**Device - LORAWAN – App**

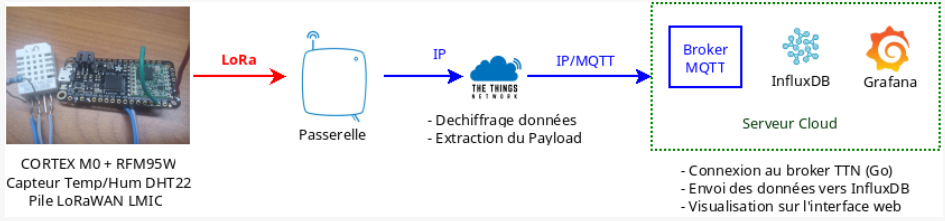
**Etapes de développement :**

* Assemblage et programmation du device
  + Savoir si il vibre (rouge / vert)
  + Temps d’action
  + Géolocalisation du device
* Si création d’une passerelle :
  + Téléchargement de Raspbian sur le Raspberry PI 3
  + Activer SPI sur le Rapsberry Pi 3 pour activer le module passerelle RHF0M301
  + Installation et démarrage de la passerelle
* The Things Network :
  + Connexion du nœud (capteur) à la passerelle
  + Enregistrer la passerelle sur The Things Network
* Récupération des données :

## Utilisation de  [l’API The Things Network](http://thethingsnetwork.org/api/v0/) pour la récupération des données. Le payload arrive sous forme de json.

## Enregistrement des données vers le cloud

## Création d’une application web pour le contrôle et l’affichage des données



**1. Device : Capteur de vibration**

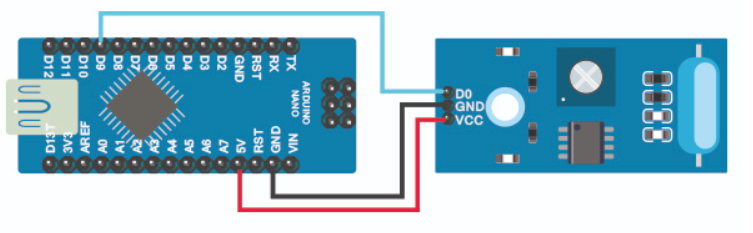
- Les nœuds : il s’agit des objets connectés disposant d’un émetteur récepteur LoRaWAN. Ils disposent de capteurs afin d’effectuer de la remontée d’informations et peuvent recevoir des messages pour déclencher des opérations. Ils sont déployés dans l’environnement après avoir été configurés et doivent être autonomes. Ils disposent d’un identifiant unique DevEUI de 64 bits. Les informations remontées sont associées à un numéro d’application unique AppEUI de 64 bits. Selon la méthode d’activation choisie, une clé AppKey ou deux clés NwkSKey et AppSKey devront être choisies. Elles ont une taille de 128 bits.

