

Orange IoT Starter KiT

User Guide



Document Control

| | |
|----------------------|--|
| Title: | User Guide for IoT Starter Kit |
| Issue: | Version 1.0 |
| Date: | 17 January 2016 |
| Author: | T. Ansanay |
| Distribution: | Orange Labs Products and Services |

Document Sign Off

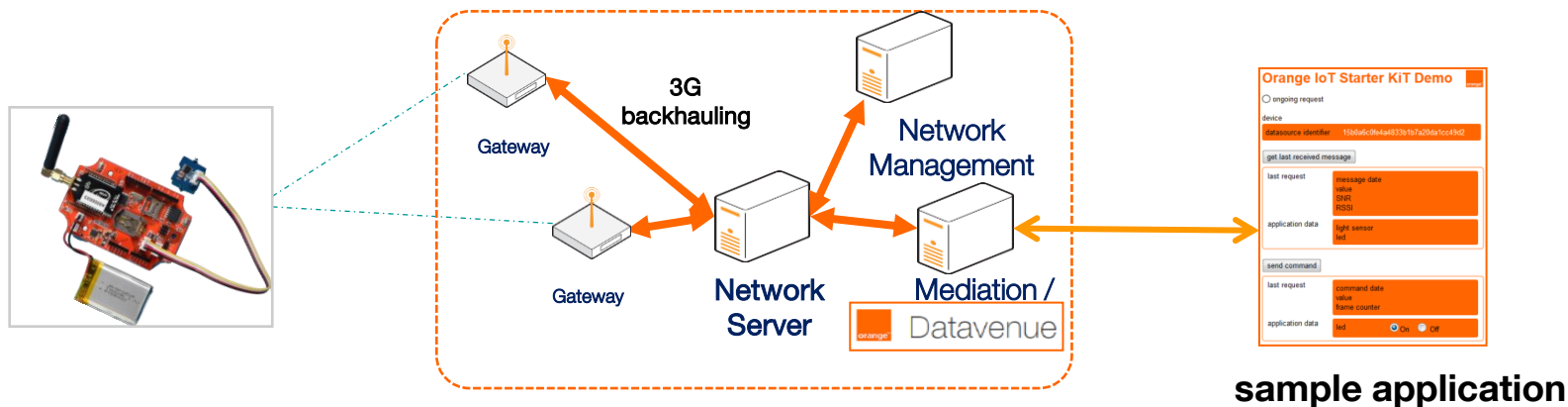
| | | | | |
|--------------------------|-------------------|--------------------|-------------------|------------------------|
| Nature of Signoff | Person | Department | Date | Role |
| Authors | T. Ansanay | Orange Labs | 29/02/2016 | Project Member |
| Reviewers | S. Quélard | Orange Labs | 15/03/2016 | Project Manager |

Document Changes

| | | | |
|------------------------|--------------------|-------------------|-----------------------------|
| Date | Version | Author | Change Details |
| 17 January 2016 | Draft 0.1 | T. Ansanay | First complete draft |
| 15 March 2016 | Draft 0.2 | T. Ansanay | Review and update |
| 21 March 2016 | Version 1.0 | T. Ansanay | |

Purpose

- This document presents the **Orange IoT Starter Kit**, that is based on an Arduino platform and integrates LoRa[®]* connectivity of Orange LoRa[®] experimental network.
- The kit is provided with a sample application.



Contents

1. Get your starter kit up with the web application running
2. Starter Kit components
3. Orange IoT Starter Kit LoRa[®] connectivity
4. Setting up of the Arduino environment
5. Getting the debug option work

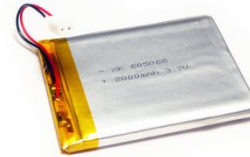
1. Get your starter kit up & web application running

Orange IoT Starter Kit components

The kit is composed with

- A Seeeduino Stalker board v2.3
- A radio shield integrating LoRa®* connectivity, format Xbee, provided by ATIM
- A LIPO battery 3.0 volts
- A Light Sensor with I2C cable
- A FTDI serial converter from Spakrfun
- One USB cable
- An antenna

For more information about each of these components, have a look at the [Starter Kit Components part](#).



Sample application

Description of the sample application

- ✓ the device sends periodically -every **3 minutes**- a message containing the last luminosity sensor value. The default periodicity is set to 3 minutes in order to respect the duty cycle imposed on the unlicensed bands, considering the device is in a constrained radio environment (SF 12). (this periodicity can be decreased if your device is in good radio conditions).
- ✓ The web application can display the last received luminosity sensor information.
- ✓ The web application can be used to send a message to set on/ off or in blink state the led of the Arduino board.

The sample application code can be founded on
<https://github.com/Orange-OpenSource/lpwa-iot-kit>

The sample application is composed of

- a program to run on the Arduino :

“DemoStarterkit_Stalker_Lightsensor_Led_V1.0_nodebug.ino”

or *“DemoStarterkit_Stalker_Lightsensor_Led_V1.0_debugmode.ino”*

- a web application program, that uses Datvenue APIs

“DemoStarterkit”

Sample Arduino program code

The sample program code is a simple Arduino code written in C language including a Setup function and a main loop.

The **setup function** initialized the UART used for Debug and the serial software link used to exchange data with the modem by calling two methods:

- ✓ `initXbeeDebugSerials (19200, 19200);` // set the hardware and software Serial speeds of the Atmega 328
- ✓ `initXbeeNanoN8 ();` // init xbee serial speed and read the DevEUI version

The **main loop** sequences the exchanges with the modem on the IoT Network :

- **Reading** Light Sensor value
- **Transmitting** frame payload of 5 bytes (one byte for led state and four bytes for light Sensor Value)
- **Temporization** 2,1 seconds
- **Listening** and **Reading** data from ATIM modem.
- **Tempo.** ~ 3 minutes

Sample web application code

The sample web application is a simple JavaScript application, that can be launched from a PC.


It uses Datvenue APIs, to send command to the device and retrieve messages sent by the device.

The application also handles the AES encryption of the data, as defined in LoRaWAN™ specification (application data encryption) . IoT Territory network exposes the encrypted data to the user.

Your device is a LoRa® class A device, it is not in permanent reception. It will receive the sent command just after an emission (emission every 3 minutes)

Important information about LoRa® technology and the the IoT territory network are available in the part “[Orange IoT Starter Kit LoRa® connectivity / IoT Territory experimental network](#)”. Do not forget to read it.

Orange IoT Starter Kit Demo



☐ ongoing request

datasource identifier

36199223a6334087b91e53fa60ca85d0

get last received message

last request

message date

17/3/2016 10:47:05

value

0x00000003ff

SNR

11 dB

RSSI

-45 dBm

application data

light sensor

1023

led

Off

send command

last request

command date

17/3/2016 10:47:33

value

0x02

frame counter

892

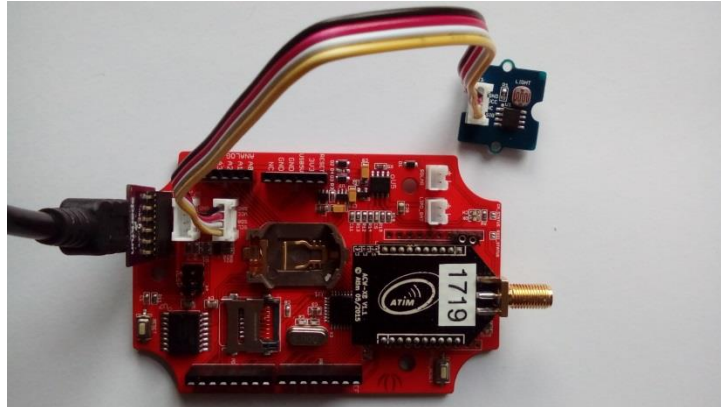
application data

led

☐ On ☒ Off ☐ Blink

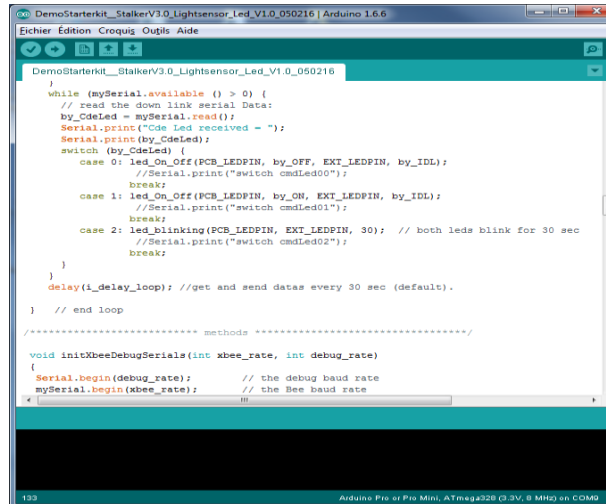
Starter Kit set up

- 1 Plug the ATIM radio modem on the stalker 2.3 platform on the bee socket.
- 2 Plug the Groove Light sensor on the I²C socket.
- 3 Then connect the stalker kit to the USB serial of your computer through the Sparkfun FTDI board which is plugged to the UART socket as shown on the following photo.



