IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS
TYLER DIVISION

LUCIO DEVELOPMENT LLC,	§	
	§	
Plaintiff,	§	Case No:
	§	
vs.	§	PATENT CASE
	§	
SEMTECH CORPORATION	§	
	§	
Defendant.	§	
	§	

COMPLAINT

Plaintiff Lucio Development LLC ("Plaintiff" or "Lucio") files this Complaint against Semtech Corporation ("Defendant" or "Semtech") for infringement of United States Patent No. 7,069,546 (hereinafter "the '546 Patent").

PARTIES AND JURISDICTION

1. This is an action for patent infringement under Title 35 of the United States Code. Plaintiff is seeking injunctive relief as well as damages.

2. Jurisdiction is proper in this Court pursuant to 28 U.S.C. §§ 1331 (Federal Question) and 1338(a) (Patents) because this is a civil action for patent infringement arising under the United States patent statutes.

3. Plaintiff is a Texas limited liability company with its office address at 555 Republic Dr., Suite 200, Plano, Texas 75074.

4. On information and belief, Defendant is a Delaware corporation with a principal place of business at 200 Flynn Rd., Camarillo, CA 93012. On information and belief, Defendant may be served through its registered agent, United States Corporation Company,

2711 Centerville Rd., Suite 400, Wilmington, DE 19808.

5. This Court has personal jurisdiction over Defendant because Defendant has committed, and continues to commit, acts of infringement in this District, has conducted business in this District, and/or has engaged in continuous and systematic activities in this District.

6. On information and belief, Defendant's instrumentalities that are alleged herein to infringe were and continue to be used, imported, offered for sale, and/or sold in this District.

VENUE

7. Venue is proper in this District pursuant to 28 U.S.C. §1400(b) because acts of infringement are occurring in this District and Defendant has a regular and established place of business in this District. For instance, on information and belief, Defendant has a regular and established place of business at 1101 Resource Dr., Suite 121, Plano, Texas 75074.

COUNT I
(INFRINGEMENT OF UNITED STATES PATENT NO. 7,069,546)

8. Plaintiff incorporates paragraphs 1 through 7 herein by reference.

9. This cause of action arises under the patent laws of the United States and, in particular, under 35 U.S.C. §§ 271, *et seq*.

10. Plaintiff is the owner by assignment of the '546 Patent with sole rights to enforce the '546 Patent and sue infringers.

11. A copy of the '546 Patent, titled "Generic Framework for Embedded Software Development," is attached hereto as Exhibit A.

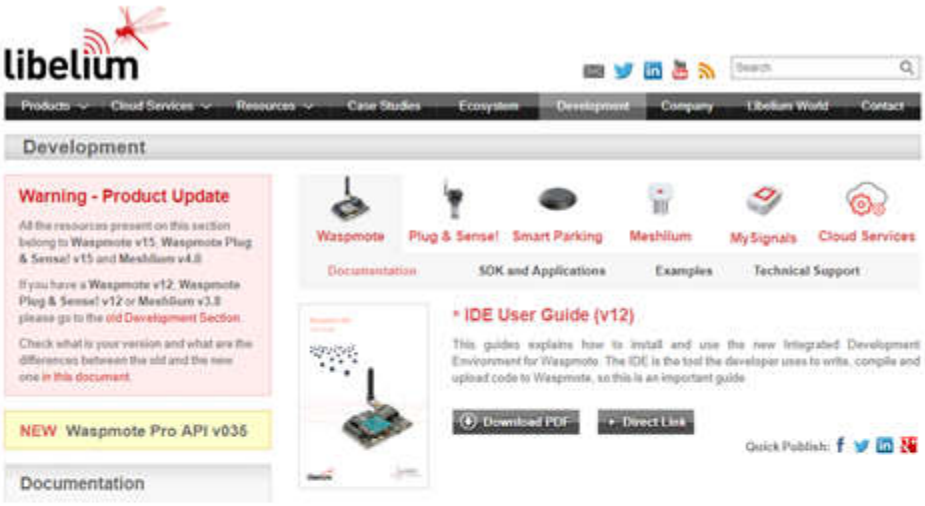
12. The '546 Patent is valid, enforceable, and was duly issued in full compliance with Title 35 of the United States Code.

13. On information and belief, Defendant has infringed and continues to infringe

one or more claims, including at least Claim 1, of the '546 Patent by making, using, importing, selling, and/or offering for sale a software platform for embedded software development, which is covered by at least Claim 1 of the '546 Patent. Defendant has infringed and continues to infringe the '546 Patent directly in violation of 35 U.S.C. § 271.

14. Defendant, sells, offers to sell, and/or uses embedded software development packages including, without limitation, the Libelium Waspmote Integrated Development Environment (IDE), and any similar products ("Product"), which infringe at least Claim 1 of the '546 Patent.

15. Libelium's Waspmote IDE is a software development kit which is used for multiple operating systems (such as Linux, Windows and Mac) and used for writing and uploading code to Waspmote LoRaWAN hardware modules and Waspmote Plug and Sense. Waspmote IDE provides one or more generic application handler programs, each such program comprising computer program code for performing generic application functions common to multiple types of hardware modules used in a communication system. For example, Waspmote IDE provides one or more generic application handler programs, such as Waspmote API, which include files such as pre-loaded libraries for Sketches (Waspmote code files). Waspmote libraries include source code comprising functions and data structure, which are common and uniform across all supported Waspmote hardware modules. Certain elements of this limitation are illustrated in the screenshots below and in the screenshots referenced in connection with other elements herein.