* 1. **Matériel :**

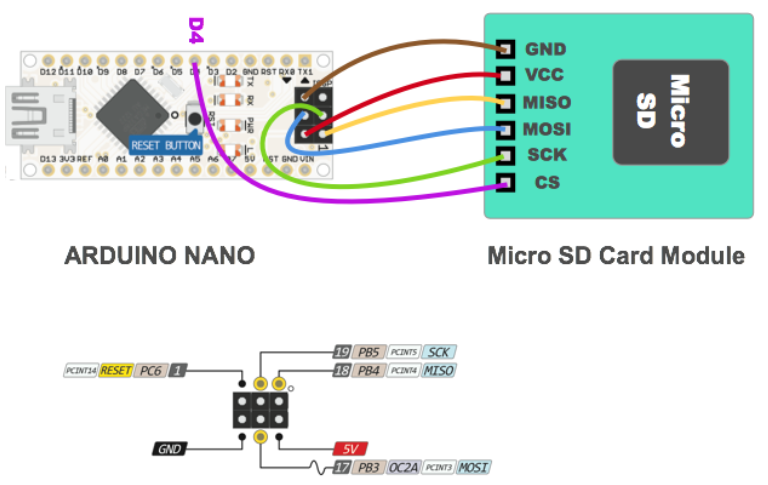
**Europe LORAWAN Frequencies : 863 MHz – 870 MHz**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nom** | **Prix** | **Site intenet** | **Image** |
| Arduino Nano | **8 €** | [Arduino nano](https://www.amazon.fr/s?k=arduino+nano&i=computers&adgrpid=73466641868&hvadid=324630940981&hvdev=c&hvlocphy=9055443&hvnetw=g&hvpos=1t1&hvqmt=e&hvrand=12671751019505078350&hvtargid=kwd-15608517219&tag=googhydr0a8-21&ref=pd_sl_78dnr6rr2g_e) |  |
| Lora Module reyax rylr896 | **13 €** | <https://www.ebay.fr/str/reyax/> |  |
| SW-420 vibration sensor | **1 à 2 €** | [**Vibration sensor**](https://www.amazon.fr/s?k=SW-420+arduino&i=computers&__mk_fr_FR=%C3%85M%C3%85%C5%BD%C3%95%C3%91&ref=nb_sb_noss) |  |
| **Micro TF Card Memory Shield Module SPI Micro Storage Card Adapter** | **1 à 2 €** | [Micro Storage Card Adapter](https://fr.banggood.com/Micro-SD-TF-Card-Memory-Shield-Module-SPI-Mciro-SD-Adapter-For-Arduino-p-919914.html?gmcCountry=FR&currency=EUR&createTmp=1&utm_source=googleshopping&utm_medium=cpc_bgcs&utm_content=zouzou&utm_campaign=pla-frg-ele-mob-shield-pc&gclid=EAIaIQobChMI8eS106Gs4QIV1ZrVCh35gQcCEAQYAiABEgKnh_D_BwE&cur_warehouse=CN) |  |
| Carte SD 256MO | **2 €** | [**Micro SD**](https://www.amazon.fr/s?k=carte+sd&i=electronics&adgrpid=57767312818&hvadid=275389391035&hvdev=c&hvlocphy=20321&hvnetw=g&hvpos=1t1&hvqmt=e&hvrand=3170431456315032745&hvtargid=kwd-543997013&tag=googhydr0a8-21&ref=pd_sl_3juw0znaxg_e) |  |
| Support battery connector 4 x AA | **2 €** | [**Support battery**](https://fr.farnell.com/keystone/2478/battery-holder-leaded-4-x-aa/dp/1650685?gclid=EAIaIQobChMIn_etw8ud4QIVCkTTCh28nAAKEAQYAiABEgJZG_D_BwE&gross_price=true&mckv=1g2ZGxeV_dc|pcrid|80993908142|&CAWELAID=120185620000084557&CAGPSPN=pla&CAAGID=13038038102&CMP=KNC-GFR-GEN-SHOPPING-1650685&CATCI=pla-45226921478) |  |
| Jumper Wire | **2 € les 40 pieces** | [**Jumper wire**](https://fr.banggood.com/40pcs-20cm-Male-to-Female-Color-Breadboard-Cable-Jump-Wire-Jumper-p-992837.html?gmcCountry=FR&currency=EUR&createTmp=1&utm_source=googleshopping&utm_medium=cpc_bgcs&utm_content=zouzou&utm_campaign=pla-frg-all-pc&gclid=EAIaIQobChMItuCb9aOs4QIVCxHTCh04TQP8EAkYAiABEgJmpfD_BwE&cur_warehouse=CN) |  |
|  |  |  |  |

* 1. **Montage du Device**
     1. **Arduino – Vibration sensor**

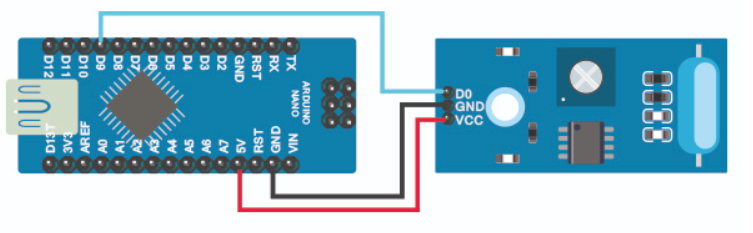


* Pin 9 de l’Arduino -> D0 du module.
* GND de l’Arduino -> GND du module.
* VCC de l’Arduino -> VCC du module
  + 1. **Arduino – Slot Card Micro SD**



Youtube : Jim & Joe's or Arduino Pro Mini - Turning on lights with the SW-420 Vibration Sensor

* + 1. **Arduino - Lora Module reyax rylr896**



GND

TX

RX

VDD

Gauche 123456 Droite

* 1 : VDD Power Pin 3V
* 2 :
* 3 : RX
* 4 : TX
* 5 :
* 6 : GND (Ground)

Arduino RX ->Lora module TX

Arduino TX ->Lora module RX

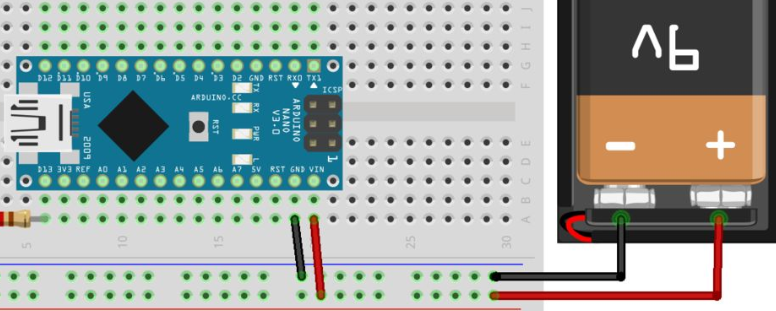
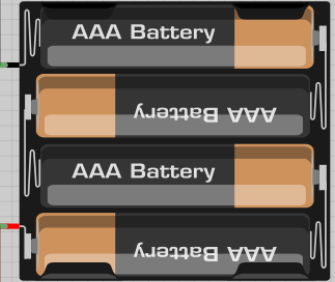
<https://www.youtube.com/watch?v=T4vPc4EN_O8&t=277s>

### La ligne série

Les 2 broches étiquetées TX et RX sont respectivement la ligne d’émission série (T pour Transmit, émettre) et la ligne de réception série (R pour receive, recevoir). Cette ligne est principalement employée pour dialoguer avec l’ordinateur hôte, votre Mac ou votre PC. Des fonctions permettent d’afficher des messages dans une fenêtre de l’ordinateur hôte et de lire le clavier de l’ordinateur hôte.

