



Real-time Internet of Things – Wireless Motion Sensor

User Manual V1.2



---

### ATTENTION

Please read this manual before  
using your BITalino R-IoT

---

The information contained in this manual has been carefully checked and we made every effort to ensure its quality. PLUX reserves the right to make changes and improvements to this manual and products referenced at any time without notice.

Please check your kit after receiving and before using it the first time to confirm if it contains all the ordered sensors, accessories and other components. Contact our support if there are any variations from your original order.

**PLUX Wireless Biosignals S.A.**

email: [plux@plux.info](mailto:plux@plux.info)

web: <http://www.plux.info>

**Headquarters**

Zona Industrial das Corredouras, Lt. 14 – 1º

2630-369 Arruda dos Vinhos

Portugal

tel.: +351 263 978 572

**Lisbon Office**

Av. 5 de Outubro, nº 79 – 2º

1050-059 Lisboa

Portugal

tel.: +351 211 956 542

**Institut de Recherche et Coordination Acoustique/Musique (IRCAM)**

Web: <https://www.ircam.fr/>

1 Place Igor Stravinsky

75004 Paris

France

tel: +33 1 44 78 48 43

## TABLE OF CONTENTS

<b>1</b>	<b>Regulatory &amp; Legal Information .....</b>	<b>6</b>
1.1	Disclaimer .....	6
1.2	Guarantee of Quality & Warranty .....	6
1.2.1	Warranty Voidance .....	7
1.3	Contact & Support.....	7
<b>2</b>	<b>List of Acronyms .....</b>	<b>8</b>
<b>3</b>	<b>Introduction.....</b>	<b>9</b>
3.1	Overview .....	9
3.1.1	The CC3200 MCU .....	9
3.1.2	The Energia IDE .....	10
3.2	BITalino R-IoT.....	11
<b>4</b>	<b>Hardware Description.....</b>	<b>12</b>
4.1	Port A .....	13
4.2	Port B.....	14
4.3	Battery Considerations .....	14
<b>5</b>	<b>R-IoT Configuration &amp; Setup.....</b>	<b>15</b>
5.1	Module Default Configuration .....	15
5.2	Changing the Default Configuration .....	15
5.3	Network Connection Details .....	17
5.4	ID when using Multiple Devices .....	18
<b>6</b>	<b>Wi-Fi and Computer setup .....</b>	<b>19</b>
6.1	Change Computer IP.....	19
6.2	Configure the Wi-Fi Access Point and Router .....	23
6.2.1	How to find the router IP address. ....	23
6.2.2	Access the Wi-Fi Access Point for the first time .....	25
6.3	Local Network Conventions and Standards .....	26
<b>7</b>	<b>The OSC structure .....</b>	<b>28</b>
7.1	OSC Syntax.....	28
7.1.1	Atomic Data Types.....	28
7.1.2	OSC Packets .....	28
7.1.3	OSC Messages.....	29

---

7.1.4	OSC Bundles.....	29
<b>8</b>	<b>R-IoT Streaming, Sensor Fusion and Analysis.....</b>	<b>30</b>
8.1	Sensor Fusion.....	30
8.2	R-IoT Code Repository .....	30
8.3	Receive Sensors Data from the Module.....	30
8.4	Max Abstractions for the R-IoT Bitalino.....	32
<b>9</b>	<b>Programming the R-IoT.....</b>	<b>33</b>
9.1	Install the Energia IDE.....	33
9.2	Install the USB Serial Port Driver .....	33
9.3	Customize the IDE .....	33
9.4	Install the Firmware and Examples .....	34
9.5	Use Energia IDE .....	34
<b>10</b>	<b>Glossary .....</b>	<b>40</b>
<b>11</b>	<b>Bibliography .....</b>	<b>42</b>

## LIST OF FIGURES

<i>Figure 1-1 – Functional Block Diagram.....</i>	<i>9</i>
<i>Figure 1-2 – Energia IDE interface.....</i>	<i>10</i>
<i>Figure 2-1 – Top View.....</i>	<i>12</i>
<i>Figure 2-2 – Bottom View .....</i>	<i>12</i>
<i>Figure 2-3 – Port A Pinout detail.....</i>	<i>13</i>
<i>Figure 2-4 – Port B Pinout detail.....</i>	<i>14</i>
<i>Figure 3-1 – R-IoT Configuration.....</i>	<i>15</i>
<i>Figure 3-2 – AP Mode details.....</i>	<i>15</i>
<i>Figure 3-3 – AP SSID Example. ....</i>	<i>16</i>
<i>Figure 3-4 – R-IoT Webpage.....</i>	<i>17</i>
<i>Figure 4-1 – Change IP MacOS.....</i>	<i>19</i>
<i>Figure 4-2 – Change IP (Step1). ....</i>	<i>20</i>
<i>Figure 4-3 – Change IP (Step 2). ....</i>	<i>20</i>
<i>Figure 4-4 – Change IP (Step 3). ....</i>	<i>21</i>
<i>Figure 4-5 – Change IP (Step 4). ....</i>	<i>21</i>
<i>Figure 4-6 – Finding IP (Step1). ....</i>	<i>23</i>
<i>Figure 4-7 – Finding IP (Step 2). ....</i>	<i>24</i>
<i>Figure 4-8 – Finding IP (Step 3). ....</i>	<i>24</i>
<i>Figure 4-9 – TP-link MR3020 3G/Wi-Fi router. ....</i>	<i>25</i>
<i>Figure 4-10 – AP router Configuration.....</i>	<i>25</i>
<i>Figure 4-11 – AP router Configuration, SSID setup.....</i>	<i>26</i>
<i>Figure 5-1 – Max/MSP example.....</i>	<i>31</i>
<i>Figure 5-2 – Abstraction analysis-example.maxpat.....</i>	<i>32</i>
<i>Figure 6-1 – Blink Code example.....</i>	<i>35</i>
<i>Figure 6-2 – Selection of the Board. ....</i>	<i>36</i>
<i>Figure 6-3 – Communication Port configuration. ....</i>	<i>37</i>
<i>Figure 6-4 – Flashing firmware.....</i>	<i>38</i>

# 1 Regulatory & Legal Information

## 1.1 Disclaimer

BITalino R-IoT devices are intended for use in life science education and research applications only; they are not medical devices, nor medical software solutions, nor are they intended for medical diagnosis, cure, mitigation, treatment or prevention of disease and is provided to you “as is”.

We expressly disclaim any liability whatsoever for any direct, indirect, consequential, incidental or special damages, including, without limitation, lost revenues, lost profits, losses resulting from business interruption or loss of data, regardless of the form of action or legal theory under which the liability may be asserted, even if advised of the possibility of such damages.

## 1.2 Guarantee of Quality & Warranty

The BITalino R-IoT acquisition system has a two years warranty from the date of purchase. R-IoT sensors have three months warranty from the date of purchase. PLUX guarantees that the system, sensors and accessories will be free from material or manufacturing defects for the mentioned time periods following date of purchase.

If PLUX receives a notification of any such defects within the warranty period, it will repair or replace with the same unit/model, any products with proven defects at no cost to the client. During the repair period PLUX promises to provide a temporary replacement under the same specification. Repairs will be carried out at PLUX's premises after the equipment has been received.

### 1.2.1 Warranty Voidance

Usage of the device that is not in accordance with the handling instructions indicated in the manual or use with accessories other than those manufactured by PLUX will invalidate the warranty of your devices.

Be careful when connecting your BITalino R-IoT devices, sensors and/or accessories to any third party device including the usage of the third party connection components that are available for BITalino R-IoT systems as **the usage of these components will void the electrical warranty of your device** and sensors and, if not indicated otherwise, the warranty of the third party system you're connecting to the device. Check the electrical specifications of both systems you want to connect to prevent any damage to the user(s) or the systems.

### 1.3 Contact & Support

Contact us if you're experiencing any problems that cannot be solved with the information given in the BITalino R-IoT or manual. We'll get back to you as soon as possible to find the best solution for your problem.

Please send us an e-mail with precise information about the error occurrence, device configuration, and, if possible, screenshots of the problem to [support@plux.info](mailto:support@plux.info).



