

This guide will explain how to use the KNoT meta-platform for building IoT solutions in an easy way. The first two sections show a brief introduction to IoT platforms and how to build an IoT solution using an IoT platform. Then, there is a quick start guide showing how to use the KNoT meta-platform.

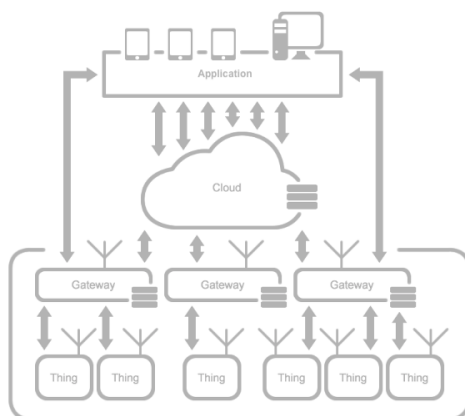
Introduction to IoT Platforms

An IoT platform can be defined as a set of software and/or hardware that abstracts the key operations of an IoT solution. This means that a platform is expected to provide the means to collect device data, send it via internet to a server in the cloud, store, process, present, and manage such data in a way that provides useful information to the users.

KNoT is called a meta-platform because it was built by using other open source platforms, to address interoperability. The main objective is to connect all existing IoT platforms, acting as a glue, and enabling them to talk to each other.

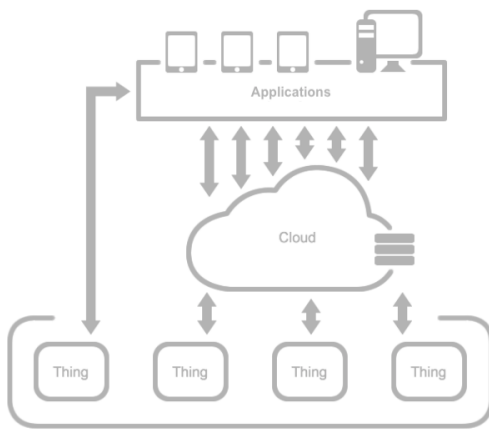
According to the processing power and communication capabilities, an IoT platform can behave as one of the three architectures explained below.

1. Architecture 1.0 - Hub and Spoke



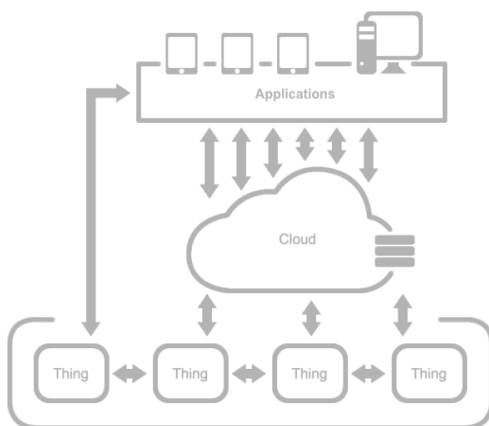
This architecture considers devices with low processing power and no IP. So, those devices have not enough processing capabilities to make decisions or run complex algorithms. For this reason, they must rely on other architectural elements like cloud services, for storage or data analysis. Usually, they are low cost devices with very limited microcontrollers to make the product economically viable. For instance, smart lights. The device generally uses a cheap radio to connect to a gateway, that forwards its data to the cloud.

2. Architecture 2.0 - Front Loaded



This architecture considers devices with low processing power, but with enough connectivity capabilities to have an IP Stack. For this reason, they do not need to rely on other architectural elements like gateways, for protocol translation nor internet connectivity. In this scenario, the devices can directly communicate with the applications or cloud services.