Elle est également employer pour dialoguer avec certains modules comme les modules radio XBee ou certains modules WiFi ou encore bluetooth.

* + 1. **Arduino Nano - Battery**



Pole négatif (noir) -> GND

Pole Positif (rouge) -> VIN

* 3.3V. Un arduino Uno est équipé d’un régulateur 3,3V dont la sortie est connectée à cette broche. On peut s’en servir pour alimenter un circuit externe en 3,3V.
* 5V. L’Arduino, si il fonctionne sous une tension de 5V, est équipé d’un régulateur 5V. La sortie de ce régulateur est connecté sur cette broche qui peut donc être employée pour alimenter des circuits externes en 5V. Si on désire construire sa propre alimentation 5V, on peut également fournir cette tension à l’Arduino via cette broche.
* GND. La masse ou 0V
* Vin. L’entrée du régulateur embarqué sur l’Arduino. On peut choisir d’alimenter l’Arduino via cette broche au lieu d’utiliser la prise d’alimentation.

Source : <https://www.locoduino.org/spip.php?article57>

* 1. **Simulation**

Simulation avant achat des pièces.

Source : <https://www.autodesk.com/products/eagle/free-download>

**Attente des pièces….**

**1.3. Software : Arduino**

Téléchargement :

Arduino IDE :

<https://www.arduino.cc/en/main/software>

Inclure bibliothèque : [MCCI\_LoRaWAN\_LMIC\_library-2.3.2.zip](http://downloads.arduino.cc/libraries/github.com/mcci-catena/MCCI_LoRaWAN_LMIC_library-2.3.2.zip)

Library : <https://www.arduinolibraries.info/libraries/mcci-lo-ra-wan-lmic-library>

Ajout direct dans les fichiers : C:\Users\Jchapol\Documents\Arduino\libraries\MCCI\_LoRaWAN\_LMIC\_library-2.3.2

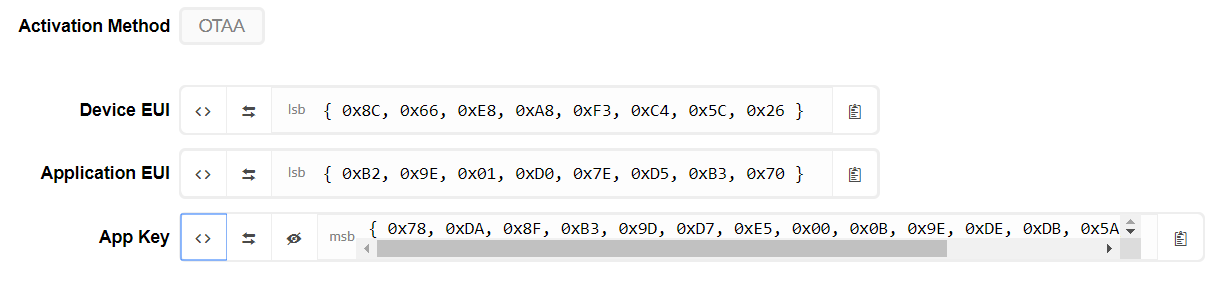
Source : <https://www.arduino.cc/en/Guide/Libraries>

Ouverture de la classe de ttn-otaa (Over-the-air activation)

Application Aduino/Fichier/Exemples/ MCCI\_LoRaWAN\_LMIC\_library/ ttn-otaa

Remplacement des variables FILLMEIN :

* APPEUI (lsb)
* DEVEUI (lsb)
* APPKEY (msb)



* + 1. **Arduino code: Vibration Sensor mode Emission**

|  |
| --- |
| int EP =9;  void setup(){  pinMode(ledPin, OUTPUT);  pinMode(EP, INPUT); //set EP input for measurement  Serial.begin(9600); //init serial 9600  Serial.println("----------------------Vibration demo------------------------");  }  void loop(){  long measurement =TP\_init();  delay(50);  Serial.print("measurement = ");  Serial.println(measurement);  if (measurement > 1000){  digitalWrite(ledPin, HIGH);  }  else{  digitalWrite(ledPin, LOW);  }  }  long TP\_init(){  delay(10);  long measurement=pulseIn (EP, HIGH); //wait for the pin to get HIGH and  //returns measurement  return measurement;  } |

### Les entrées-sorties numériques

Quasiment toutes les broches d’un Arduino peuvent être programmées en entrée ou sortie numérique (et non les deux en même temps). Par exemple, sur l’Arduino Uno, il s’agit d’une part des broches numérotées de 0 à 13 mais également des broches A0 à A5. Une broche programmée ainsi peut être :