## 2 List of Acronyms

**AP** – Access Point

**DHCP** - Dynamic Host Configuration Protocol

**I2C** - Inter-Integrated Circuit

**IC** - Integrated Circuit

**IDE** - Integrated Development Environment

**IP** – Internet Protocol

**IMU** - Inertial Measurement Unit

**IoT** – Internet of Things

**JTAG** – Joint Test Action Group

**LAN** – Local Area Network

**MCU** – Microcontroller Unit

**OSC** - Open Sound Control

**SPI** – Serial Peripheral Interface

**SSID** - Service Set Identifier

**TCP/IP** - Internet Protocol Suite

**TI** – Texas Instruments

**UART** - Universal Asynchronous Receiver/Transmitter

**UDP** - User Datagram Protocol

**UPnP** – Universal Plug and Play

**WAN** – Wide Area Network

**WISH** – Wireless Intelligent Stream Handling

**WPS** – Wi-Fi Protected Setup

## 3 Introduction

The BITalino R-IoT module is the 7th generation of IRCAM's wireless sensor digitizing unit, an essential tool linking motion sensing, gestural recognition and Live Performance Arts.

Since 2001, IRCAM aimed to provide a low latency, high data-rate & resolution and stage compatible devices, capable of streaming gestural information to the computer from multiple performers.

The BITalino R-IoT embeds a 9-axis digital IMU sensor (LSM9DS1) [1] featuring a 3-axis accelerometer, a 3-axis gyroscope and a 3-axis magnetometer, allowing for instance the onboard computation of the absolute orientation of the module in space. The sensor is attached to the SPI port to sample the 16-bit motion data at high speed.

The core of the BITalino R-IoT module is based upon the CC3200 [2] chip from Texas Instruments (TI) and it is compatible with Energia [3], a programming tool for TI processors with the easiness of the Arduino [4] look & feel.

### 3.1 Overview

#### 3.1.1 The CC3200 MCU

Created for the Internet of Things (IoT), the SimpleLink WiFi CC3200 device is a wireless MCU that integrates a high-performance ARM®Cortex-M4 MCU allowing developers to build an entire application with a single IC. With on-chip Wi-Fi, internet and robust security protocols, no prior Wi-Fi experience is needed for faster development.

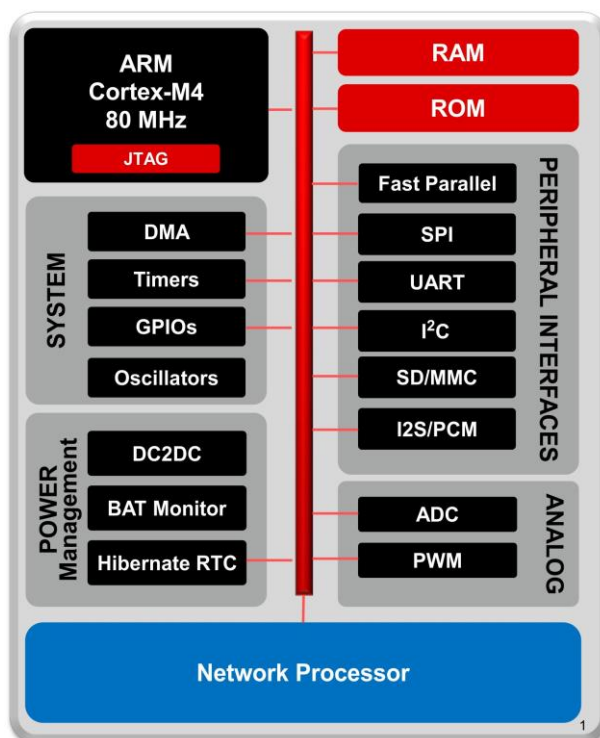


Figure 1-1 – Functional Block Diagram.

### 3.1.2 The Energia IDE

Energia is an open-source platform for electronics prototyping that bring the Wiring [5] and Arduino framework to the TI MSP430, MSP432x, TM4C, C2000, **CC32xx** and CC13xx MCUs based on the LaunchPad [6] board. The Energia IDE is cross platform and supported on Mac OS, Windows and Linux.

The foundation of Energia is the Wiring framework that was developed by Hernando Barragan [3]. That was thoughtfully created with designers and artists in mind to encourage a community where both beginners and experts from around the world share ideas, knowledge and their collective experience.



Figure 1-2 – Energia IDE interface.

## 3.2 BITalino R-IoT

The BITalino R-IoT is a slightly different version of the former R-IoT device from IRCAM's lab team ISMM. It's aimed to refine a few aspects of the original design to match with other populations of users, and some new use cases too.

The BITalino R-IoT was made compatible with the BITalino ecosystem and sensor platform as a Wi-Fi transmitter, bridge and data hub. It captures pre-digitized BITalino sensor data and packs it in an Open Sound Control (OSC) message in addition to the IMU data, which are sent in a separate OSC message (see [Glossary](#)).

In addition, a new form factor of the module allows for soldering additional sensors or accessories (e.g. switch, LED, piezo) to expand its possibilities. Finally, an RGB LED was added for an enhanced visual interaction with the user, easing the boot & connection sequence (Wi-Fi connection, streaming). Furthermore, you can facilitate your experience with the R-IoT by purchasing a pre-configured TP-Link router with all the necessary parameters to ease the connection of your BITalino R-IoT without any additional intervention needed (<https://plux.info/accessories/403-wireless-route-bitalino-riot-810121713.html>). If you prefer to configure a network yourself or change the settings on the device, we advise you to read the following chapters.

The BITalino R-IoT exports only some of the available I/O's from the CC3200 for matters of size form factor. It remains small enough to be installed anywhere: on the body, on an instrument or in an object and it can be adapted to various contexts.

An I2C bus is also exported allowing for a complete chain of accessories to be talked to, from LED controllers to additional sensors or I/O extenders, it also embeds all the features from the original design.

Lastly, the BITalino R-IoT edition is meant to be used either as a replacement of the Bluetooth module provided with the BITalino sensor board or in standalone mode (as a wireless Wi-Fi IMU). The pinout of the module reflects this specificity compared to the original R-IoT module and features a 6 pin 2.54mm header that provides the power supply and serial communication to the BITalino board or to a FTDI USB serial cable (with crossed / null-modem wiring).

To ease the reading of this document, we will refer from now to the BITalino R-IoT as **R-IoT**.