Starter Kit set up

- 4 Launch Arduino IDE, if you are not used to this environment refer to the “Setting up the Arduino environment” part to set up the IDE and get your first Arduino program running.
- 5 Unless already done, set the board (Tools -> board) to ATmega328 Arduino Pro or Pro Mini , The processor (Tools -> Processor) to ATmega328 (3,3V 8 MHz) and the port to the right port (depends on your computer).



```
Arduino IDE - DemoStarterkit_StalkerV3.0_Lightsensor_Led_V1.0_050216 | Arduino 1.6.6
Echier Edition Croquis Outils Aide

DemoStarterkit_StalkerV3.0_Lightsensor_Led_V1.0_050216
}
while (mySerial.available () > 0) {
  // read the down link serial Data:
  by_CdeLed = mySerial.read();
  Serial.print("Cde Led received = ");
  Serial.print(by_CdeLed);
  switch (by_CdeLed) {
    case 0: led_On_Off(PCB_LEDPIN, by_OFF, EXT_LEDPIN, by_IDL);
            //Serial.print("switch cmdLed00");
            break;
    case 1: led_On_Off(PCB_LEDPIN, by_ON, EXT_LEDPIN, by_IDL);
            //Serial.print("switch cmdLed01");
            break;
    case 2: led_Blinking(PCB_LEDPIN, EXT_LEDPIN, 30); // both leds blink for 30 sec
            //Serial.print("switch cmdLed02");
            break;
  }
  delay(1_delay_loop); //get and send datas every 30 sec (default).
} // end loop

----- methods -----
void initXBeeDebugSerials(int xbee_rate, int debug_rate)
{
  Serial.begin(debug_rate); // the debug baud rate
  mySerial.begin(xbee_rate); // the Bee baud rate
}
```

Starter Kit set up

The kit can be used with or without a debug mode.

The debug mode enables you to visualize the received and sent commands (to and from Nano-N8) To configure the kit with debug mode refer to the “[Getting the debug work](#)” part

6

No debug

Upload the program "DemoStarterKit_LightSensor_Led_V1.0_nodebug.ino"

Take off the USB serial of your computer and use the Lipo battery

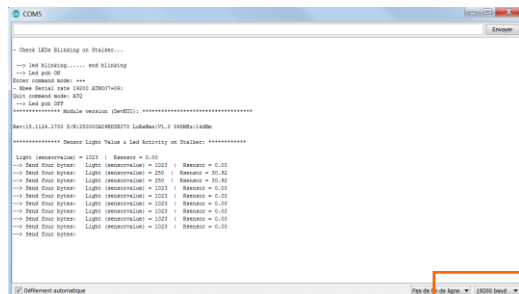
The serial monitor can not be used if you use no debug mode.

6 bis

With debug

Upload the program "DemoStarterKit_LightSensor_Led_V1.0_debugmode.ino"

Open the serial monitor available in IDE or putty and enjoy the view



Set the baud rate to 19200

Starter Kit set up

- 8 Copy the **web application** repertory DemoStarterKit on your PC
- 9 Copy the configuration file.js that we provided to you in the repertory demoStarterKit_files and rename it in '**configuration.js**'
You can check that it uses your device parameters (radio and Datavenue parameters).

- ✓ your device Datavenue Datasource Identifier.
- ✓ your default appSKey:
"321c0c42338290c81b6bee2c8a34957c ».
If you need to change this key have a look at the part : « [Changing the kit's parameters](#) ».
- ✓ your datavenue OAPI-Key and ISS- Key.

```
//=====
//
//      CONFIGURATION CONSTANTS
//
//=====

var _CONFIG = {

  //----- datasource identifier for the concerned device
  datasource: "f7b3517eb2964541a73b5e238836d38a",

  //----- AES encryption/decryption cipher application session key (use "" for no encryption)
  appSKey: "321c0c42338290c81b6bee2c8a34957c",

  //----- datavenue server url
  datavenueUrl: "https://api.orange.com/datavenue/v1",

  //----- security keys
  orangeAdminKey: "mr58c01jXw1bYd3koGv00BwntDZIak12",
  masterKey: "dcc5dff2c3ef4680b4555dcd5be3ac4e",

  //----- maximum time for a request (in milliseconds, use 0 for no timeout)
  requestTimeout: 8000,

  //----- Command FPort
  CmdFPort: 5,

  //----- light sensor measurement range
  lightMin: 40000,    // illuminated
  lightMax: 1000000   // in darkness
};
```

← X-OAPI-Key

← X-ISS-Key


Starter Kit set up

- 10 Launch the web application by double clicking “**DemoStarterKit.htm** ” , the following window should appear in the browser:

Browsers running with the demo:

| navigateur | versions |
|-------------------|--------------|
| Firefox | 41.0 et 42.0 |
| Chrome | 46.0.2490 |
| Internet Explorer | 11.0.9600 |

Orange IoT Starter Kit Demo



☐ ongoing request

datasource identifier cd331e50a6f14d11b6387d994347c209

get last received message

last request

message date
value
SNR
RSSI

application data

light sensor
led

send command

last request

command date
value
frame counter


application data

led ☐ On ☒ Off ☐ Blink

Starter Kit set up

- 11 Read the up link data
Activate the button « get last received message » you should see the data source identifier number and the light sensor value should appear in the browser as shown on the following figure

Orange IoT Starter Kit Demo



☐ ongoing request

datasource identifier

36199223a6334087b91e53fa60ca85d0

get last received message

last request

message date

17/3/2016 16:45:35

value

0x00000000f6

SNR

8 dB

RSSI

-102 dBm

application data

light sensor

246

led

Off

send command

last request

command date

17/3/2016 16:46:02

value

0x02

frame counter

936

application data

led

☐ On ☐ Off ☒ Blink

Starter Kit set up

- 12 Send a command from the web application
- Activate either the « On », « Off » or « Blink » button on the web application then send the command and check the corresponding action of the led on the board as described.
- As your device is a LoRa Class A device, it will receive the command just after its next message sending (can take 3 minutes, as the frequency of message emission is 3 minutes)

Orange IoT Starter Kit Demo

☐ ongoing request

datasource identifier cd331e50a6f14d11b6387d994347c209

get last received message

last request

| | |
|--------------|-------------------|
| message date | 5/2/2016 16:48:47 |
| value | 0x00000003ff |
| SNR | 3 dB |
| RSSI | -102 dBm |

application data

| | |
|--------------|------|
| light sensor | 1023 |
| led | Off |

send command

last request

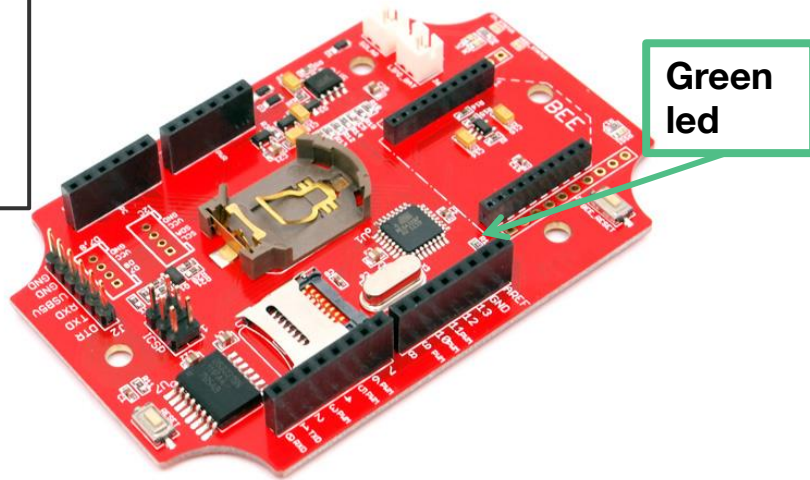
| | |
|---------------|--------------------|
| command date | 25/2/2016 15:12:26 |
| value | 0x02 |
| frame counter | 1149 |

application data

| | |
|-----|---|
| led | <input type="radio"/> On <input type="radio"/> Off <input checked="" type="radio"/> Blink |
|-----|---|