3. Architecture 3.0 - Smart Objects



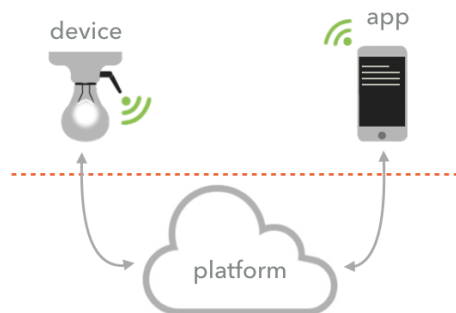
This architecture considers devices with high processing power and also connectivity capabilities to have an IP Stack. For this reason, they can run complex algorithms (e.g. artificial intelligence), get data from other devices directly and make decisions autonomously. They also can directly coordinate other devices. Usually, they are high cost devices in the first place. Thus, embedding a powerful microcontroller is economically viable since its cost can be diluted on the product cost.

Building IoT Solutions

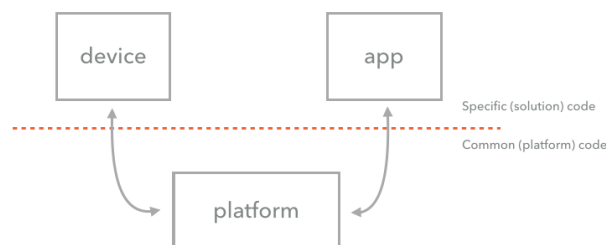
An IoT solution, from the perspective of the user, is composed by a device and a controlling app, according to the image below.



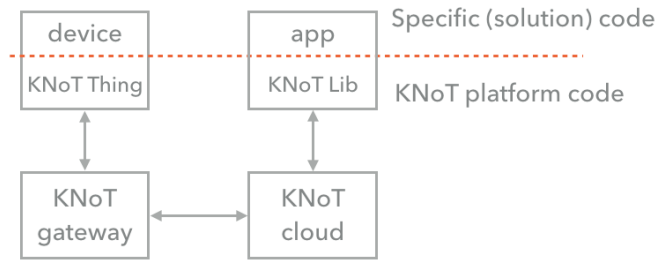
However, from the developer's perspective, an IoT solution needs to connect a device to the Internet, send data to a server, store that data, and make it available so that an application can generate value from it. According to that, it is possible to realize that there are several operations of an IoT solution that are generic, and applicable to any solution in fact. To encapsulate these operations, we use "development platforms". These platforms group the generic operations required for a given problem class.



By using a platform, the developer can focus his efforts on what really matters, which are the specific code of his solution, leaving all the effort with connectivity, protocols, transmission and data storage in charge of the platform.



The KNoT platform, built based on IoT Architecture 1.0 explained above, also implements a multi-radio gateway, that is responsible for abstracting the way that the devices connect to the internet, serving as a proxy between the simple and cheap radios and the internet protocol (IP).



According to the image above, the KNoT platform implements 4 modules for deploying an IoT solution:

- **KNoT Thing:** Represents a device and is composed by hardware (microcontroller, power and RF Radio) and a software stack that handles the sensors, actuators and manages the incoming and outgoing messages exchanged with the gateway.
- **KNoT Gateway:** Acts as a proxy by connecting multiple devices with different wireless stacks to the internet, translate device's messages to a common protocol and send the data to the cloud. It also has a small instance of the cloud that is called fog, for local storage and processing.
- **KNoT Cloud:** It is a server that provides an API for data/message exchange between devices and apps. It also stores the device's data and performs temporal operations on it.
- **KNoT Libs:** Libraries for Android, iOS and Javascript that abstract the cloud and help the application development.

The section below will explain how to use the KNoT platform to develop IoT solutions in a very easy way. It will be shown, step-by-step, how to compile the KNoT Linux Gateway Distro, how to compile the KNoTThing library and how to setup a KNoT Cloud on a server.

Quick Start Guide

In this section, you will find all resources you need to have a KNoT Hello World (a.k.a. lamp demo) working. A KNoT solution is composed by a KNoTThing, a KNoT Gateway, the KNoT Cloud server and an application to interact with the thing. You can experience the KNoT platform by building a Hello World device and a gateway that connects to a cloud server.