## 4 Hardware Description

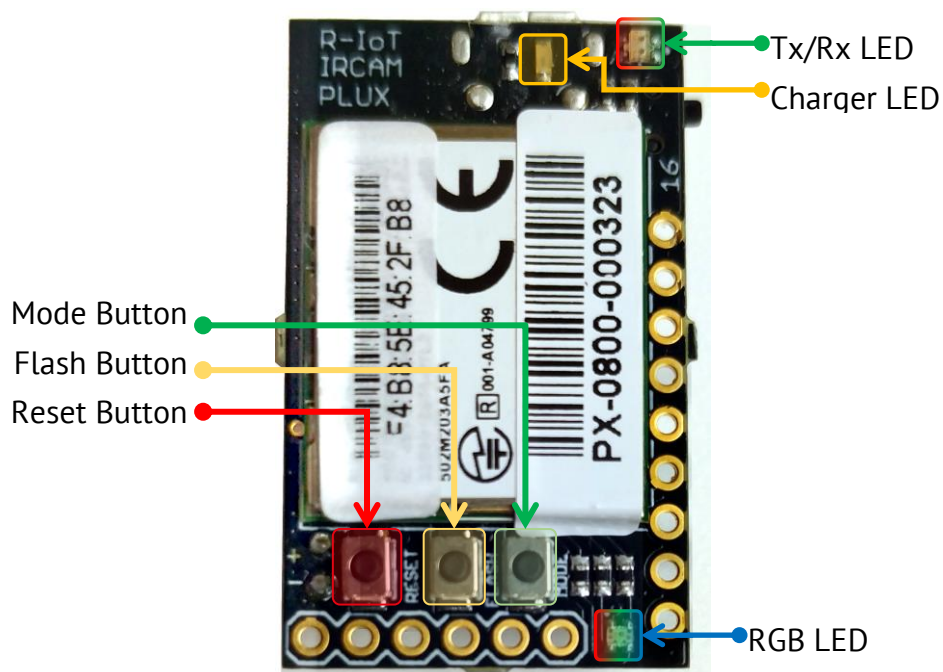


Figure 2-1 – Top View

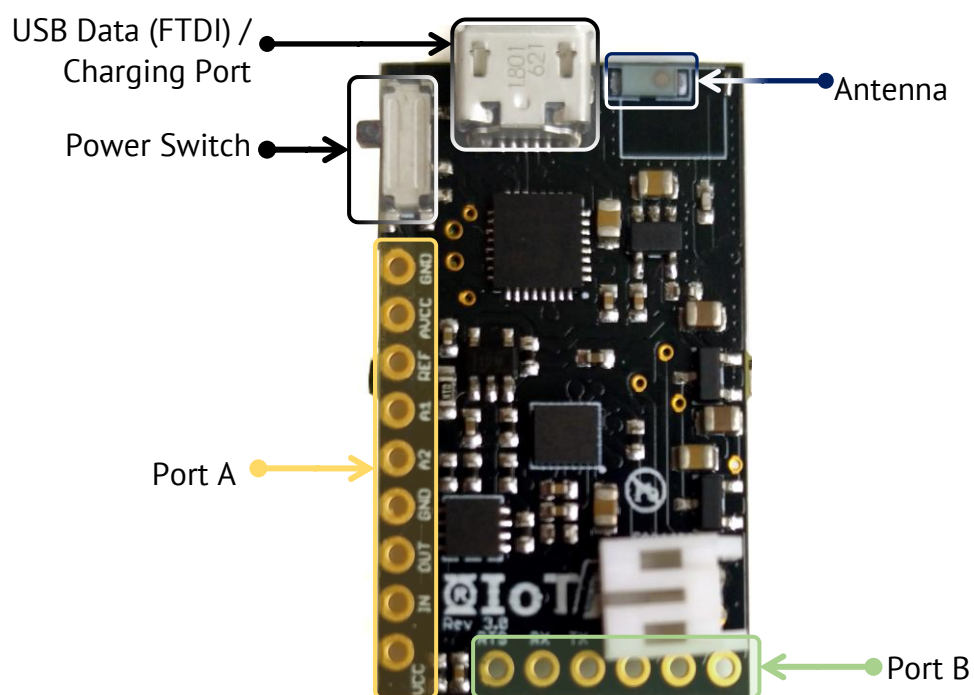


Figure 2-2 – Bottom View

## 4.1 Port A



Figure 2-3 – Port A Pinout detail

Reference	Pin Alias	Signal Name (CC3200)	Signal Description
VCC	V <sub>CC</sub>	-	3.3V General Power Source.
IN	GPIO_13	GPIO_13	Digital Input.
OUT	GPIO_12	GPIO_12	Digital Output.
GND	GND	-	
A2	GPIO_05	ADC_CH3	12-bit ADC Input (1.5V <sub>Max</sub> ).
A1	GPIO_04	ADC_CH2	12-bit ADC Input (1.5V <sub>Max</sub> ). Can be selected as a “Wake-up Source”.
REF	V <sub>REF</sub>	-	1.65V Isolated Reference.
AVCC	AV <sub>CC</sub>	-	3.3V Sensors Power Source.
GND	GND	-	

Table 1 – Port A Pin Attributes

## 4.2 Port B



Figure 2-4 – Port B Pinout detail

Reference	Pin Alias	Signal Name (CC3200)	Signal Description
RTS	RESET	-	External Reset.
RX	GPIO_11	I2C_SDA <sup>(1)</sup>	I2C Data.
		UART1_RX <sup>(2)</sup>	UART RX Data.
TX	GPIO_10	I2C_SCL <sup>(1)</sup>	I2C Clock.
		UART1_TX <sup>(2)</sup>	UART TX Data.
DVCC	DV <sub>CC</sub>	-	3.3V Digital Power Source.
CTS	NC	-	Not Connected.
GND	GND	-	

(1) Config Mode = 7.

(2) Config Mode = 0.

Table 2 – Port B Pin Attributes

## 4.3 Battery Considerations

- Polymer Lithium Ion Battery
- Weight: 10g
- Size: 29 x 36 x 4.75 mm
- Output: 3.7V
- Capacity: 700mAh
- Connector: 2-pin JST should be charged using R-IoT via Battery connector on the PCB, using the provided USB cable.

## 5 R-IoT Configuration & Setup

### 5.1 Module Default Configuration

R-IoT's default firmware implements a Wi-Fi Access Point (AP mode) and a Web server (Station mode) to configure its wireless parameters such as the Wi-Fi network name (SSID), the Open Sound Control UDP port to send data to the recipient computer IP address.

The default SSID is "*riot*"; for a quick install, the simplest is to configure your router and computer rather than edit the R-IoT configuration in first place.

By default, the module will be sending OSC data (*The OSC message structure is described in [section 7](#)*) to the **DEST IP** (destination IP address) **192.168.1.100** on port **8888**. The remaining data can be seen on Figure 3-4.

IP:	192	.	168	.	1	.	40
DEST IP:	192	.	168	.	1	.	100
GATEWAY:	192	.	168	.	1	.	1
MASK:	255	.	255	.	255	.	0
PORT:	8888						
ID:	0						
SAMPLERATE:	5						

Figure 3-1 – R-IoT Configuration.

### 5.2 Changing the Default Configuration

To trigger the configuration mode, press the **Mode button** while the R-IoT is turned off. Then, turn the device on while holding down the mode button. The **RGB LED** will show a red light that will flash rapidly, then turns green and finally it will turn and remain blue, meaning that the device is in AP mode.

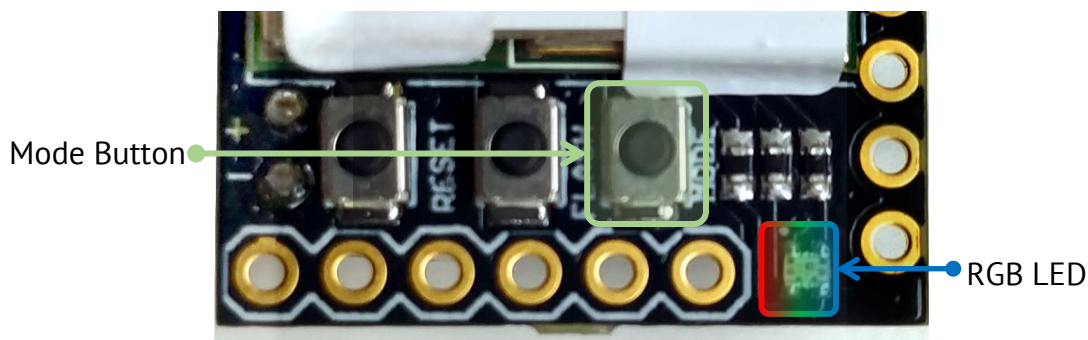
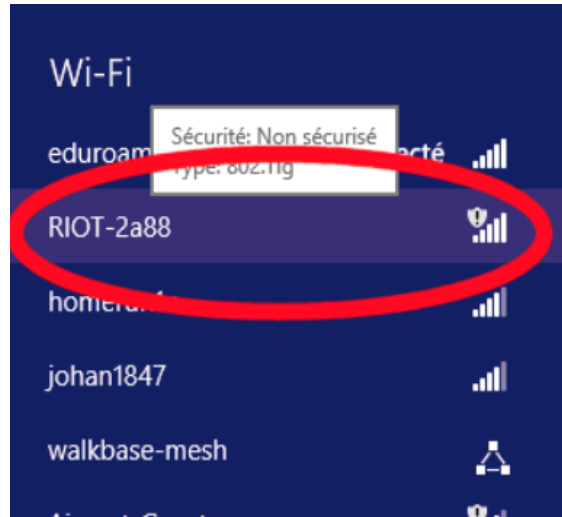


Figure 3-2 – AP Mode details.



The Wi-Fi network name can be easily identified in the list of available SSID and starts with RIOT (in uppercase) followed by a 4 character alphanumeric, as demonstrated in figure 3-3.



*Figure 3-3 – AP SSID Example.*

To continue, please follow these steps.

