Low Powered Network for the Internet of Things
Use LoRa network with the SAP Hana Cloud

Author: Laurent Gomez, Laurent.gomez@sap.com

Abstract

In this document, we guide you step by step toward the establishment of a Low Powered connectivity between a device, and the SAP HANA Cloud. You will do the following:

- Enable LoRa communication on an Arduino Uno
- Setup your own LoRa gateway on a RaspberryPi
- Setup your IoT platform account to push sensor data from your gateway

At the end of this session, you will have an Arduino device pushing data to the SAP Hana Cloud platform through the LoRa network.

Contents

Introduction 3

Setup 4

Setup the Internet of Things Services on your our own HCP trial account 5

Activate the Internet of Things Services 5

Deploy the Message Management Service 8

Configure devices in the Internet of Things Services 11

Test the MMS via HTTP API 18

Arduino 21

Install the Arduino IDE 21

Install the LoRa libraries 23

Check the LoRa module 24

LoRa gateway 32

Send messages from Arduino to HCP 35

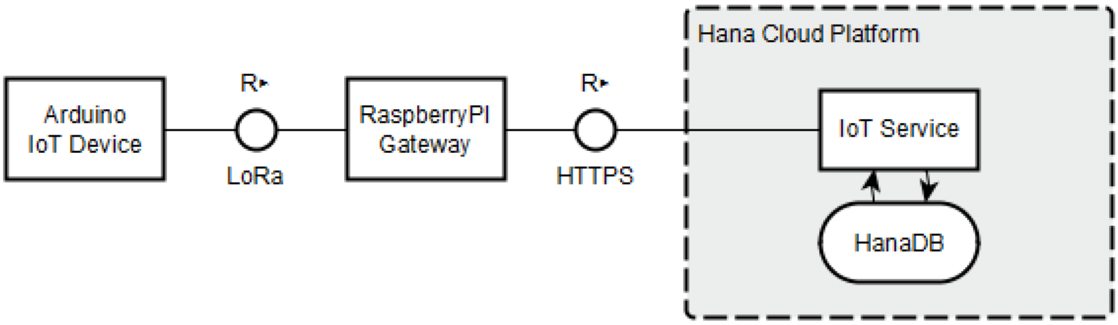
Push data from Arduino 35

Receive data on the gateway 37

Send message to HCP 41

Introduction

In this lab session, we will establish the connection between an IoT device and the SAP IoT service hosted on the SAP Hana Cloud Platform. The overall architecture is depicted as follows:



Data generated by the Arduino device is sent over LoRa to a raspberryPi which serves as a LoRa gateway. LoRa is a communication protocol over ultra-narrow-band radio.

LoRaWAN™ *is a Low Power Wide Area Network (LPWAN) specification intended for wireless battery operated Things in a regional, national or global network. LoRaWAN targets key requirements of Internet of Things such as secure bi-directional communication, mobility and localization services. The LoRaWAN specification provides seamless interoperability among smart Things without the need of complex local installations and gives back the freedom to the user, developer, businesses enabling the roll out of Internet of Things. [reference]*

LoRa is part of the communication protocol for Low Power Network Wide Area. *Low-Power Wide-Area Network (LPWAN) or Low-Power Network (LPN) is a type of wireless telecommunication network designed to allow long range communications at a low bit rate among things (connected objects), such as sensors operated on a battery.[reference]*

Our LoRa gateway then forwards the data packet to the SAP IoT service on SAP HCP. SAP HANA Cloud Platform is an open platform-as-a-service that provides unique in-memory database and application services. [reference]

Setup

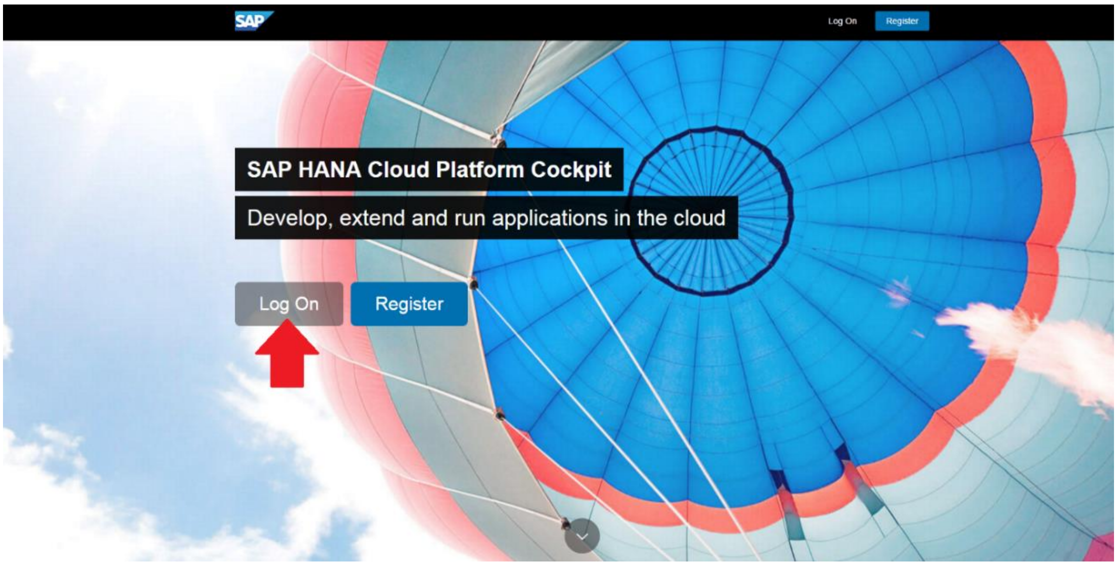
In this section, we will guide through

- the setup of your own IoT Service on HCP,
- the setup of your Arduino device with the LoRa module
- the setup of your LoRa gateway on raspberryPI

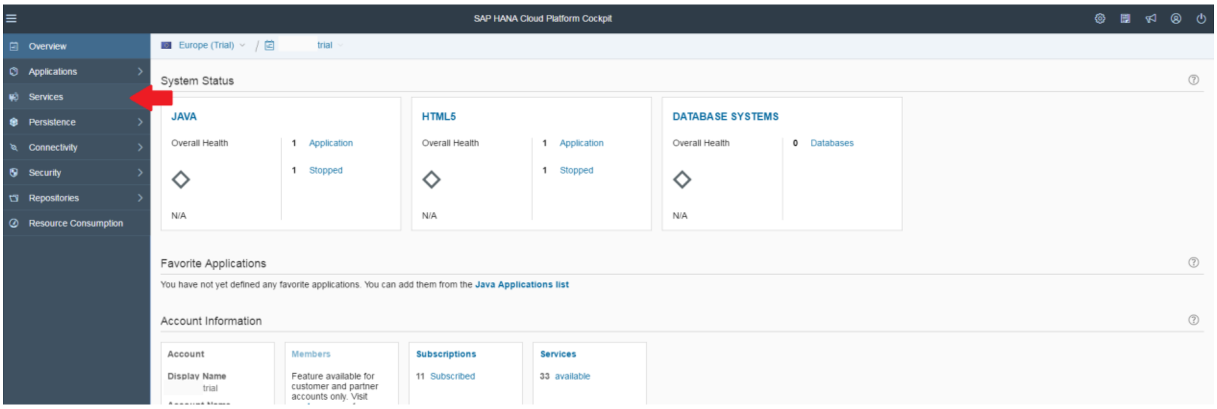
Setup the Internet of Things Services on your our own HCP trial account

Activate the Internet of Things Services

1. Go to https://account.hanatrial.ondemand.com/ and log on. If you don't have an account yet you may have to register using your @sap.com email address.

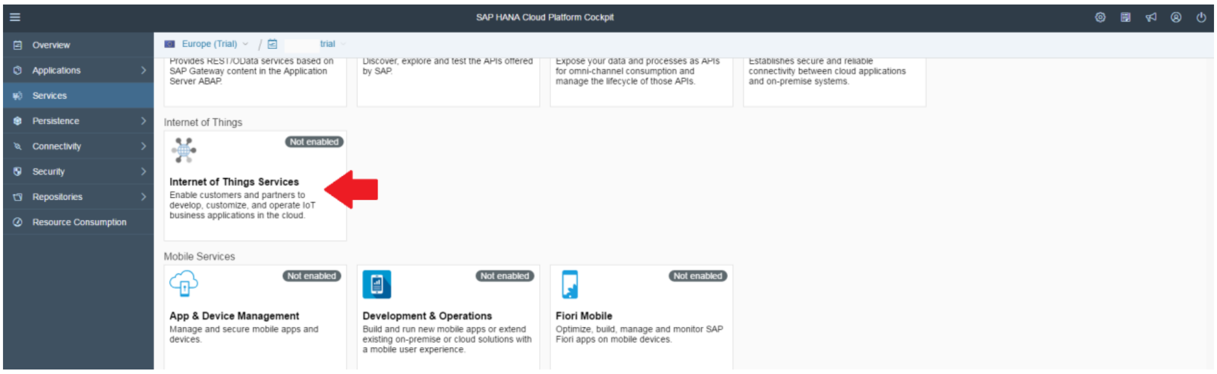

SAP
Log On
Register
SAP HANA Cloud Platform Cockpit
Develop, extend and run applications in the cloud
Log On
Register

2. Go to *Services* and enable the *Internet of Things Services*.



You have not yet defined any favorite applications. You can add them from the **Java Applications list**

Feature available for customer and partner accounts only. Visit



Provides REST/OData services based on SAP Gateway content in the Application Server ABAP.

Discover, explore and test the APIs offered by SAP.

Expose your data and processes as APIs for omni-channel consumption and manage the lifecycle of those APIs.

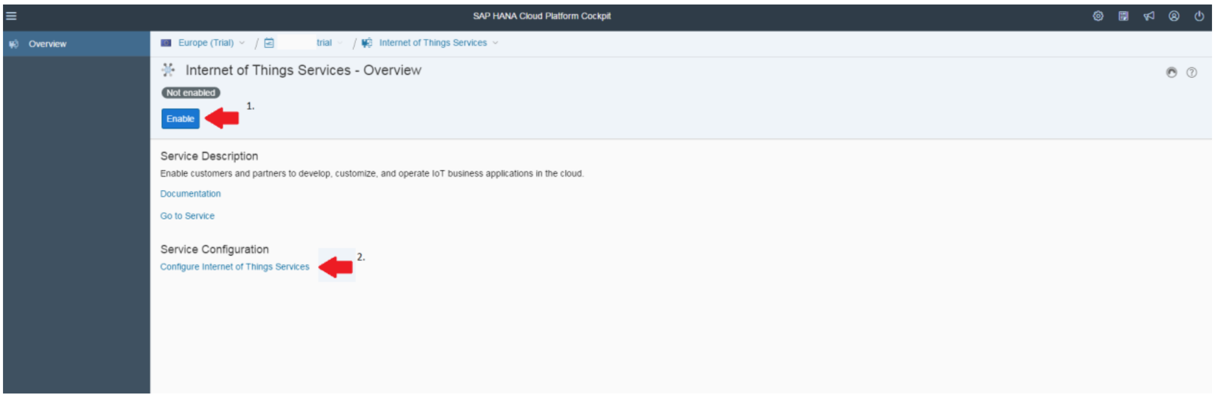
Establishes secure and reliable connectivity between cloud applications and on-premise systems.

Internet of Things Services
Enable customers and partners to develop, customize, and operate IoT business applications in the cloud.

App & Device Management
Manage and secure mobile apps and devices.

Development & Operations
Build and run new mobile apps or extend existing on-premise or cloud solutions with a mobile user experience.

Fiori Mobile
Optimize, build, manage and monitor SAP Fiori apps on mobile devices.


SAP HANA Cloud Platform Cockpit
Overview
Europe (Trial) / trial / Internet of Things Services
Not enabled
1.
Enable
Documentation
Go to Service
Configure Internet of Things Services
2.

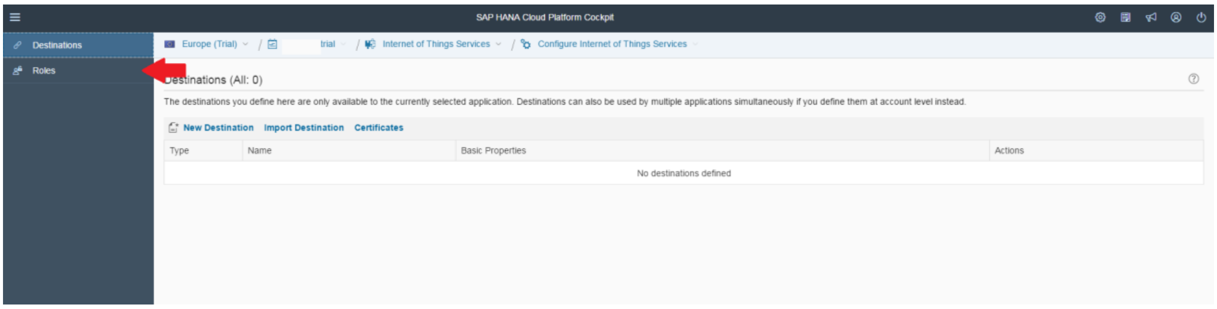
Internet of Things Services - Overview

Service Description

Enable customers and partners to develop, customize, and operate IoT business applications in the cloud.

Service Configuration

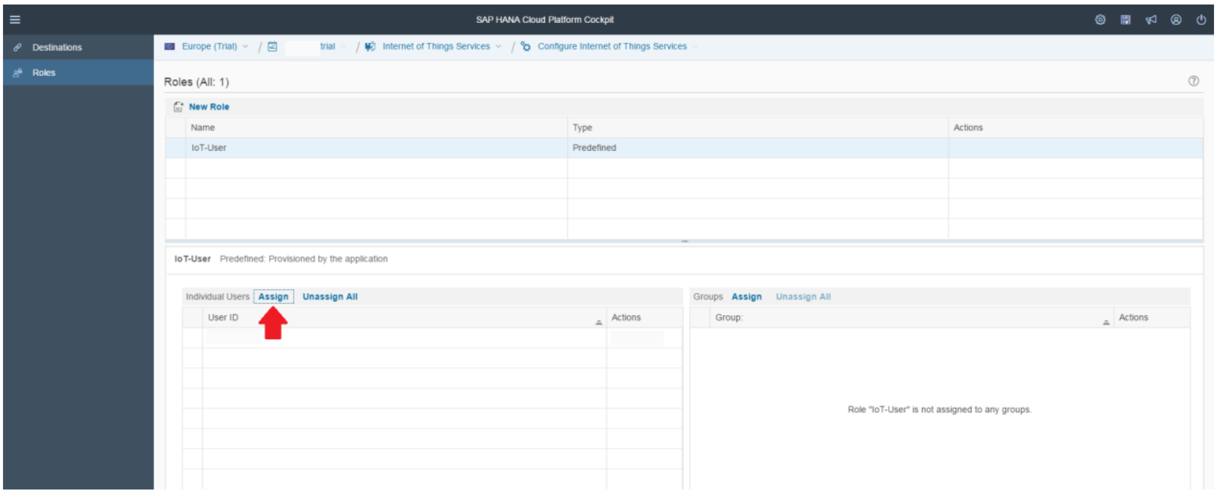
3. In the *Internet of Things Services* open *Configure Internet of Things Services* and assign the *IoT-User* role to your user.


SAP HANA Cloud Platform Cockpit
Destinations
Roles
Europe (Trial) / trial / Internet of Things Services / Configure Internet of Things Services
New Destination
Import Destination
Certificates

Destinations (All: 0)

The destinations you define here are only available to the currently selected application. Destinations can also be used by multiple applications simultaneously if you define them at account level instead.

Type	Name	Basic Properties	Actions
No destinations defined			


SAP HANA Cloud Platform Cockpit
Destinations
Roles
Europe (Trial) / trial / Internet of Things Services / Configure Internet of Things Services
New Role
Individual Users
Assign
Unassign All
Groups
Assign
Unassign All

Roles (All: 1)

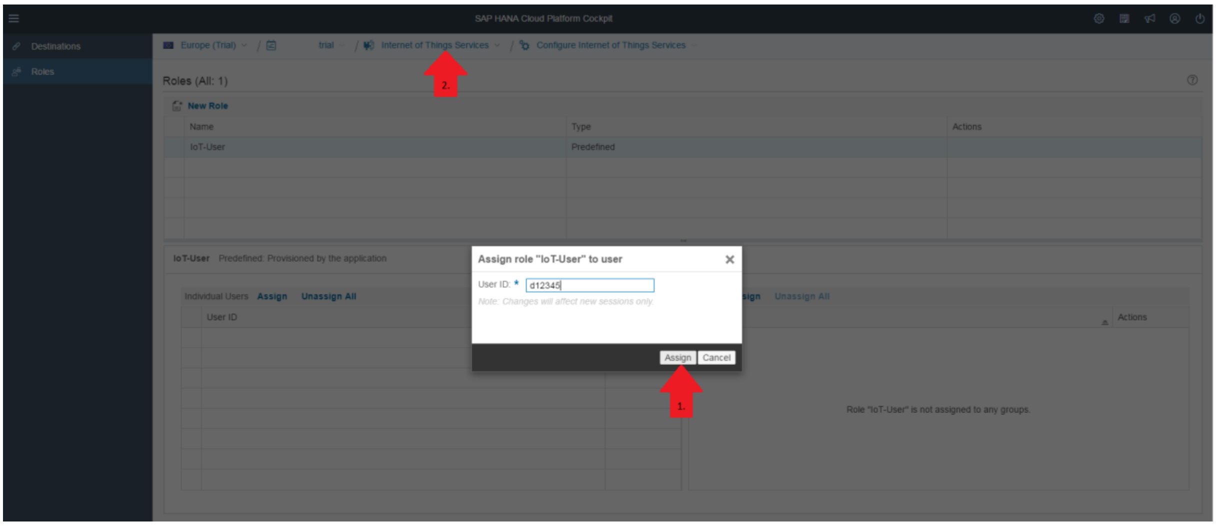
Name	Type	Actions
IoT-User	Predefined	

IoT-User Predefined: Provisioned by the application

User ID	Actions

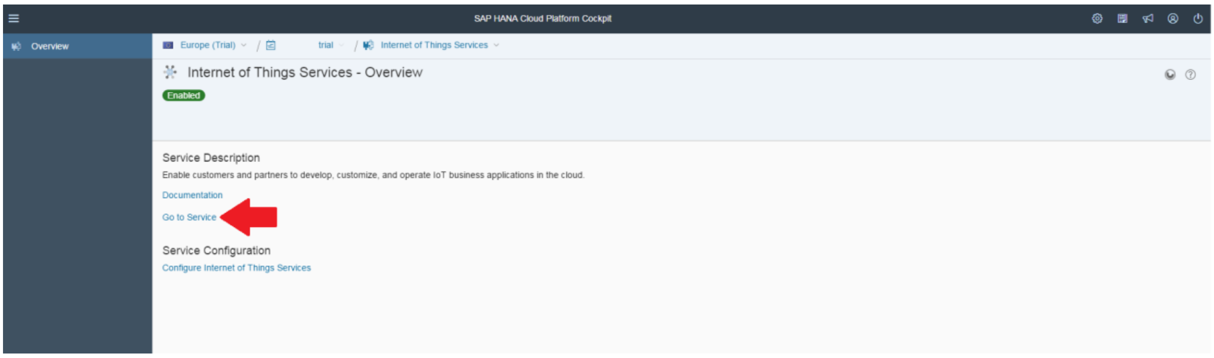
Group	Actions

Role "IoT-User" is not assigned to any groups.


SAP HANA Cloud Platform Cockpit
Destinations
Roles
Europe (Trial) / trial / Internet of Things Services / Configure Internet of Things Services
Roles (All: 1)
New Role
Name
Type
Actions
IoT-User
Predefined
IoT-User Predefined: Provisioned by the application
Individual Users Assign Unassign All
User ID
Actions
Role "IoT-User" is not assigned to any groups.
Assign role "IoT-User" to user
Note: Changes will affect new sessions only.
Assign
Cancel
1.
2.