* Une entrée. Le programme peut lire une tension présente sur cette broche en utilisant digitalRead(...). Comme cette tension est interprétée comme un chiffre binaire (0 ou 1), la datasheet du MCU de l’Arduino Uno garantit que toute tension inférieure à 0,3 x Vcc, Vcc étant égale à 5V, soit 1,5V sera comprise comme un 0 et que toute tension supérieure à 0,6 x Vcc, soit 3V, sera comprise comme un 1. Entre les deux, c’est flou. digitalRead(...) renverra de toutes façons un 0 ou un 1 mais de manière plus ou moins aléatoire et variable selon l’Arduino utilisé.
* Une sortie. Le programme peut écrire un chiffre binaire, au moyen de digitalWrite(...), chiffre qui dans le programme sont nommées HIGH pour le 1 et LOW pour le 0, qui sera traduit en une tension de 5V pour le 1 et de 0V pour le 0. Attention toutefois, cette tension peut respectivement être plus basse ou plus haute si le courant qui est tiré de la broche commence à être important.
  + 1. **Arduino code : Micro SD**

Source : <http://lab.dejaworks.com/arduino-nano-sd-card-connection/>

|  |
| --- |
| /\*  SD card test  This example shows how use the utility libraries on which the'  SD library is based in order to get info about your SD card.  Very useful for testing a card when you're not sure whether it’s working or not.  The circuit:  \* SD card attached to SPI bus as follows:  \*\* MOSI - pin 11 on Arduino Uno/Duemilanove/Diecimila  \*\* MISO - pin 12 on Arduino Uno/Duemilanove/Diecimila  \*\* CLK - pin 13 on Arduino Uno/Duemilanove/Diecimila  \*\* CS - depends on your SD card shield or module.  Pin 4 used here for consistency with other Arduino examples  \*/  // include the SD library:  #include <SPI.h>  #include <SD.h>  // set up variables using the SD utility library functions:  Sd2Card card;  SdVolume volume;  SdFile root;  // change this to match your SD shield or module;  // Arduino Ethernet shield: pin 4  // Adafruit SD shields and modules: pin 10  // Sparkfun SD shield: pin 8  const int chipSelect = 4;  void setup() {  // Open serial communications and wait for port to open:  Serial.begin(9600);  while (!Serial) {  ; // wait for serial port to connect. Needed for native USB port only  }  Serial.print("\nInitializing SD card...");  // we'll use the initialization code from the utility libraries  // since we're just testing if the card is working!  if (!card.init(SPI\_HALF\_SPEED, chipSelect)) {  Serial.println("initialization failed. Things to check:");  Serial.println("\* is a card inserted?");  Serial.println("\* is your wiring correct?");  Serial.println("\* did you change the chipSelect pin to match your shield or module?");  return;  } else {  Serial.println("Wiring is correct and a card is present.");  }  // print the type of card  Serial.print("\nCard type: ");  switch (card.type()) {  case SD\_CARD\_TYPE\_SD1:  Serial.println("SD1");  break;  case SD\_CARD\_TYPE\_SD2:  Serial.println("SD2");  break;  case SD\_CARD\_TYPE\_SDHC:  Serial.println("SDHC");  break;  default:  Serial.println("Unknown");  }  // Now we will try to open the 'volume'/'partition' - it should be FAT16 or FAT32  if (!volume.init(card)) {  Serial.println("Could not find FAT16/FAT32 partition.\nMake sure you've formatted the card");  return;  }  // print the type and size of the first FAT-type volume  uint32\_t volumesize;  Serial.print("\nVolume type is FAT");  Serial.println(volume.fatType(), DEC);  Serial.println();  volumesize = volume.blocksPerCluster(); // clusters are collections of blocks  volumesize \*= volume.clusterCount(); // we'll have a lot of clusters  volumesize \*= 512; // SD card blocks are always 512 bytes  Serial.print("Volume size (bytes): ");  Serial.println(volumesize);  Serial.print("Volume size (Kbytes): ");  volumesize /= 1024;  Serial.println(volumesize);  Serial.print("Volume size (Mbytes): ");  volumesize /= 1024;  Serial.println(volumesize);  Serial.println("\nFiles found on the card (name, date and size in bytes): ");  root.openRoot(volume);  // list all files in the card with date and size  root.ls(LS\_R | LS\_DATE | LS\_SIZE);  }  void loop(void) {  } |