1. Set your computer as host ([Chapter 6](#)) and lookup the Wi-Fi network proposed by the module;
2. Connect to the network (no security needed);
3. Open your web browser and open the URL <http://192.168.1.1> (which is the pre-configured IP);

After these steps you should be redirected to the web page hosted by the module, like the one on [Figure 3-4](#), where you can configure its network behaviour & parameters.

# {SOUND MUSIC MOVEMENT} INTERACTION IRCAM - PLUX - 2017

## R-IoT Configuration Page

### Module Information

MAC: f4:b8:5e:45:2f:b8  
ID: 0  
Beta = 0.40  
Firmware: R-IoT Bitalino v2.041 - IRCAM-PLUX 2017-2018

### Network Configuration

WIFI MODE:	<input type="text" value="station"/>
IP TYPE:	<input type="text" value="DHCP"/>
SSID:	<input type="text" value="riot"/>
SECURITY:	<input type="text" value="None"/>
PASSWD:	<input type="text" value="12345678"/>
IP:	<input type="text" value="192"/> . <input type="text" value="168"/> . <input type="text" value="1"/> . <input type="text" value="40"/>
DEST IP:	<input type="text" value="192"/> . <input type="text" value="168"/> . <input type="text" value="1"/> . <input type="text" value="100"/>
GATEWAY:	<input type="text" value="192"/> . <input type="text" value="168"/> . <input type="text" value="1"/> . <input type="text" value="1"/>
MASK:	<input type="text" value="255"/> . <input type="text" value="255"/> . <input type="text" value="255"/> . <input type="text" value="0"/>
PORT:	<input type="text" value="8888"/>
ID:	<input type="text" value="0"/>
SAMPLERATE:	<input type="text" value="5"/>
<input type="button" value="Submit"/>	

Figure 3-4 – R-IoT Webpage.

### 5.3 Network Connection Details

The R-IoT connects to the network name set in the SSID field above. It should match the network name set in your Wi-Fi access point (see [Chapter 6](#)).

Most of the default settings should work with your Wi-Fi infrastructure. We suggest keeping the R-IoT in station mode (AP is provided by the infrastructure router).

If some security is absolutely needed, WPA-2 can be enabled along with a password or passphrase, however security uses more bandwidth and can be delicate to configure for new users. If the goal is to prevent unwanted access to the local network during sensor

streaming and collection, it is easier to setup a MAC address filtering in the Access Point itself.

In addition, the SSID beaconing can be disabled (also on the Access Point) to prevent finding the name of the network, therefore allowing connection only to parties who know the said SSID name.

In DHCP mode, there's no need to fill in the gateway and mask fields as those will be provided by their DHCP server.

The only other important parameters to be chosen are the recipient IP address (the IP of the computer receiving the data) and the UDP port.

#### **5.4 ID when using Multiple Devices**

The ID is used to ease the data routing when identifying the R-IoT in the software application. We suggest using both a port and ID based routing in order to ensure the proper identification of the data flow source. A simple module convention makes it easy to remember.

For example:

- R-IoT module #0 uses port #8000
- R-IoT module #1 uses port #8001
- R-IoT module #8 uses port #8008

In the case the software application has limited flexibility for choosing the UDP port (or when there's a single UDP port available for all input flows), the ID is then used to separate and route the different modules.

Once the parameters are all set in the configuration page, click on submit button to save the changes in the non-volatile memory of the R-IoT module, which can be rebooted to apply the new settings (turn the device off and then on again or press the reset switch).

## 6 Wi-Fi and Computer setup

As explained above, the R-IoT can work as out-of-the-shelf product when it is acquired with our TP-Link Wi-Fi router which have been pre-configured to work together.

The R-IoT can operate on both a local network and over the internet if necessary. As already mentioned above, the R-IoT will be sending data to its DEST IP (destination IP address), identified in the field of the configuration web page, [Figure 3-4](#), which is by default 192.168.1.100.

Therefore, a local network must be created using the AP router, and the proper destination address set on the computer.

### 6.1 Change Computer IP

On **MacOS**, go to the system preferences then network preferences. Select the Wi-Fi connection and connect to **Wi-Fi (riot)** instead of your home Wi-Fi router; change the IP address method to "manually" and set it to 192.168.1.100. Set the network mask to 255.255.255.0.

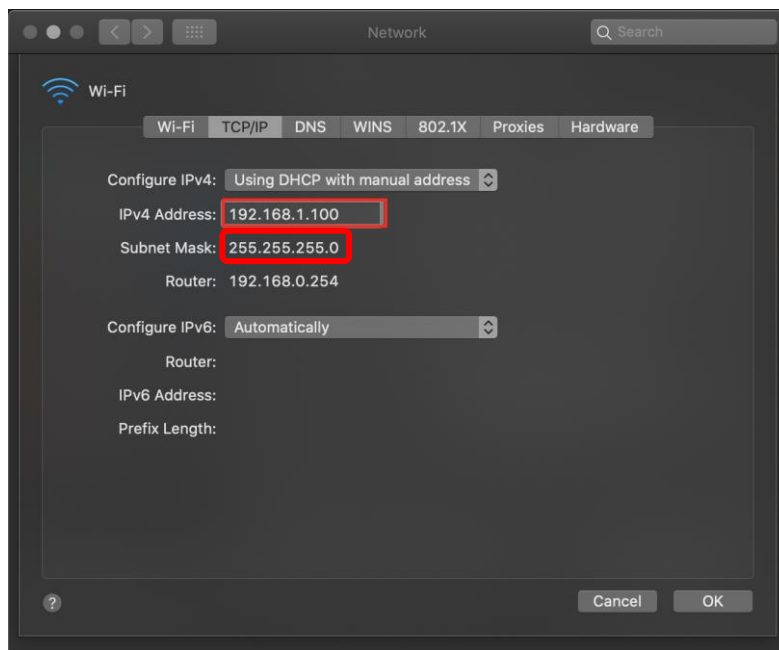
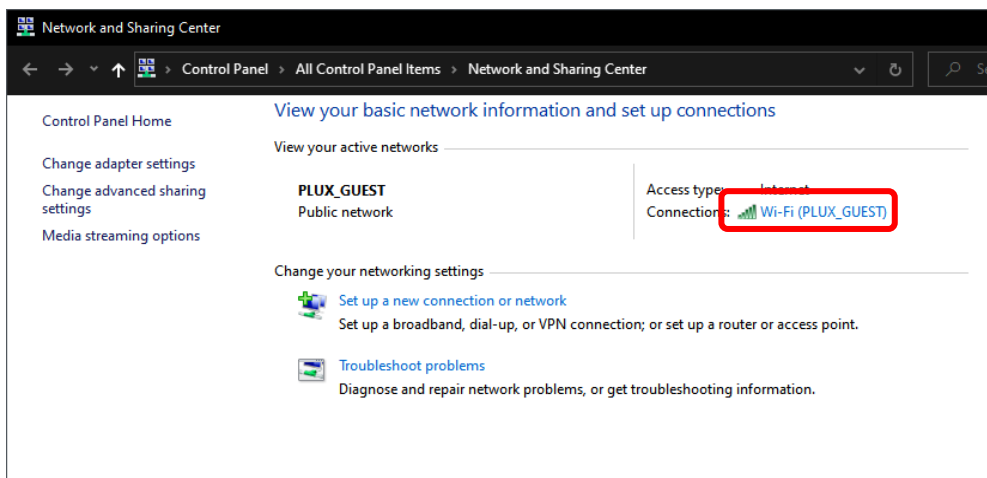


Figure 4-1 – Change IP MacOS

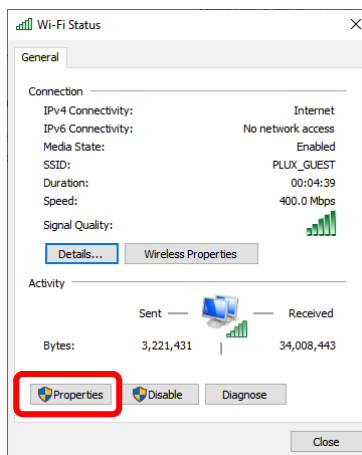
On **Windows**, the procedure is as follows:

1. Open the **Network and Sharing Center** and select **Wi-Fi**;



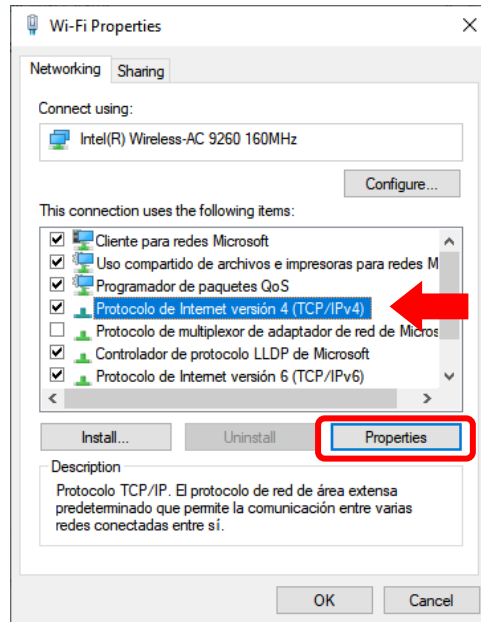
*Figure 4-2 – Change IP (Step1).*

2. A window will open. Then you need to select **Properties**;



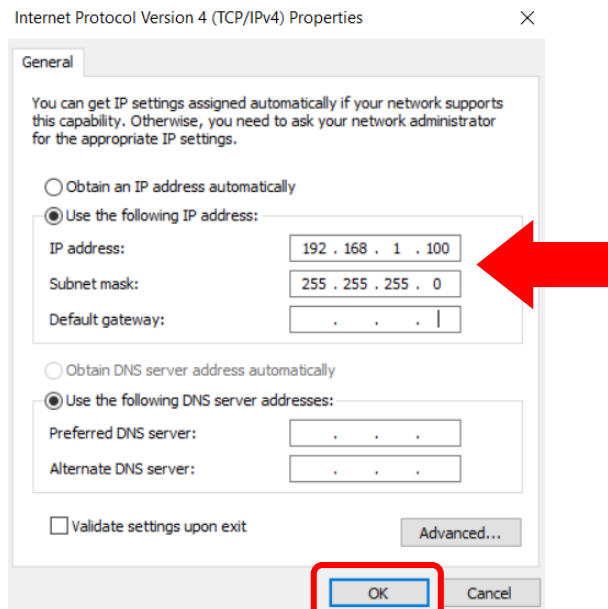
*Figure 4-3 – Change IP (Step 2).*

- Another window will show up. Now, select **Internet Protocol Version 4 (TCP/IPv4)** and go to **Properties**;



*Figure 4-4 – Change IP (Step 3).*

- The final step is to fill the final windows as shown in the next figure and click **OK** (the IP address should be the same as the DEST IP set in the configuration of the R-IoT);



*Figure 4-5 – Change IP (Step 4).*

- After finishing using the R-IoT, you should change back this setting to **Obtain an IP address automatically**.

On both operating systems, you can also use network profiles that allows quick switching between your regular internet configuration (i.e. using DHCP for most use cases) and a dedicated sensor configuration to use the R-IoT over a local network (using a manually set IP address). MacOS has a profile system in its network preferences. On Windows you might have to install a free software to manage the profiles, like **NetsetMan**.<sup>1</sup>

If you further need to change the destination IP, just make sure the R-IoT module is also configured to send to the right one. The selected IP address must match the network base address (192.168.1.XXX in our case) so that the OSC messages reach the recipient.

---

<sup>1</sup> For further information please visit <https://www.netsetman.com/en/freeware>.

## 6.2 Configure the Wi-Fi Access Point and Router

The router to be used as an access point is to be connected to a host router.

Many routers have the option to convert it to an access point by selecting just one option. Just select the Access Point mode and you are good to go.

Nevertheless, not all the routers include this feature and may have to be configured manually.

Take care of the following things before you configure your router to be an Access Point:

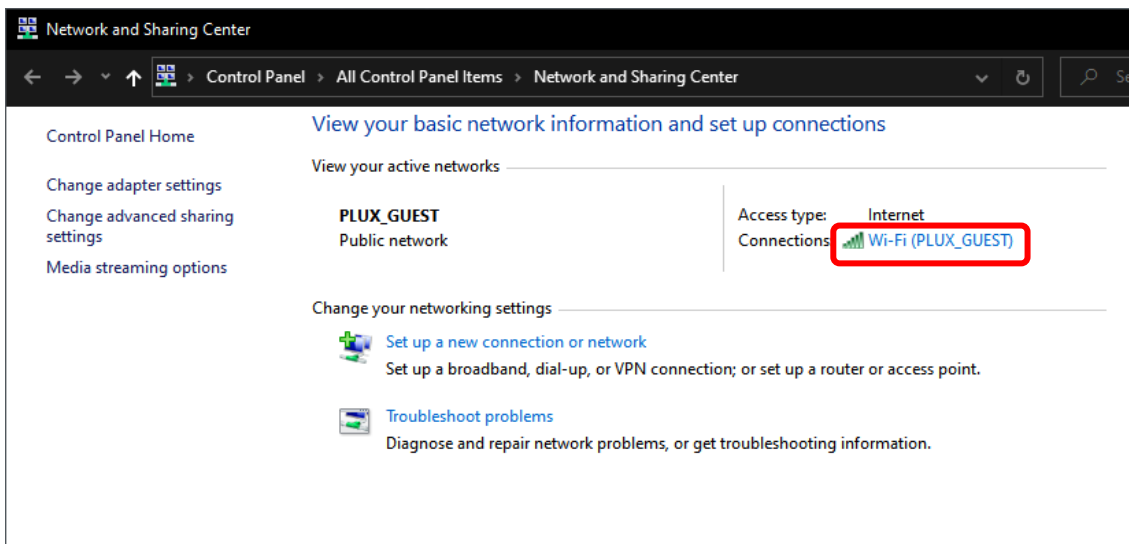
- Disable UPnP.
- Disable DHCP.
- Disable features that depend upon WAN: Virtual Server, SPI and Firewall, Application and Port Forward Rules, Access Control and Web Filters, Time, WISH and WPS.
- Change the IP address of the LAN on your network to an available address.

The essential information to know (provided by the user's manual of the device) is its IP address, so that you can connect to the device configuration page using your favourite browser.

### 6.2.1 How to find the router IP address.

On Windows, the procedure is as follows:

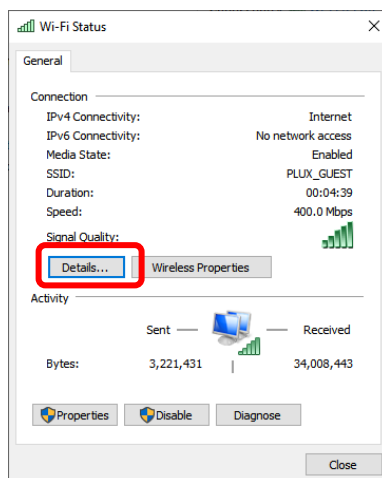
1. Open the **Network and Sharing Center** and select **Wi-Fi**;



*Figure 4-6 – Finding IP (Step1).*

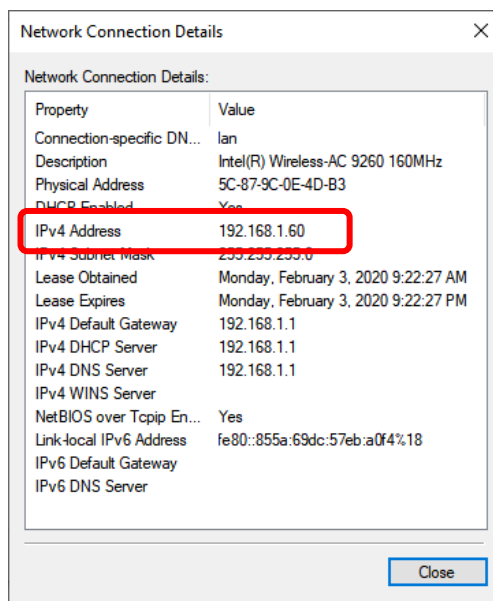


2. A window will open. Then you need to select **Details**;



**Figure 4-7 – Finding IP (Step 2).**

3. The recently open window will show all the **Network Connection Details**. The IP address will be displayed.



**Figure 4-8 – Finding IP (Step 3).**

Each brand or manufacturer tend to have its own default configuration. Further configuration steps may be specific to each Access Point and, thus, you should refer to the user manual of the one you own. Nonetheless, we will explain the procedure for the one we provide in our catalogue.