Source:
http://www.libelium.com/development/waspmote/documentation/ide-guide-v12/

Waspmote Hardware

2. Hardware

2.1. Specifications

The LoRaWAN module is managed via UART and it can be connected to SOCKET0 or SOCKET1.

2.1.1. LoRaWAN EU

The main features of the module are listed below:

- **Protocol:** LoRaWAN 1.0, Class A
- **LoRaWAN-ready**
- **Frequency:** EU 863-870 MHz and EU 433 MHz ISM frequency bands.
- **TX power:** up to +14 dBm
- **Sensitivity:** down to -136 dBm
- **Range:** >15 km at suburban and >5 km at urban area. Typically, each base station covers some km. Check the LoRaWAN Network in your area.
- **Chipset consumption:** 38.9 mA
- **Radio bit rate:** from 250 to 5470 bps
- **Receiver:** purchase your own base station or use networks from LoRaWAN operators

Source: http://www.libelium.com/downloads/documentation/waspmote-lorawan-networking-guide.pdf, page 10.

1. Introduction

The aim of this Guide is to introduce the user to the new Integrated Development Environment for Waspmote (Waspmote IDE). This Integrated Development Environment (IDE) is used for writing the code and uploading it to Waspmote and Waspmote Plug & Sense!. It also used to monitor serial output and for debugging. This IDE contains the Waspmote API (the API is the set of all libraries Waspmote needs for compiling programs). New API versions are released instantly by Libelium whenever improvements are made or bugs fixed.

This manual is **only intended** for Waspmote IDE versions higher or equal to v03, and Waspmote API versions higher or equal to v003. Waspmote IDE versions higher or equal to v03 are valid **only for** Waspmote PRO (v1.2) and Waspmote Plug & Sense!.

1.1. New features

There are several benefits to use the new Waspmote IDE:

- Faster compilation
- New API structure
- More debug messages
- RAM memory used information
- Easier installation
- More preferences
- 30+ different languages
- New automatic updates
- OTA compatibility
- Scrollable editor tabs

We want to specially thank all developers that have given us their feedback to improve the Development Environment. We hope developers can take advantage of all these new features.

Source: http://www.libelium.com/downloads/documentation/v12/waspmote_ide_user_guide.pdf, page 3.

2. Installation

This section explains how to install the Waspmote v12/Pro Integrated Development Environment (Waspmote IDE in short) in your Operating System.

First, download the Waspmote v12 IDE from the Waspmote software webpage:

http://www.libelium.com/development/waspmote/sdk_applications/

and select your OS:

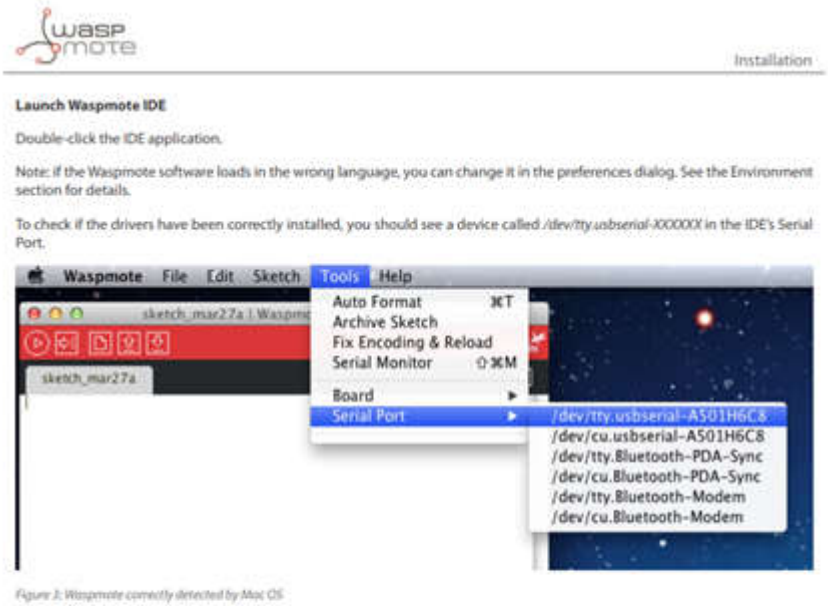
» Waspmote Pro IDE - v.XX

The Waspmote Pro IDE is Waspmote's software development kit. It is used for writing and uploading code to Waspmote. It also can monitor the serial output and be used in debugging.

Linux 32 bits | Linux 64 bits | Mac OS | Windows

Figure 1: Waspmote IDE download links

Source: http://www.libelium.com/downloads/documentation/v12/waspmote_ide_user_guide.pdf, page 4.