Led commands sent from the web application:

| | |
|--------------|----|
| Led OFF | 00 |
| Led ON | 01 |
| Led Blinking | 02 |



Starter Kit set up

13 You can check the received command from the web application

- ✓ If you have activated the debug option, open a debug window
- ✓ You should be able to read the light sensor values, sent (uplink) to the application and the led commands (downlink) issued by the web application.

```
- Check LEDs Blinking on Stalker...
--> led blinking..... end blinking
--> Led pcb ON

- Enter command mode: +++
--> Xbee Serial rate 19200 ATM007=06:

- Quit command mode: ATQ
--> Led pcb OFF

***** Module version (DevEUI): *****

Rev:15.1124.1700 S/N:220000A09BD5B370 LoRaWan:V1.0 868MHz:14dBm

***** Sensor Light Value & Led Activity on Stalker: *****

--> Send four bytes: Light (sensorvalue) = 1023 | Rsensor = 0.00
--> Send four bytes: Light (sensorvalue) = 947 | Rsensor = 0.80
--> Send four bytes: Light (sensorvalue) = 933 | Rsensor = 0.96

--> Cde Led received = 2 --> led blinking..... end blinking
--> Send four bytes: Light (sensorvalue) = 1023 | Rsensor = 0.00

--> Cde Led received = 1 --> Led pcb ON
--> Send four bytes: Light (sensorvalue) = 1023 | Rsensor = 0.00

--> Cde Led received = 2 --> led blinking..... end blinking
--> Send four bytes: Light (sensorvalue) = 1023 | Rsensor = 0.00
--> Send four bytes: Light (sensorvalue) = 1023 | Rsensor = 0.00
--> Send four bytes: Light (sensorvalue) = 1023 | Rsensor = 0.00
--> Send four bytes: Light (sensorvalue) = 453 | Rsensor = 12.58

--> Cde Led received = 2 --> led blinking..... end blinking
```

initialisation
phase

Orange IoT Starter Kit Demo

☐ ongoing request

datasource identifier 1cc07401e00140fd91498a2355bb35db

get last received message

last request

message date 21/3/2016 10:41:44
value 0x00000001c5
SNR 8 dB
RSSI -88 dBm

application data

light sensor 453
led Off

send command

last request

command date 21/3/2016 10:39:39
value 0x02
frame counter 431

application data

led ☐ On ☐ Off ☒ Blink

2. Starter Kit components

Orange IoT Starter Kit components

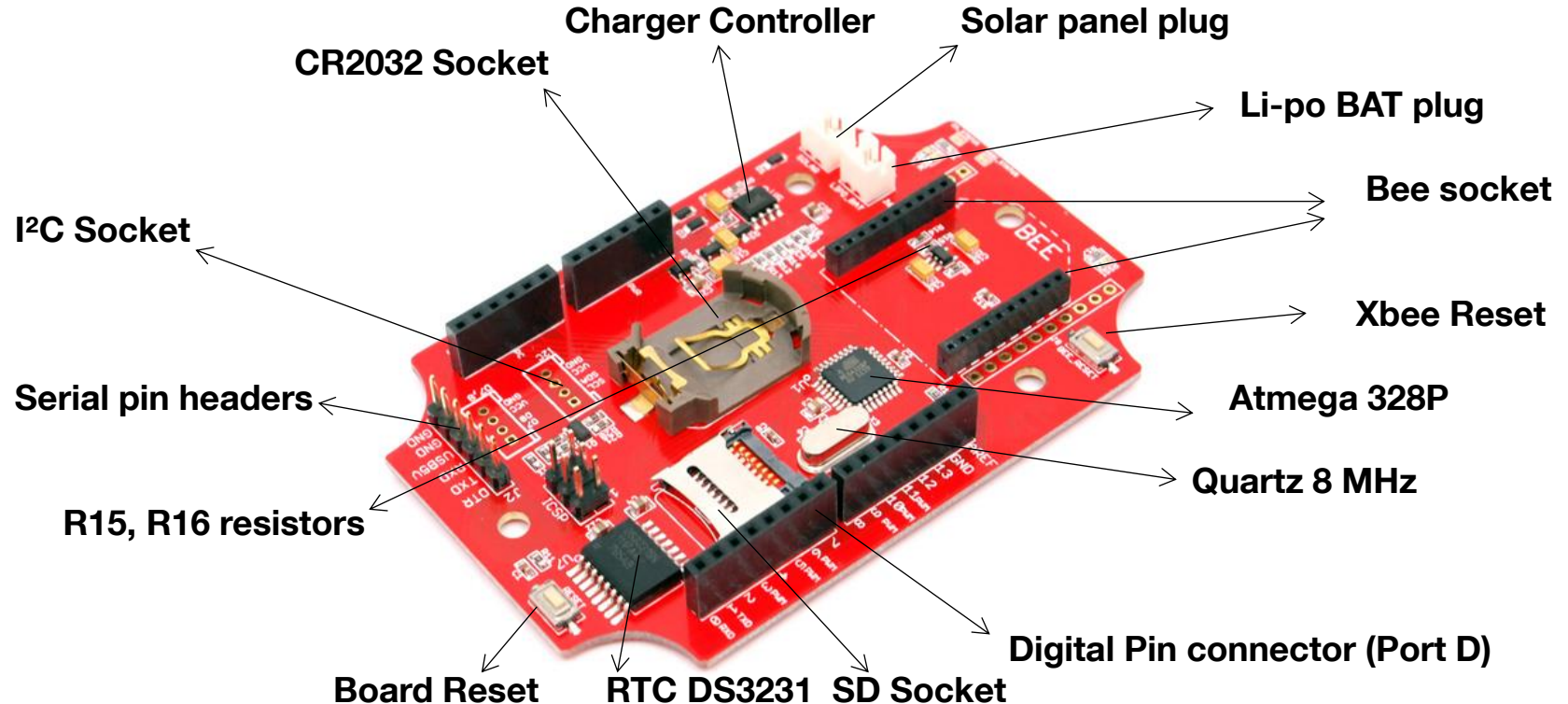
Arduino Stalker Board Specifications

This platform is compatible Arduino with Pro and Pro mini which is architected around an ATMEGA 328 micro-controller:

| Parameter | Value |
|------------------------------|---|
| MCU | ATmega328 |
| Crystal Oscillator | 8 MHz |
| RTC | DS3231 |
| I/O Logic | 3.3v |
| Board for Arduino IDE | Arduino Pro or Pro Mini (3.3v , 8 MHz)/ATmega328 |
| Power Supply | 3.7v LiPo Battery, Use 5VDC solar panel for charging the battery. |
| Power Connector | 2 pin JST/ USB |
| Connectivity | I2C, UART, SPI |
| Open Circuit Current | 6 mA max |
| Charging Current | 300mA |
| Maximum Current on 3.3v port | 800mA |
| PCB Size | 92.71mm X 60.96mm |