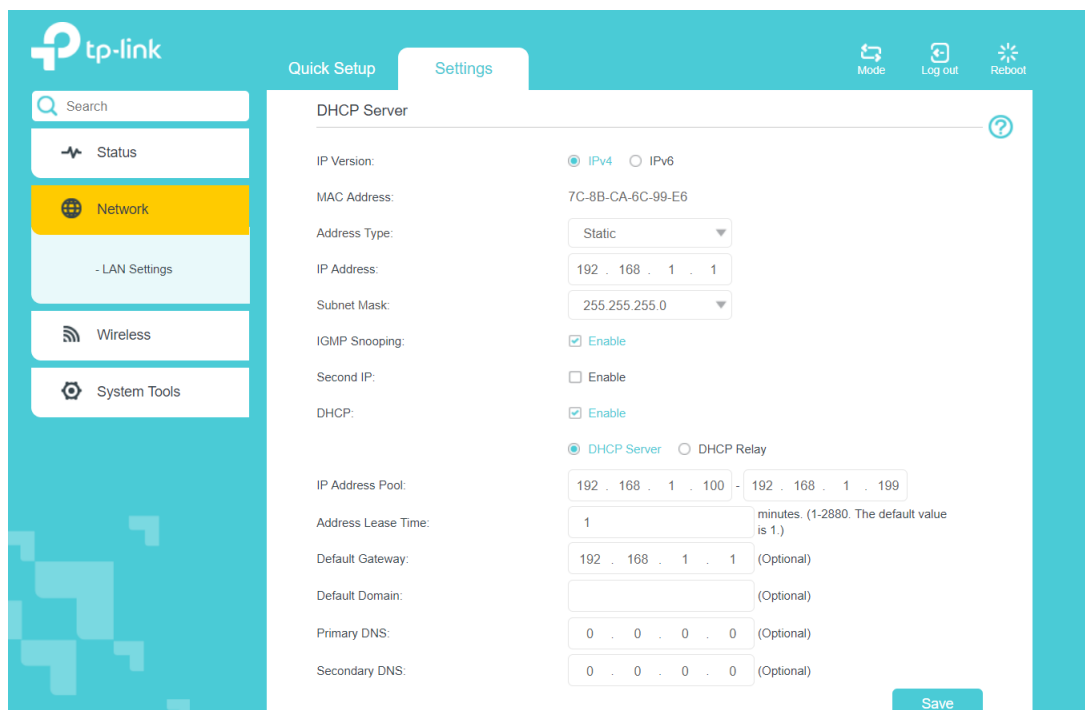
## 6.2.2 Access the Wi-Fi Access Point for the first time



**Figure 4-9 – TP-link MR3020 3G/Wi-Fi router.**

The configuration of the access point can be reached by connecting the computer to it, and then go to [tplinkwifi.net](http://tplinkwifi.net). This will prompt you for a login page where you will need to insert the pre-configured password that is on the user's manual of the Access Point.

The default configuration of the R-IoT is to connect to a computer with the IP address 192.168.1.100. This value can be set between 192.168.1.100 up to 192.168.1.199, as shown on the next figure:



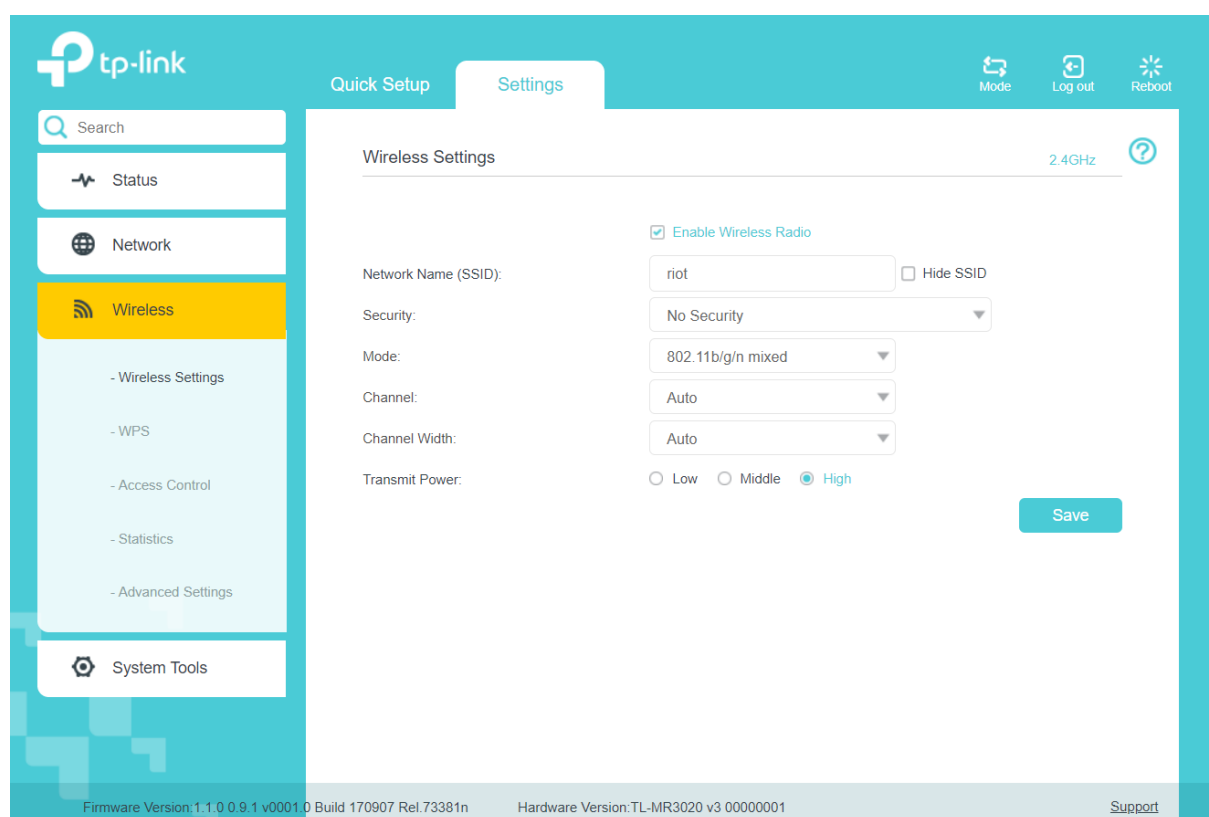
**Figure 4-10 – AP router Configuration.**

That will usually require to log in once again after a reboot of the AP.

Tune the DHCP settings. Most of the time the DHCP will be active on addresses from 192.168.1.100 to 192.168.1.199. Change this from 192.168.1.110 to 192.168.1.150, anything to avoid having the default destination IP address (192.168.1.100) to be in the DHCP lease pit. Alternatively, you can leave it in the DHCP area and reserve the address based on the MAC address of your computer (see DHCP address reservation).

Finally, go to the Wireless settings and change the SSID of your network. If you are using the R-IoT defaults, use **riot** as the network name.

Select a Wi-Fi channel that is possibly different than your neighbour. Finally, go to the security section and disable encryption (WPA-2 encryption is supported but not recommended on first use).



**Figure 4-11 – AP router Configuration, SSID setup.**

Once the router is configured with this 192.168.1.XXX address scheme and DHCP, the computer IP address can be set to 192.168.1.100 (as explained above), the default destination address the R-IoT module sends data to.

### 6.3 Local Network Conventions and Standards

This IP address configuration requires a few steps, but it is a onetime process. You've just created what is possibly your first local network (congratulations!) and moving between Internet and your sensors will soon become some easy gym!

There's no absolute standard on what to use as your IP address and local network IP scheme. "Our" IRCAM standard using a 192.168.1.100 recipient IP address inherits from the hardware we designed back in 2004 using Linksys Wi-Fi routers which were on 192.168.1.XXX by default.