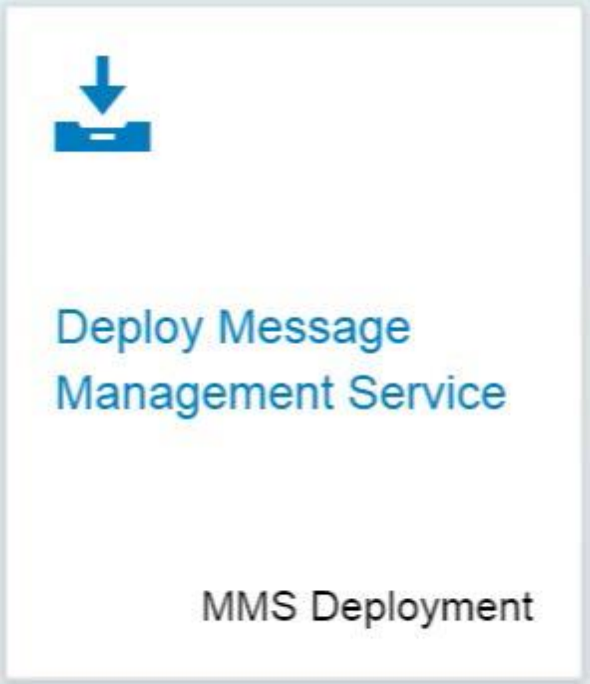
User ID: * d12345

4. Go back to the Internet of Things Services Overview and open *Go to Service.*


SAP HANA Cloud Platform Cockpit
Overview
Europe (Trial) / trial / Internet of Things Services
Internet of Things Services - Overview
Enabled
Service Description
Enable customers and partners to develop, customize, and operate IoT business applications in the cloud.
Documentation
Go to Service
Service Configuration
Configure Internet of Things Services

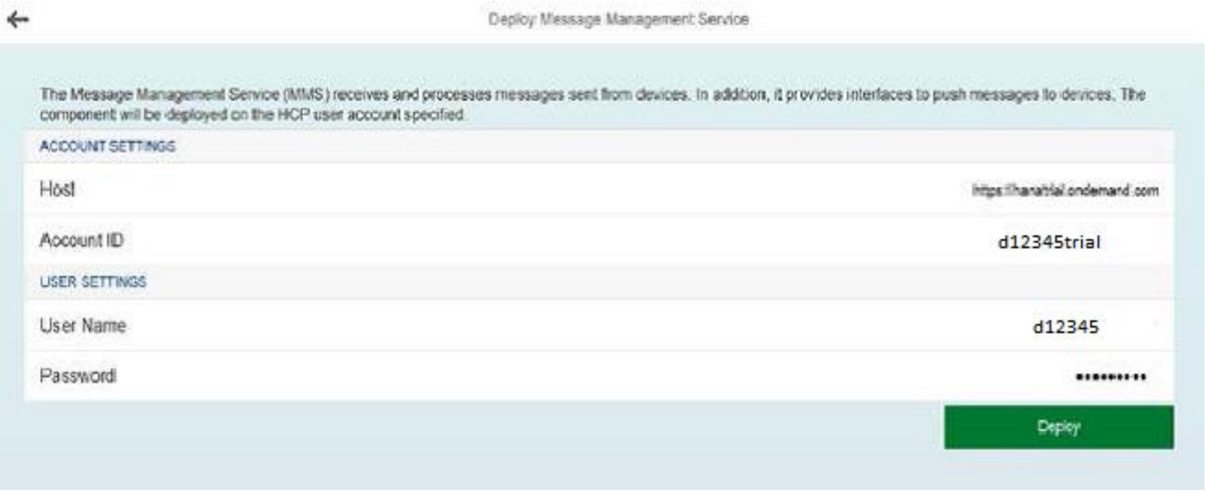
Deploy the Message Management Service

5. In the Internet of Things Services Cockpit select *Deploy Message Management Service* enter your username and password and press *Deploy*. (In our case the Message Management Service (MMS) is responsible for receiving messages from IoT devices.)



Deploy Message Management Service

MMS Deployment



Deploy Message Management Service

The Message Management Service (MMS) receives and processes messages sent from devices. In addition, it provides interfaces to push messages to devices. The component will be deployed on the HCP user account specified.

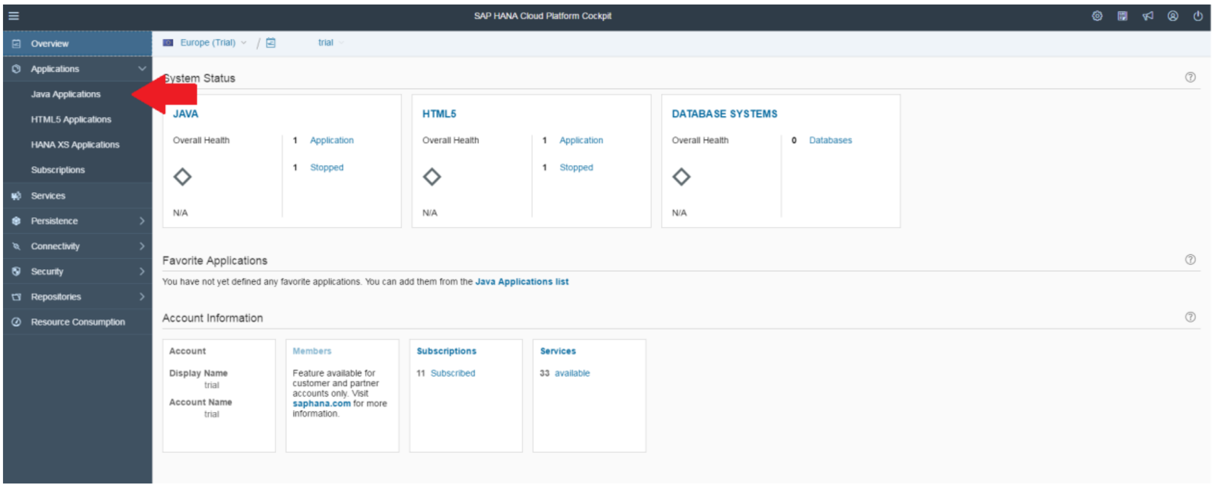
ACCOUNT SETTINGS

Host	https://hanatrial.ondemand.com
Account ID	d12345trial

USER SETTINGS

User Name	d12345
Password	••••••••••

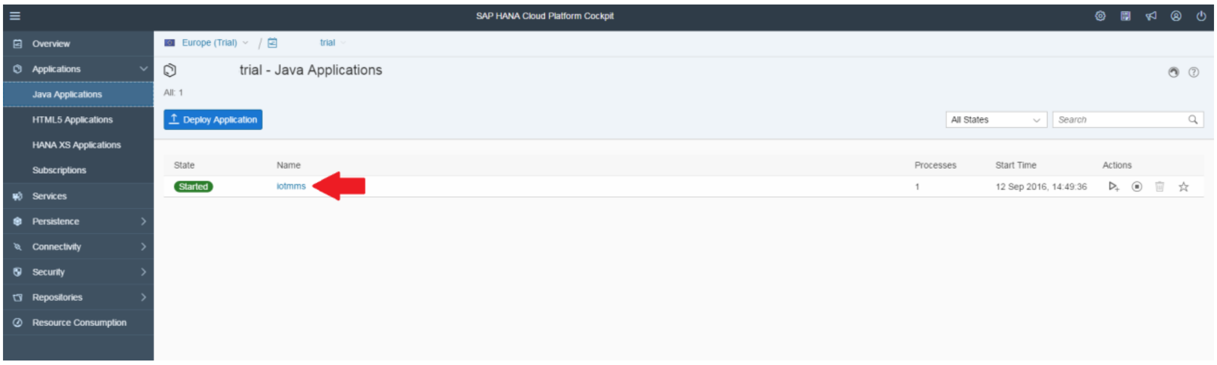
6. Go back to your HCP-trial-Cockpit Overview and go to *Applications* > *Java Applications.*



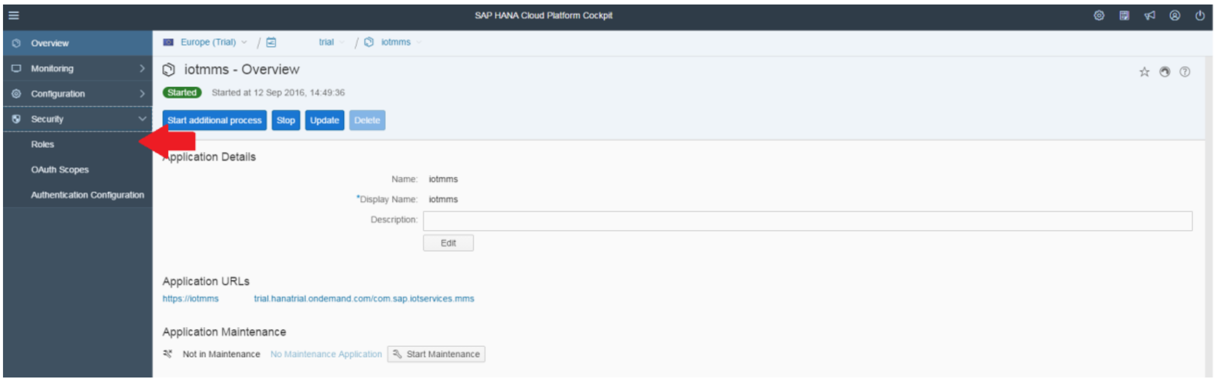
You have not yet defined any favorite applications. You can add them from the Java Applications list

Feature available for customer and partner accounts only. Visit saphana.com for more information.

7. Select the Application *iotmms* and go to *Security > Roles* and assign the *IoT-MMS-User* role to your user.



State	Name	Processes	Start Time	Actions
Started	iotmms	1	12 Sep 2016, 14:49:36	

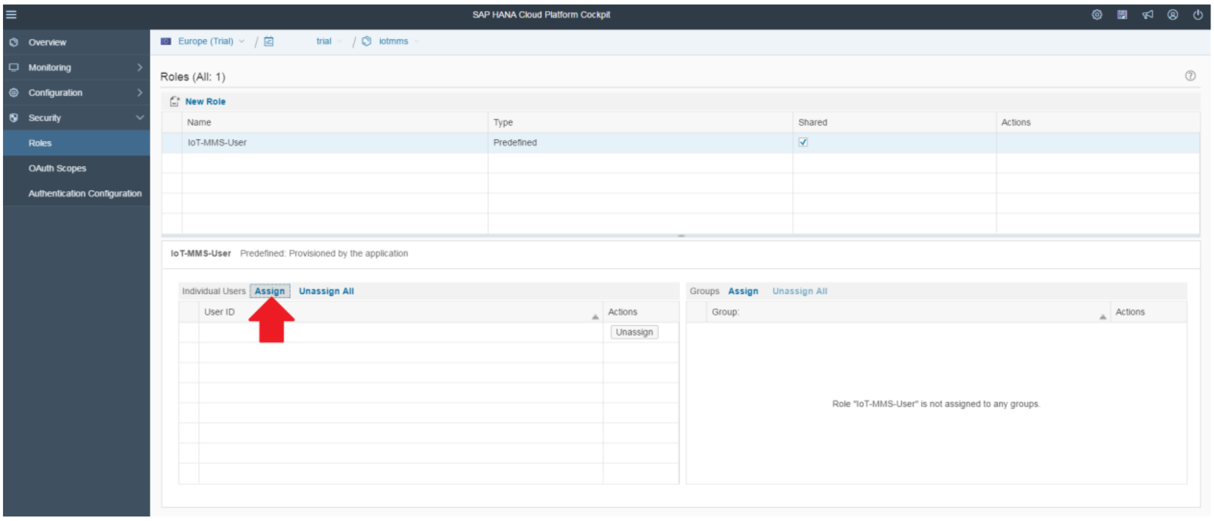


Started at 12 Sep 2016, 14:49:36

Name: iotmms
*Display Name: iotmms
Description:

https://iotmms trial.hanatrial.ondemand.com/com.sap.iotservices.mms

Not in Maintenance No Maintenance Application

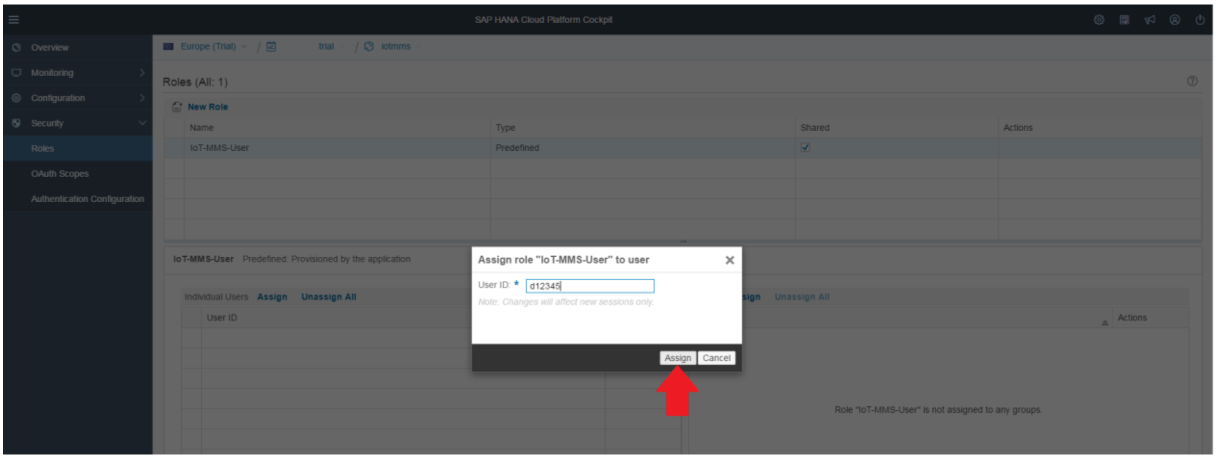


Name	Type	Shared	Actions
IoT-MMS-User	Predefined	☑	

IoT-MMS-User Predefined: Provisioned by the application

User ID	Actions

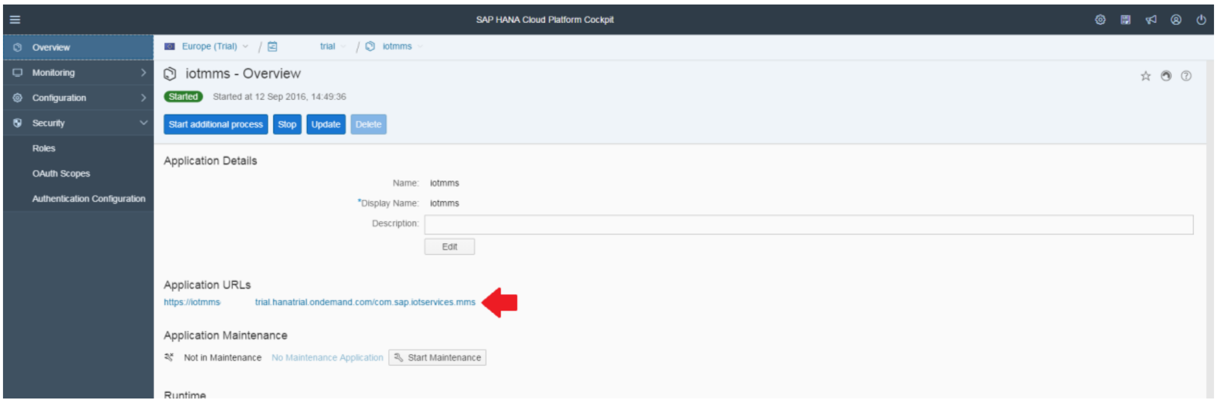
Group:	Actions


SAP HANA Cloud Platform Cockpit
Assign role "IoT-MMS-User" to user
Assign
Cancel

User ID: * d12345

Note: Changes will affect new sessions only.

8. Go back to the *iotmms Overview* and click on the URL below *Application URLs* (e.g. https://iotmmsd0xxxxxtrial.hanatrial.ondemand.com/com.sap.iotservices.mms). This takes you to the MMS-Cockpit (let this page remain open in a separate tab).



Started at 12 Sep 2016, 14:49:36

Application Details

Name: iotmms

*Display Name: iotmms

Description:

Application URLs

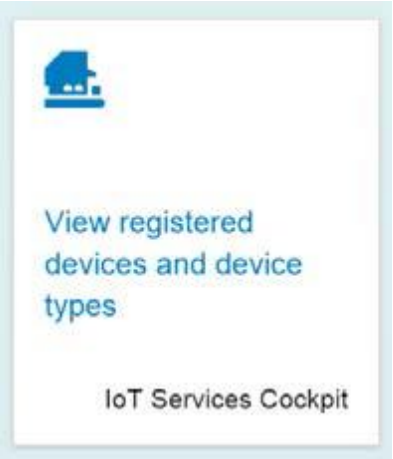
https://iotmms trial.hanatrial.ondemand.com/com.sap.iotservices.mms

Application Maintenance

Not in Maintenance No Maintenance Application

Configure devices in the Internet of Things Services

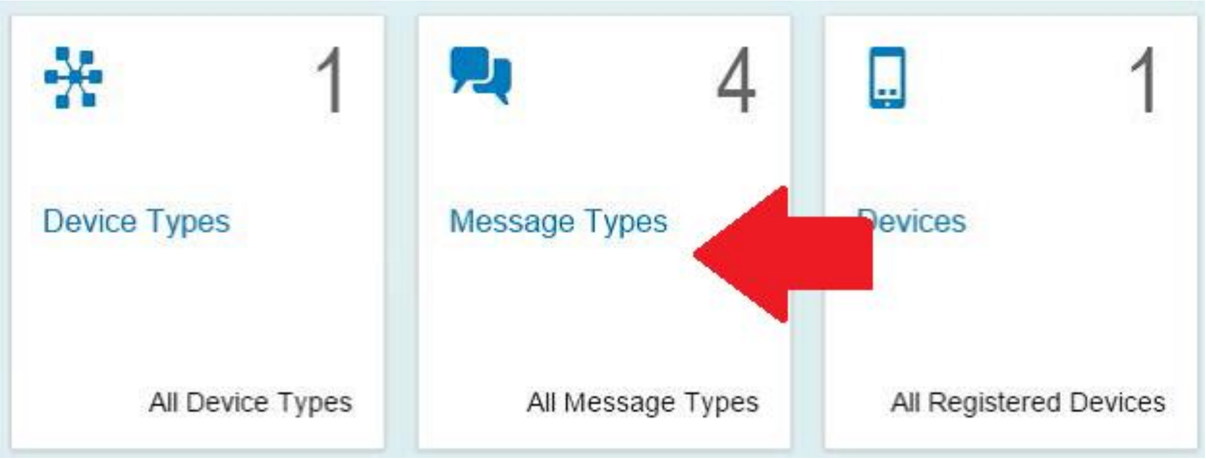
9. Go to *View registered devices and device types* which takes you back to the IoT Services Cockpit.