Installation

Launch Waspmote IDE

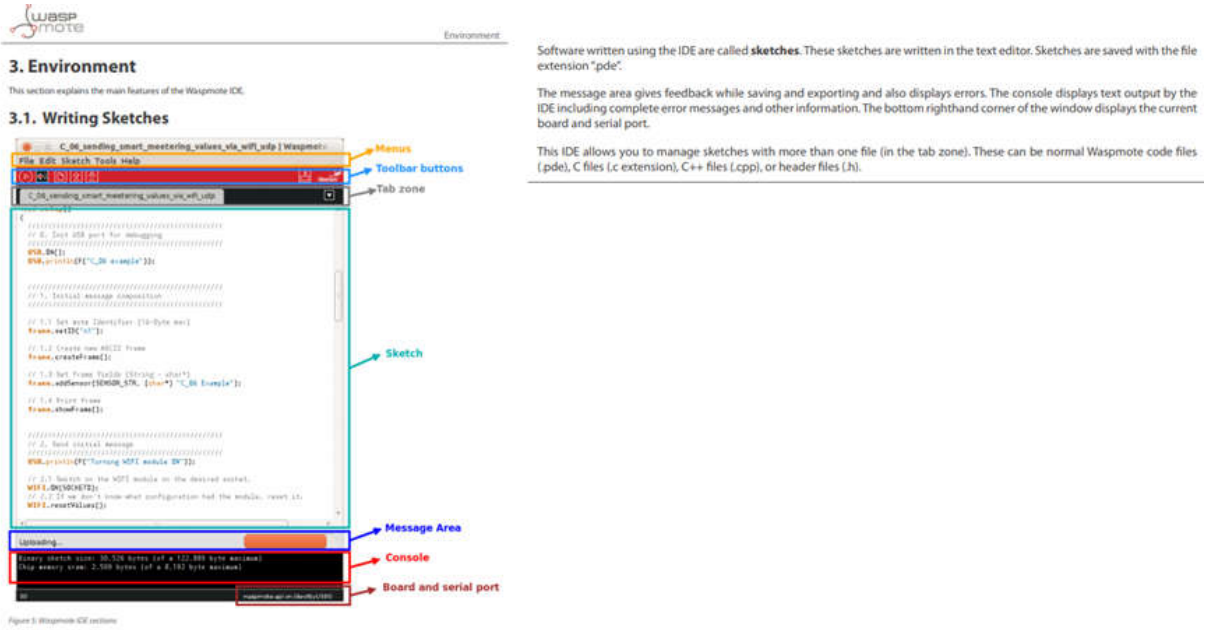
Double-click the IDE application.

Note: if the Waspmote software loads in the wrong language, you can change it in the preferences dialog. See the Environment section for details.

To check if the drivers have been correctly installed, you should see a device called */dev/tty.usbserial-XXXXXX* in the IDE's Serial Port.

Figure 3: Waspmote correctly detected by Mac OS

Source: http://www.libelium.com/downloads/documentation/v12/waspmote_ide_user_guide.pdf, page 6.



Environment

3. Environment

This section explains the main features of the Waspmote IDE.

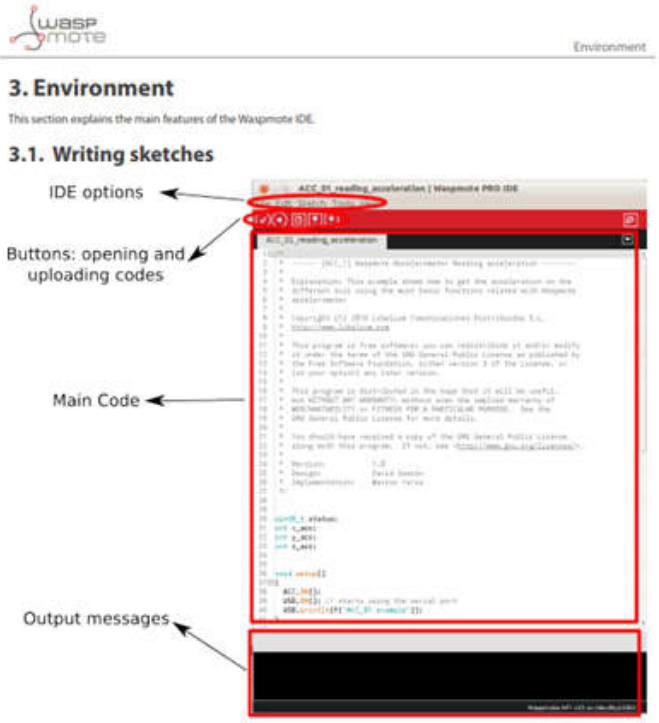
3.1. Writing Sketches

Software written using the IDE are called **sketches**. These sketches are written in the text editor. Sketches are saved with the file extension ".pde".

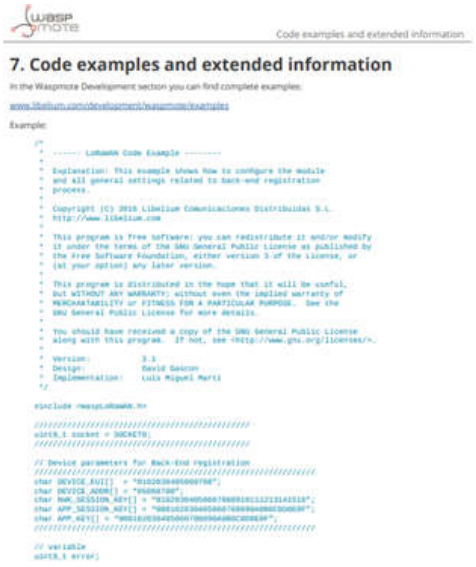
The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the IDE including complete error messages and other information. The bottom righthand corner of the window displays the current board and serial port.