* + 1. **Arduino Code : Lora Module reyax rylr896**

|  |
| --- |
| #include <lmic.h>  #include <hal/hal.h>  #include <SPI.h>  //  // For normal use, we require that you edit the sketch to replace FILLMEIN  // with values assigned by the TTN console. However, for regression tests,  // we want to be able to compile these scripts. The regression tests define  // COMPILE\_REGRESSION\_TEST, and in that case we define FILLMEIN to a non-  // working but innocuous value.  //  #ifdef COMPILE\_REGRESSION\_TEST  # define FILLMEIN 0  #else  # warning "You must replace the values marked FILLMEIN with real values from the TTN control panel!"  # define FILLMEIN (#dont edit this, edit the lines that use FILLMEIN)  #endif  // This EUI must be in little-endian format, so least-significant-byte  // first. When copying an EUI from ttnctl output, this means to reverse  // the bytes. For TTN issued EUIs the last bytes should be 0xD5, 0xB3,  // 0x70.  static const u1\_t PROGMEM APPEUI[8]={ 0xB2, 0x9E, 0x01, 0xD0, 0x7E, 0xD5, 0xB3, 0x70 };  void os\_getArtEui (u1\_t\* buf) { memcpy\_P(buf, APPEUI, 8);}  // This should also be in little endian format, see above.  static const u1\_t PROGMEM DEVEUI[8]={ 0x8C, 0x66, 0xE8, 0xA8, 0xF3, 0xC4, 0x5C, 0x26 };  void os\_getDevEui (u1\_t\* buf) { memcpy\_P(buf, DEVEUI, 8);}  // This key should be in big endian format (or, since it is not really a  // number but a block of memory, endianness does not really apply). In  // practice, a key taken from ttnctl can be copied as-is.  static const u1\_t PROGMEM APPKEY[16] = { 0x78, 0xDA, 0x8F, 0xB3, 0x9D, 0xD7, 0xE5, 0x00, 0x0B, 0x9E, 0xDE, 0xDB, 0x5A, 0x09, 0x2D, 0x85 };  void os\_getDevKey (u1\_t\* buf) { memcpy\_P(buf, APPKEY, 16);}  static uint8\_t mydata[] = "Hello, world!";  static osjob\_t sendjob;  // Schedule TX every this many seconds (might become longer due to duty  // cycle limitations).  const unsigned TX\_INTERVAL = 60;  // Pin mapping  const lmic\_pinmap lmic\_pins = {  .nss = 10,  .rxtx = LMIC\_UNUSED\_PIN,  .rst = 5,  .dio = {2, 3, LMIC\_UNUSED\_PIN},  };  void onEvent (ev\_t ev) {  Serial.print(os\_getTime());  Serial.print(": ");  switch(ev) {  case EV\_SCAN\_TIMEOUT:  Serial.println(F("EV\_SCAN\_TIMEOUT"));  break;  case EV\_BEACON\_FOUND:  Serial.println(F("EV\_BEACON\_FOUND"));  break;  case EV\_BEACON\_MISSED:  Serial.println(F("EV\_BEACON\_MISSED"));  break;  case EV\_BEACON\_TRACKED:  Serial.println(F("EV\_BEACON\_TRACKED"));  break;  case EV\_JOINING:  Serial.println(F("EV\_JOINING"));  break;  case EV\_JOINED:  Serial.println(F("EV\_JOINED"));  {  u4\_t netid = 0;  devaddr\_t devaddr = 0;  u1\_t nwkKey[16];  u1\_t artKey[16];  LMIC\_getSessionKeys(&netid, &devaddr, nwkKey, artKey);  Serial.print("netid: ");  Serial.println(netid, DEC);  Serial.print("devaddr: ");  Serial.println(devaddr, HEX);  Serial.print("artKey: ");  for (int i=0; i<sizeof(artKey); ++i) {  Serial.print(artKey[i], HEX);  }  Serial.println("");  Serial.print("nwkKey: ");  for (int i=0; i<sizeof(nwkKey); ++i) {  Serial.print(nwkKey[i], HEX);  }  Serial.println("");  }  // Disable link check validation (automatically enabled  // during join, but because slow data rates change max TX  // size, we don't use it in this example.  LMIC\_setLinkCheckMode(0);  break;  /\*  || This event is defined but not used in the code. No  || point in wasting codespace on it.  ||  || case EV\_RFU1:  || Serial.println(F("EV\_RFU1"));  || break;  \*/  case EV\_JOIN\_FAILED:  Serial.println(F("EV\_JOIN\_FAILED"));  break;  case EV\_REJOIN\_FAILED:  Serial.println(F("EV\_REJOIN\_FAILED"));  break;  case EV\_TXCOMPLETE:  Serial.println(F("EV\_TXCOMPLETE (includes waiting for RX windows)"));  if (LMIC.txrxFlags & TXRX\_ACK)  Serial.println(F("Received ack"));  if (LMIC.dataLen) {  Serial.print(F("Received "));  Serial.print(LMIC.dataLen);  Serial.println(F(" bytes of payload"));  }  // Schedule next transmission  os\_setTimedCallback(&sendjob, os\_getTime()+sec2osticks(TX\_INTERVAL), do\_send);  break;  case EV\_LOST\_TSYNC:  Serial.println(F("EV\_LOST\_TSYNC"));  break;  case EV\_RESET:  Serial.println(F("EV\_RESET"));  break;  case EV\_RXCOMPLETE:  // data received in ping slot  Serial.println(F("EV\_RXCOMPLETE"));  break;  case EV\_LINK\_DEAD:  Serial.println(F("EV\_LINK\_DEAD"));  break;  case EV\_LINK\_ALIVE:  Serial.println(F("EV\_LINK\_ALIVE"));  break;  /\*  || This event is defined but not used in the code. No  || point in wasting codespace on it.  ||  || case EV\_SCAN\_FOUND:  || Serial.println(F("EV\_SCAN\_FOUND"));  || break;  \*/  case EV\_TXSTART:  Serial.println(F("EV\_TXSTART"));  break;  default:  Serial.print(F("Unknown event: "));  Serial.println((unsigned) ev);  break;  }  }  void do\_send(osjob\_t\* j){  // Check if there is not a current TX/RX job running  if (LMIC.opmode & OP\_TXRXPEND) {  Serial.println(F("OP\_TXRXPEND, not sending"));  } else {  // Prepare upstream data transmission at the next possible time.  LMIC\_setTxData2(1, mydata, sizeof(mydata)-1, 0);  Serial.println(F("Packet queued"));  }  // Next TX is scheduled after TX\_COMPLETE event.  }  void setup() {  Serial.begin(9600);  Serial.println(F("Starting"));  #ifdef VCC\_ENABLE  // For Pinoccio Scout boards  pinMode(VCC\_ENABLE, OUTPUT);  digitalWrite(VCC\_ENABLE, HIGH);  delay(1000);  #endif  // LMIC init  os\_init();  // Reset the MAC state. Session and pending data transfers will be discarded.  LMIC\_reset();  // Start job (sending automatically starts OTAA too)  do\_send(&sendjob);  }  void loop() {  os\_runloop\_once();  } |