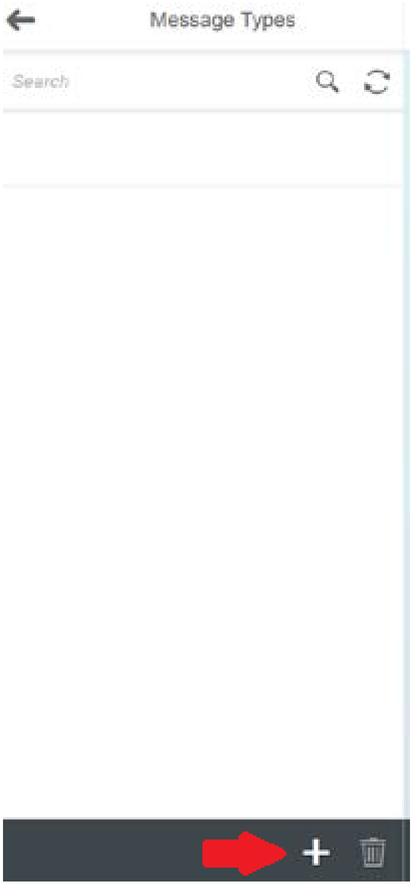


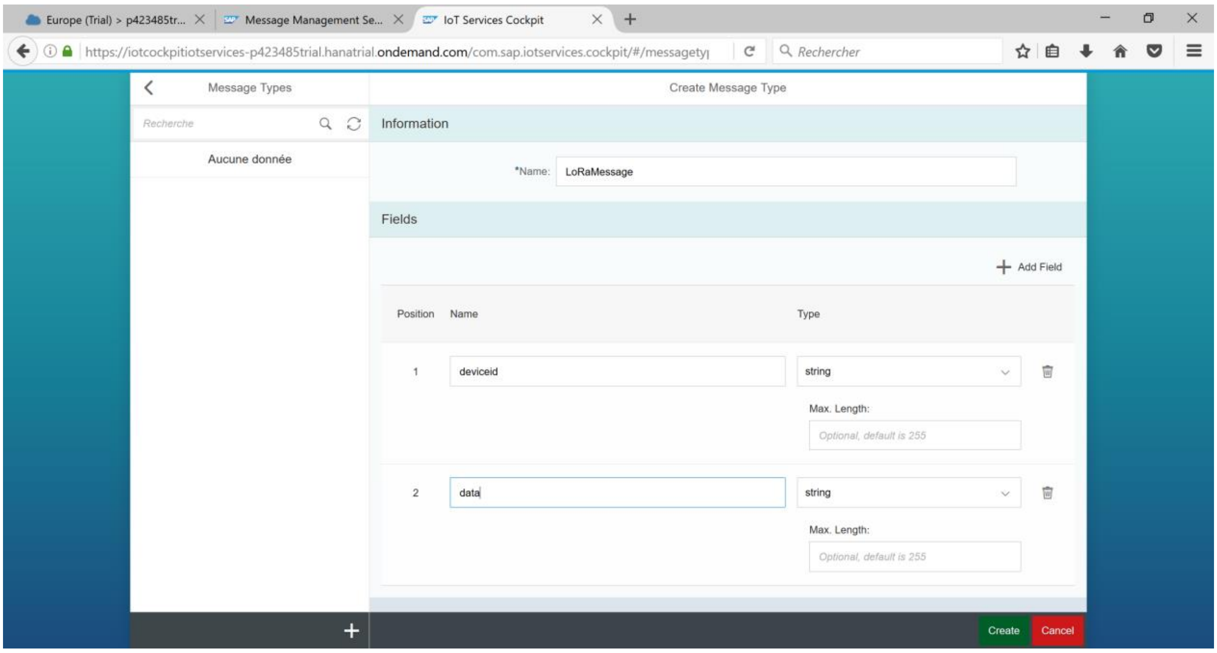
10. Go to *Message Types* and create a new message type called *LoRaMessage*.

    The first field of the message type is **deviceid**

    The second field of the message type is **data**.







https://iotcockpitiotservices-p423485trial.hanatrial.ondemand.com/com.sap.iotservices.cockpit/#/messagety

Create Message Type

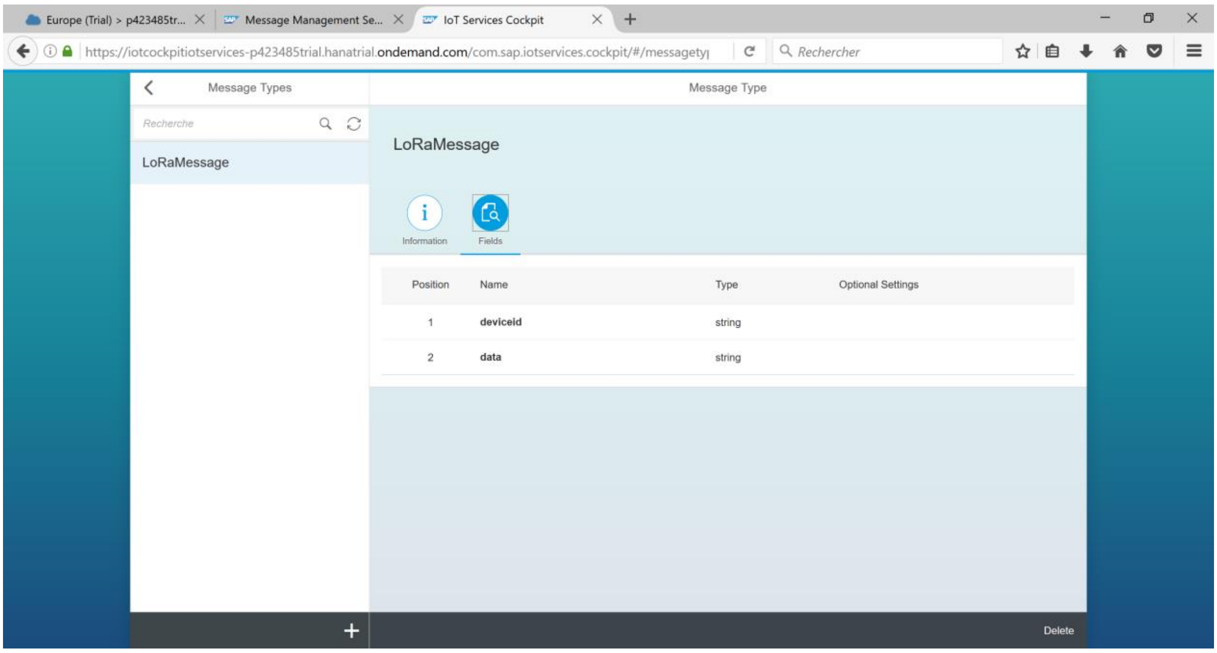
Information

*Name: LoRaMessage

Fields

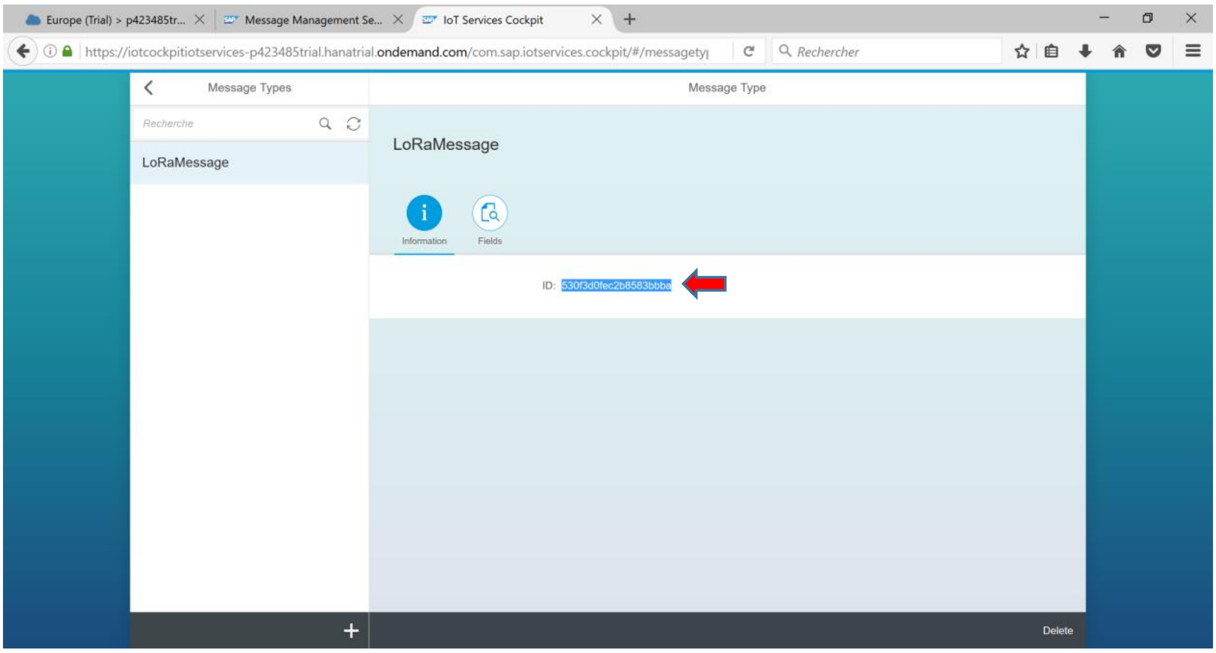
Add Field

Position	Name	Type
1	deviceid	string Max. Length: Optional, default is 255
2	data	string Max. Length: Optional, default is 255


Europe (Trial) > p423485tr...
Message Management Se...
IoT Services Cockpit
https://iotcockpitiotservices-p423485trial.hanatrial.ondemand.com/com.sap.iotservices.cockpit/#/messagety
Rechercher
Message Types
Message Type
Recherche
LoRaMessage
Information
Fields
Delete

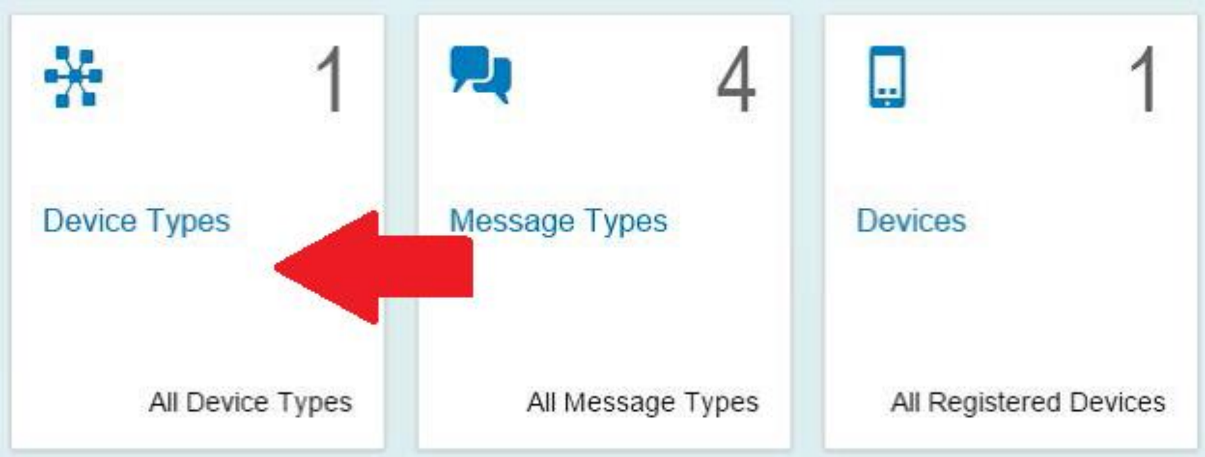
Position	Name	Type	Optional Settings
1	deviceid	string	
2	data	string	

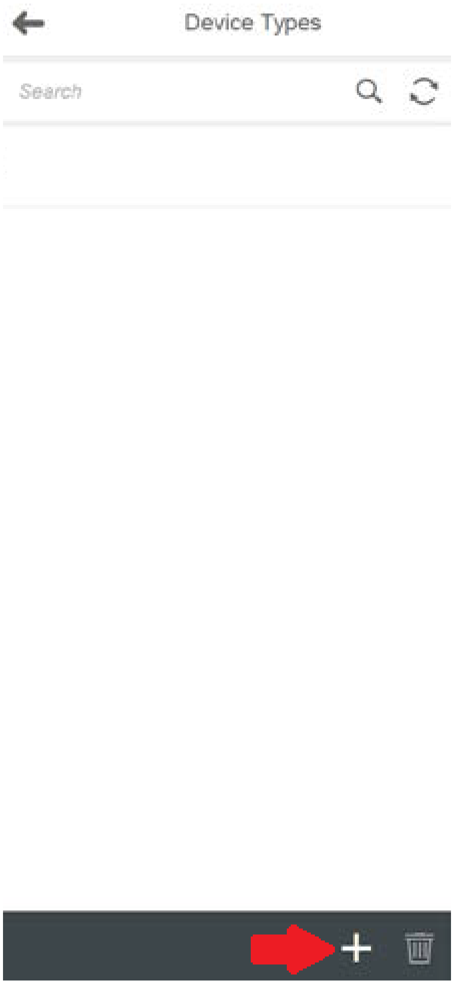
11. Open a text editor and paste the now appearing *message type ID* in a new text file.

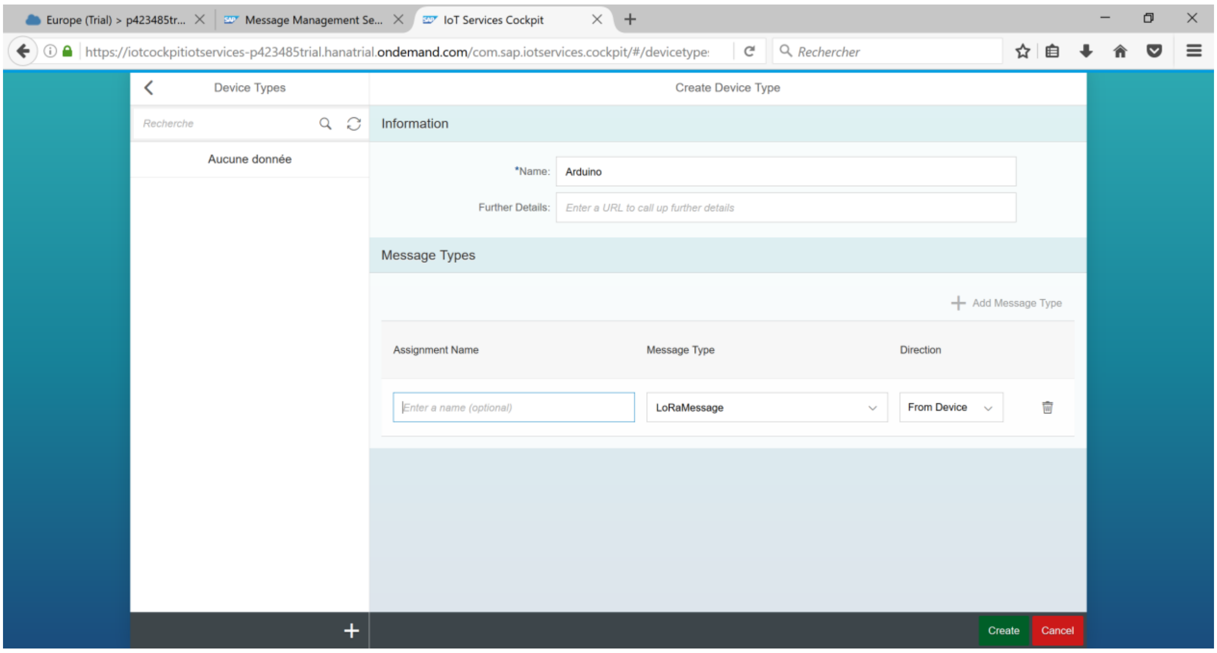

Europe (Trial) > p423485tr...
Message Management Se...
IoT Services Cockpit
https://iotcockpitiotservices-p423485trial.hanatrial.ondemand.com/com.sap.iotservices.cockpit/#/messagety
Rechercher
Message Types
Message Type
Recherche
LoRaMessage
Information
Fields
Delete

ID: 530f3d0fec2b8583bbba

12. Go to *Device Types* and create a new device type called *Arduino* and add the message type *LoRaMessage* with the direction *from device.*


1
Device Types
All Device Types
4
Message Types
All Message Types
1
Devices
All Registered Devices

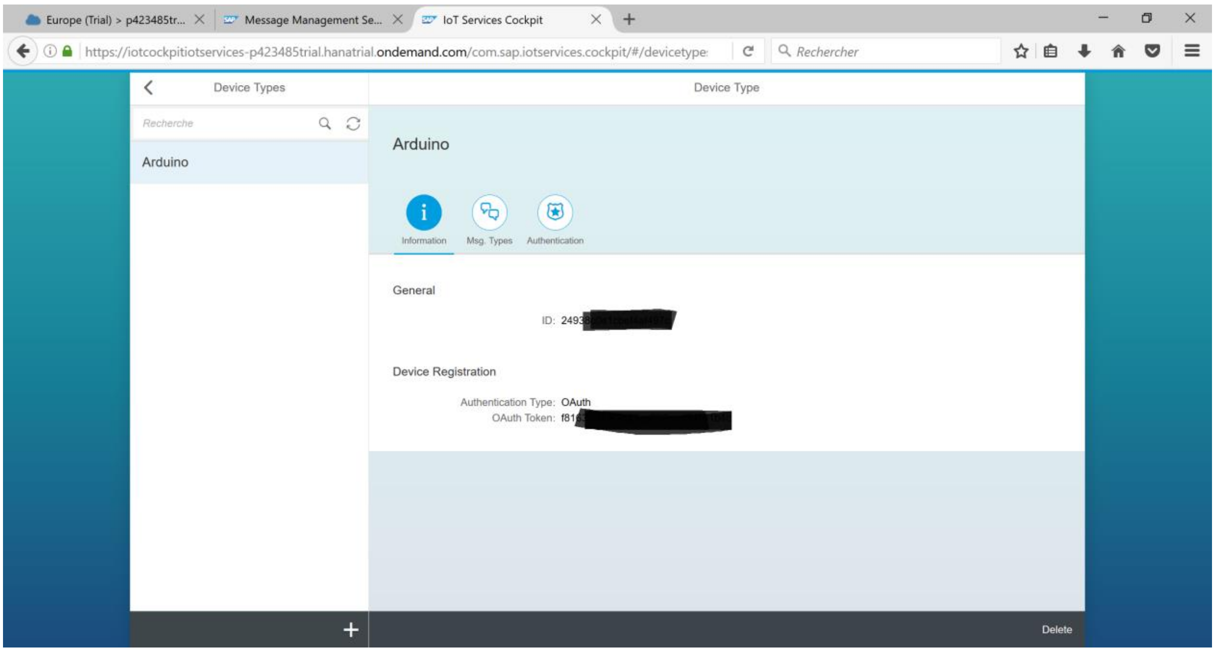

Device Types
Search
+



*Name: Arduino

Message Type: LoRaMessage

Direction: From Device

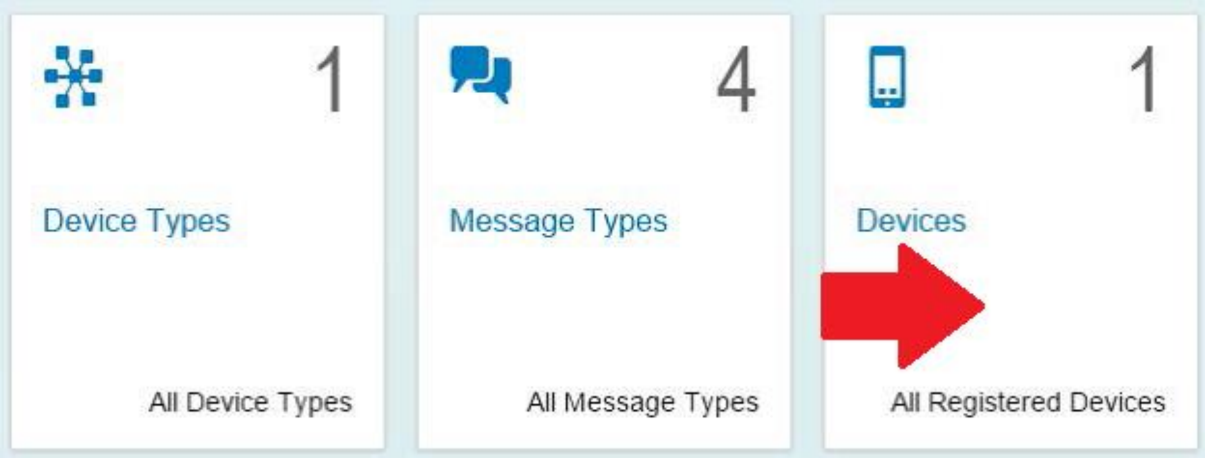


ID: 2493[redacted]