This IDE allows you to manage sketches with more than one file (in the tab zone). These can be normal Waspmote code files (.pde), C files (.c extension), C++ files (.cpp), or header files (.h).

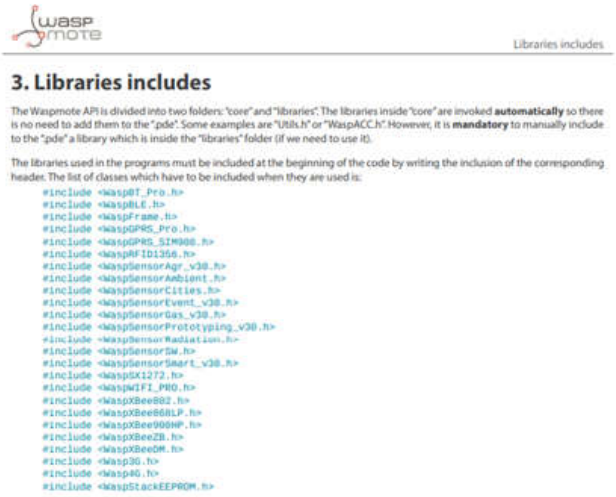
Source: http://www.libelium.com/downloads/documentation/v12/waspmote_ide_user_guide.pdf, page 8.



Source: http://www.libelium.com/downloads/documentation/waspmote_ide_user_guide.pdf, page 8.



Source: http://www.libelium.com/downloads/documentation/waspmote-lorawan-networking-guide.pdf, page 61.



Source: http://www.libelium.com/downloads/documentation/waspmote_programming_guide.pdf, page 6.

3.4. Libraries

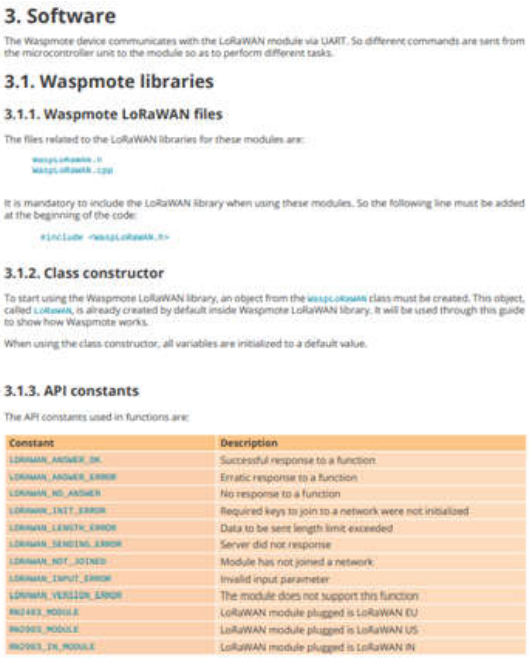
Libraries provide extra functionality for use in sketches. To use a library in a sketch, select it from the *Sketch → Import Library* menu. This will insert one or more #include statements at the top of the sketch and compile the library with your sketch. There is a list of libraries in the reference. Some libraries are included with the Waspmote API. Others can be downloaded from a variety of sources.

To install a third-party library: Libraries are often distributed as a ZIP file or folder. The name of the folder is the name of the library. Inside the folder will be a .cpp file, a .h file and often a keywords.txt file, examples folder, and other files required by the library.

To install the library, first quit the Waspmote IDE. Then uncompress the ZIP file containing the library. It should contain a folder called MyLibrary, with files like MyLibrary.cpp and MyLibrary.h inside. If the .cpp and .h files aren't in a folder, you'll need to create one. In this case, you'd make a folder called "MyLibrary". Drag the MyLibrary folder into the IDE libraries folder. There may be more files than just the .cpp and .h files, just make sure they're all there. Restart the IDE. Make sure the new library appears in the *Sketch → Import Library* menu item.

Note: If you are not advanced user, we recommend to use only Libelium's official libraries. Other libraries can cause damages to the electronics, and their use is obviously out of the warranty scope.

Source: http://www.libelium.com/downloads/documentation/v12/waspmote_ide_user_guide.pdf, page 9.



3. Software

The Waspmote device communicates with the LoRaWAN module via UART. So different commands are sent from the microcontroller unit to the module so as to perform different tasks.

3.1. Waspmote libraries

3.1.1. Waspmote LoRaWAN files

The files related to the LoRaWAN libraries for these modules are:

WaspLoRaWAN.h
WaspLoRaWAN.cpp

It is mandatory to include the LoRaWAN library when using these modules. So the following line must be added at the beginning of the code:

#include <WaspLoRaWAN.h>

3.1.2. Class constructor

To start using the Waspmote LoRaWAN library, an object from the WaspLoRaWAN class must be created. This object, called LoRaWAN, is already created by default inside Waspmote LoRaWAN library. It will be used through this guide to show how Waspmote works.

When using the class constructor, all variables are initialized to a default value.