Deploying the cloud

What you will need:

- Linux server computer or a virtual machine on a cloud service.
- Git version control tool, that can be found here: <https://git-scm.com/download/linux>

Steps to deploy the cloud:

- Step 1: Install & Start MongoDB on the server. Instructions here: <https://docs.mongodb.com/manual/administration/install-on-linux/>.
- Step 2: Open a terminal and clone the knot-cloud-source (<https://github.com/CESARBR/knot-cloud-source/>) repository in the desired directory:

```
# git clone https://github.com/CESARBR/knot-cloud-source.git
```

- Step 3: Install Node.js and npm. Instructions here: <https://www.npmjs.com/get-npm/>.
- Step 4: Install cloud dependencies:

```
# cd knot-cloud-source  
# sudo npm install --unsafe-perm --production
```

If you find some issues, try using Node 8.12.0.

- Step 5: Modify the config.js file, line 59 as following:

```
59   mongo: {  
60     databaseUrl: 'mongodb://localhost:27017/knot_cloud'  
61   },
```

- Step 6: Run server:

```
# node server.js --http &
```

- Step 7: Verify if the cloud is running. Use *curl* to display the cloud status:

```
# curl 127.0.0.1:3000/status
```

The result should be:

```
{"meshblu": "online" }
```

Enjoy your brand new KNoT Cloud Server!

Building the Gateway

What you will need:

- Raspberry pi 3



- 5V 2A power supply with micro USB connector.
- 8GB or more SD Card and a card reader.
- Ethernet cable with internet connection.
- NRF24L01 radio or the KNoT Gateway Board: https://github.com/CESARBR/knot-hardware-pcbs/tree/master/KNoT_Gateway/KNoT-Gateway%20Starter%20Board.
- Linux or Mac computer.

Steps to build the gateway:

- Step 1: Download the gateway image at:
<http://knot-devel.cesar.org.br/releases/v01.03-rc05/RPi3-KNOT-v01.03-rc05.img.gz> to your computer.
- Step 2: Connect and find the SD card device name for Mac or Linux:

- **For mac:** Find you the SD Card device's name, {DEVICE}, with the command:

```
# diskutil list
```

Usually the device name is something like *rdisk2* or *rdisk3*.

- **For linux:** Find you the SD Card device's name, {DEVICE}, with the command:

```
# lsblk
```

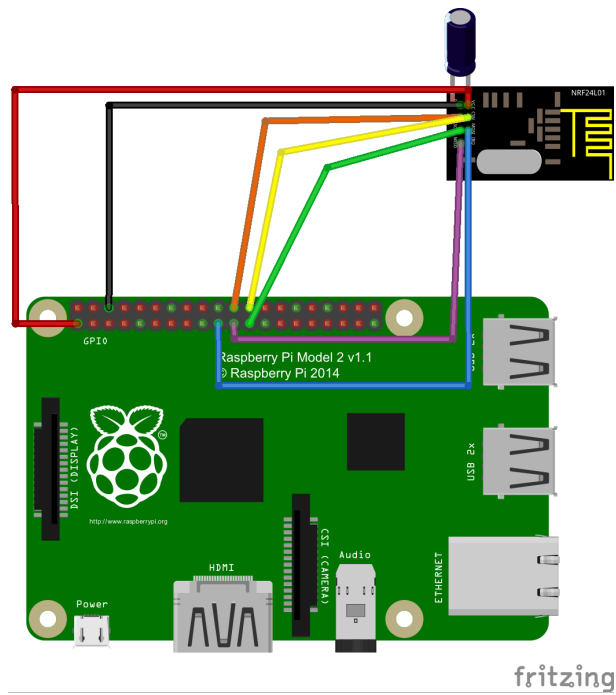
Usually the device name is something like *sdb* or *sdc*.