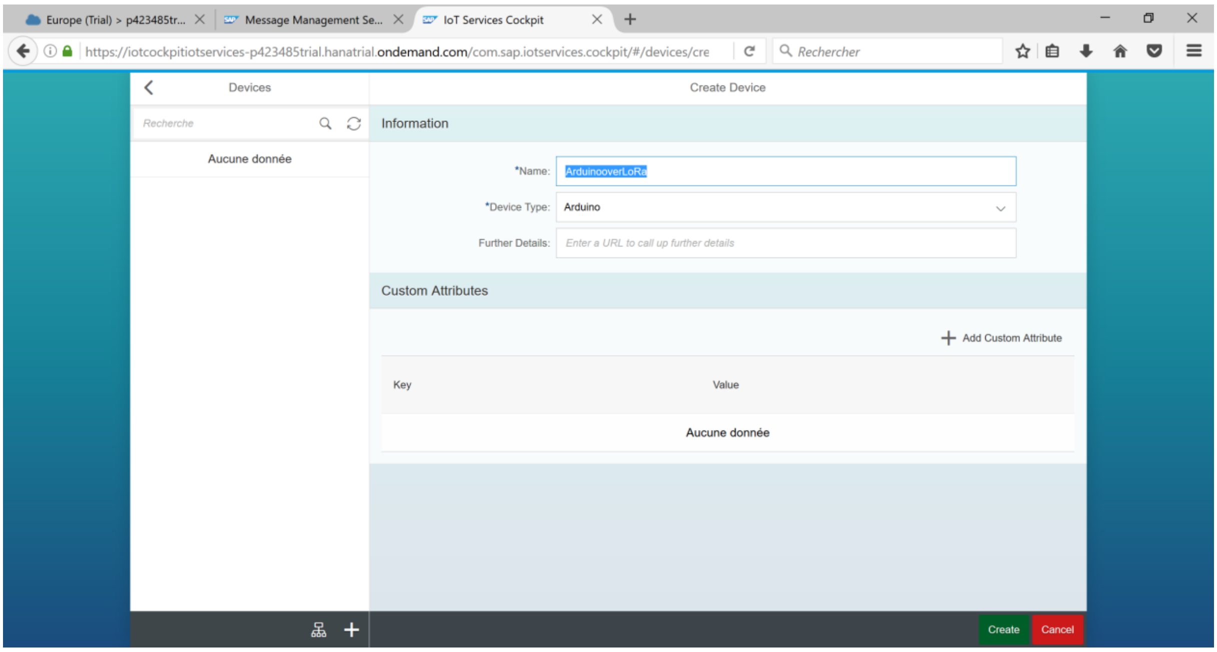
Authentication Type: OAuth

OAuth Token: f81[redacted]

13. Go back to *Internet of Things Services Cockpit* and then go to *Devices*.


1
Device Types
All Device Types
4
Message Types
All Message Types
1
Devices
All Registered Devices

14. Create a new device called *ArduinooverLoRa* and choose *Arduino* as device type.

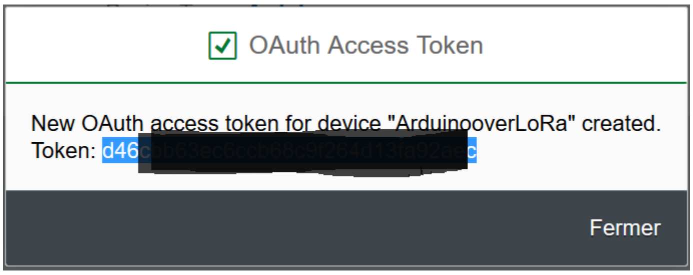


*Name: ArduinooverLoRa

*Device Type: Arduino

Further Details: Enter a URL to call up further details

15. Paste the now appearing *token* in your already opened text file.

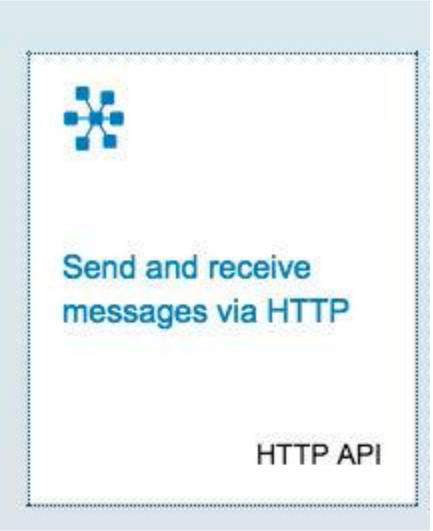


New OAuth access token for device "ArduinooverLoRa" created.
Token: d46[redacted]

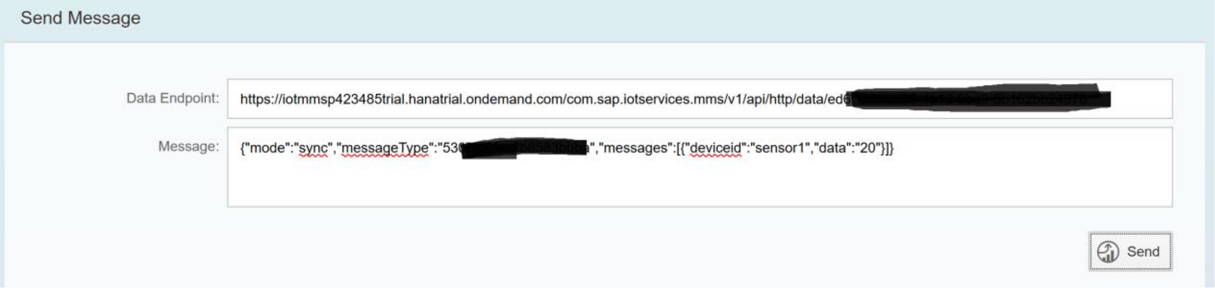
16. Also paste the ***device ID*** appearing after closing the token window in your text file.

Test the MMS via HTTP API

17. Go to the *Message Management Service Cockpit* (separate tab) and now go to *Send and receive messages via HTTP*.



18. Below *Send message* change the last part of the URL associated to *HTTP endpoint* [usually: d000-e000-v000-i000-c000-e001] to your devices' ID (last pasted string in text file).

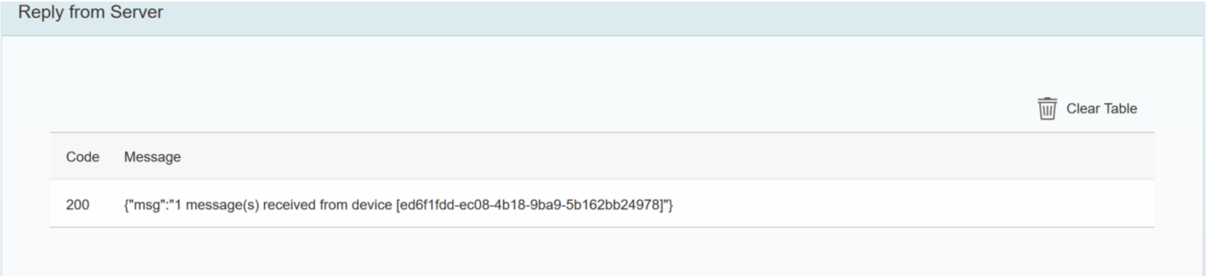

Send Message

Data Endpoint: https://iotmmsp423485trial.hanatrial.ondemand.com/com.sap.iotservices.mms/v1/api/http/data/ed[redacted]

Message: {"mode":"sync","messageType":"53[redacted]","messages":[{"deviceid":"sensor1","data":"20"}]}

19. Now replace the value from *messageType* in the *message* [usually: m0t0y0p0e1] to your message type ID (first pasted string in text file) and replace *[{"sensor":"sensor1","value":"20","timestamp":1413191650}]}* by *[{"deviceid":"sensor1","data":"20"}]*

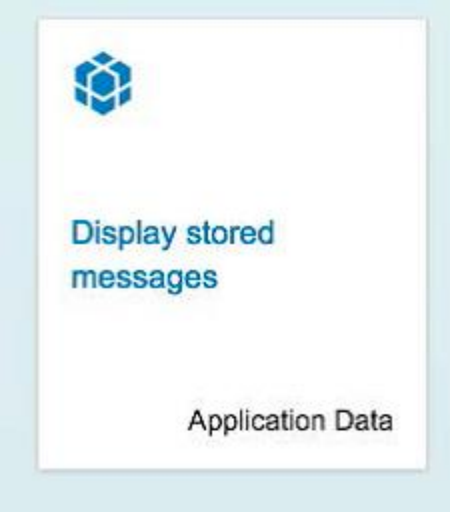
20. Press *Send* and the *Reply from server* console should now show a message like this: 200 {"msg":"1 message(s) received from device [<your Device ID>]"}.



Reply from Server

Code	Message
200	{"msg":"1 message(s) received from device [ed6f1fdd-ec08-4b18-9ba9-5b162bb24978]"}

Congratulations! Now your IoT Services are set up properly and you can see your messages by going back to the *Message Management Service Cockpit* and going to *Display stored messages*. Your messages are usually stored in the first appearing table which is named
T_IOT_<your message type ID>.



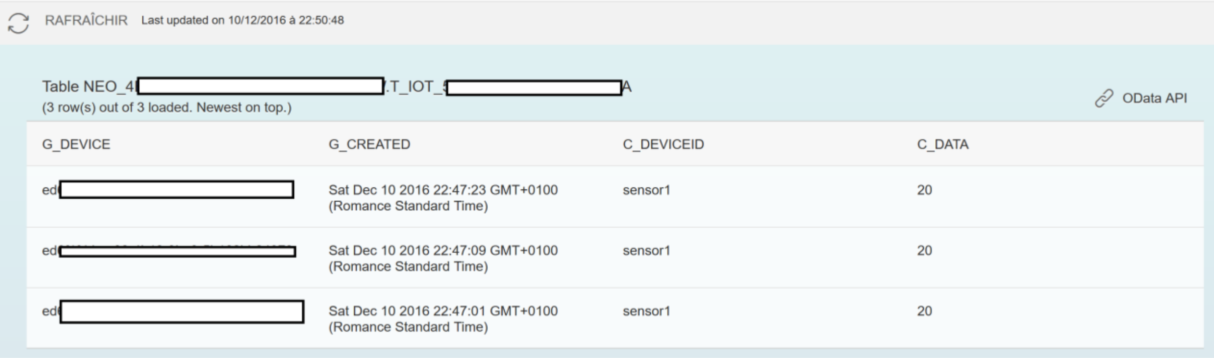

RAFRAÎCHIR Last updated on 10/12/2016 à 22:50:48

Table NEO_4[redacted].T_IOT_[redacted]A
(3 row(s) out of 3 loaded. Newest on top.)

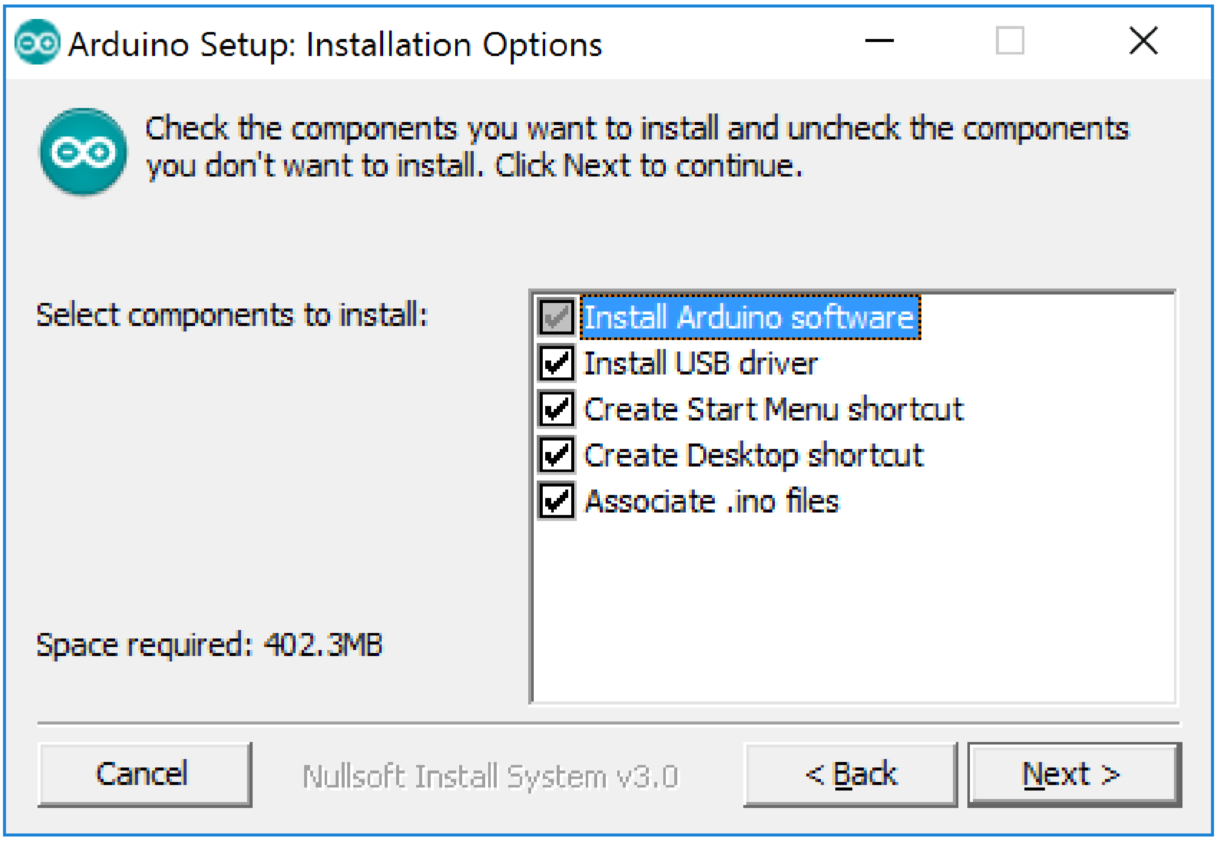
OData API

G_DEVICE	G_CREATED	C_DEVICEID	C_DATA
ed[redacted]	Sat Dec 10 2016 22:47:23 GMT+0100 (Romance Standard Time)	sensor1	20
ed[redacted]	Sat Dec 10 2016 22:47:09 GMT+0100 (Romance Standard Time)	sensor1	20
ed[redacted]	Sat Dec 10 2016 22:47:01 GMT+0100 (Romance Standard Time)	sensor1	20

Arduino

Install the Arduino IDE

1. Get the Arduino IDE installation form here.
2. Start the installation



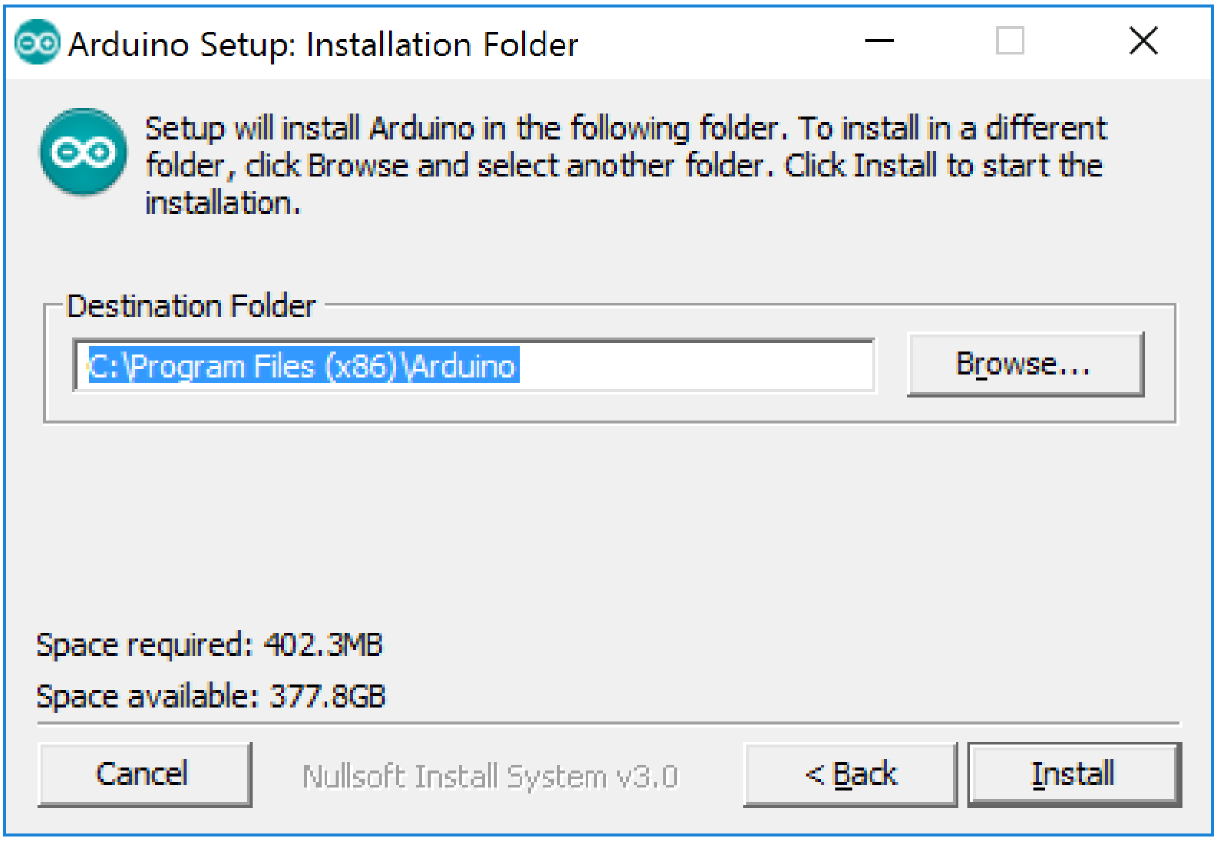
Arduino Setup: Installation Options

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:

- ☑ Install Arduino software
- ☑ Install USB driver
- ☑ Create Start Menu shortcut
- ☑ Create Desktop shortcut
- ☑ Associate .ino files

Space required: 402.3MB



Arduino Setup: Installation Folder

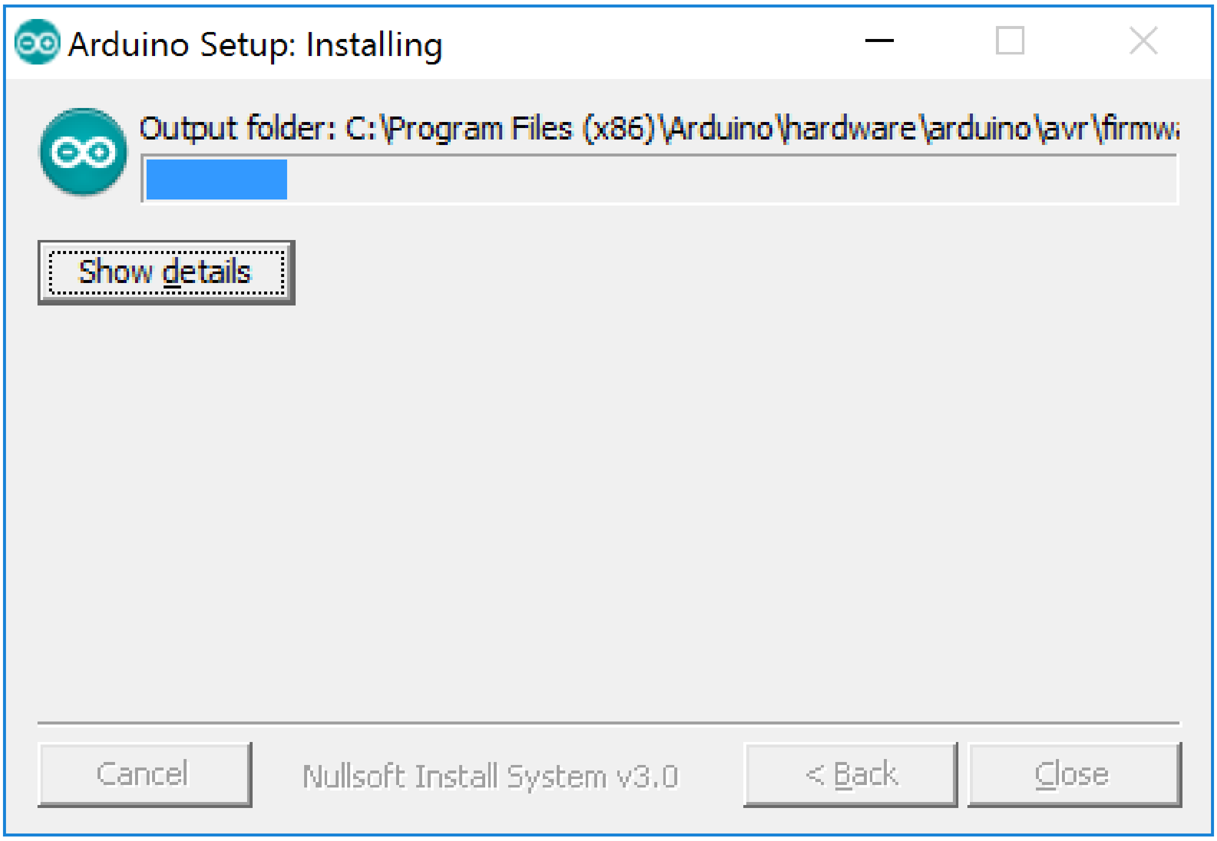
Setup will install Arduino in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

Destination Folder: C:\Program Files (x86)\Arduino

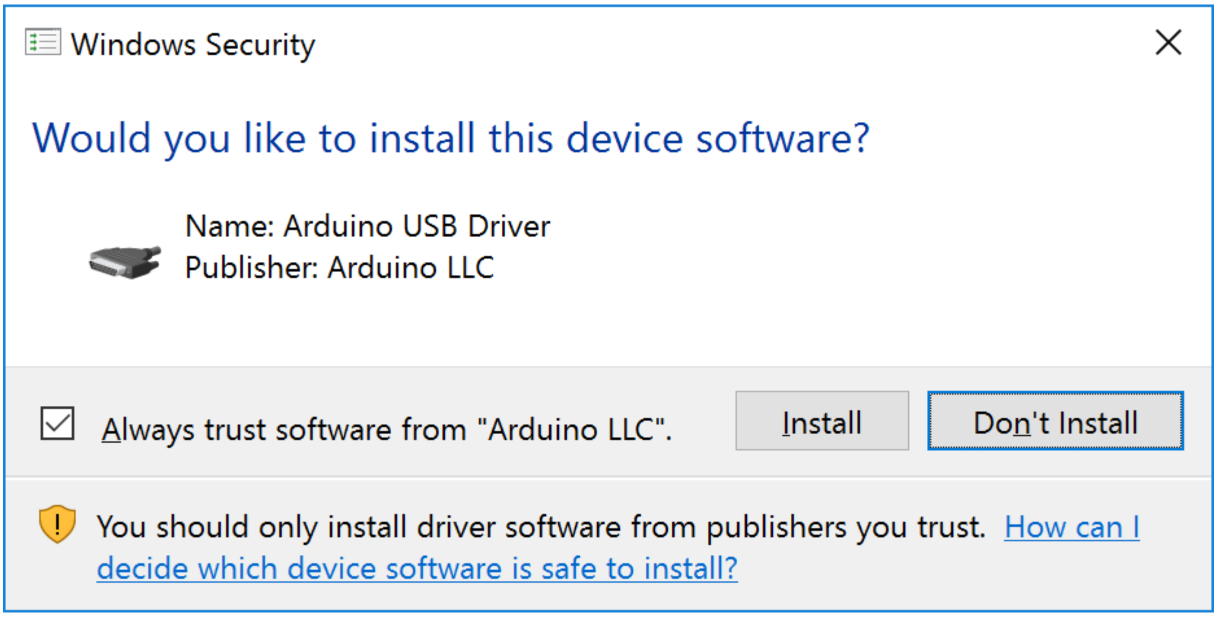
Space required: 402.3MB

Space available: 377.8GB

Cancel
Nullsoft Install System v3.0
< Back
Install



Output folder: C:\Program Files (x86)\Arduino\hardware\arduino\avr\firmw



Windows Security

Would you like to install this device software?