* 1. **Suggestion en plus :**
* Arduino : DataLoger module DS1307

<https://www.carnetdumaker.net/articles/fabriquer-un-systeme-dacquisition-de-mesures-datalogger-avec-une-carte-arduino-genuino/>

Module de timer :afin d'avoir l'heure exacte de chaque prise de mesures.

Pas sûr que cela soit utile. Dès que la donnée est transmise et enregistré, les calculs se feront côté Java

* Ecrire sur une carte SD

<https://www.carnetdumaker.net/articles/lire-et-ecrire-des-donnees-sur-une-carte-sd-avec-une-carte-arduino-genuino/>

**2. The Thing Network**

Source : <https://www.youtube.com/watch?v=d8q-hZLIf6I&index=22&list=PLmL13yqb6OxdeOi97EvI8QeO8o-PqeQ0g>

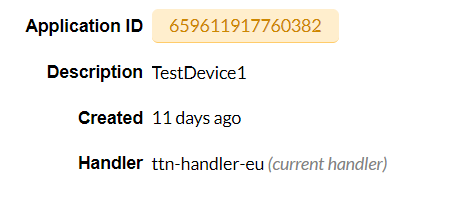
Site : <https://www.thethingsnetwork.org/>

Créer un compte : JonathanC / jonathan1985

**2.1. Création Application**

Aller sur CONSOLE / Applications /add application

* Application ID : Nom de l’application
* Description
* Handler registration : ttn-handler-eu



**2.2. Création Device**

L’application est créée avec :

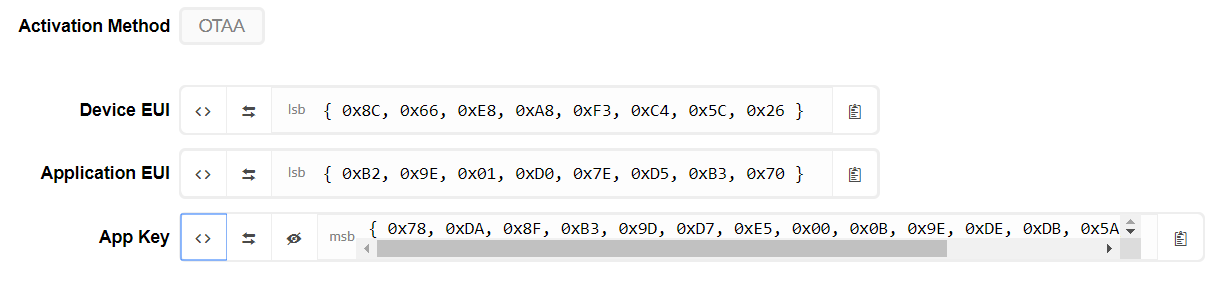
* Application EUIS
* Access keys

Enregistrement d’un device :

* Device ID : nom du device
* Device EUI : génération : <https://www.mobilefish.com/services/guid/guid.php>
* Rien dans App Key
* App EUI : déjà rempli
* Register

Le device est créé avec :

* Device EUI (lsb) = { 0x8C, 0x66, 0xE8, 0xA8, 0xF3, 0xC4, 0x5C, 0x26 };
* Application EUI (lsb) = { 0xB2, 0x9E, 0x01, 0xD0, 0x7E, 0xD5, 0xB3, 0x70 };
* App Key (msb) = { 0x78, 0xDA, 0x8F, 0xB3, 0x9D, 0xD7, 0xE5, 0x00, 0x0B, 0x9E, 0xDE, 0xDB, 0x5A, 0x09, 0x2D, 0x85 };