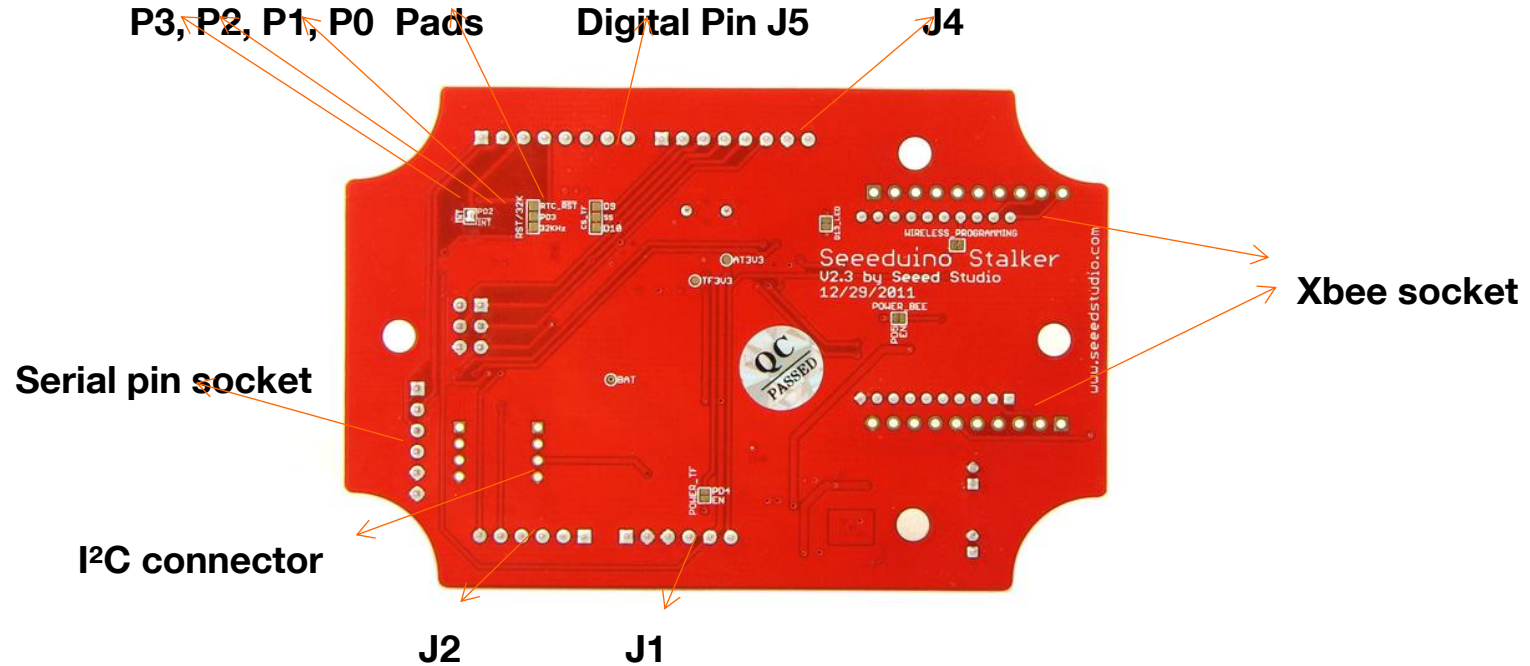
Orange IoT Starter Kit components

Arduino Stalker Board hardware view



Orange IoT Starter Kit components

Arduino Stalker Board bottom view



Orange IoT Starter Kit components

LoRa[®] radio shield



The LoRa[®] radio shield provided in the starter kit is a shield developed by ATIM. (<http://www.atim.com/en/>)

This shield integrates the ATIM Nano N8 LoRa[®] modem that is mounted on a Xbee form factor board of size 50x25x13 mm.

This board is powered under 3.3 v and has SMA connector for the antenna.

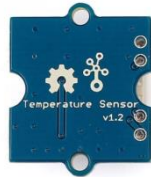
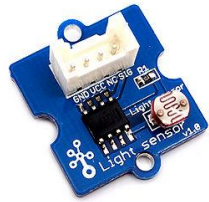
To connect the board to an Arduino platform two Xbee 10-pin socket with 2mm pitch are used.

Orange IoT Starter Kit components

Light sensor Groove version 1.1

The Light sensor module uses the GL5528 photo-resistor to detect the light intensity of the environment. The resistance of the sensor decreases when the light intensity increases.

The light sensor uses a thermistor which returns the ambient light in the form of a resistance value, which is then used to alter the voltage signal issue from Vcc supply of the Seeeduino. The Atmega 328 then converts this voltage value measured on an analog input pin and after converting it to digital compute the temperature.



Orange IoT Starter Kit components

FTDI board



This is a basic breakout board for the FTDI FT232RL USB to serial IC. The pinout of this board matches the FTDI cable to work with official Arduino and cloned 5V Arduino boards. It can also be used for general serial applications. The major difference with this board is that it brings out the DTR pin as opposed to the RTS pin of the FTDI cable. The DTR pin allows an Arduino target to auto-reset when a new Sketch is downloaded. This is a really nice feature to have and allows a sketch to be downloaded without having to hit the reset button.

This board has a jumper on the back of the board that allows the board to be configured to either 3.3V or 5V (both power output and IO level). This board ship default to 5V, and works on the Stalker v2.3 with both configuration.

We recommend to cut the default trace and add a solder jumper to switch to 3.0V for the Stalker 2.3. (but this is not mandatory for a proper functioning)

3. Orange IoT Starter KiT LoRa[®] connectivity

**IoT Territory experimental
network**

LoRa[®] , LoRaWAN[™]

A LoRaWAN[™] network is a Low-Power, Wide-Area Network (LPWAN), designed for **long range**, **low power** and **low data rates** applications.

LoRa[®] stands for the physical layer of the protocol. This radio technology has been developed and is owned by Semtech Corporation.

LoRaWAN[™] stands for the MAC layer protocol, and is specified by the LoRa[®] Alliance.

A LoRaWAN[™] network is a star network , in which gateways relays messages from devices to a network server.

LoRa[®] data rates ranges from 300 bps to ~50 kbps.

LoRaWAN[™] protocol uses an adaptive data rate algorithm, that enables to optimize device consumption, and network capacity.

LoRa[®] technology operates on unlicensed band. (863-870 MHz in Europe). These bands are regulated.

Useful LoRa[®] Alliance Website documents :
[LoRaWAN[™] protocol specification](#)
[What is LoRaWAN[™]](#)

Orange IoT Starter Kit LoRa® Set Up

The Orange IoT Starter Kit is compliant to LoRaWAN™ 1.0 specification.

The kit is configured to be a **LoRa® class A** device.

It supports APB activation mode only (it is provisioned on Orange experimental network).

The Adaptive Data Rate algorithm is activated.

The device is preconfigured to send unconfirmed UL messages.

The following LoRa® parameters are preconfigured on the device:

- devEUI
- devAddress : network address of the device.
- nwkSKey : network key of the device
- appSKey : application key (used to encrypt/decrypt applicative payload)

The default set up of the Orange IoT Starter Kit guarantees a proper functioning of the device on the Orange experimental network.

We highly recommend not to modify the default radio configuration of the device.

Orange IoT Starter KiT LoRa[®] connectivity

The Orange IoT Starter Kit is provisioned on Orange experimental LoRa network. (IoT Territory pilot network).

Orange IoT Territory network is an experimental network opened to clients since May 2015th. More information can be found on IoT Communities site in the document :

IoT Territory - Presentation EN.pdf

Information on LoRaWAN[™] functionalities supported on Orange IoT Territory network can be found in the document

LoRaWAN[™] functionalities on Orange IoT Territory network v1-0.pdf

IoT Territory Datavenue APIs

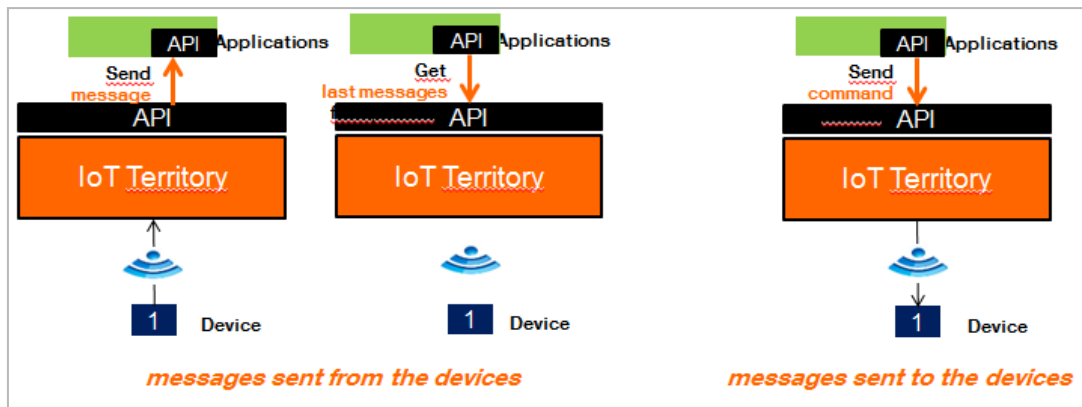
Communication with the device is done via **Orange Datavenue APIs**.