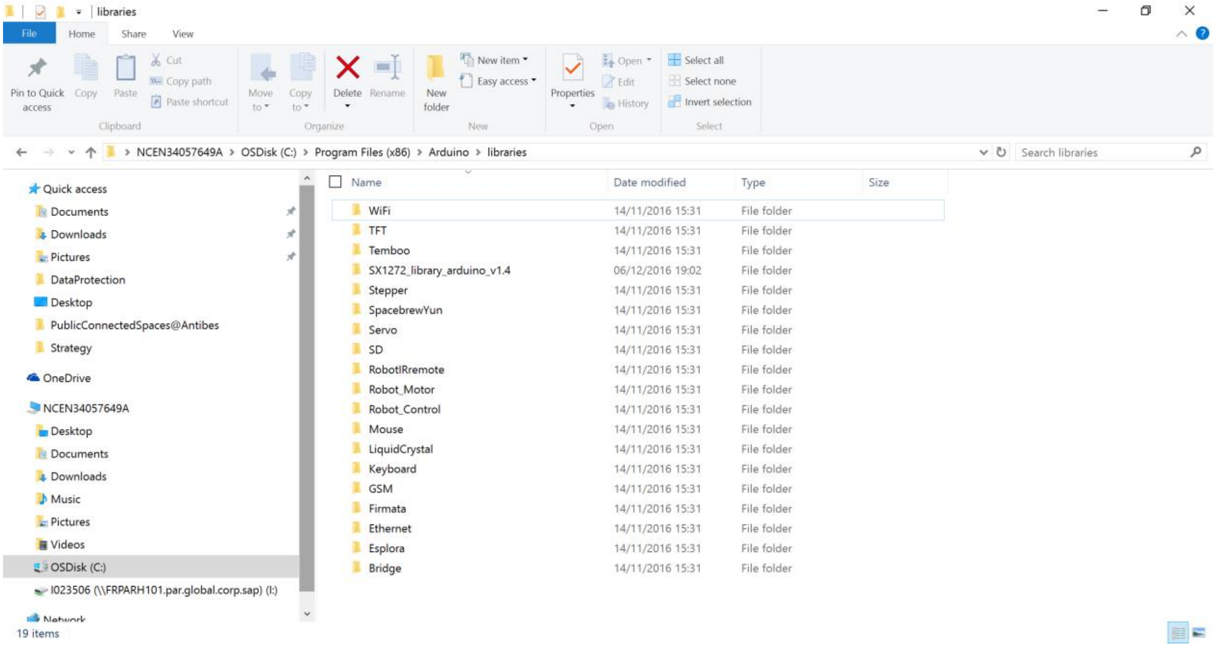
Name: Arduino USB Driver
Publisher: Arduino LLC

☑ Always trust software from "Arduino LLC".

You should only install driver software from publishers you trust. How can I decide which device software is safe to install?

Install the LoRa libraries

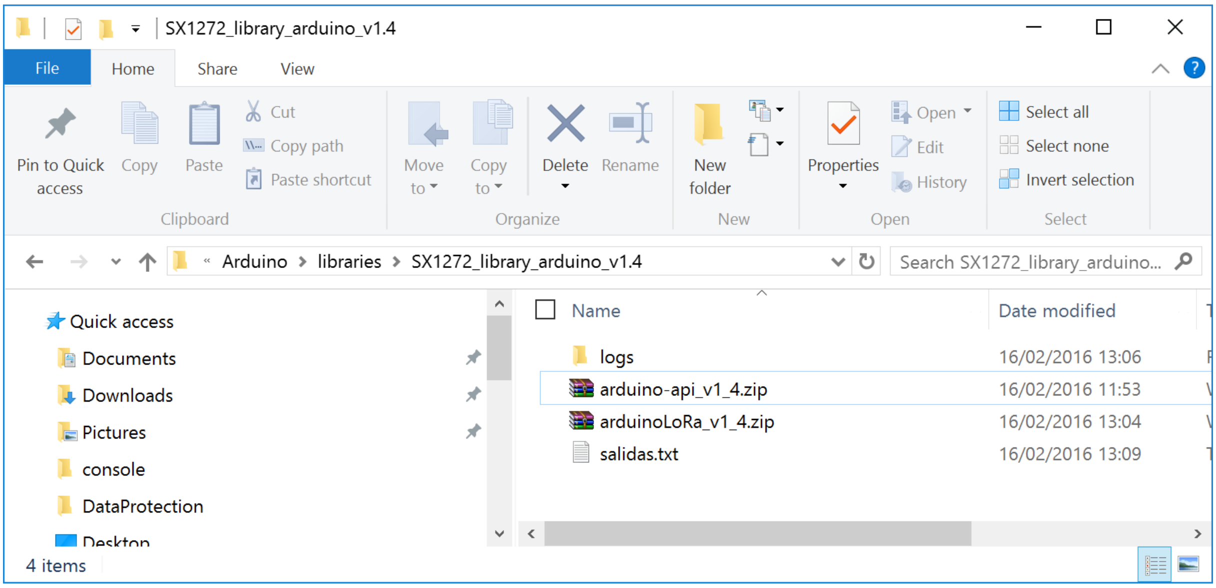
1. Copy the LoRa library into the *<ArduinoInstallationFolder>/libraries* and extract it



NCEN34057649A > OSDisk (C:) > Program Files (x86) > Arduino > libraries

Name	Date modified	Type	Size
WiFi	14/11/2016 15:31	File folder	
TFT	14/11/2016 15:31	File folder	
Temboo	14/11/2016 15:31	File folder	
SX1272_library_arduino_v1.4	06/12/2016 19:02	File folder	
Stepper	14/11/2016 15:31	File folder	
SpacebrewYun	14/11/2016 15:31	File folder	
Servo	14/11/2016 15:31	File folder	
SD	14/11/2016 15:31	File folder	
RobotIRremote	14/11/2016 15:31	File folder	
Robot_Motor	14/11/2016 15:31	File folder	
Robot_Control	14/11/2016 15:31	File folder	
Mouse	14/11/2016 15:31	File folder	
LiquidCrystal	14/11/2016 15:31	File folder	
Keyboard	14/11/2016 15:31	File folder	
GSM	14/11/2016 15:31	File folder	
Firmata	14/11/2016 15:31	File folder	
Ethernet	14/11/2016 15:31	File folder	
Esplora	14/11/2016 15:31	File folder	
Bridge	14/11/2016 15:31	File folder	

2. Unzip the two ZIP *arduino-api_v1_4.zip* and *arduinoLoRa_v_1_4.zip* in *<ArduinoInstallationFolder>/libraries*

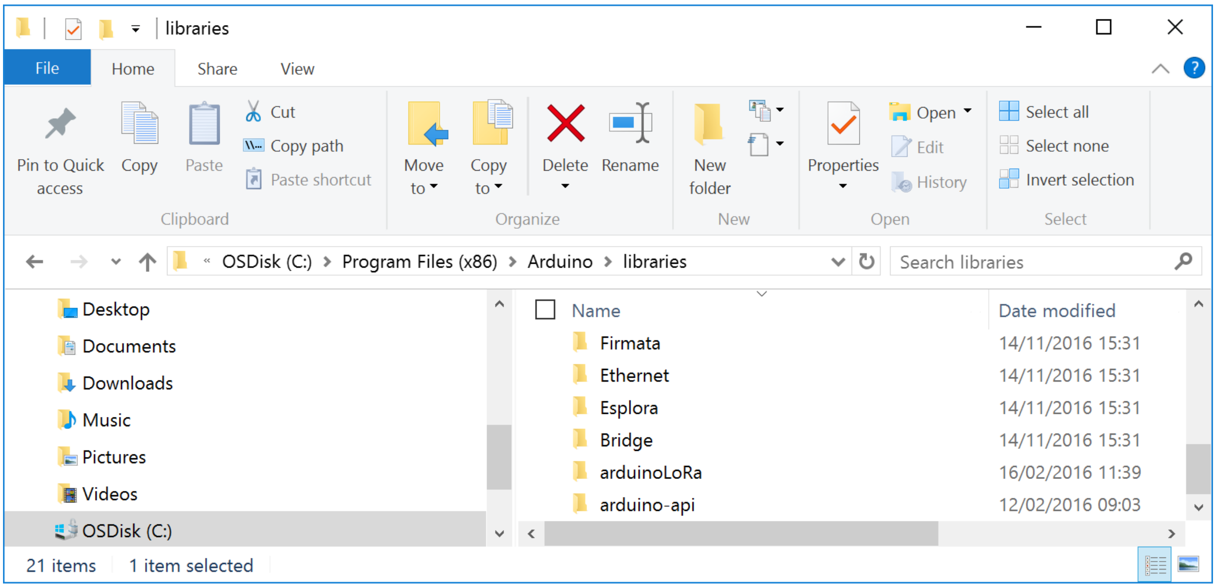


SX1272_library_arduino_v1.4

« Arduino › libraries › SX1272_library_arduino_v1.4

Search SX1272_library_arduino...

Name	Date modified
logs	16/02/2016 13:06
arduino-api_v1_4.zip	16/02/2016 11:53
arduinoLoRa_v1_4.zip	16/02/2016 13:04
salidas.txt	16/02/2016 13:09



libraries

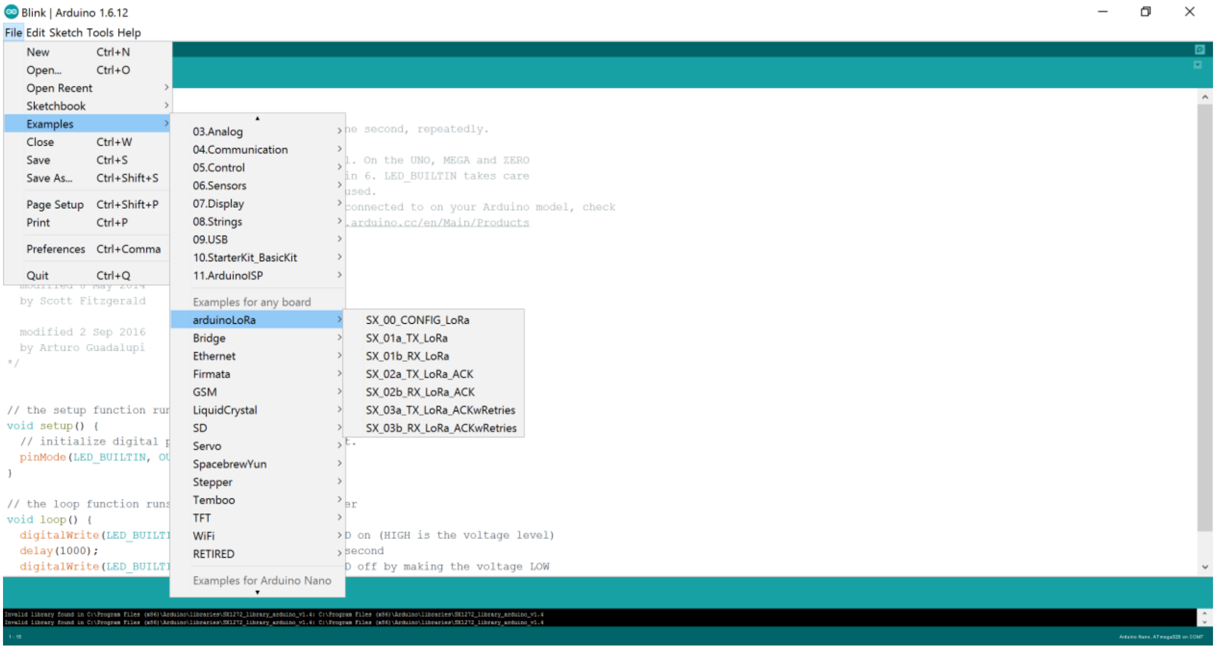
« OSDisk (C:) › Program Files (x86) › Arduino › libraries

Search libraries

Name	Date modified
Firmata	14/11/2016 15:31
Ethernet	14/11/2016 15:31
Esplora	14/11/2016 15:31
Bridge	14/11/2016 15:31
arduinoLoRa	16/02/2016 11:39
arduino-api	12/02/2016 09:03

Check the LoRa module

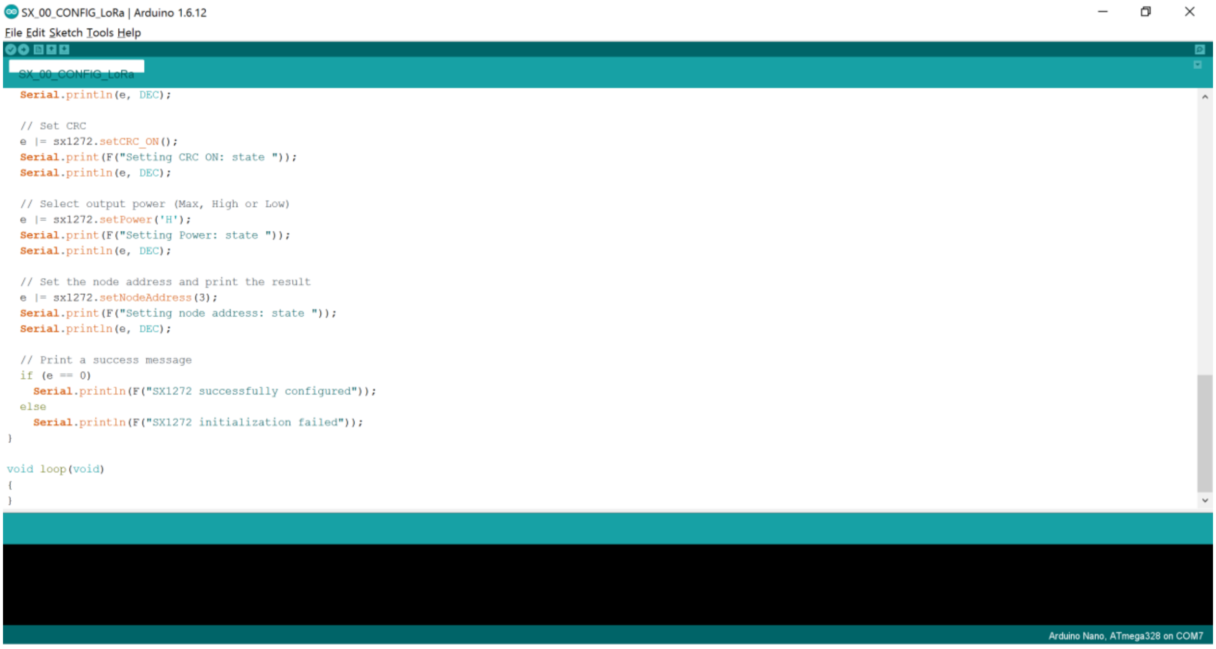
1. Start the Arduino IDE
2. Check for LoRa libraries examples



```
he second, repeatedly.

1. On the UNO, MEGA and ZERO
in 6. LED_BUILTIN takes care
used.
connected to on your Arduino model, check
.arduino.cc/en/Main/Products

  by Scott Fitzgerald

  modified 2 Sep 2016
  by Arturo Guadalupi
*/

// the setup function rur
void setup() {
  // initialize digital p
  pinMode(LED_BUILTIN, OU
}

// the loop function runs
void loop() {
  digitalWrite(LED_BUILTI               D on (HIGH is the voltage level)
  delay(1000);                          second
  digitalWrite(LED_BUILTI               D off by making the voltage LOW
```