**2.3. HTTP Intégration**

Source : <https://www.youtube.com/watch?v=Uebcq7xmI1M&index=2&list=PLM8eOeiKY7JVwrBYRHxsf9p0VM_dVapXl>

Url of the end point : <https://requestbin.fullcontact.com/>

Maven dependence :

<dependency>

<groupId>commons-httpclient</groupId>

<artifactId>commons-httpclient</artifactId>

<version>3.1</version>

</dependency>

**3. Application Web**

Source : <https://www.thethingsnetwork.org/docs/applications/java/quick-start.html>

Installation :

* IDE : Eclipse
* Serveur Apache Tomcat
* JDK 1.8
* Apache maven

**3.1. Maven**

Installation du Framework Maven sur Eclipse

Source : <https://www.vogella.com/tutorials/EclipseMaven/article.html#create-maven-project>

* Sur Eclipse, File/New/Dynamic Web Project.
* Choix du nom du projet
* Clic droit/Configure/Convert to Maven project
* Open pom.xml
* Commenter le plugin : maven-war-plugin
* Ajouter les dépendances : ttn et gson

<dependency>

<groupId>org.thethingsnetwork</groupId>

<artifactId>data-mqtt</artifactId>

<version>2.1.0</version>

</dependency>

<dependency>

<groupId>com.google.code.gson</groupId>

<artifactId>gson</artifactId>

<version>2.8.5</version>

</dependency>

* Clic droit sur projet /Maven/Update project
* Right-click the pom.xml file / Run As / Maven build…
  + Goals : clean install
* Rafraichir le projet
* Ajout Servlet : File/New/Servlet
* Choix du Java package et du nom de la Classe
* Si les imports ne sont pas bons : Clic droit sur le projet/Properties/Targeted Tuntimes : sélectionner le serveur « Tomcat ».
* Les vues sont créées dans Web content : File/JSP File

URL : http://localhost:8080/nomProjet/nomFile.jsp

**4. La Passerelle (gateway)**

- Les passerelles : elles sont déployées par les opérateurs ou par des associations souhaitant déployer un réseau IoT LoRaWAN. Placées de préférence en hauteur afin de maximiser la réception, elles sont chargées de recevoir les paquets LoRa et de transmettre ceux qui sont des paquets LoRaWAN valides vers les différents serveurs applicatifs. Pour y parvenir, elles sont connectées à ceux-ci au moyen de liens réseaux classiques : réseaux câblés ou mobiles. Elles sont identifiées par un identifiant de 8 octets.

Kit de montage :

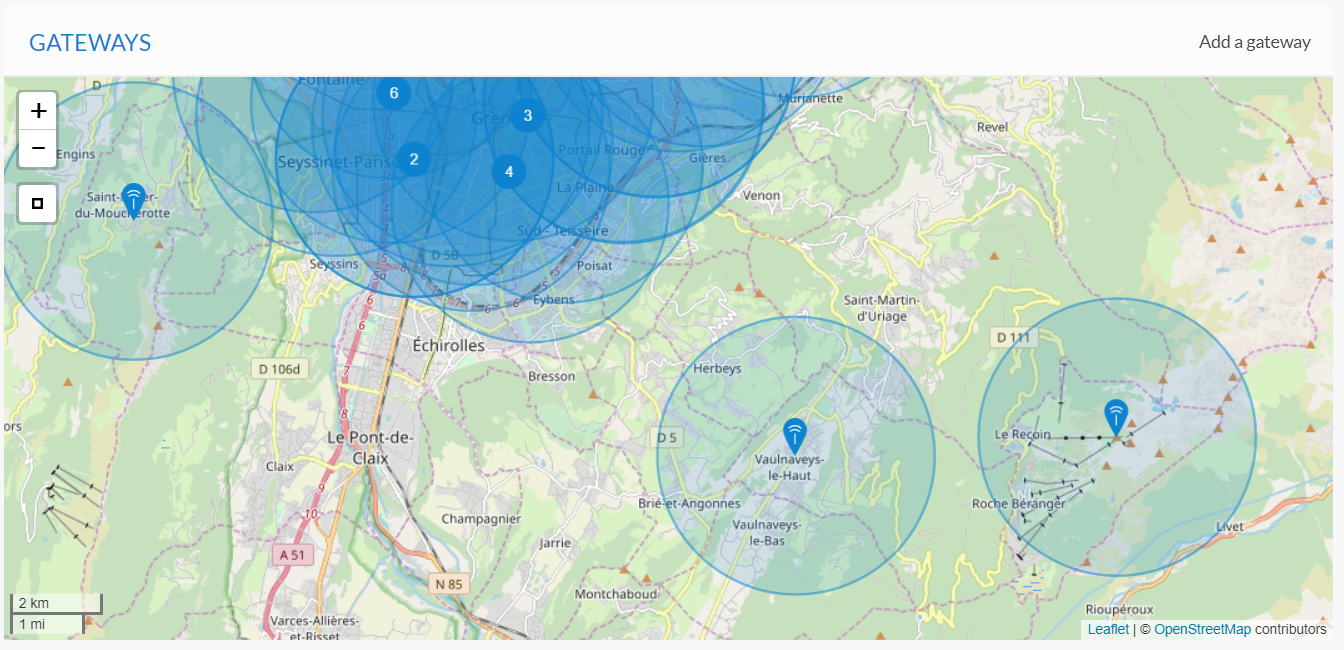
 - 1 x [**Raspberry Pi 3**](https://www.gotronic.fr/art-carte-raspberry-pi3-b-1-gb-24532.htm) - 1 x module [Seeeduino LoRaWAN](http://wiki.seeed.cc/Seeeduino_LoRAWAN/" \t "_blank) avec GPS  
 - 1 x module passerelle RHF0M301  
 - 1 x pont PRI 2 RHF4T002  
 - 1 x antenne 868 MHz  
 - 1 x alimentation microUSB 5 Vcc/2,5 A  
 - 1 x cordon USB-microUSB 20 cm  
 - 1 x cordon convertisseur USB-série  
 - 1 x cordon JST 10 cm  
 - 1 x cordon RJ45 20 cm  
 - 1 x carte microSD 16 Go