Flash the found device with:

```
# gunzip RPi3-KNOT-v01.03-rc05.img.gz
# sudo dd bs=1M if=RPi3-KNOT-v01.03-rc05.img of=/dev/{DEVICE}
```

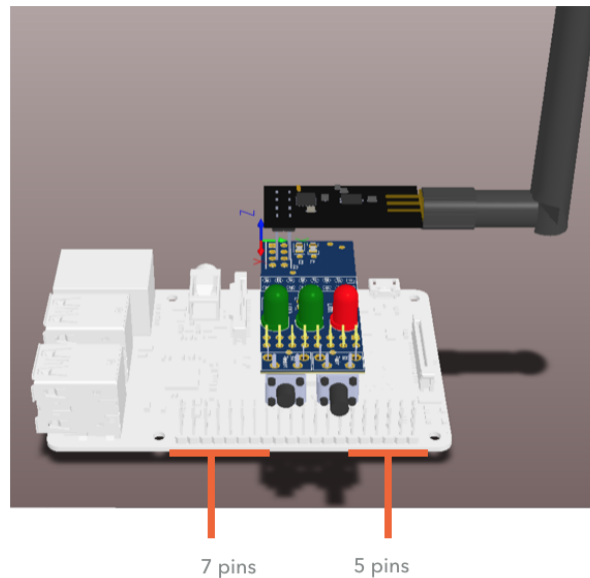
Don't forget to replace {DEVICE} with your found device name.

- Step 3 (a): If you don't have the KNoT Gateway Board, connect the NRF24L01 radio to the gateway according to the picture and table below:



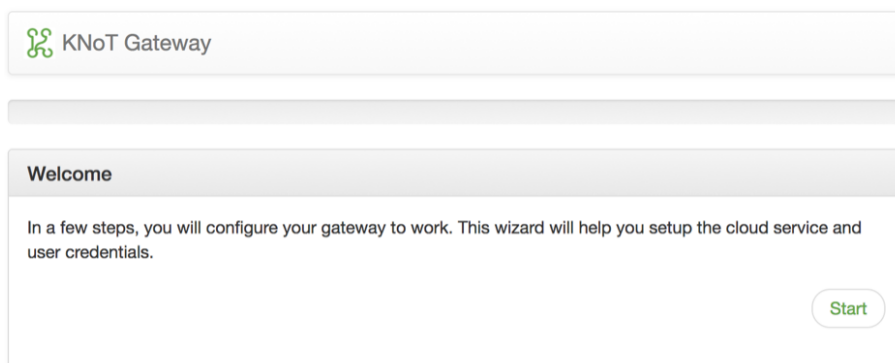
Arduino	NRF24L01+	Color
6/GND	1/GND	Black
1/3.3V	2/VCC	Red
22/GPIO25	3/CE	Orange
24/GPIO8	4/CSN	Yellow
23/SPI_CLK	5/SCK	Green
19/SPI_MOSI	6/MOSI	Blue
21/SPI_MISO	7/MISO	Violet

- Step 3 (b): If you have the KNoT Gateway Board, connect the KNoT Gateway Board according to the picture below:




You can find the Gerber files to print your own KNoT Gateway Board on Github link: https://github.com/CESARBR/knot-hardware-pcbs/tree/master/KNoT_Gateway/KNoT-Gateway%20Starter%20Board

- Step 4: Remove SD Card from PC and insert it into the raspberry pi.
- Step 5: Connect your PC to the Raspberry Pi using an Ethernet cable.
- Step 6: Power the Raspberry Pi. Attention: plug the raspberry pi into a 2A power supply. Even if it turns on and boot by plugging it on a PC USB port, the radio will be very unstable and the communication will be prejudiced. The 3.3V power on the Raspberry Pi is rated for a maximum of 50mA. A regular NRF24L01+ only needs 15mA, but if you are using a power amplified version you might exceed what the Raspberry Pi can output. In that case, an external 3.3V power supply might be required. If you use an external power source, GND from external power must be connected to the Raspberry Pi's GND.
- Step 7: Access <http://knot.local> from your PC and click on "Start" to start configuring the gateway.