3.1.3. API constants

The API constants used in functions are:

Constant	Description
LORAWAN_ANSWER_OK	Successful response to a function
LORAWAN_ANSWER_ERROR	Erratic response to a function
LORAWAN_NO_ANSWER	No response to a function
LORAWAN_INIT_ERROR	Required keys to join to a network were not initialized
LORAWAN_LENGTH_ERROR	Data to be sent length limit exceeded
LORAWAN_SENDING_ERROR	Server did not response
LORAWAN_NOT_JOINED	Module has not joined a network
LORAWAN_INPUT_ERROR	Invalid input parameter
LORAWAN_VERSION_ERROR	The module does not support this function
RN2483_MODULE	LoRaWAN module plugged is LoRaWAN EU
RN2903_MODULE	LoRaWAN module plugged is LoRaWAN US
RN2903_IN_MODULE	LoRaWAN module plugged is LoRaWAN IN

Source: http://www.libelium.com/downloads/documentation/waspmote-lorawan-networking-guide.pdf, page 19.

16. Waspmote IDE generates specific application handler code to associate the generic application functions with specific functions of a device driver for at least one of the types of the hardware modules. For example in addition to the generic drivers and Waspmote

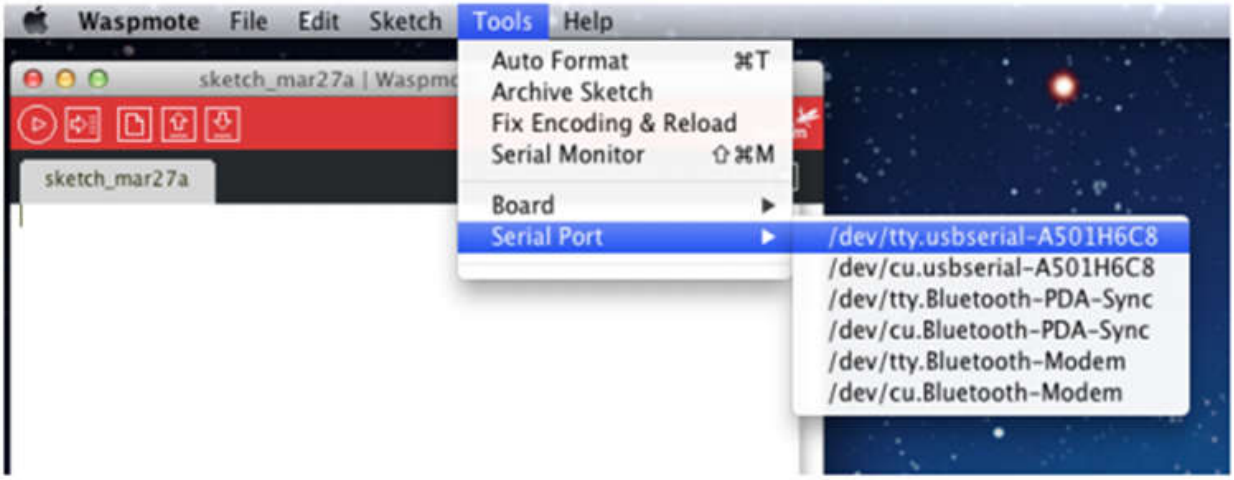
API, Waspmote IDE also includes specific application handler code which is specific to the application (such as application on Windows, Linux and iOS etc.) and specific to particular Waspmote hardware modules such as LoRaWAN modules, Sigfox module, WiFi PRO module etc. Certain elements of this limitation are illustrated in the screenshots below and in the screenshots referenced in connection with other elements herein.

Launch Waspmote IDE

Double-click on the IDE application.

Note: If the Waspmote software loads in the wrong language, you can change it in the preferences menu. See the "Environment" section for more details.

To check if the drivers have been correctly installed, you should see a device called */dev/tty.usbserial-XXXXXX* in the IDE's Serial Port.