Source : <https://wikifab.org/wiki/D%C3%A9ployer_une_passerelle_LoRaWAN_pour_The_Things_Network>

S’il existe déjà une passerelle active sur site, la création d’une passerelle n’est pas obligatoire. Cependant, si cette passerelle appartient à un privé, nous sommes dépendants de cette tiers personne.

La commune où j’habite possède une passerelle LoraWan. Avec le nœud, je devrais pouvoir configurer la plateforme ‘‘The Things Network’’ et tester la récupération les données.

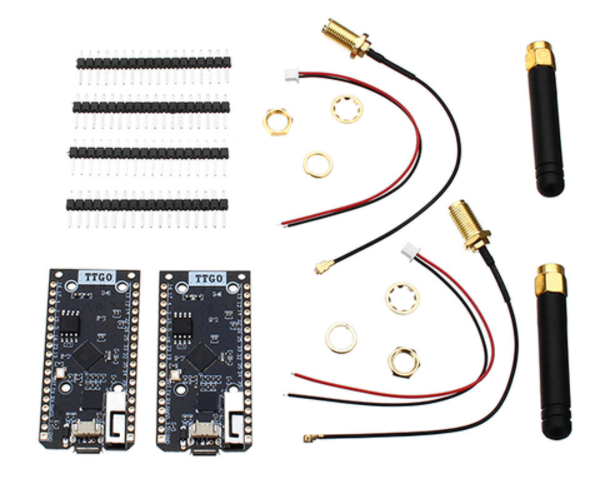


<https://www.instructables.com/id/Make-Lora-Node-and-Lora-Gateway-by-Using-DORJI-DRF/>

<https://letmeknow.fr/blog/2015/10/27/tutomodulelora/>

**Wemos® TTGO 2 Pcs / lot ESP32 SX1276 LoRa 868/915 MHz Bluetooth WI-FI Lora Internet Antenne Carte de Développement Pour Arduino 20€**

**De 10 à 100 € (à creuser)**

868MHz 915MHz SX1276 data radio modem DRF1276DM 15€

Sources utiles pour les configurations :

<https://connect.ed-diamond.com/MISC/MISCHS-015/LORAWAN-deploiement-d-une-infrastructure-de-test>

<https://www.rs-online.com/designspark/building-a-raspberry-pi-powered-lorawan-gateway-cn-de-fr>

<http://wiki.seeedstudio.com/LoRa_LoRaWan_Gateway_Kit/>

Lien matériel Node :

<https://www.youtube.com/watch?v=vMMFCxzSP_s&list=PLmL13yqb6OxdeOi97EvI8QeO8o-PqeQ0g&index=19>

<https://www.banggood.com/fr/868MHz-LoRa-Radio-Node-V1_0-Wireless-IOT-Lora-Module-RFM95-SX1276-ATmega328P-3_7-12V-uFL-Antenna-Internet-of-Things-FZ3338-p-1636480.html?gmcCountry=FR&currency=EUR&createTmp=1&utm_source=googleshopping&utm_medium=cpc_bgs&utm_content=frank&utm_campaign=ssc-fr-all-0716&ad_id=367115636049&gclid=EAIaIQobChMI1ev87O3v6AIVPYBQBh0lwQahEAQYByABEgIziPD_BwE&cur_warehouse=CN>

<https://fr.farnell.com/the-things-network/ttn-un-868/the-things-uno-eu/dp/2675815?gclid=EAIaIQobChMIjZ6Ige3v6AIVDe3tCh2GAAqtEAYYASABEgLbJPD_BwE&gross_price=true&mckv=s_dc|pcrid|80994017342|plid||kword||match||slid||product|2675815|pgrid|18029979182|ptaid|pla-41477300408|&CMP=KNC-GFR-GEN-SHOPPING_TEST_93_08-APR-2020_T>