3 generic uses cases are available through the API

- *send a command to a device*
- *retrieve datas coming from a device (Polling mode)*
- *get datas coming from a device on your server (Push Mode)*

Data models

A device is identified by a *datasourceId* , and 4 streams associated to the datasource are defined : *message*, *command*, *battery*, *fcoun*t.



More information on the APIs for Orange IoT Territory network can be found on IoT Communities site in the documents:
[OrangelIoT API Startup guide v1.7.pdf](#)
<https://datavenue.orange.com/#/restapidoc>

Your starter kit credentials

A credential file containing LoRa parameters of your kit will be provided

Orange IoT Starter Kit

Your device credentials

LoRa® Parameters

devEUI : 70B3D59BA0123456

devAdress : 123456A0

nwkSkey : d2c4123454e8cf550c923b64267267e

appSkey : d4f11e40e52d219e09b3838d4f406782

Datavenue Parameters

X-OAPI-Key : mr58c01jXW1b123456ntDZlak12

X-ISS-Key : dcc5dff126787755444434ac4e

DataSourceId : 6e7c1dc74d8a4e9699b4f6e8155261f3

Stream Message Id : 36199223a6334087b91e53fa12345678

Stream Command Id : 6e7c1dc74d8a4e9699b4f6e8155261f3

Stream Battery Id : 3619123456334087b91e53fa12345678

Stream Downlink Id : 6123456785f4437e895d8f8a7ff2718c

the appSkey is needed to decrypt the applicative payload exposed by Datavenue

your Datavenue Keys

the datavenue Ids required to get your device data and send commands to your device

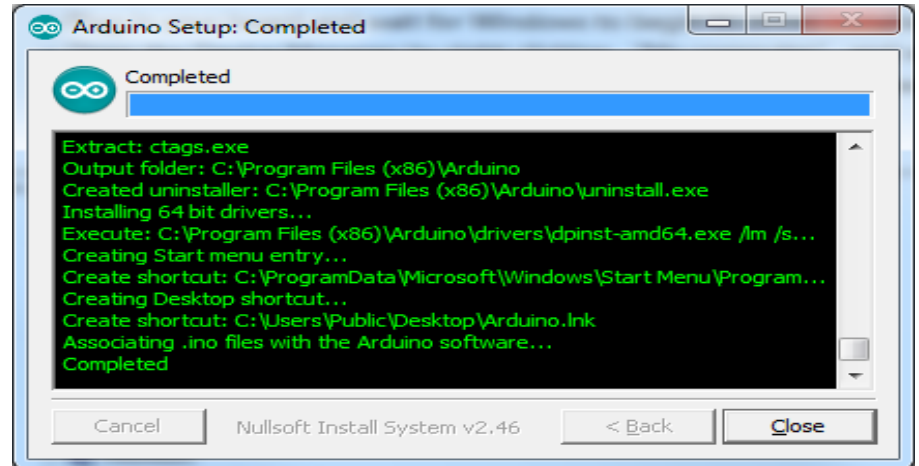
4. Setting up the Arduino environment

Arduino software (IDE): Download

Download the Arduino Environment version Software 1.6.6

<https://www.arduino.cc/en/Main/Software>

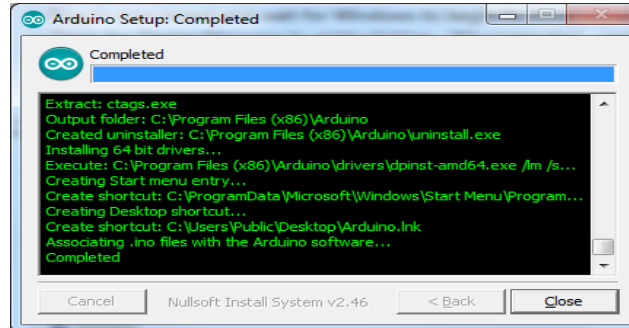
The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux.



Arduino software (IDE): Seeeduino driver installation

Install the driver for the Seeeduino with Windows 7

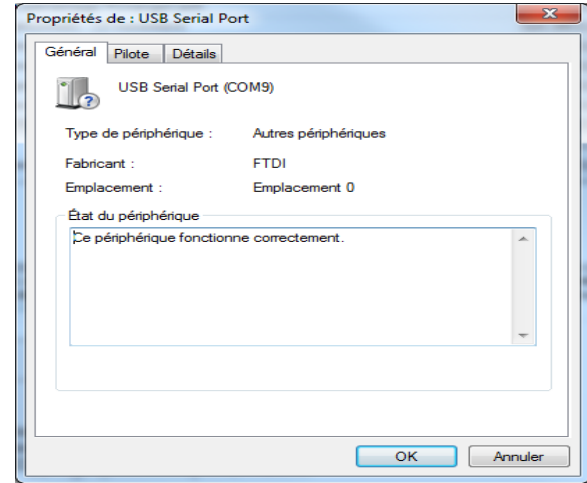
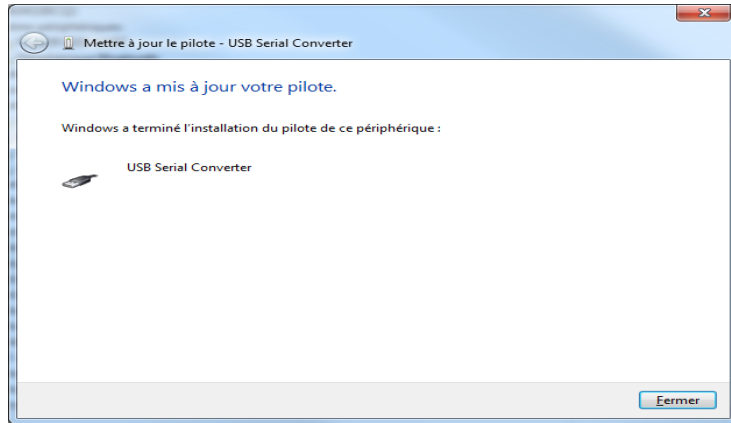
- ✓ Connect the Seeeduino board to your computer using the USB cable. The green power LED (labeled PWR) should go on.
- ✓ Plug in your board and wait for Windows to begin its driver installation process. After a few moments, the process will fail.
- ✓ Open the Device Manager by right clicking "My computer" and selecting control panel.
- ✓ Look under Ports (COM & LPT). You should see an open port named "USB Serial Port" Right click on the "USB Serial Port" and choose the "Update Driver Software" option.
- ✓ Select the driver file named "FTDI USB Drivers", located in the "Drivers" folder of the Arduino Software download



Arduino software (IDE): Seeeduino driver installation

Install the driver for the Seeeduino with Windows 7

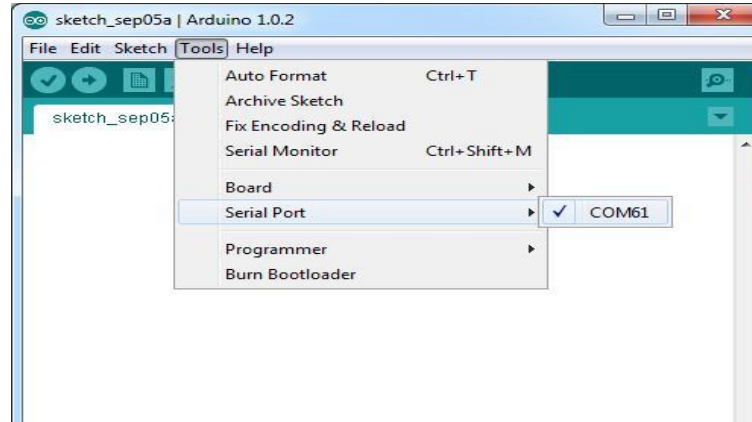
- ✓ The below dialog boxes automatically appears if you have installed driver successfully.:



Arduino software (IDE): Seeeduino driver installation

Install the driver for the Seeeduino with Windows 7

- ✓ You can also see the serial port in Arduino environment:



[http://www.seeedstudio.com/wiki/How to use Seeeduino#Installing drivers for the Seeeduino with window7](http://www.seeedstudio.com/wiki/How_to_use_Seeeduino#Installing_drivers_for_the_Seeeduino_with_window7)

Arduino software (IDE): program example



Getting Started with Seeeduino

- ✓ 0. Hello world
- ✓ 1. Connect Seeeduino to PC
- ✓ 2. Open the Blink example
- ✓ 3. Select your board
- ✓ 4. Select your Serial Port
- ✓ 5. Upload the program
- ✓ 6. Result

http://www.seeedstudio.com/wiki/Seeeduino_v3.0

Arduino software (IDE): program example

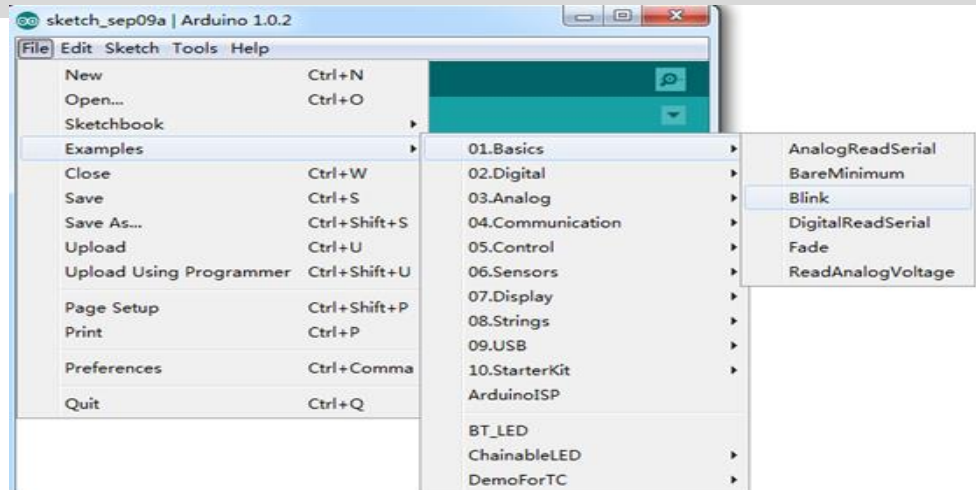


Getting Started with Seeeduino

1 . Connect Seeeduino to PC

Connect the Seeeduino board to your computer using the USB cable. The green power LED (labeled PWR) should go on. (When Seeeduino works independently, you can select USB or power adapter to power for Seeeduino.)

2 . Open the Blink example



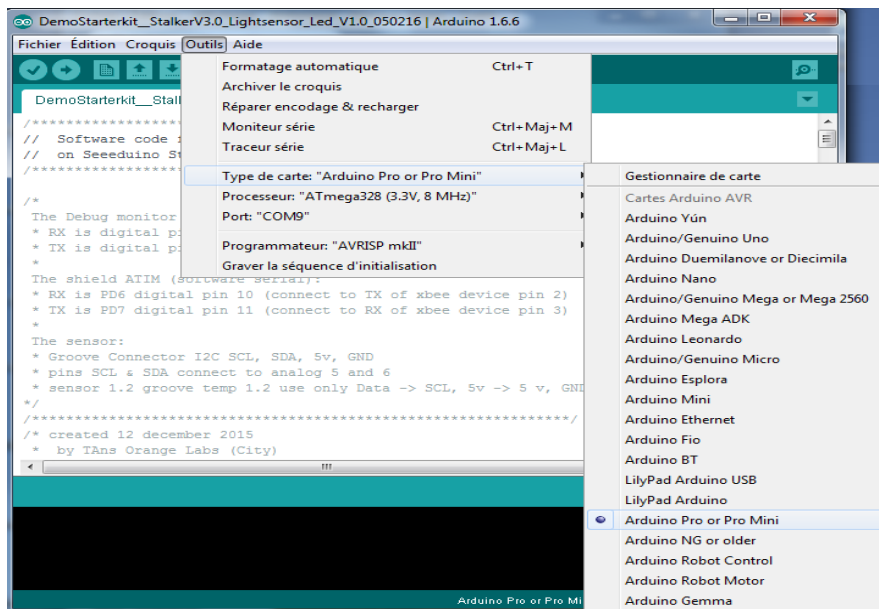
Arduino software (IDE): board type selection



Getting Started with Seeeduino

3. Select your board

You'll need to select the entry in the Tools > Board menu that corresponds to your Arduino.
Here we need to select ATmega328 Arduino Pro or Pro Mini.



Arduino software (IDE): Processor type selection

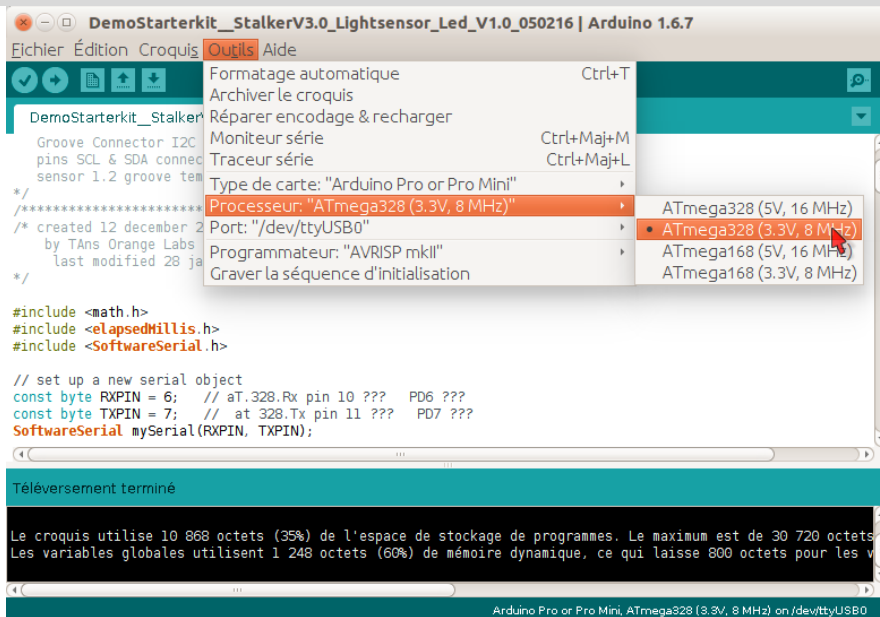


Getting Started with Seeeduno

4. Select your Processor

You'll need to select the entry in the Tools > Processor.

Here we need to select ATmega328 (3.3V 8 MHz).



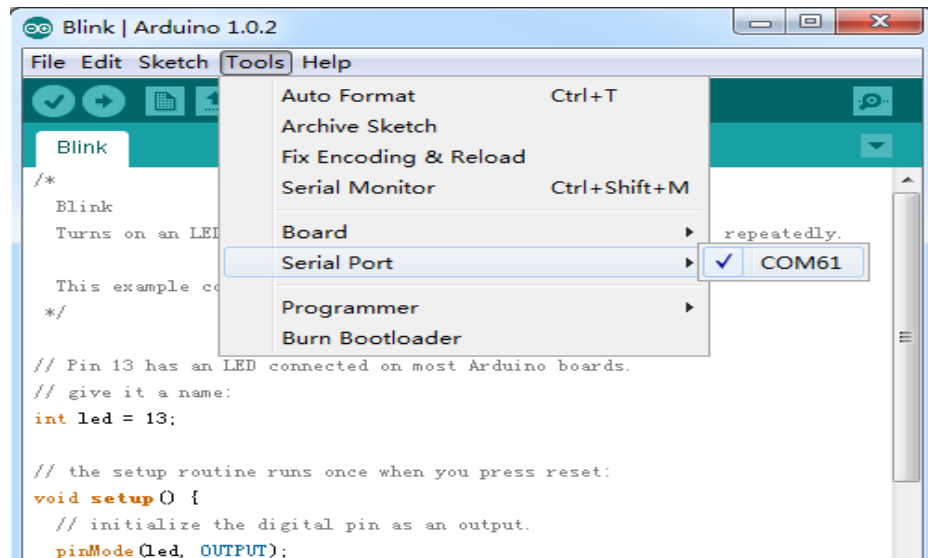
Arduino software (IDE): board type selection