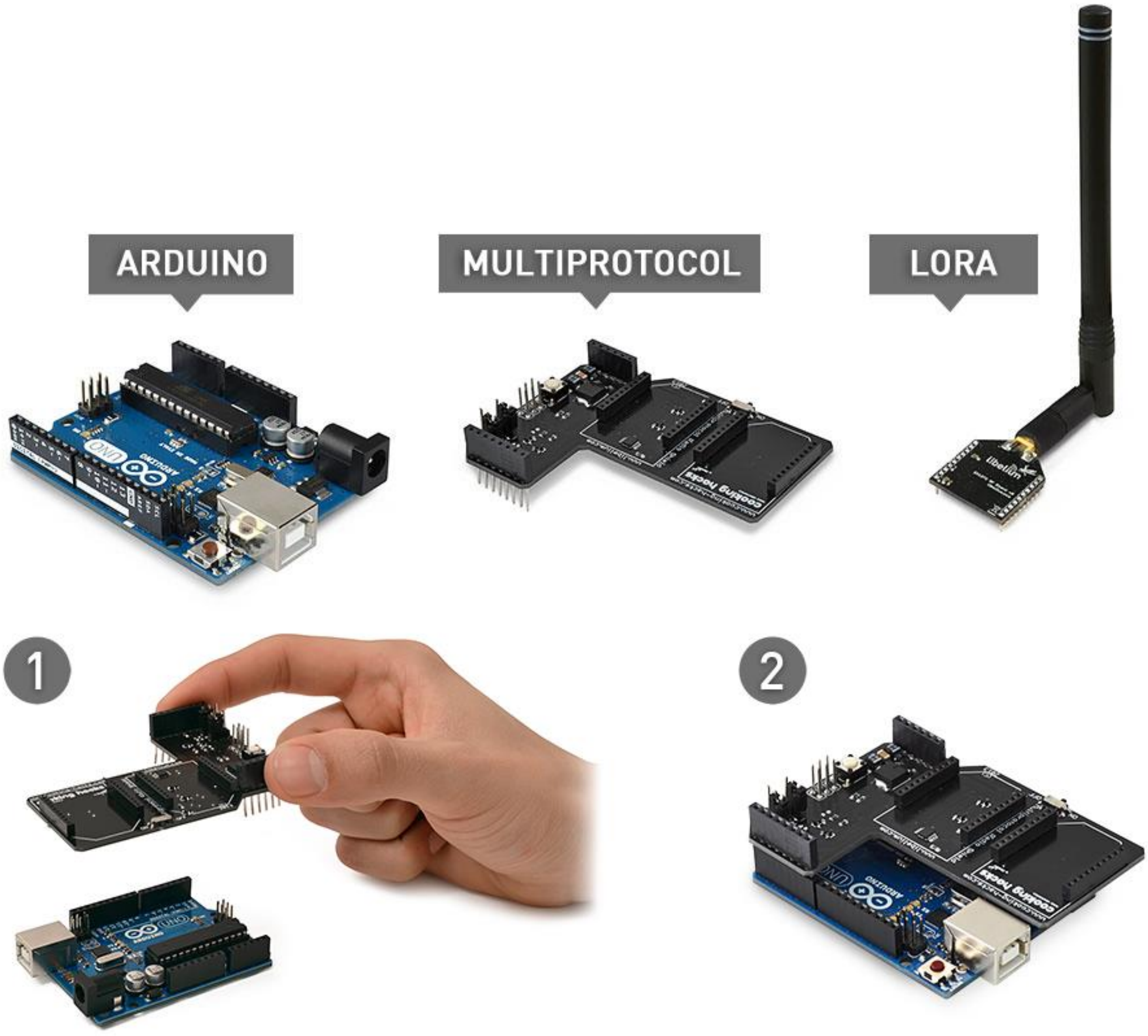
```
Invalid library found in C:\Program Files (x86)\Arduino\libraries\SX1272_library_arduino_v1.4: C:\Program Files (x86)\Arduino\libraries\SX1272_library_arduino_v1.4
Invalid library found in C:\Program Files (x86)\Arduino\libraries\SX1272_library_arduino_v1.4: C:\Program Files (x86)\Arduino\libraries\SX1272_library_arduino_v1.4
```

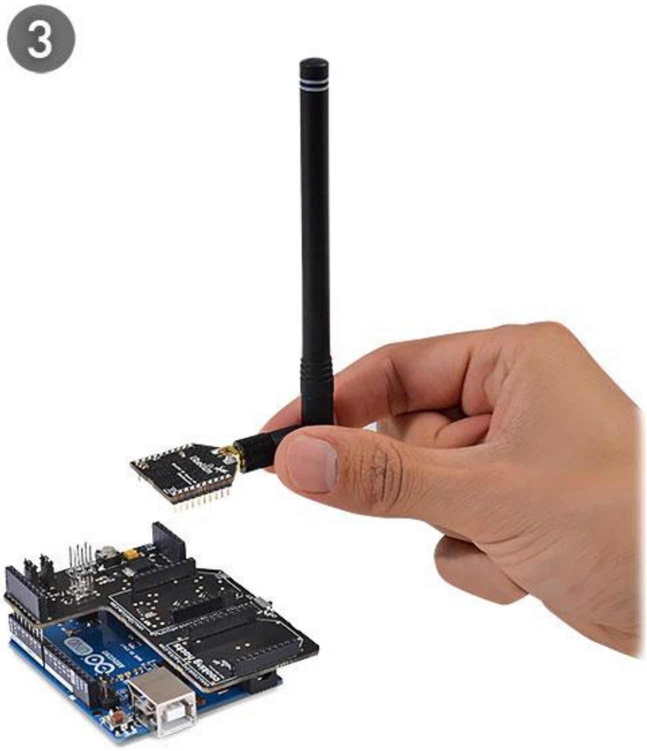
3. Open *SX_00_Config_LoRa*

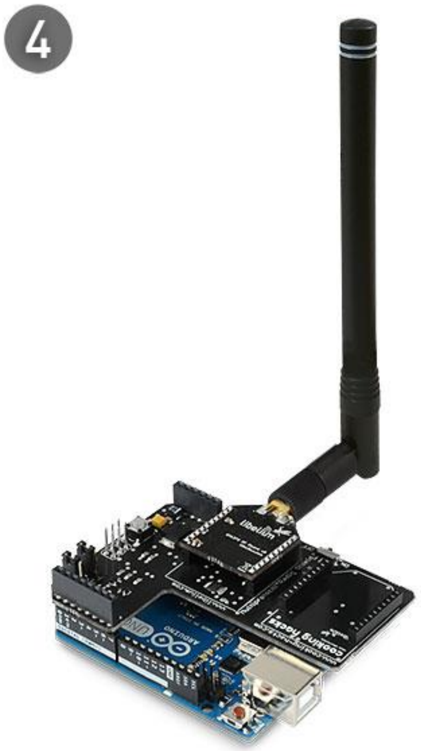

SX_00_CONFIG_LoRa | Arduino 1.6.12
File Edit Sketch Tools Help
SX_00_CONFIG_LoRa
Arduino Nano, ATmega328 on COM7

```
  Serial.println(e, DEC);

  // Set CRC
  e |= sx1272.setCRC_ON();
  Serial.print(F("Setting CRC ON: state "));
  Serial.println(e, DEC);

  // Select output power (Max, High or Low)
  e |= sx1272.setPower('H');
  Serial.print(F("Setting Power: state "));
  Serial.println(e, DEC);

  // Set the node address and print the result
  e |= sx1272.setNodeAddress(3);
  Serial.print(F("Setting node address: state "));
  Serial.println(e, DEC);

  // Print a success message
  if (e == 0)
    Serial.println(F("SX1272 successfully configured"));
  else
    Serial.println(F("SX1272 initialization failed"));
}

void loop(void)
{
}
```

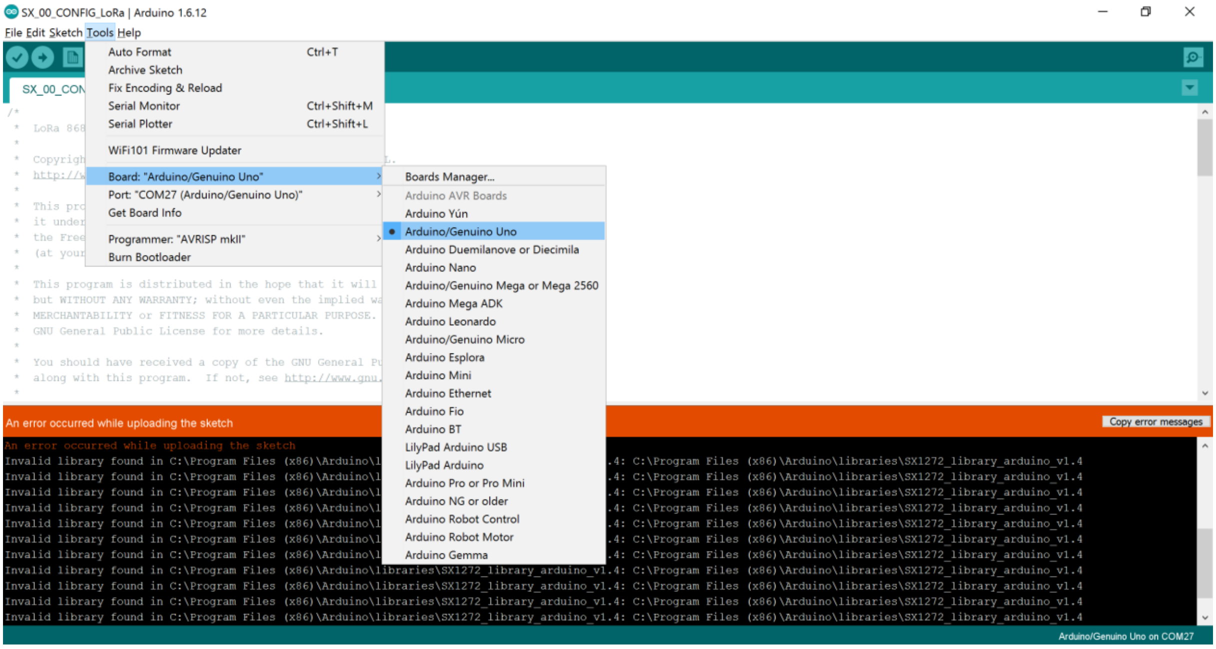
4. Connect the LoRa Arduino module to the Arduino Uno


ARDUINO
MULTIPROTOCOL
LORA
1
2





5. Configure the board as *Arduino Uno*

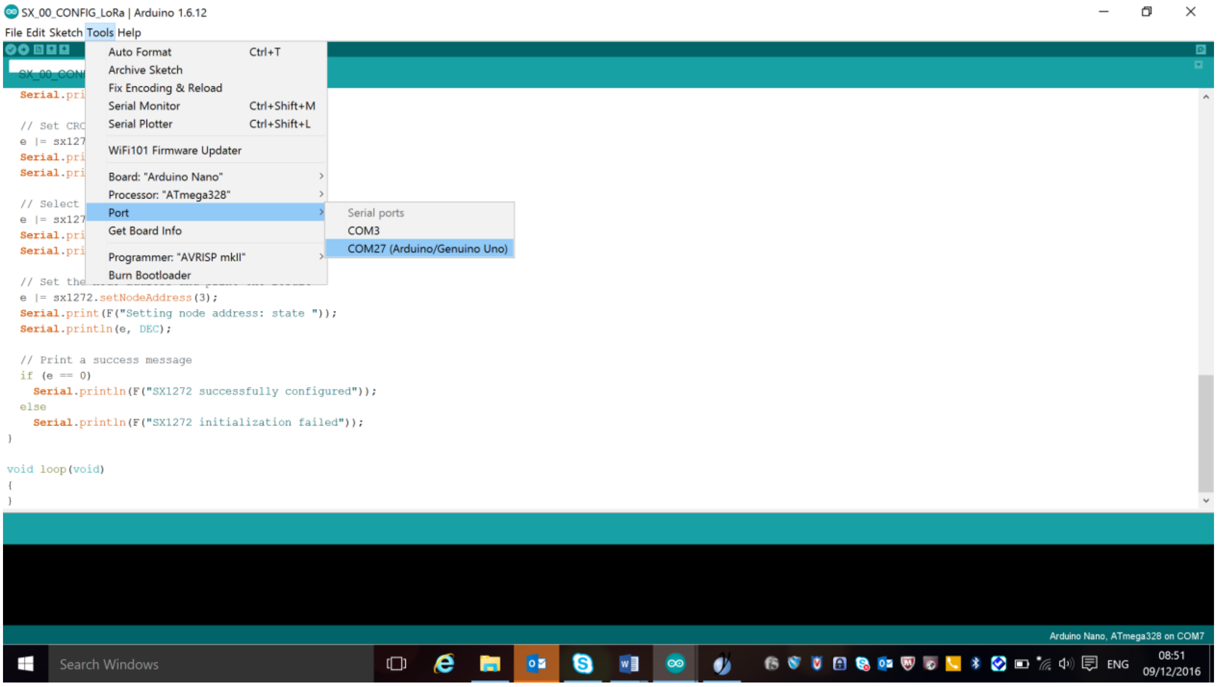


```
/*
 *  LoRa 868
 *
 *  Copyrigh
 *  http://w
 *
 *  This pro
 *  it under
 *  the Free
 *  (at your
 *
 *  This program is distributed in the hope that it will
 *  but WITHOUT ANY WARRANTY; without even the implied wa
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 *  GNU General Public License for more details.
 *
 *  You should have received a copy of the GNU General Pu
 *  along with this program.  If not, see http://www.gnu.
 *
```

An error occurred while uploading the sketch

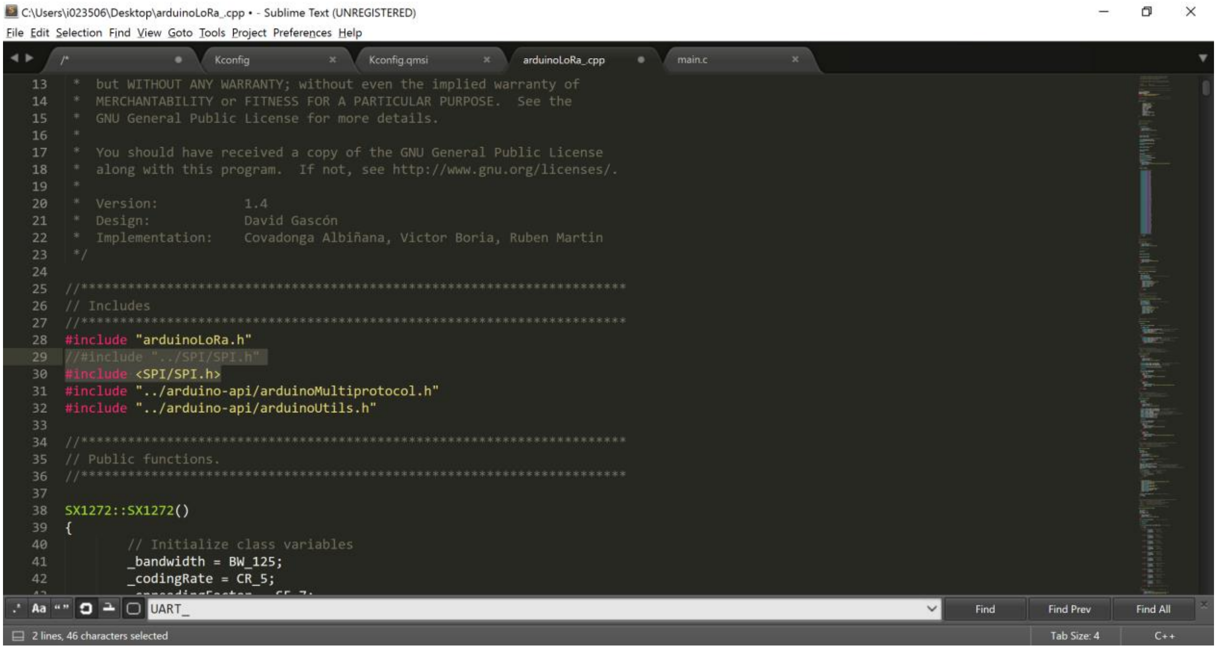
```
An error occurred while uploading the sketch
Invalid library found in C:\Program Files (x86)\Arduino\l                      .4: C:\Program Files (x86)\Arduino\libraries\SX1272_library_arduino_v1.4
Invalid library found in C:\Program Files (x86)\Arduino\l                      .4: C:\Program Files (x86)\Arduino\libraries\SX1272_library_arduino_v1.4
Invalid library found in C:\Program Files (x86)\Arduino\l                      .4: C:\Program Files (x86)\Arduino\libraries\SX1272_library_arduino_v1.4
Invalid library found in C:\Program Files (x86)\Arduino\l                      .4: C:\Program Files (x86)\Arduino\libraries\SX1272_library_arduino_v1.4
Invalid library found in C:\Program Files (x86)\Arduino\l                      .4: C:\Program Files (x86)\Arduino\libraries\SX1272_library_arduino_v1.4
Invalid library found in C:\Program Files (x86)\Arduino\l                      .4: C:\Program Files (x86)\Arduino\libraries\SX1272_library_arduino_v1.4
Invalid library found in C:\Program Files (x86)\Arduino\l                      .4: C:\Program Files (x86)\Arduino\libraries\SX1272_library_arduino_v1.4
Invalid library found in C:\Program Files (x86)\Arduino\libraries\SX1272_library_arduino_v1.4: C:\Program Files (x86)\Arduino\libraries\SX1272_library_arduino_v1.4
Invalid library found in C:\Program Files (x86)\Arduino\libraries\SX1272_library_arduino_v1.4: C:\Program Files (x86)\Arduino\libraries\SX1272_library_arduino_v1.4
Invalid library found in C:\Program Files (x86)\Arduino\libraries\SX1272_library_arduino_v1.4: C:\Program Files (x86)\Arduino\libraries\SX1272_library_arduino_v1.4
Invalid library found in C:\Program Files (x86)\Arduino\libraries\SX1272_library_arduino_v1.4: C:\Program Files (x86)\Arduino\libraries\SX1272_library_arduino_v1.4
```

6. Configure the Port



```
  Serial.pri

  // Set CRC
  e |= sx127
  Serial.pri
  Serial.pri

  // Select
  e |= sx127
  Serial.pri
  Serial.pri

  // Set the
  e |= sx1272.setNodeAddress(3);
  Serial.print(F("Setting node address: state "));
  Serial.println(e, DEC);

  // Print a success message
  if (e == 0)
    Serial.println(F("SX1272 successfully configured"));
  else
    Serial.println(F("SX1272 initialization failed"));
}

void loop(void)
{
}
```

7. Modify *arduinoLoRa.ccp* as follows:

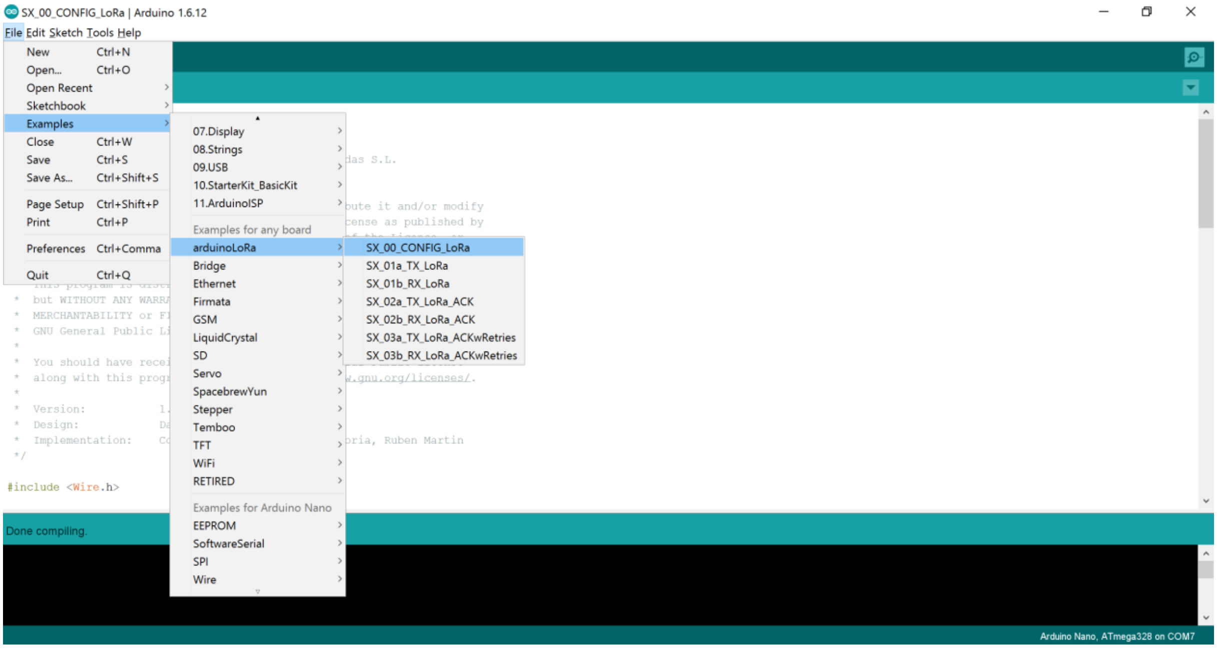

C:\Users\i023506\Desktop\arduinoLoRa_.cpp • - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
/*
Kconfig
Kconfig.qmsi
arduinoLoRa_.cpp
main.c
Find
Find Prev
Find All
Tab Size: 4
C++

```
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *  GNU General Public License for more details.
 *
 *  You should have received a copy of the GNU General Public License
 *  along with this program.  If not, see http://www.gnu.org/licenses/.
 *
 *  Version:           1.4
 *  Design:            David Gascón
 *  Implementation:    Covadonga Albiñana, Victor Boria, Ruben Martin
 */

//**********************************************************************
// Includes
//**********************************************************************
#include "arduinoLoRa.h"
//#include "../SPI/SPI.h"
#include <SPI/SPI.h>
#include "../arduino-api/arduinoMultiprotocol.h"
#include "../arduino-api/arduinoUtils.h"

//**********************************************************************
// Public functions.
//**********************************************************************

SX1272::SX1272()
{
	// Initialize class variables
	_bandwidth = BW_125;
	_codingRate = CR_5;
```

UART_

2 lines, 46 characters selected

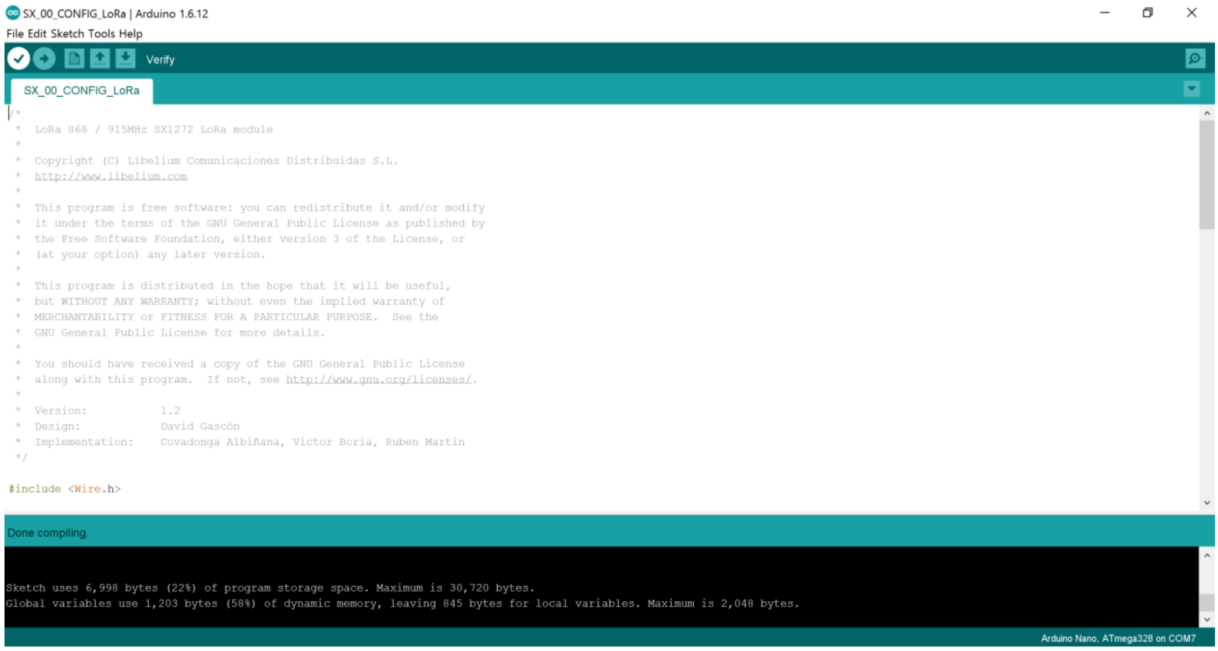
8. Load SX_00_CONFIG_LoRa



```
#include <Wire.h>
```

Done compiling.