- Step 8: Enter the cloud IP address as Hostname and port (default is 3000) and then click "Next". You can find the ip address of your machine with "ifconfig | grep inet" for Mac or "ip add show" for Linux.

 KNoT Gateway

Step 1. Cloud service


Set the KNoT Cloud server address that this gateway will connect to. This is the server where your device data and user credentials will be stored.

Hostname

Port

Previous
Next

- Step 9: Create a user (owner of the gateway and devices connected to this gateway) by providing an e-mail and password and click "Finish".

 KNoT Gateway

Step 2. User credentials

Create a user account to sign in on this gateway. This account will be saved in the KNoT Cloud server configured in the previous step.

E-mail

Password

Password confirmation

Previous
Finish

You have your brand new KNoT Gateway ready to connect to the KNoTThings!

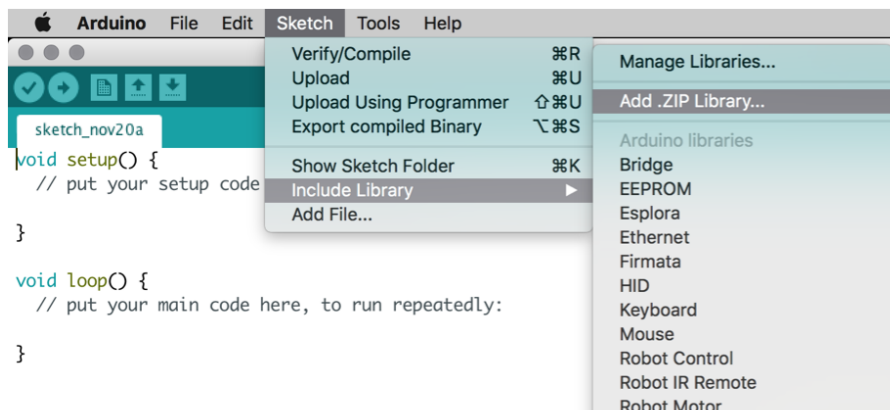
Building the Thing

What you will need:

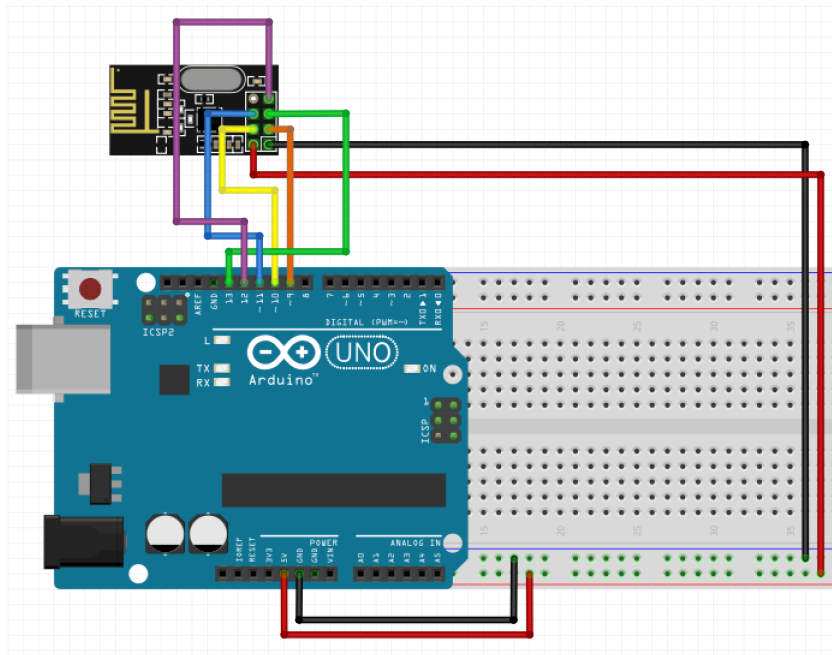
- Arduino UNO, pro mini or nano (KNoTThing library was tested on those Arduino versions. Probably it will work with other versions but we did not test on other versions).
- NRF24L01 radio or the KNoTThing PCB.
- One LED, one switch and the corresponding resistors (220Ω and $10k\Omega$).
- Linux, Mac or Windows PC.
- Arduino IDE that can be downloaded here: <https://www.arduino.cc/en/Main/Software>.