Getting Started with Seeeduino

5. Select your Serial Port

Select the serial device of the Arduino board from the Tools | Serial Port menu:



Arduino software (IDE): compilation / uploading

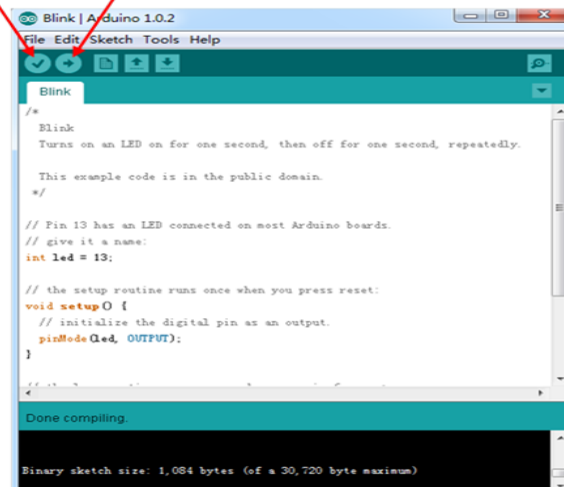
Getting Started with Seeeduino

6. Upload the program

Click the "Upload" button in the environment. Wait a few seconds - you should see the RX and TX leds on the board flashing. If the upload is successful, the message "Done uploading." will appear in the status bar.

Compile

Upload

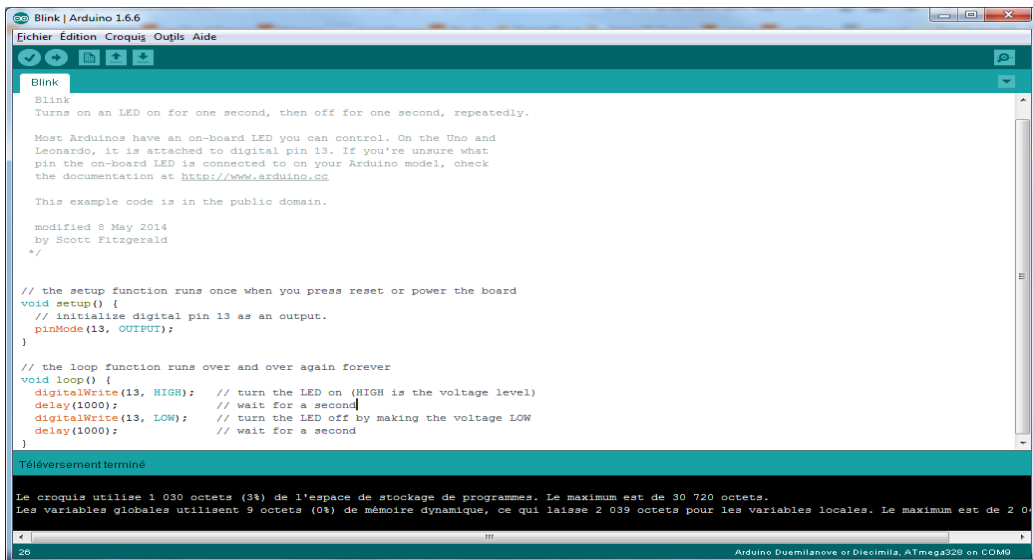


Arduino software (IDE): compilation / uploading

Getting Started with Seeeduno

7. Upload the program

If the upload is successful, the message "Done uploading." will appear in the status bar.



```
Blink | Arduino 1.6.6
Fichier Édition Croquis Outils Aide

Blink
Blink
Turns on an LED on for one second, then off for one second, repeatedly.

Most Arduinos have an on-board LED you can control. On the Uno and Leonardo, it is attached to digital pin 13. If you're unsure what pin the on-board LED is connected to on your Arduino model, check the documentation at http://www.arduino.cc

This example code is in the public domain.

modified 8 May 2014
by Scott Fitzgerald
*/

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);             // wait for a second
  digitalWrite(13, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);             // wait for a second
}

Téléversement terminé

Le croquis utilise 1 030 octets (3%) de l'espace de stockage de programmes. Le maximum est de 30 720 octets.
Les variables globales utilisent 9 octets (0%) de mémoire dynamique, ce qui laisse 2 039 octets pour les variables locales. Le maximum est de 2 048 octets.
```

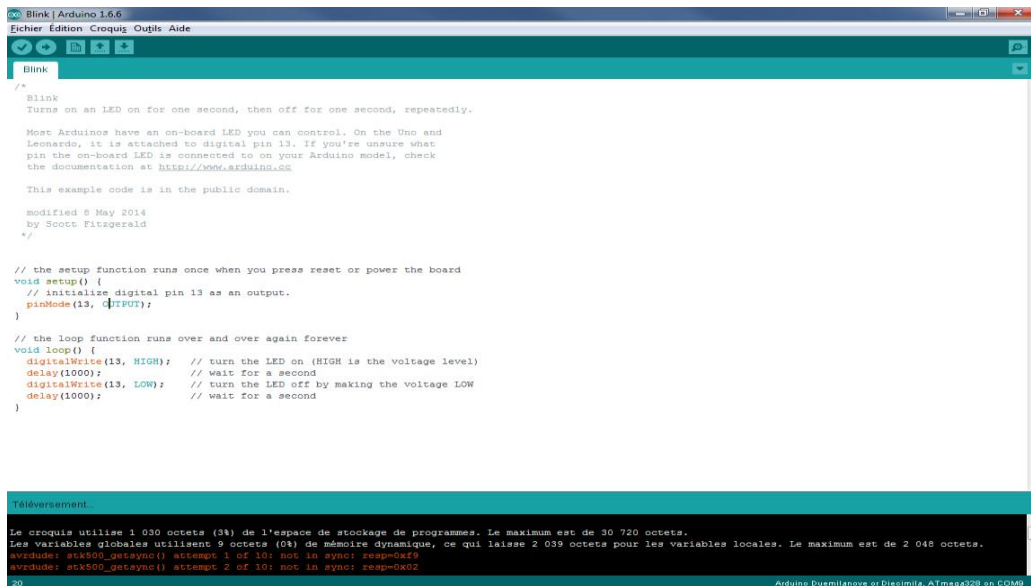
Arduino software (IDE): compilation / uploading

Getting Started with Seeeduno

8. Upload the program

An error of uploading might appear:

avrdude stk500_getsync(): not in sync resp=0x30 error → disconnect the Xbee shield.



```
Arduino IDE - Blink | Arduino 1.6.6
Echier Edition Croquis Outils Aide

Blink

/*
 * Blink
 * Turns on an LED on for one second, then off for one second, repeatedly.
 *
 * Most Arduinos have an on-board LED you can control. On the Uno and
 * Leonardo, it is attached to digital pin 13. If you're unsure what
 * pin the on-board LED is connected to on your Arduino model, check
 * the documentation at http://www.arduino.cc.
 *
 * This example code is in the public domain.
 *
 * modified 8 May 2014
 * by Scott Fitzgerald
 */

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);            // wait for a second
  digitalWrite(13, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);            // wait for a second
}

Téléversement...

Le croquis utilise 1 030 octets (3%) de l'espace de stockage de programmes. Le maximum est de 30 720 octets.
Les variables globales utilisent 9 octets (0%) de mémoire dynamique, ce qui laisse 2 039 octets pour les variables locales. Le maximum est de 2 048 octets.
avrdude: stk500_getsync() attempt 1 of 10: not in sync: resp=0x30
avrdude: stk500_getsync() attempt 2 of 10: not in sync: resp=0x30
20
Arduino Due/Arduino Uno or Diecimila, ATmega328 on COM9
```

Arduino software (IDE): compilation / uploading



Getting Started with Seeedduino

9. Result

A few seconds after the upload finishes, you should see the pin 13 (L) LED on the board start to blink (in green). If it does, congratulations! You've gotten Arduino up-and-running.