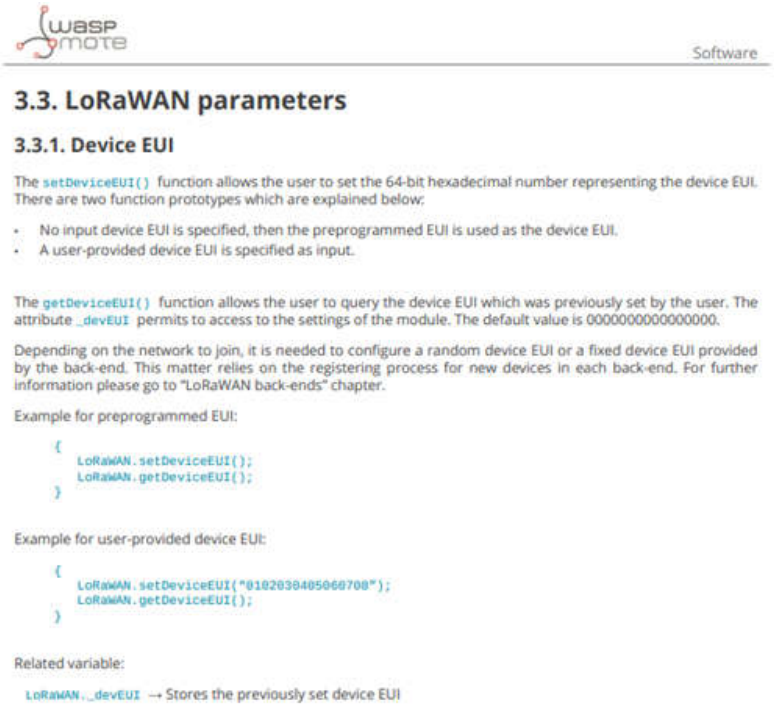


Source: http://www.libelium.com/downloads/documentation/waspmote_ide_user_guide.pdf, page 6.

```
////////////////////////////////////////////////
// 2. Reset to factory default values
////////////////////////////////////////////////

error = LoRaWAN.factoryReset();

// Check status
if( error == 0 )
{
  USB.println(F("2. Reset to factory default values OK"));
}
else
{
  USB.print(F("2. Reset to factory error = "));
  USB.println(error, DEC);
}
```

```
////////////////////////////////////////////////
// 3. Set/Get Device EUI
////////////////////////////////////////////////

// Set Device EUI
error = LoRaWAN.setDeviceEUI(DEVICE_EUI);

// Check status
if( error == 0 )
{
  USB.println(F("3.1. Set Device EUI OK"));
}
else
{
  USB.print(F("3.1. Set Device EUI error = "));
  USB.println(error, DEC);
}

// Get Device EUI
error = LoRaWAN.getDeviceEUI();

// Check status
if( error == 0 )
{
  USB.print(F("3.2. Get Device EUI OK. "));
```

Source: http://www.libelium.com/downloads/documentation/waspmote-lorawan-networking-guide.pdf, page 62.



wasp mote Software

3.3. LoRaWAN parameters

3.3.1. Device EUI

The `setDeviceEUI()` function allows the user to set the 64-bit hexadecimal number representing the device EUI. There are two function prototypes which are explained below:

- No input device EUI is specified, then the preprogrammed EUI is used as the device EUI.
- A user-provided device EUI is specified as input.

The `getDeviceEUI()` function allows the user to query the device EUI which was previously set by the user. The attribute `_devEUI` permits to access to the settings of the module. The default value is 0000000000000000.

Depending on the network to join, it is needed to configure a random device EUI or a fixed device EUI provided by the back-end. This matter relies on the registering process for new devices in each back-end. For further information please go to "LoRaWAN back-ends" chapter.

Example for preprogrammed EUI:

```
{
  LoRaWAN.setDeviceEUI();
  LoRaWAN.getDeviceEUI();
}
```

Example for user-provided device EUI:

```
{
  LoRaWAN.setDeviceEUI("0102030405060708");
  LoRaWAN.getDeviceEUI();
}
```

Related variable:

`LoRaWAN._devEUI` → Stores the previously set device EUI

Source: http://www.libelium.com/downloads/documentation/waspmote-lorawan-networking-guide.pdf, page 24.

17. Waspmote IDE generates specific application handler code and defines a

specific element in the specific code to be handled by one of the generic application functions for that hardware module. For example, Waspmote IDE generates system-specific application handler code by defining specific elements such as functions and data structures corresponding to specific hardware modules (such as LoRaWAN modules, Sigfox module, WiFi PRO module etc.) that extend or otherwise connect the system-specific application handler code and data structures made available by the generic application handler code of the Waspmote IDE. Certain elements of this limitation are illustrated in the screenshots below and in the screenshots referenced in connection with other elements herein.

3.3.5. Application EUI

The `setAppEUI()` function allows the user to set the 64-bit hexadecimal number representing the application identifier. This parameters is a global application identifier that uniquely identifies the application provider (i.e., owner) of the module.

Example of use:

```
{
    LoRaWAN.setAppEUI("1112131415161718");
}
```

Related variable:

`LoRaWAN._appEUI` → Stores the previously set application EUI

Source: https://ip.Semtech.com/uploads/103/SWdev-pdf, page 7.

4.4.3. Waspmote programming

LORIOT.io portal provides by default the whole configuration required by the module to connect to a network.

The mandatory field to field are the ones which where mentioned in the previous section. They can be copied as is to the code so module is configured the same way it was created in the portal.

Inside the device description in the LORIOT.io portal they warn to copy **EUI** and **Address** in little endian format, but it won't be necessary for the Waspmote code, it can be copied big endian format.

Examples of setting configuration necessary to connect into a network and send packets:
www.libelium.com/development/waspmote/examples/lorawan-02-send-unconfirmed
www.libelium.com/development/waspmote/examples/lorawan-03-send-confirmed

Source: http://www.libelium.com/downloads/documentation/waspmote-lorawan-networking-guide.pdf, page 58.

4.4. LORIOT

LORIOT.io is a provider of a LoRaWAN Network Server and Application Server software, which is commercially offered through a set of business models.

They provide:

- Software for the supported LoRa gateways
- Cloud-based LoRaWAN Network Server
- Programming interface (APIs) for Internet of Things applications to access the end node data
- Output of end node data to number of 3rd party services

As a gateway owner, users can use LORIOT.io software on gateways to connect them to their cloud. From then on, all data received by the gateways will be relayed to the user through the LORIOT.io APIs or 3rd party services.

The network servr components fulfills to role of protocol processor. It is a TLS connection end-point for the gateways and the customer applications. It is responsible for processing the incoming end node data according to the LoRaWAN protocol.