Steps to build the Thing:

- Step 1: Download the KNoTThing Arduino library here: <http://knot-devel.cesar.org.br/releases/v01.03-rc05/KNoTThing-v01.03-rc05.zip>.
- Step 2: Install the arduino library in the Arduino IDE, by accessing the menu Sketch→Include Library→Add .ZIP Library:

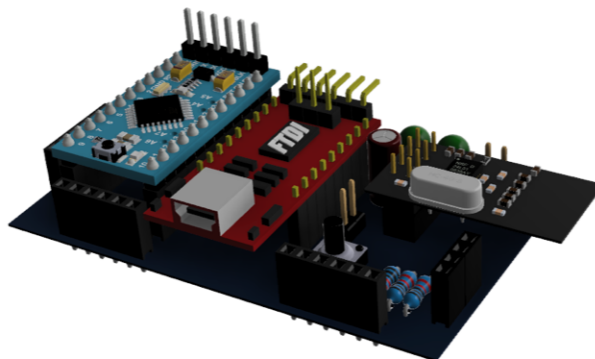


- Step 3(a): If you don't have the KNoTThing PCB, connect the NRF24L01 to the Arduino according to the picture and table below:

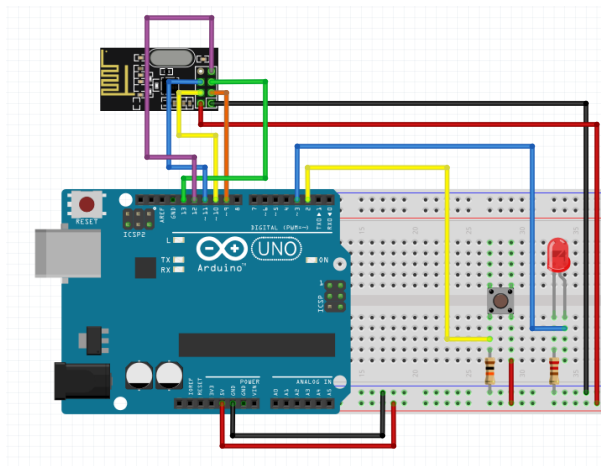


Arduino	NRF24L01+	Color
GND	1/GND	Black
3.3V	2/VCC	Red
D9	3/CE	Orange
D10	4/CSN	Yellow
D13/SPI_CLK	5/SCK	Green
D11/SPI_MOSI	6/MOSI	Blue
D12/SPI_MISO	7/MISO	Violet