9. Compile it

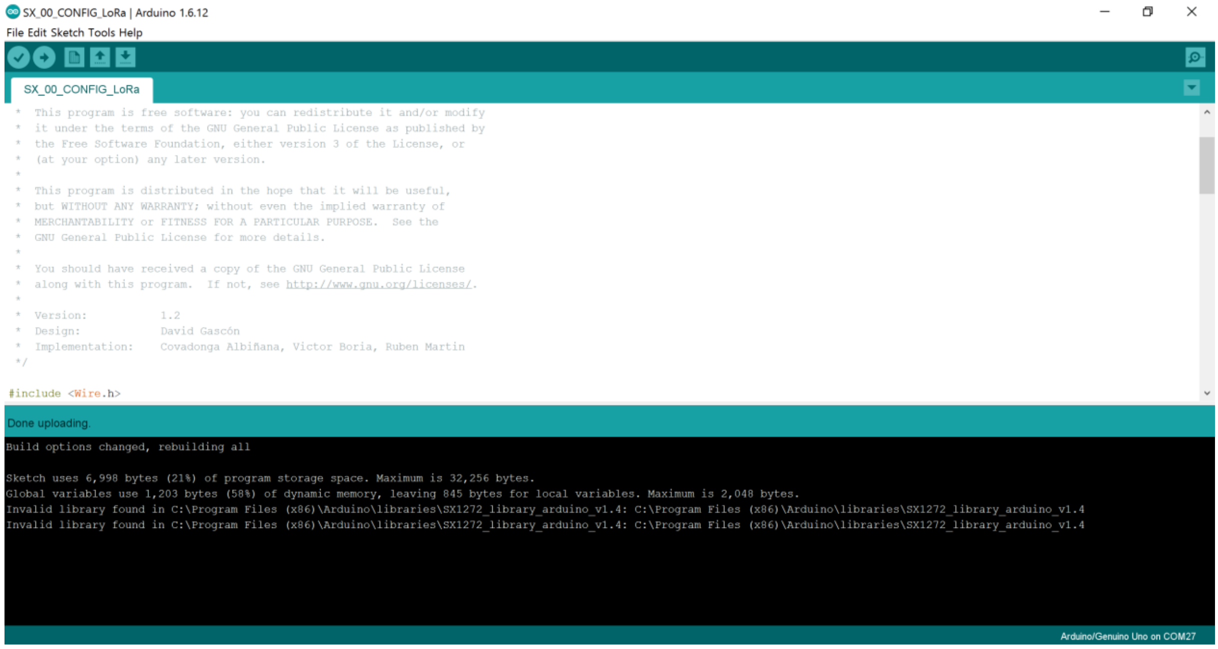

SX_00_CONFIG_LoRa | Arduino 1.6.12
File Edit Sketch Tools Help
Verify
SX_00_CONFIG_LoRa

```
/*
 *  LoRa 868 / 915MHz SX1272 LoRa module
 *
 *  Copyright (C) Libelium Comunicaciones Distribuidas S.L.
 *  http://www.libelium.com
 *
 *  This program is free software: you can redistribute it and/or modify
 *  it under the terms of the GNU General Public License as published by
 *  the Free Software Foundation, either version 3 of the License, or
 *  (at your option) any later version.
 *
 *  This program is distributed in the hope that it will be useful,
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *  GNU General Public License for more details.
 *
 *  You should have received a copy of the GNU General Public License
 *  along with this program.  If not, see http://www.gnu.org/licenses/.
 *
 *  Version:           1.2
 *  Design:            David Gascón
 *  Implementation:    Covadonga Albiñana, Victor Boria, Ruben Martin
 */

#include <Wire.h>
```

Done compiling.

```
Sketch uses 6,998 bytes (22%) of program storage space. Maximum is 30,720 bytes.
Global variables use 1,203 bytes (58%) of dynamic memory, leaving 845 bytes for local variables. Maximum is 2,048 bytes.
```

10. Load it on the board

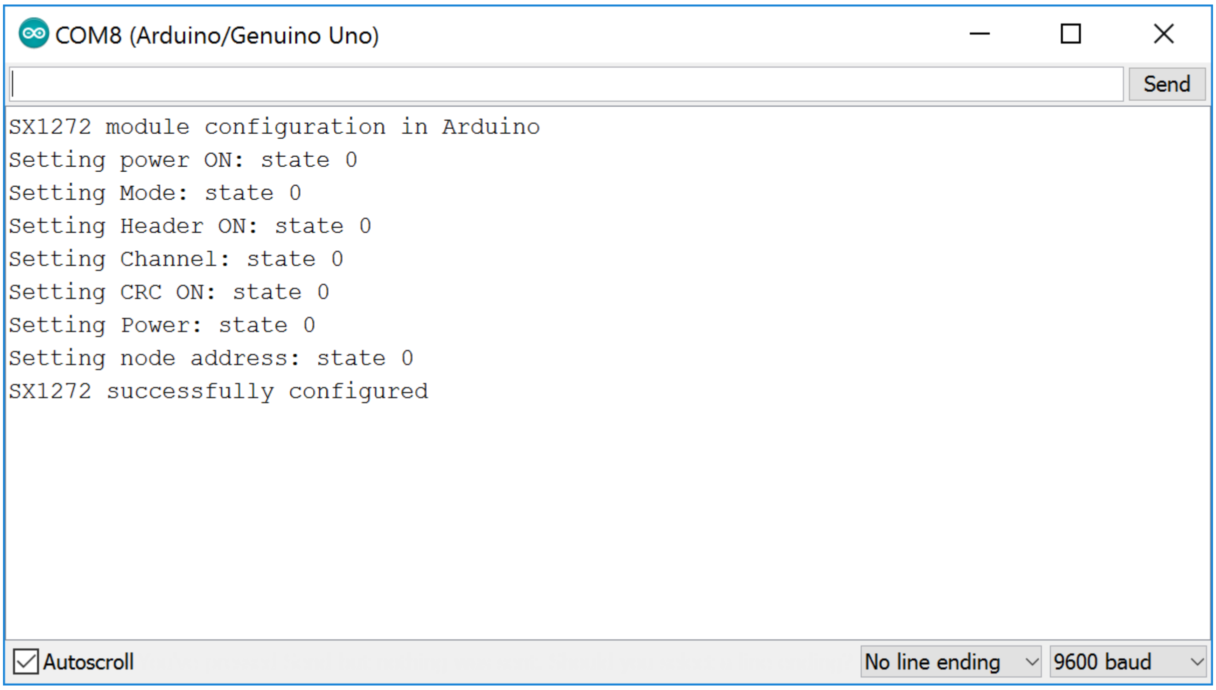

SX_00_CONFIG_LoRa | Arduino 1.6.12
File Edit Sketch Tools Help
SX_00_CONFIG_LoRa

```
 *  This program is free software: you can redistribute it and/or modify
 *  it under the terms of the GNU General Public License as published by
 *  the Free Software Foundation, either version 3 of the License, or
 *  (at your option) any later version.
 *
 *  This program is distributed in the hope that it will be useful,
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *  GNU General Public License for more details.
 *
 *  You should have received a copy of the GNU General Public License
 *  along with this program.  If not, see http://www.gnu.org/licenses/.
 *
 *  Version:           1.2
 *  Design:            David Gascón
 *  Implementation:    Covadonga Albiñana, Victor Boria, Ruben Martin
 */

#include <Wire.h>
```

Done uploading.

```
Build options changed, rebuilding all

Sketch uses 6,998 bytes (21%) of program storage space. Maximum is 32,256 bytes.
Global variables use 1,203 bytes (58%) of dynamic memory, leaving 845 bytes for local variables. Maximum is 2,048 bytes.
Invalid library found in C:\Program Files (x86)\Arduino\libraries\SX1272_library_arduino_v1.4: C:\Program Files (x86)\Arduino\libraries\SX1272_library_arduino_v1.4
Invalid library found in C:\Program Files (x86)\Arduino\libraries\SX1272_library_arduino_v1.4: C:\Program Files (x86)\Arduino\libraries\SX1272_library_arduino_v1.4
```

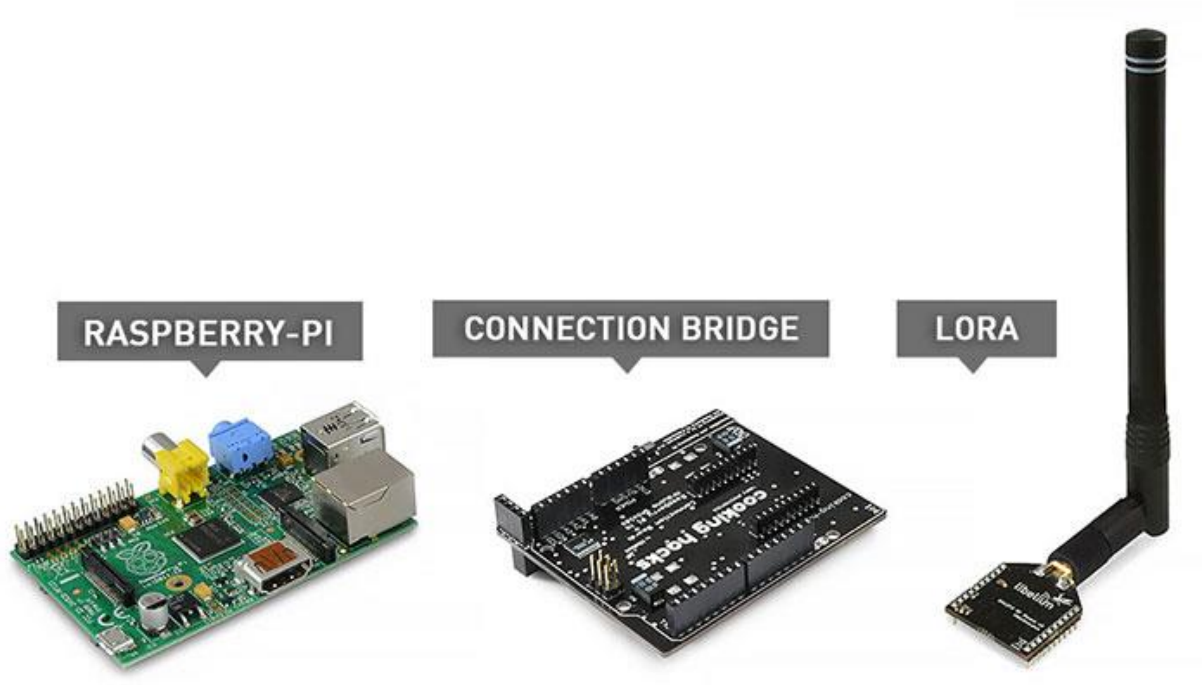
11. Check the output in the Serial Monitor
    Start the serial monitor with *CTRL+SHIFT+M*

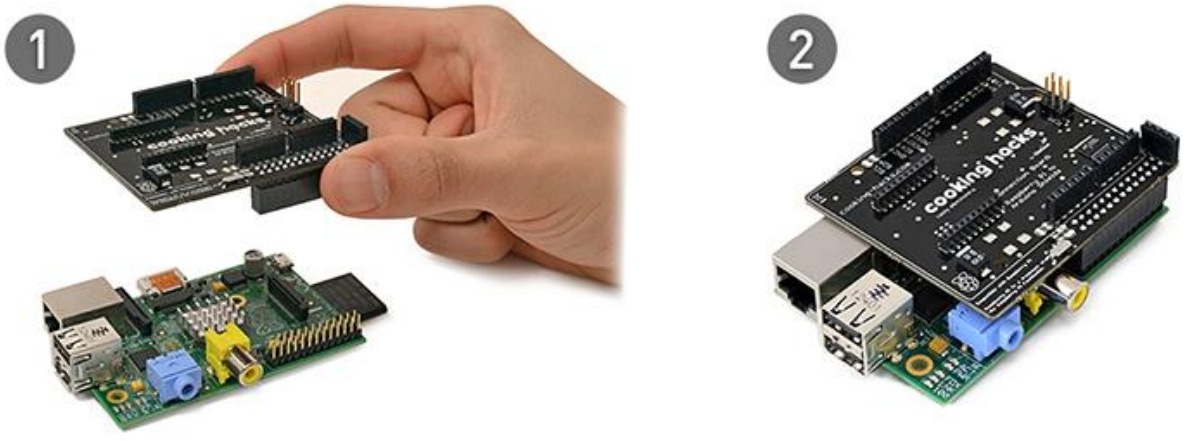


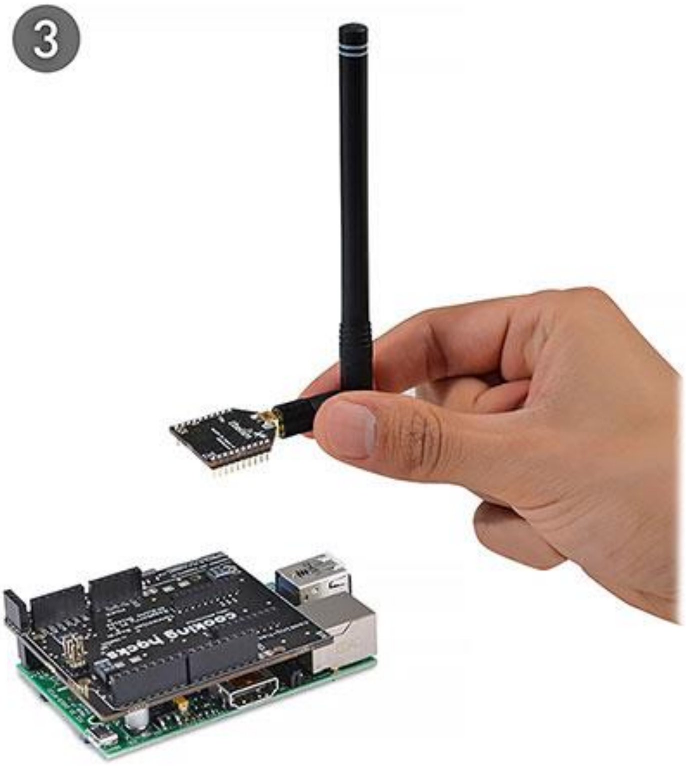
```
SX1272 module configuration in Arduino
Setting power ON: state 0
Setting Mode: state 0
Setting Header ON: state 0
Setting Channel: state 0
Setting CRC ON: state 0
Setting Power: state 0
Setting node address: state 0
SX1272 successfully configured
```

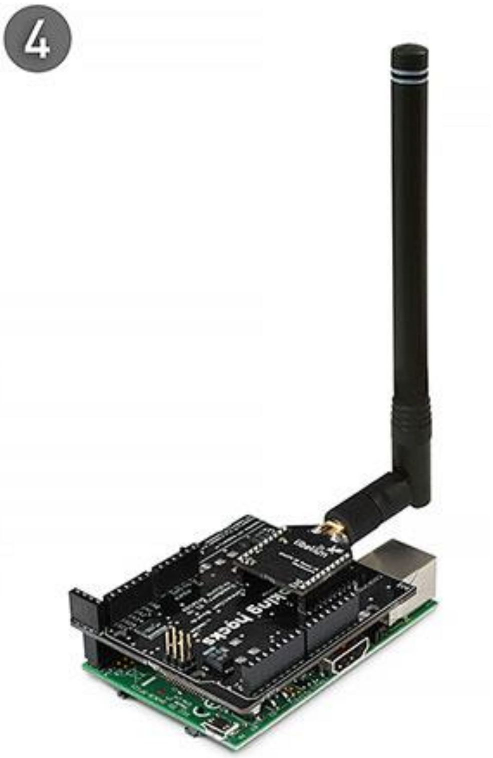
LoRa gateway

1. Plug the LoRa module on the rapsberrypi









2. Log to the raspberry pi

3. Install the library on the raspberrypi.

The following instructions can be found here.

The SX1272 library for Raspberry Pi requires the ArduPi library and both libraries should be in the same path.