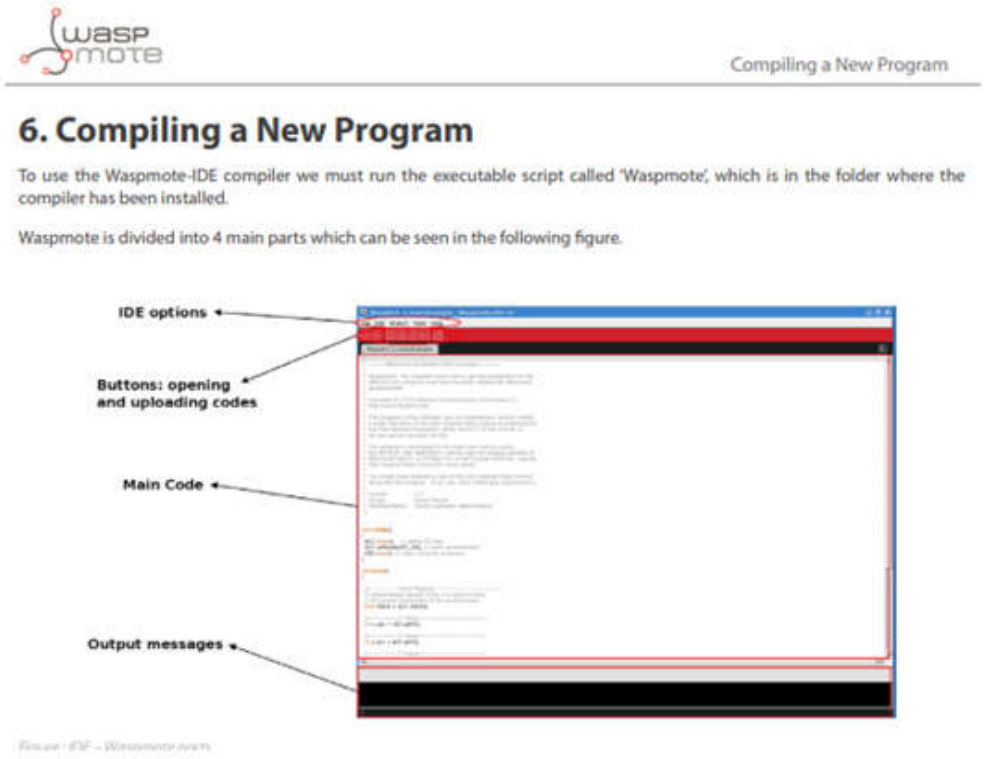
The specific roles of LORIOT.io Network Server are:

- Gateway population management
- Application population management
- Device population management
- Collection of billing records
- Security management
- Data distribution

Source: https://ip.Semtech.com/uploads/103/SWdev-pdf, page 1-2.

18. Waspmote IDE compiles the generic application handler programs together

with the specific application handler code to produce machine-readable code to be executed by an embedded processor in the at least one of the types of the hardware modules. For example, when a specific application is needed for a particular hardware, the generic functions and the specific functions are compiled together to yield a machine readable code. Waspmote IDE and/or its customers compile the generic functions and the specific functions using Waspmote IDE and/or any other compiling SDK supported by Semtech. Certain elements of this limitation are illustrated in the screenshots below and in the screenshots referenced in connection with other elements herein.



Source: http://www.libelium.com/downloads/documentation/quickstart_guide.pdf, page 13.

- The first part is the menu which allows configuration of general parameters such as the selected serial port.
- The second part is a button menu which allows verification, opening, saving or loading the selected code on the board.
- The third part contains the main code which will be loaded to Waspmote.
- The fourth part shows us the possible compilation and load errors, as well as the success messages if the process is carried out correctly.

The Waspmote-IDE buttons panel allows certain functions to be carried out such as opening a previously saved code, creating a new one or loading the code on the board. The following figure shows the panel and the functions of each button.

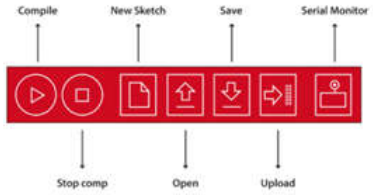
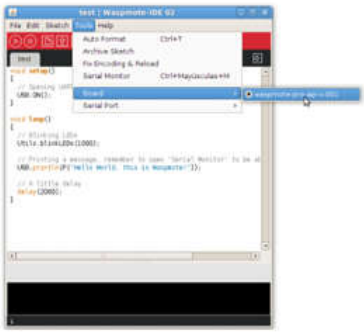


Figure: IDE - Waspmote panel of buttons

Compiling a New Program

Once the program has been opened correctly some configuration changes must be made so that the programs load correctly in Waspmote.

In the **'Tools/Board'** tab the Waspmote board must be selected. This refers to the API selected.



Source: http://www.libelium.com/downloads/documentation/quickstart_guide.pdf, page 13-14.

In the **'Tools/Serial Port'** tab, the USB to which Waspmote has been connected to the computer must be selected.