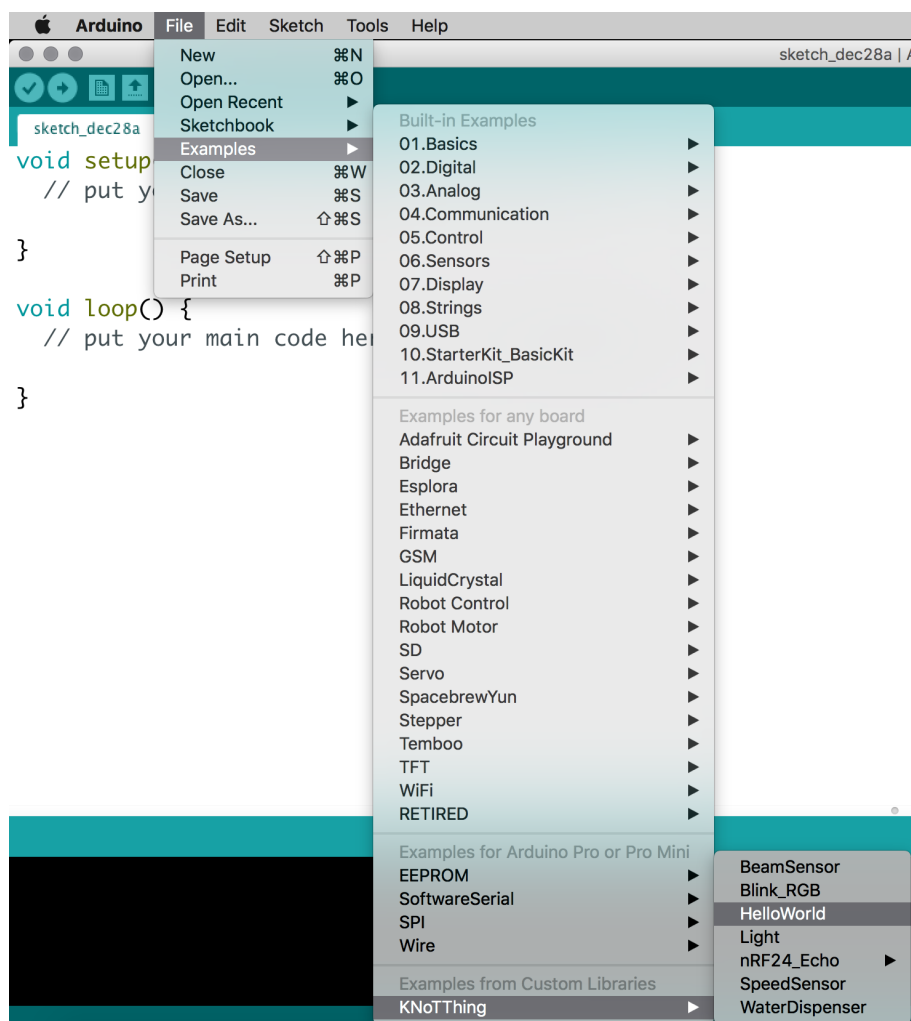
- Step 3(b): If you have the KNoTThing PCB, use it as below:



- Step 4(a): If you don't have the KNoTThing PCB, connect the switch and the LED to the Arduino pins 2 and 3 respectively:



- Step 4(b): If you have the KNoTThing PCB, you should see something like "KNoT StarterKit Rev xx.xx" printed on your board, where the "xx.xx" should be your thing's version. The version 01.02 has an integrated user LED at pin 2. Use that LED and connect a switch connected at pin 3. The version 01.03 has an integrated user LED at pin 5 and an user switch at pin 2. You can use the Figure from Step 4 (a) as reference.
- Step 5: Open the KNoTThing HelloWorld Demo example:



- Step 5.1: If you have the KNoTThing PCB, update the directive line showed below by replacing the VXX_XX with your respective board's version.

```
#define KNOT_BOARD_VXX_XX
```

For example, if you have a "KNoT StarterKit Rev 01.02" it should be replaced as showed below.

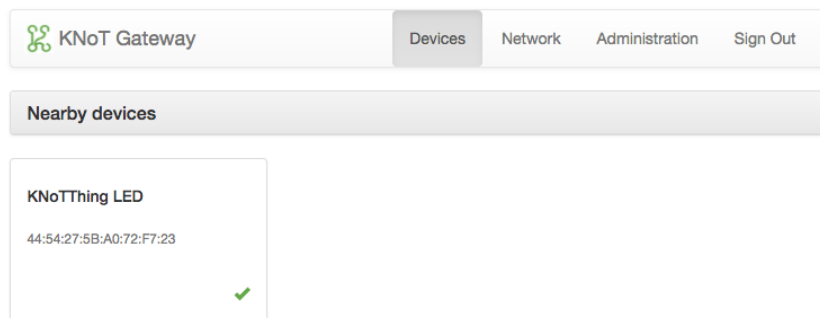
```
#define KNOT_BOARD_V01_02
```

- Step 6: Upload the sketch to the arduino board.