5. Getting the debug option work

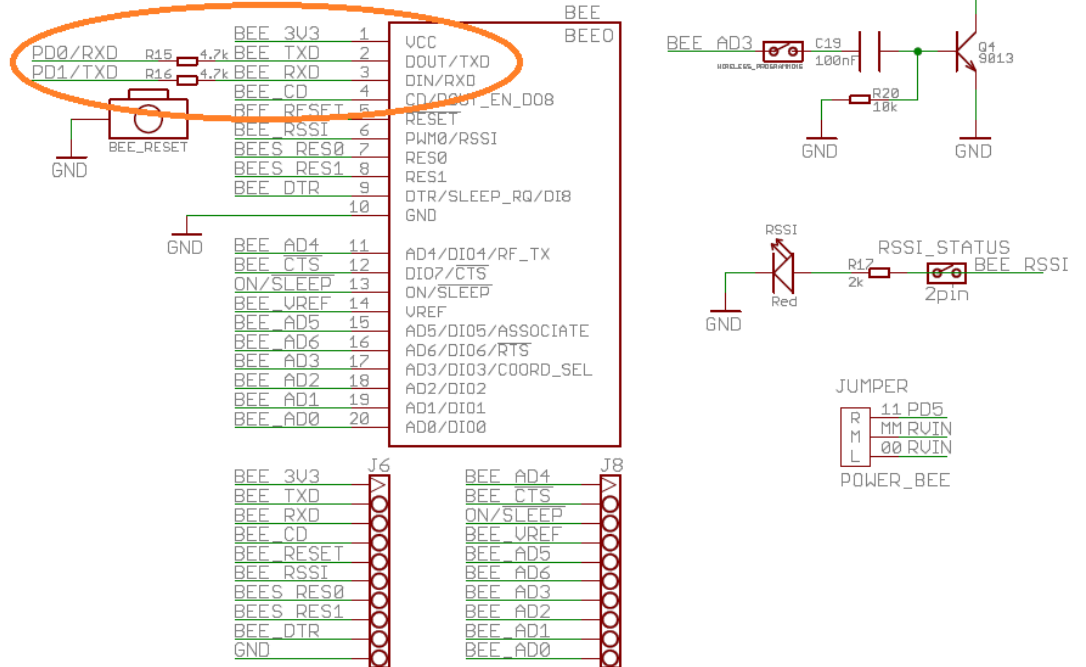
Orange IoT Starter Kit components

Arduino Stalker Board Software and Hardware Serial

Stalker 2.3 platform handles only one hardware serial which is already used by the Xbee shield and for uploading the software via USB interface.

So to add a serial connection for Debug with the PC, some hardware modifications needs to be performed.

XBee Interface



Orange IoT Starter Kit components

Arduino Stalker Board Software and Hardware Serial

The following tables show the initial hardware configuration which handles default serial hardware of Atmega for both program uploading and connection with xbee shield.

| Device Nano-N8 Modem ATIM | | FTDI USB serial via J2 connector | | PCB Route (via) | Hardware Serial (default) | |
|----------------------------------|---------------|----------------------------------|------------|--------------------|--|----------------------|
| Xbee modem connection name | pin number | J2 connection name | pin number | Resistor | PCB routing | Atmega pin number |
| Xbee.Tx | 2 | | | R15 | R15 Route to PD0 of digital connector (See note 1) | 30 |
| Xbee.Rx | 3 | | | R16 | R16 Route to PD1 of Digital connector (See note 1) | 31 |
| | | RxD of FTDI | J2.3 | | Route to PD0 of Digital connector | 30 |
| | | TXD of FTDI | J2.2 | | Route to PD1 of digital connector | 31 |

47

Note 1 : Digital Pin connector Port D

Orange IoT Starter Kit components

Arduino Stalker Board Software and Hardware Serial

The following tables show the new hardware configuration to have both the default hardware UART connection between Atmega328 MCU and the PC for Debug function along with an additional software serial for connecting the radio shield and keeping the uploading link on hardware serial:

| Device Nano-N8 Modem ATIM | | FTDI USB serial via J2 connector | | Hardware Serial (default) | | Additional Software Serial (Replace initial xbee /Atmega connection) | | |
|----------------------------------|---------------|-------------------------------------|---------------|---------------------------|--------------------------------------|---|---------------------------|---|
| Xbee modem connection name | pin number | J2 connection name | pin number | Atmega pin number | PCB routing | Atmega pin number | Resisto r to remove | Wire to add for Software serial |
| Xbee.Tx | 2 | | | | | 10 (PD6) | R15 | Wire(Note1) xbee.Tx to PD6 (pin 6 of Digital connector) |
| Xbee.Rx | 3 | | | | | 11 (PD7) | R16 | Wire(Note1) xbee.Rx to PD7 (pin 7 of Digital connector) |
| | | RxD of FTDI | J2.3 | 30 | Route to PD0 of Digital connector | | | |
| | | TXD of FTDI | J2.2 | 31 | Route to PD1 of Digital connector | | | |

Note 1 : see figure 1 and 2 on the following slide

Orange IoT Starter Kit components

Arduino Stalker Board Software and Hardware Serial

The following figures show the hardware modifications to have both the default hardware UART connection and the PC for Debug function along with an additional software serial for connecting the radio shield :

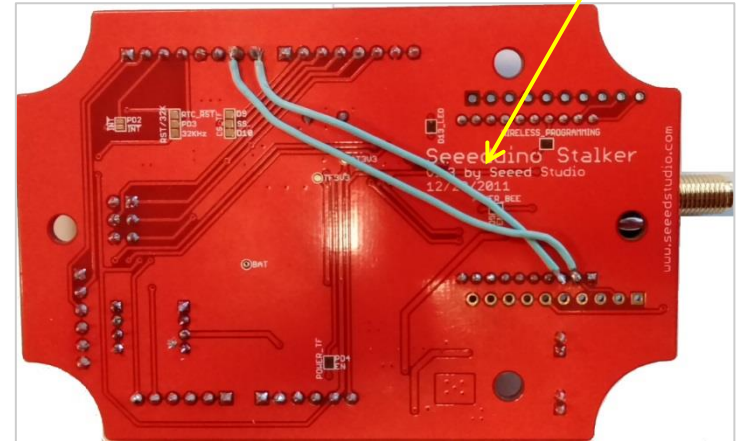
→ remove resistors R15, R16



Wires to solder

→ Xbee.Rx to PD7

→ Xbee.Tx to PD6

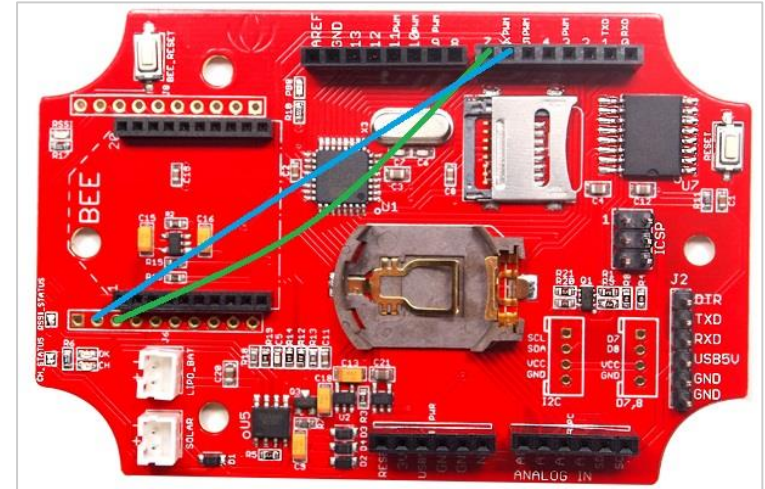


Orange IoT Starter Kit components

Arduino Stalker Board Software and Hardware Serial

If you want to have optional debug you can bypass the soldering and use two jumper wires as shown in the image below:

→ remove resistors R15, R16



5. Changing the kit's parameters

Changing your application Key

In case you want to change this key, First use the ATO command 074 by adding **mySerial.print("ATO074=AppSKey\n");** to the Arduino program if you have activated the debug or **Serial.print("ATO074=AppSKey\n");** else.

Now change the application session key in the web-application configuration file (key referred as AppSKey)



The default configuration of the ATIM module allows its access to the network. Any change you make can lead to your kit not functioning anymore.