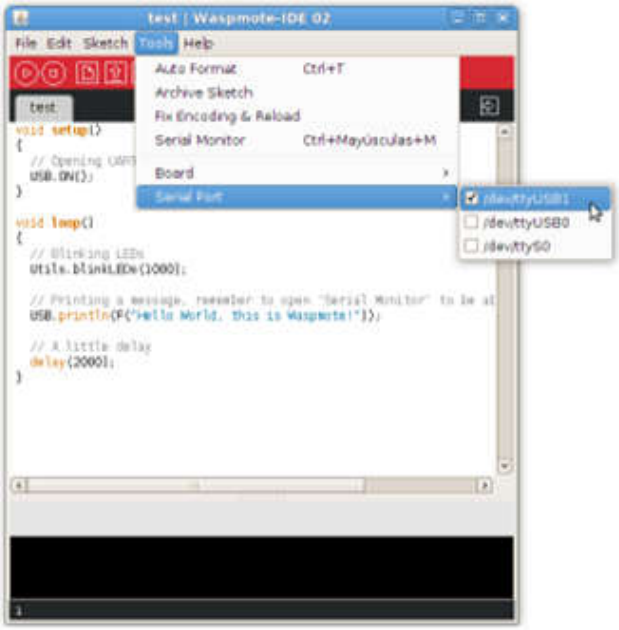
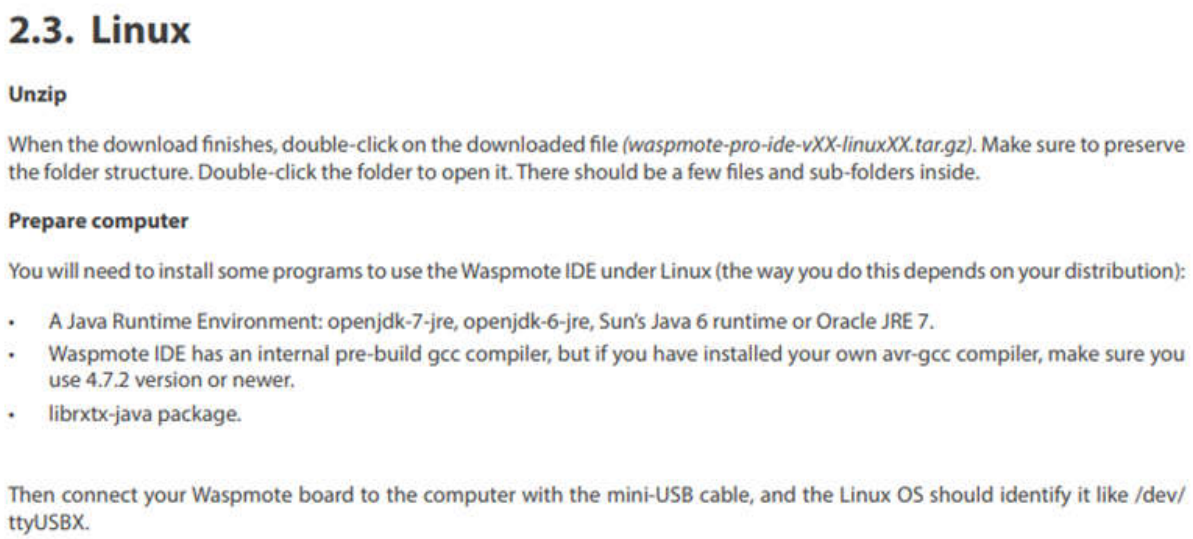


Figure : Select USB port

Source: http://www.libelium.com/downloads/documentation/quickstart_guide.pdf, page 14.



2.3. Linux

Unzip

When the download finishes, double-click on the downloaded file *(waspmote-pro-ide-vXX-linuxXX.tar.gz)*. Make sure to preserve the folder structure. Double-click the folder to open it. There should be a few files and sub-folders inside.

Prepare computer

You will need to install some programs to use the Waspmote IDE under Linux (the way you do this depends on your distribution):

- A Java Runtime Environment: openjdk-7-jre, openjdk-6-jre, Sun's Java 6 runtime or Oracle JRE 7.
- Waspmote IDE has an internal pre-build gcc compiler, but if you have installed your own avr-gcc compiler, make sure you use 4.7.2 version or newer.
- librxtx-java package.

Then connect your Waspmote board to the computer with the mini-USB cable, and the Linux OS should identify it like /dev/ttyUSBX.

Source: http://www.libelium.com/downloads/documentation/waspmote_ide_user_guide.pdf, page 6.

19. Defendant's actions complained of herein will continue unless Defendant is enjoined by this court.

20. Defendant's actions complained of herein are causing irreparable harm and monetary damage to Plaintiff and will continue to do so unless and until Defendant is enjoined and restrained by this Court.

21. Plaintiff is in compliance with 35 U.S.C. § 287.

PRAYER FOR RELIEF

WHEREFORE, Plaintiff asks the Court to:

(a) Enter judgment for Plaintiff on this Complaint on all causes of action asserted herein;

(b) Enter an Order enjoining Defendant, its agents, officers, servants, employees, attorneys, and all persons in active concert or participation with Defendant who receive notice of the order from further infringement of United States Patent No. 7,069,546 (or, in the

alternative, awarding Plaintiff a running royalty from the time of judgment going forward);

(c) Award Plaintiff damages resulting from Defendant's infringement in accordance with 35 U.S.C. § 284;

(d) Award Plaintiff pre-judgment and post-judgment interest and costs; and

(e) Award Plaintiff such further relief to which the Court finds Plaintiff entitled under law or equity.

Dated: January 31, 2019

Respectfully submitted,

/s/ Jay Johnson

JAY JOHNSON
State Bar No. 24067322
D. BRADLEY KIZZIA
State Bar No. 11547550
KIZZIA JOHNSON, PLLC
1910 Pacific Ave., Suite 13000
Dallas, Texas 75201
(214) 451-0164
Fax: (214) 451-0165
jay@kjpllc.com
bkizzia@kjpllc.com

ATTORNEYS FOR PLAINTIFF

EXHIBIT A