Download the SX1272 Libraries for Raspberry Pi.

```
wget http://www.cooking-
hacks.com/media/cooking/images/documentation/tutorial_SX1272/arduPi-
api_LoRa_v1_4.zip && unzip -u arduPi-api_LoRa_v1_4.zip && cd
cooking/examples/LoRa && chmod +x cook.sh && cd ../../..
```

4. Install ArduPi library

ArduPi For Raspberry Pi:

```
wget http://www.cooking-
hacks.com/media/cooking/images/documentation/raspberry_arduino_shield
/raspberrypi.zip && unzip raspberrypi.zip && cd cooking/arduPi &&
chmod +x install_arduPi && ./install_arduPi && rm install_arduPi &&
cd ../..
```

ArduPi For Raspberry Pi 2:

```
wget http://www.cooking-
hacks.com/media/cooking/images/documentation/raspberry_arduino_shield
```

```
/raspberrypi2.zip && unzip raspberrypi2.zip && cd cooking/arduPi &&
chmod +x install_arduPi && ./install_arduPi && rm install_arduPi &&
cd ../..
```

5. Go to examples folder:

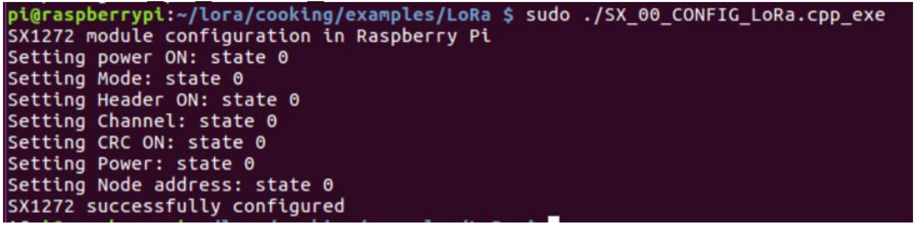
```
cd cooking/examples/LoRa/
```

6. Compile SX_00_CONFIG_LoRa.cpp:

```
./cook.sh SX_00_CONFIG_LoRa.cpp
```

7. Start *SX_00_CONFIG_LoRa.cpp.exe*

```
sudo ./SX_00_CONFIG_LoRa.cpp
```



```
pi@raspberrypi:~/lora/cooking/examples/LoRa $ sudo ./SX_00_CONFIG_LoRa.cpp_exe
SX1272 module configuration in Raspberry Pi
Setting power ON: state 0
Setting Mode: state 0
Setting Header ON: state 0
Setting Channel: state 0
Setting CRC ON: state 0
Setting Power: state 0
Setting Node address: state 0
SX1272 successfully configured
```

Send messages from Arduino to HCP

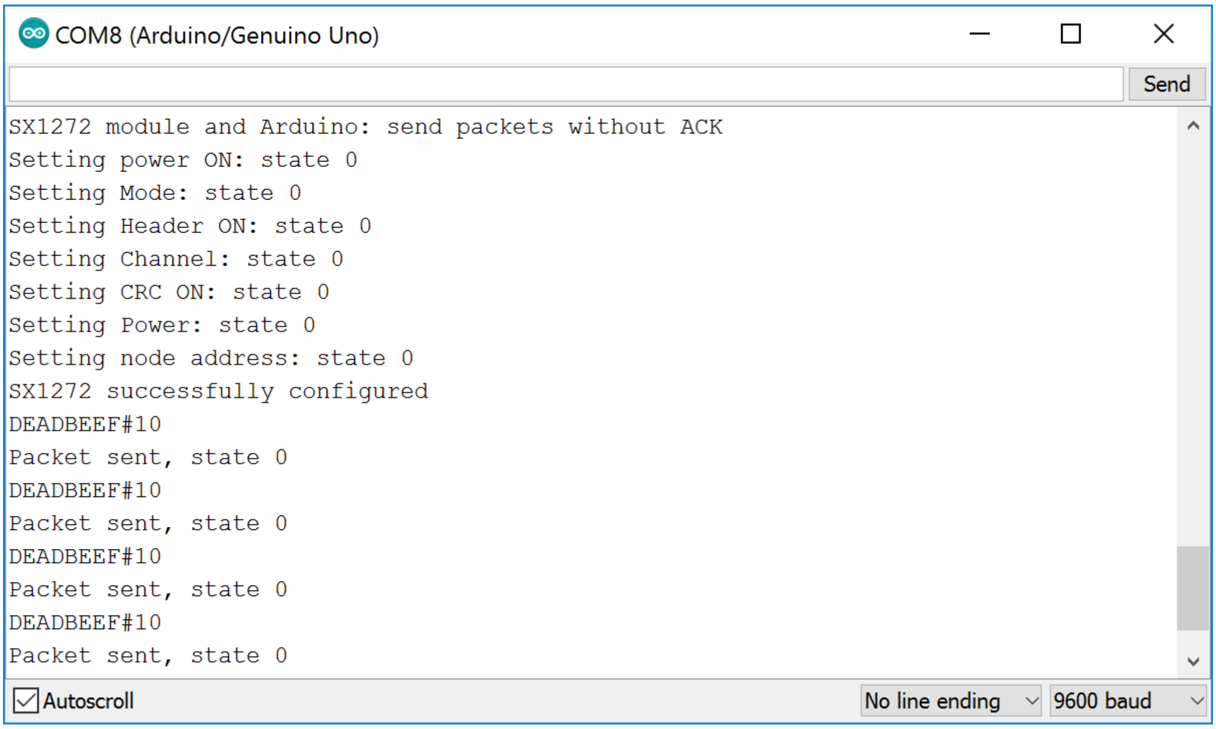
Push data from Arduino

1. Create a new sketch in the Arduino IDE
2. Copy paste the following program

```
/*
 *  LoRa 868 / 915MHz SX1272 LoRa module
 *
 *  Copyright (C) Libelium Comunicaciones Distribuidas S.L.
 *  http://www.libelium.com
 *
 *  This program is free software: you can redistribute it and/or modify
 *  it under the terms of the GNU General Public License as published by
 *  the Free Software Foundation, either version 3 of the License, or
 *  (at your option) any later version.
 *
 *  This program is distributed in the hope that it will be useful,
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *  GNU General Public License for more details.
 *
 *  You should have received a copy of the GNU General Public License
 *  along with this program.  If not, see http://www.gnu.org/licenses/.
 *
 *  Version:          1.2
 *  Design:           David Gascón
 *  Implementation:   Covadonga Albiñana, Victor Boria, Ruben Martin
 */

#include <Wire.h>

// Cooking API libraries
#include <arduinoUtils.h>

// Include the SX1272 and SPI library:
#include "arduinoLoRa.h"
#include <SPI.h>

int e;

void setup()
{
  // Open serial communications and wait for port to open:
  Serial.begin(9600);

  // Print a start message
  Serial.println(F("SX1272 module and Arduino: send packets without ACK"));

  // Power ON the module
  e = sx1272.ON();
  Serial.print(F("Setting power ON: state "));
  Serial.println(e, DEC);

  // Set transmission mode and print the result
  e |= sx1272.setMode(4);
  Serial.print(F("Setting Mode: state "));
  Serial.println(e, DEC);

  // Set header
  e |= sx1272.setHeaderON();
  Serial.print(F("Setting Header ON: state "));
  Serial.println(e, DEC);

  // Select frequency channel
  e |= sx1272.setChannel(CH_10_868);
  Serial.print(F("Setting Channel: state "));
  Serial.println(e, DEC);

  // Set CRC
  e |= sx1272.setCRC_ON();
  Serial.print(F("Setting CRC ON: state "));
  Serial.println(e, DEC);

  // Select output power (Max, High or Low)
  e |= sx1272.setPower('H');
  Serial.print(F("Setting Power: state "));
  Serial.println(e, DEC);

  // Set the node address and print the result
  e |= sx1272.setNodeAddress(3);
  Serial.print(F("Setting node address: state "));
  Serial.println(e, DEC);

  // Print a success message
  if (e == 0)
    Serial.println(F("SX1272 successfully configured"));
  else
    Serial.println(F("SX1272 initialization failed"));
}

void loop(void)
{
  /////////////////////////////////
  // Send data and print the result in the console
  // Prepare the message
  long data = random(100);
  char deviceid[] = "DEADBEEF";
  char message[15]= "";
  strcpy(message, deviceid);
  strcat(message, "#");
  strcat(message, "10");
  Serial.println(message);
```

```
        // Send the message to the lora module
        e = sx1272.sendPacketTimeout(8, message);
        // Debug output
        Serial.print(F("Packet sent, state "));
        Serial.println(e, DEC);

        // Wait 4 seconds
        delay(4000);
}
```

3. Change the *deviceid*
4. Compile and load the program



```
SX1272 module and Arduino: send packets without ACK
Setting power ON: state 0
Setting Mode: state 0
Setting Header ON: state 0
Setting Channel: state 0
Setting CRC ON: state 0
Setting Power: state 0
Setting node address: state 0
SX1272 successfully configured
DEADBEEF#10
Packet sent, state 0
DEADBEEF#10
Packet sent, state 0
DEADBEEF#10
Packet sent, state 0
DEADBEEF#10
Packet sent, state 0
```

☑ Autoscroll   No line ending   9600 baud

Receive data on the gateway

1. Create the following cpp code under <*cooking/examples/LoRa*>

```
/*  LoRa 868 / 915MHz SX1272 LoRa module
 *
 *  Copyright (C) Libelium Comunicaciones Distribuidas S.L.
 *  http://www.libelium.com
 *
 *  This program is free software: you can redistribute it and/or modify
 *  it under the terms of the GNU General Public License as published by
 *  the Free Software Foundation, either version 3 of the License, or
 *  (at your option) any later version.
 *
 *  This program is distributed in the hope that it will be useful,
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *  GNU General Public License for more details.
 *
 *  You should have received a copy of the GNU General Public License
 *  along with this program.  If not, see http://www.gnu.org/licenses/.
 *
 *  Version:          1.2
 *  Design:           David Gascón
 *  Implementation:    Covadonga Albiñana, Victor Boria, Ruben Martin
 */

// Include the SX1272 and SPI library:
#include "arduPiLoRa.h"

#include <cstdlib>
#include <iostream>


int e;
char my_packet[100];

void setup()
{
  // Print a start message
  printf("SX1272 module and Raspberry Pi: receive packets without ACK\n");

  // Power ON the module
  e = sx1272.ON();
  printf("Setting power ON: state %d\n", e);

  // Set transmission mode
  e |= sx1272.setMode(4);
  printf("Setting Mode: state %d\n", e);

  // Set header
  e |= sx1272.setHeaderON();
  printf("Setting Header ON: state %d\n", e);

  // Select frequency channel
  e |= sx1272.setChannel(CH_10_868);
  printf("Setting Channel: state %d\n", e);

  // Set CRC
  e |= sx1272.setCRC_ON();
  printf("Setting CRC ON: state %d\n", e);

  // Select output power (Max, High or Low)
  e |= sx1272.setPower('H');
  printf("Setting Power: state %d\n", e);

  // Set the node address
  e |= sx1272.setNodeAddress(8);
  printf("Setting Node address: state %d\n", e);

  // Print a success message
  if (e == 0)
    printf("SX1272 successfully configured\n");
  else
    printf("SX1272 initialization failed\n");

  delay(1000);
}

void loop(void)
{
  // Receive message
  e = sx1272.receivePacketTimeout(10000);

  // if there is no error
  if ( e == 0 )
  {

    for (unsigned int i = 0; i < sx1272.packet_received.length; i++)
    {
      my_packet[i] = (char)sx1272.packet_received.data[i];
    }

    // Change the path to your send2HCP python script
    char cmd[100];
    strcpy(cmd,"python /home/pi/cooking/examples/LoRa/send2HCP.py ");

    // Discard message containing anything but [0-9][a-z][A-Z][#]
    int validCharacters = strspn(my_packet,"0123456789abcdefunABCDEFUN#");
    if (validCharacters == strlen(my_packet)){
      // contains only listed chars
      strcat(cmd,my_packet);
      printf("C %s\n",cmd);
      printf("Message sent: %s\n", my_packet);
      system(cmd);
    } else {
      // contains other chars
      printf("Message NOT sent: %s\n", my_packet);
    }
  }
  else {
    printf("Receive packet, state %d\n",e);
  }
}

int main (){
  setup();
  while(1){
    loop();
  }
  return (0);
}
```

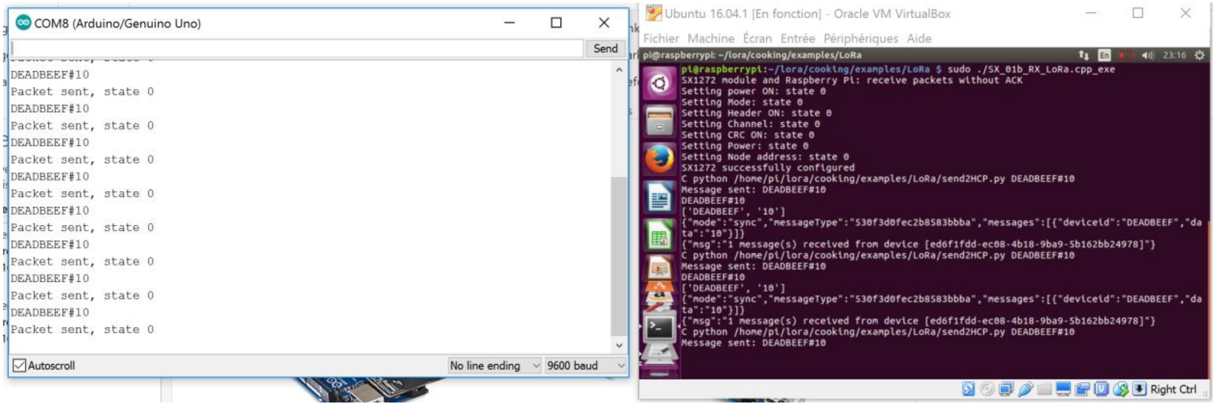
2. Create send2HCP.py under *<cooking/examples/LoRa>*

```
import requests
import sys
import time

def main():
        # Handle and split the arguments provided with the commandline
        print sys.argv[1]
        messageArr = sys.argv[1]
        messageContent = messageArr.split("#")
        print messageContent

        # Store Message Pieces separately
        messagePayload = messageContent[0]
        deviceID = messageContent[1]

        # send the message only if it matches your deviceid
        sendMessage(deviceID, messagePayload)

def sendMessage(deviceID, messagePayload):

        // TODO: SET ACCORDING TO YOUR CONFIGURATION
        url = "https://xxxxx.hana.ondemand.com/com.sap.iotservices.mms/v1/api/http/data/messag
id"

        // TODO: SET ACCORDING TO YOUR CONFIGURATION
        payload =
"{\"mode\":\"sync\",\"messageType\":\"MESSAGETYPE\",\"messages\":[{\"deviceid\":\"" +
str(deviceID) + "\",\"value\":\"" + str(messagePayload) + "\"}]}"

        // TODO: SET ACCORDING TO YOUR CONFIGURATION
                headers = {
          'authorization': "Bearer OAUTHTOKEN",
          'content-type': "application/json",
          'cache-control': "no-cache"
          }

        response = requests.request("POST", url, data=payload, headers=headers)

        print(payload)
        print(response.text)

if __name__ == "__main__":
        try:
                main()
        except (KeyboardInterrupt, SystemExit):
                raise
        except:
                print("Error detected. Check the layout of the message.")
```

3. Modify it according to your HCP IoT message and device type.
4. Modify it to discard any message not coming from your Arduino.

Send message to HCP

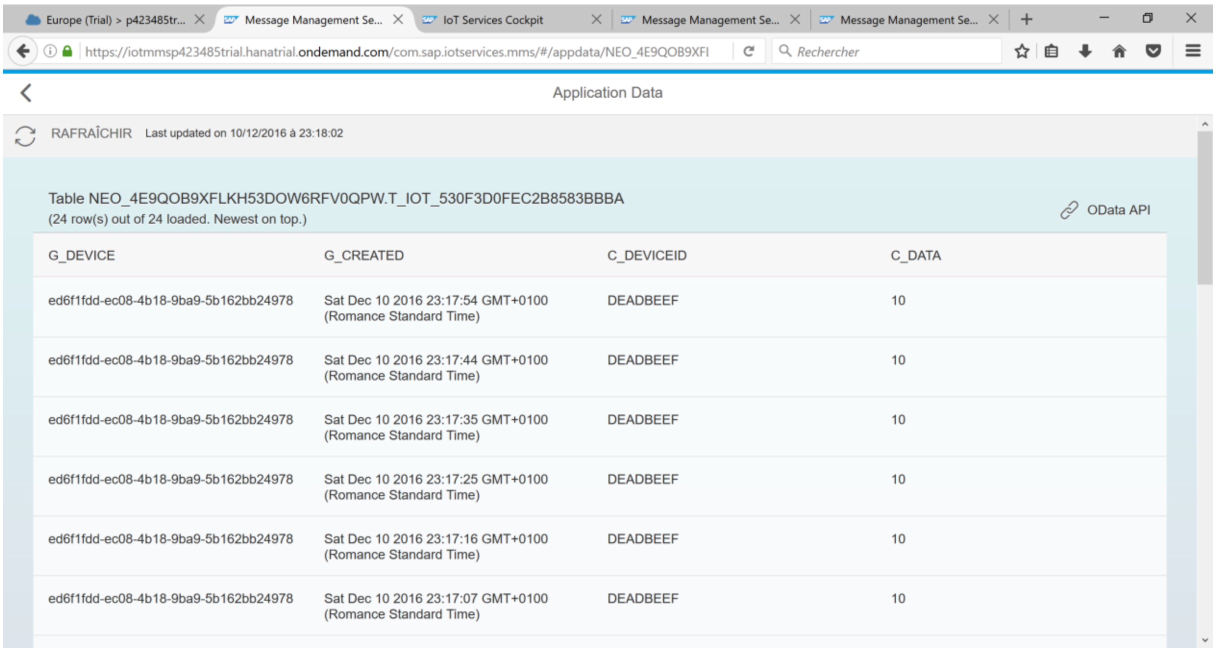
1. Start the gateway and plug your Arduino device


COM8 (Arduino/Genuino Uno)
Send
Autoscroll
No line ending
9600 baud
Ubuntu 16.04.1 [En fonction] - Oracle VM VirtualBox
Fichier Machine Écran Entrée Périphériques Aide
pi@raspberrypi: ~/lora/cooking/examples/LoRa
23:16
Right Ctrl

```
DEADBEEF#10
Packet sent, state 0
DEADBEEF#10
Packet sent, state 0
DEADBEEF#10
Packet sent, state 0
DEADBEEF#10
Packet sent, state 0
DEADBEEF#10
Packet sent, state 0
DEADBEEF#10
Packet sent, state 0
DEADBEEF#10
Packet sent, state 0
DEADBEEF#10
Packet sent, state 0
```

```
pi@raspberrypi:~/lora/cooking/examples/LoRa $ sudo ./SX_01b_RX_LoRa.cpp_exe
SX1272 module and Raspberry Pi: receive packets without ACK
Setting power ON: state 0
Setting Mode: state 0
Setting Header ON: state 0
Setting Channel: state 0
Setting CRC ON: state 0
Setting Power: state 0
Setting Node address: state 0
SX1272 successfully configured
C python /home/pi/lora/cooking/examples/LoRa/send2HCP.py DEADBEEF#10
Message sent: DEADBEEF#10
DEADBEEF#10
['DEADBEEF', '10']
{"mode":"sync","messageType":"530f3d0fec2b8583bbba","messages":[{"deviceid":"DEADBEEF","da
ta":"10"}]}
{"msg":"1 message(s) received from device [ed6f1fdd-ec08-4b18-9ba9-5b162bb24978]"}
C python /home/pi/lora/cooking/examples/LoRa/send2HCP.py DEADBEEF#10
Message sent: DEADBEEF#10
DEADBEEF#10
['DEADBEEF', '10']
{"mode":"sync","messageType":"530f3d0fec2b8583bbba","messages":[{"deviceid":"DEADBEEF","da
ta":"10"}]}
{"msg":"1 message(s) received from device [ed6f1fdd-ec08-4b18-9ba9-5b162bb24978]"}
C python /home/pi/lora/cooking/examples/LoRa/send2HCP.py DEADBEEF#10
Message sent: DEADBEEF#10
```

2. Check the sent messages in HCP



Last updated on 10/12/2016 à 23:18:02

Table NEO_4E9QOB9XFLKH53DOW6RFV0QPW.T_IOT_530F3D0FEC2B8583BBBA
(24 row(s) out of 24 loaded. Newest on top.)

G_DEVICE	G_CREATED	C_DEVICEID	C_DATA
ed6f1fdd-ec08-4b18-9ba9-5b162bb24978	Sat Dec 10 2016 23:17:54 GMT+0100 (Romance Standard Time)	DEADBEEF	10
ed6f1fdd-ec08-4b18-9ba9-5b162bb24978	Sat Dec 10 2016 23:17:44 GMT+0100 (Romance Standard Time)	DEADBEEF	10
ed6f1fdd-ec08-4b18-9ba9-5b162bb24978	Sat Dec 10 2016 23:17:35 GMT+0100 (Romance Standard Time)	DEADBEEF	10
ed6f1fdd-ec08-4b18-9ba9-5b162bb24978	Sat Dec 10 2016 23:17:25 GMT+0100 (Romance Standard Time)	DEADBEEF	10
ed6f1fdd-ec08-4b18-9ba9-5b162bb24978	Sat Dec 10 2016 23:17:16 GMT+0100 (Romance Standard Time)	DEADBEEF	10
ed6f1fdd-ec08-4b18-9ba9-5b162bb24978	Sat Dec 10 2016 23:17:07 GMT+0100 (Romance Standard Time)	DEADBEEF	10