on the server(Check Server interface setup document).

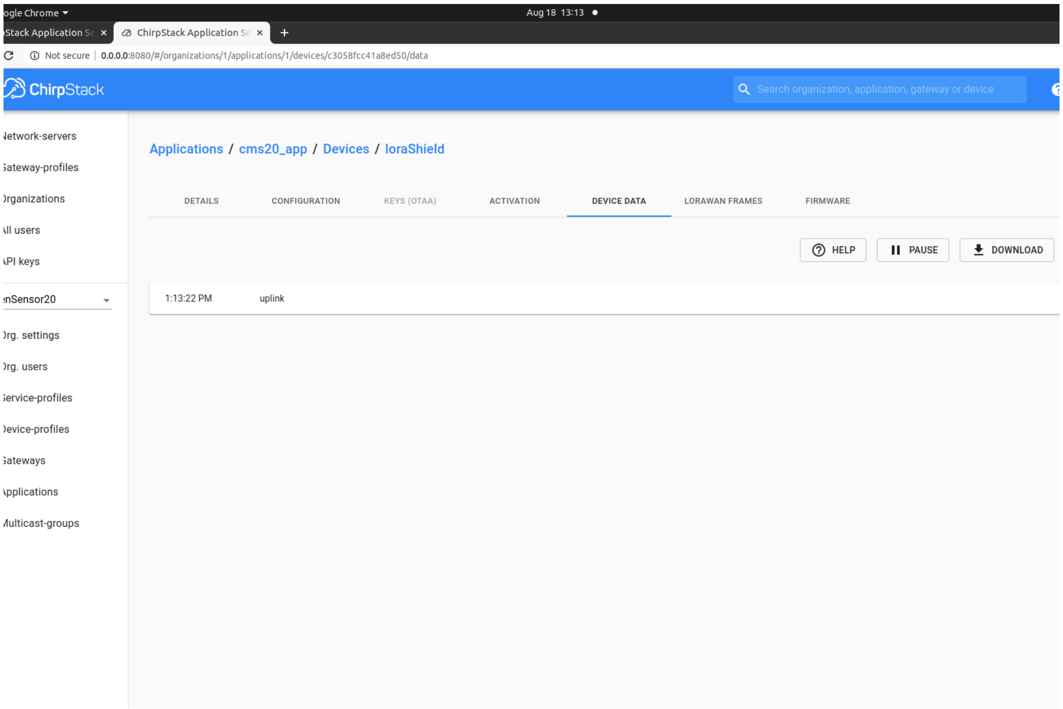
Make sure you have entered the right values. The template-sketch uses confirmation for each uplink. So make sure that the value is 1 in the last argument like this:
LMIC_setTxData2(1, mydata, sizeof(mydata), 1);

Now, you are ready. Click upload sketch and open serial monitor. You can see data sending and receiving confirmation coming in the Arduino IDE serial monitor. See figure below:



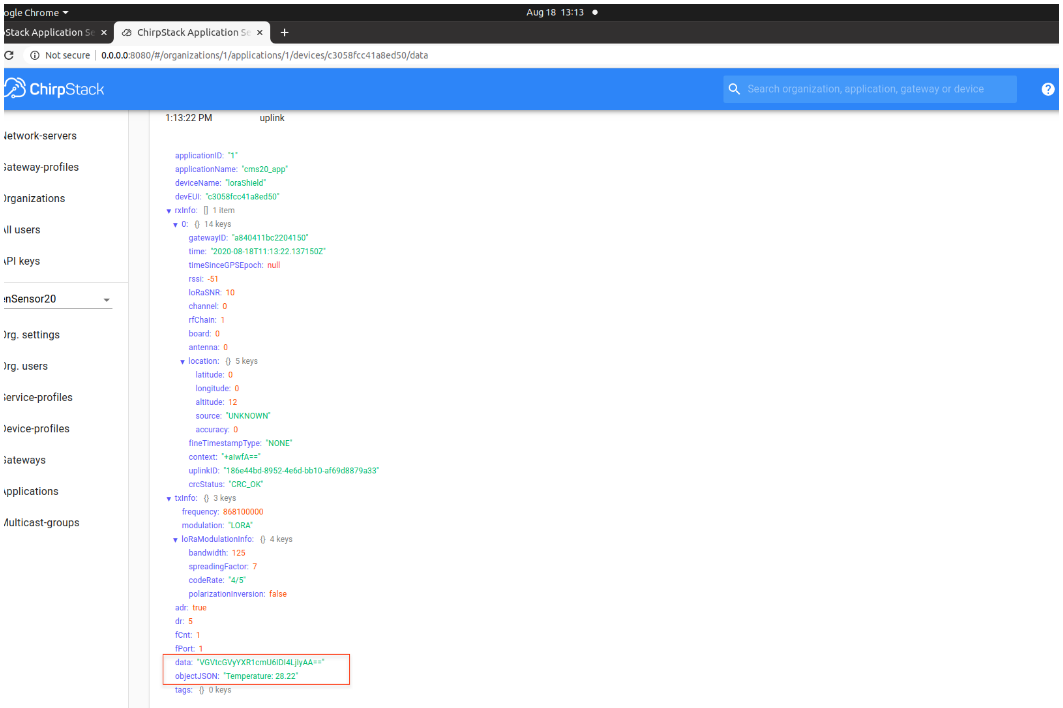
```
13:12:20.691 -> RXMODE_RSSI
13:12:20.725 -> size of molde: 19
13:12:20.725 -> 2902: engineUpdate, opmode=0x808
13:12:20.725 -> 3505: TXMODE, freq=868100000, len=32, SF=7, BW=125, CR=4/5, IH=0
13:12:20.725 -> Packet queued
13:12:21.696 -> 64424: RXMODE_SINGLE, freq=868100000, SF=7, BW=125, CR=4/5, IH=0, rxsyms=255
13:12:22.012 -> 83006: Received downlink, window=RX1, port=0, ack=1
13:12:22.012 -> 83054: EV_TXCOMPLETE (includes waiting for RX windows)
13:12:22.012 -> Received ack: 128
13:12:22.012 -> Received
13:12:22.012 -> 31
13:12:22.012 ->  bytes of payload
13:12:22.012 -> 83526: engineUpdate, opmode=0x800
13:13:22.058 -> size of molde: 19
13:13:22.058 -> 3835662: engineUpdate, opmode=0x808
13:13:22.058 -> 3836271: TXMODE, freq=868100000, len=32, SF=7, BW=125, CR=4/5, IH=0
13:13:22.058 -> Packet queued
13:13:23.024 -> 3897206: RXMODE_SINGLE, freq=868100000, SF=7, BW=125, CR=4/5, IH=0, rxsyms=255
13:13:23.340 -> 3915743: Received downlink, window=RX1, port=0, ack=1
13:13:23.340 -> 3915794: EV_TXCOMPLETE (includes waiting for RX windows)
13:13:23.340 -> Received ack: 128
13:13:23.340 -> Received
13:13:23.340 -> 30
13:13:23.340 ->  bytes of payload
13:13:23.340 -> 3916286: engineUpdate, opmode=0x800
13:14:23.360 -> size of molde: 19
13:14:23.360 -> 7668419: engineUpdate, opmode=0x808
13:14:23.394 -> 7669027: TXMODE, freq=868100000, len=32, SF=7, BW=125, CR=4/5, IH=0
13:14:23.394 -> Packet queued
13:14:24.369 -> 7729899: RXMODE_SINGLE, freq=868100000, SF=7, BW=125, CR=4/5, IH=0, rxsyms=255
13:14:24.577 -> 7742732: Received downlink, window=RX1, port=0, ack=1
13:14:24.577 -> 7742783: EV_TXCOMPLETE (includes waiting for RX windows)
13:14:24.577 -> Received ack: 128
13:14:24.577 -> Received
```