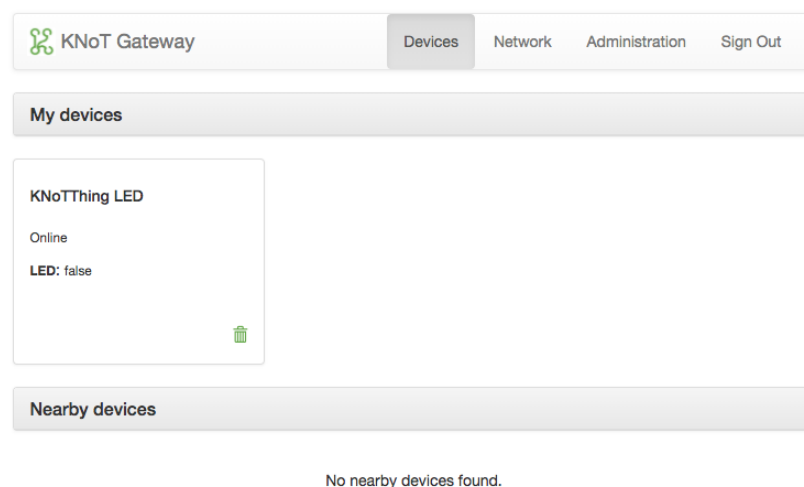
You have your brand new KNoTThing ready to connect to the KNoT Gateway!

Connecting all together

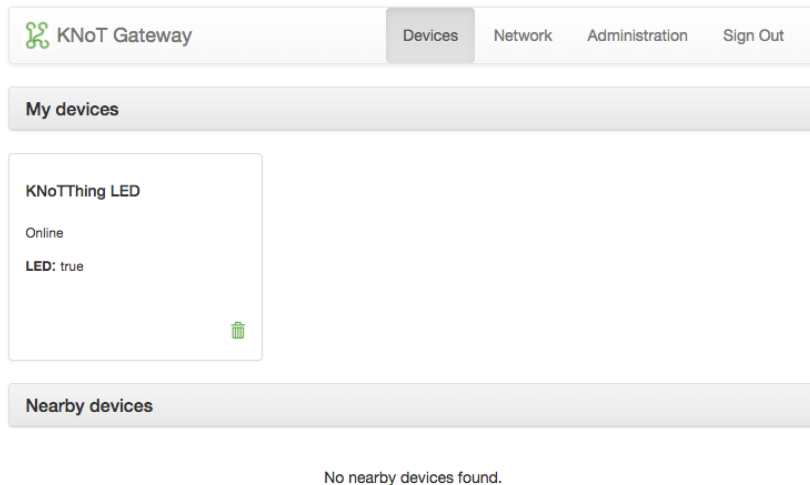
- Step 1: Open the gateway web interface (<http://knot.local>) and verify that the thing appear in the "nearby devices" list:



- Step 2: Authorize the thing to connect to the gateway by clicking on the check icon, ✓. After that, verify that the Thing is online.



- Step 3: Click on the switch and verify that the LED lights up and the LED value changes in the gateway.



You are all set! Now it is time to bring your IoT solution to life!

If the thing is displayed as offline, check the following section to get some help.

Offline Thing

If you're able to connect the thing to the gateway but it is displayed offline, follow the steps:

- Step 1: Clear the KNoTThing's EEPROM by pressing the red button for 5 seconds . Delete the Thing from your gateway connected devices. This will make it return to its original state.
- Step 2: Reset the Gateway radio and try to connect the thing again.
- Step 3: Make sure that the meshblu server is online by typing:

```
# curl 127.0.0.1:3000/status
```

- Step 4: Make sure that the MongoDB server is running on the expected port by typing:

```
# mongo --host 127.0.0.1:27017
```

The result should include, among other lines, something like:

```
"MongoDB server version:"
```

- Step 5: Make sure you changed the line 60 of config.js to:

```
59 mongo: {  
60   databaseUrl: 'mongodb://localhost:27017/knot_cloud'  
61 },
```