ABP WITH ACK TEST END NODE-RESULTS



1:13:22 PM uplink

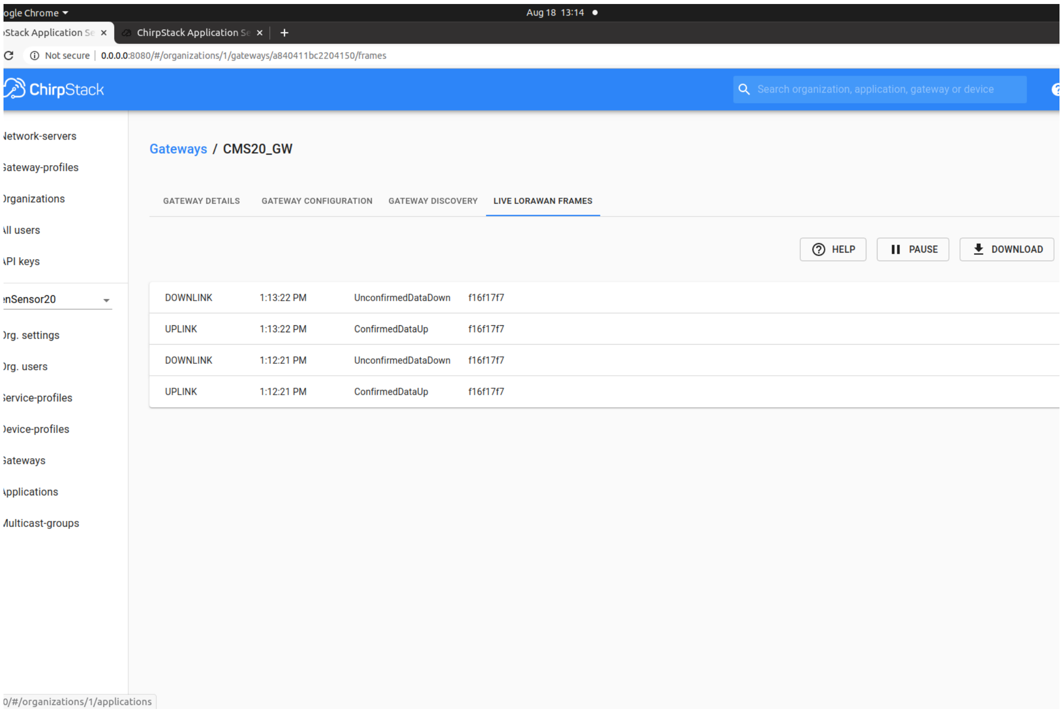
DATA ON CHIRPSTACK-WEB INTERFACE



1:13:22 PM uplink

```
  applicationID: "1"
  applicationName: "cms20_app"
  deviceName: "loraShield"
  devEUI: "c3058fcc41a8ed50"
▾ rxInfo: [] 1 item
  ▾ 0: {} 14 keys
      gatewayID: "a840411bc2204150"
      time: "2020-08-18T11:13:22.137150Z"
      timeSinceGPSEpoch: null
      rssi: -51
      loRaSNR: 10
      channel: 0
      rfChain: 1
      board: 0
      antenna: 0
    ▾ location: {} 5 keys
        latitude: 0
        longitude: 0
        altitude: 12
        source: "UNKNOWN"
        accuracy: 0
      fineTimestampType: "NONE"
      context: "+alwfA=="
      uplinkID: "186e44bd-8952-4e6d-bb10-af69d8879a33"
      crcStatus: "CRC_OK"
▾ txInfo: {} 3 keys
    frequency: 868100000
    modulation: "LORA"
  ▾ loRaModulationInfo: {} 4 keys
      bandwidth: 125
      spreadingFactor: 7
      codeRate: "4/5"
      polarizationInversion: false
  adr: true
  dr: 5
  fCnt: 1
  fPort: 1
  data: "VGVtcGVyYXR1cmU6IDI4LjIyAA=="
  objectJSON: "Temperature: 28.22"
  tags: {} 0 keys
```

UPLINK DETAILS WITH DECRYPTED DATA



DOWNLINK	1:13:22 PM	UnconfirmedDataDown	f16f17f7
UPLINK	1:13:22 PM	ConfirmedDataUp	f16f17f7
DOWNLINK	1:12:21 PM	UnconfirmedDataDown	f16f17f7
UPLINK	1:12:21 PM	ConfirmedDataUp	f16f17f7

GATEWAY LOGGING