Moreover, another classic network numbering uses the scheme 10.0.0.XXX or similar. The important thing is to understand your 3 devices (the R-IoT, the Wi-Fi access point and the computer) must be on the same scheme to be able to "see each other". In addition, the R-IoT must know to which address it has to send data to (the computer IP).

Ideally you will label your AP with a stiff sticker to remember its IP and find an easy way to remember the ID and port configuration of your R-IoT module.

Once all set, the best practice is to keep the AP and the R-IoT modules paired and stored together to avoid any mismatch.

## 7 The OSC structure

For a better understanding of the next sections of this manual, it is necessary to dedicate a few lines to the structure of the OSC messages.

### 7.1 OSC Syntax

Open Sound Control (OSC) is an open, transport-independent, message-based protocol developed for communication among computers, sound synthesizers, and other multimedia devices [7].

The syntax of OSC data is structured as follows:

#### 7.1.1 Atomic Data Types

All OSC data is composed of the following fundamental data types:

- **int32:** A 32-bit big-endian two's complement integer.
- **OSC-timetag:** A 64-bit big-endian fixed-point time tag, semantics defined below
- **float32:** A 32-bit big-endian IEEE 754 floating point number.
- **OSC-string:** A sequence of non-null ASCII characters followed by a null, followed by 0-3 additional null characters to make the total number of bits a multiple of 32. (OSC-string examples) In this document, example OSC-strings will be written without the null characters, surrounded by double quotes.
- **OSC-blob:** An int32 size count, followed by that many 8-bit bytes of arbitrary binary data, followed by 0-3 additional zero bytes to make the total number of bits a multiple of 32.

The size of every atomic data type in OSC is a multiple of 32 bits. This guarantees that if the beginning of a block of OSC data is 32-bit aligned, every number in the OSC data will be 32-bit aligned.

#### 7.1.2 OSC Packets

The unit of transmission of OSC is an OSC Packet. Any application that sends OSC Packets is an *OSC Client*; any application that receives OSC Packets is an *OSC Server*.

An OSC packet consists of its contents, a contiguous block of binary data, and its size, the number of 8-bit bytes that comprise the contents. The size of an OSC packet is always a multiple of 4.

The underlying network that delivers an OSC packet is responsible for delivering both the contents and the size to the OSC application. An OSC packet can be naturally represented by a datagram by a network protocol such as UDP. In a stream-based protocol such as TCP, the stream should begin with an int32 giving the size of the first packet, followed by the contents of the first packet, followed by the size of the second packet, etc.

The contents of an OSC packet must be either an *OSC Message* or an *OSC Bundle*. The first byte of the packet's contents unambiguously distinguishes between these two alternatives.

### 7.1.3 OSC Messages

An OSC message consists of an *OSC Address Pattern* followed by an *OSC Type Tag String* followed by zero or more OSC Arguments.

- **OSC Address Patterns:** An OSC Address Pattern is an OSC-string beginning with the character '/' (forward slash).
- **OSC Arguments:** A sequence of OSC Arguments is represented by a contiguous sequence of the binary representations of each argument.
- **OSC Type Tag String<sup>2</sup>:** An OSC Type Tag String is an OSC-string beginning with the character ',' (comma) followed by a sequence of characters corresponding exactly to the sequence of OSC Arguments in the given message. Each character after the comma is called an OSC Type Tag and represents the type of the corresponding OSC Argument. (The requirement for OSC Type Tag Strings to start with a comma makes it easier for the recipient of an OSC Message to determine whether that OSC Message is lacking an OSC Type Tag String.)

### 7.1.4 OSC Bundles

An OSC Bundle consists of the OSC-string "#bundle" followed by an OSC Time Tag, followed by zero or more OSC Bundle Elements. The OSC-timetag is a 64-bit fixed point time tag whose semantics are described below.

An *OSC Bundle Element* consists of its size and its contents. The size is an int32 representing the number of 8-bit bytes in the contents and will always be a multiple of 4. The contents are either an OSC Message or an OSC Bundle.

To be more specific a bundle may contain bundles.

---

<sup>2</sup>For more detailed information about OSC Type String please visit: [http://opensoundcontrol.org/spec-1\\_0](http://opensoundcontrol.org/spec-1_0)

## 8 R-IoT Streaming, Sensor Fusion and Analysis

### 8.1 Sensor Fusion

Calibration of the sensors and the data fusion allowing for computing the quaternions and Euler angles are provided. The data fusion is implemented using the open-source code provided by Sebastian Madgwick. [8]

### 8.2 R-IoT Code Repository

Several code examples dedicated to the R-IoT platform are available on the BITalino GitHub repository <https://github.com/BITalinoWorld> and on IRCAM's GitHub repository: <https://github.com/Ircam-R-IoT>

The repository contains the main firmware, which achieves several analyses on the sensor data and allows the configuration of the module (IP address, UDP port, module ID) via a web server or the USB serial port.

Other simpler examples show how to write dedicated code for the R-IoT platform for specific motion analysis for instance.

Being a development platform, the R-IoT module role isn't bound to motion analysis or sensor data streaming. Using the additional I/Os and analog input and the Wi-Fi modem, the unit can be turned into an efficient Internet of Things (IoT) object, a miniature web server, a car alarm or a plant watering system.

### 8.3 Receive Sensors Data from the Module

The OpenSignals software provides an easy way to acquire and visualize data from sensors (please refer to [OpenSignals User Manual Chapter 2.3](#)). Instructions on the procedures to follow are available on R-IoT's [QuickStart Guide](#).

When plugged onto the UART of a BITalino (r)evolution device (instead of the Bluetooth or BLE modules), the R-IoT forges 2 single OSC packets containing all the exported data (IMU + BITalino). When used in standalone mode, only the IMU data are streamed within a single OSC message.

The IMU OSC message starts with `/<ID>/raw` followed by a list of 21 float numbers that split as the following (physical units sent as I.S. units)

- 3 axis accelerometer (1 float per axis) {-8 ; +8} g
- 3 axis gyroscope {-2 ; +2} °/s
- 3 axis magnetometer {-2 ; +2} gauss
- Temperature °K
- Switch (GPIO28) {0 / 1}
- Analog Inputs (GPIO3 & GPIO4) {0 ; 4095}
- Quaternions {-1 ; 1}
- Euler Angles and Heading {-180 ; 180} °

The BITalino OSC message starts with `/<ID>/bitalino` followed by a list of 11 integer numbers that split like the following:

- Sequence number {0 ; 15}
- 4 digital inputs I1 to I4 {0 ; 1}
- 6 analog inputs A1 to A6 {0 ; 1023}

Below an example Max/MSP patch shows the easiness of receiving and splitting the data flow. A quick way to verify that the data is flowing correctly between the R-IoT and your computer is to use the OSC Data Monitor[9]:

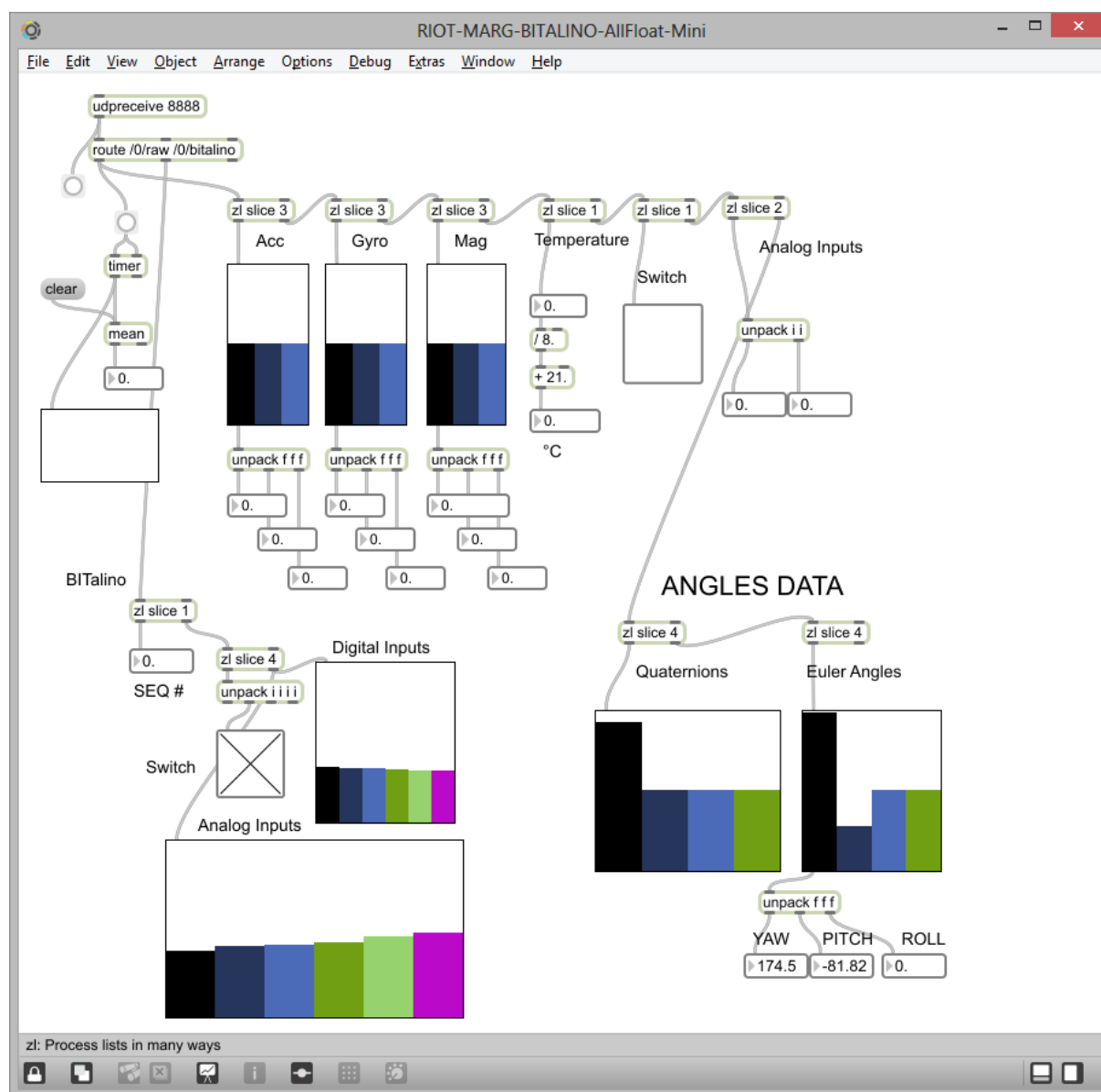


Figure 5-1 – Max/MSP example.



## 8.4 Max Abstractions for the R-IoT Bitalino

Several abstractions for Max (Cycling'74) are provided in the GitHub <https://github.com/Ircam-R-IoT>.

For example, the following abstraction *analysis-example.maxpat* shows example of various small abstraction that can be used to transform the raw accelerometer and gyroscope in intensity parameters and detecting kicks. This abstraction uses the free Max library MuBu available here: <http://forumnet.ircam.fr/product/mubu/>

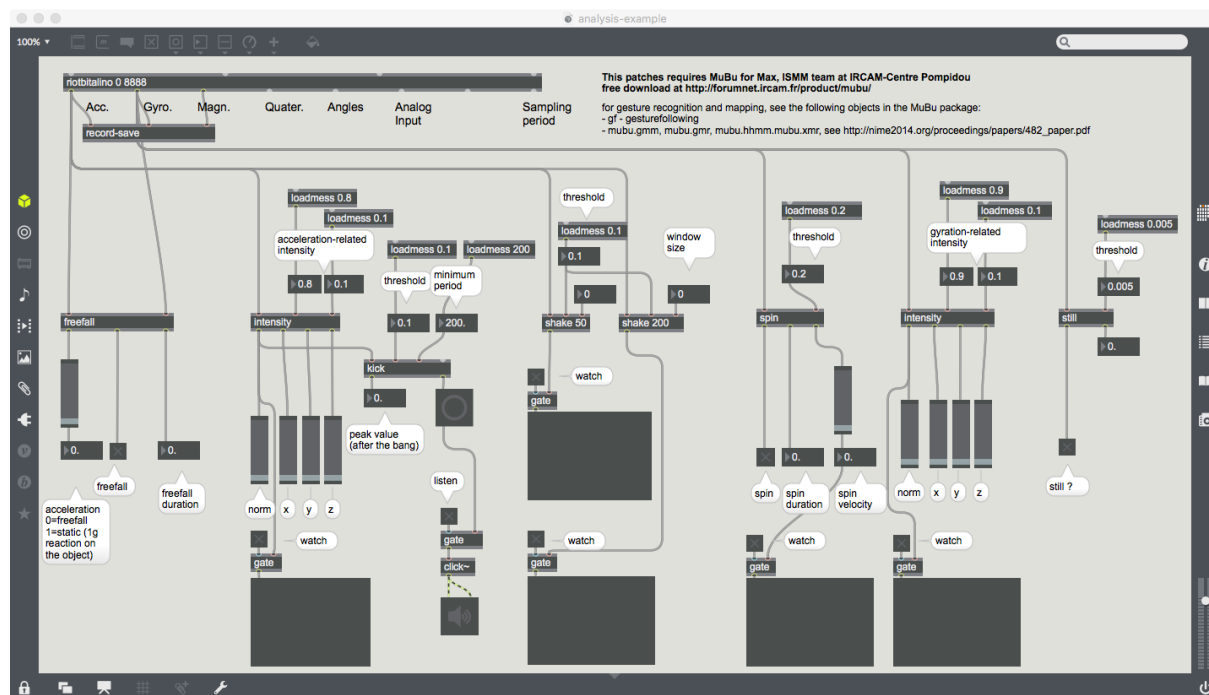


Figure 5-2 – Abstraction *analysis-example.maxpat*.

## 9 Programming the R-IoT

### 9.1 Install the Energia IDE

1. Visit <http://energia.nu/> and look for the download section
2. Download Energia Release 0101E0017
3. On Windows, simply unzip the Energia folder on your main hard drive. On Mac, place the folder-program in the Application folder.

### 9.2 Install the USB Serial Port Driver

The R-IoT device uses an external USB serial port to communicate with the computer, such as the TTL-232R-3V3 from FTDI. [10]

Install the drivers by visiting FTDI download section (look for VCP driver)<sup>3</sup>

<http://www.ftdichip.com/Drivers/VCP.htm>

On **Windows** and **Mac OS**, it will create a COM port.

On Mac OS, the user sometimes must create the port by going in the network preferences (select standard NULL modem).

### 9.3 Customize the IDE

In order to compile the "full" firmware (currently named BITalino R-IoT 2.041), the default linker file used by the Energia tool chain must be modified as the reserved heap size / stack is too high when compiling big programs.

On Windows:

- Open the location of your Energia folder, such as `C:\Program Files (x86)\energia-0101E0017\`
- Keep going into `hardware\cc3200\cores\cc3200`
- Edit the file `cc3200.ld`

On Mac OS:

- Open the location of your Energia folder, usually in the Application folder
- OSX applications are folders; right click on the app icon and select "show package contents"
- Browse `Contents/Resources/Java/hardware/cc3200/cores/cc3200` to
- Edit the file `cc3200.ld`

In the .ld file, simply edit the first line and change the default value of HEAP\_SIZE, which is 0x0007500 as follows:

```
HEAP_SIZE = 0x0008000;
```

---

<sup>3</sup> SparkFun tutorial available at: <https://learn.sparkfun.com/tutorials/how-to-install-ftdi-drivers>.

Additionally, the `WiFiUdp.cpp` source file must be modified (`hardware\cc3200\libraries\Wifi`), as its function `parsePacket()` blocks during 10ms. The code has been modified to reduce this timeout to 1ms by changing the default value of `timeout.tv_usec`, which is 10000 as follows:

```
timeout.tv_usec = 1000;
```

This will keep the low latency and a 200Hz sampling rate of the IMU streaming.

Edited files are available here:

<https://gist.github.com/wprimett/cc250921b2c7eb4121362980d9b03467> and  
<https://gist.github.com/wprimett/72768262e9b4c1a2212f71aaad4d2aa9>.

## 9.4 Install the Firmware and Examples

The full firmware uses 2 libraries that need to be installed to allow compiling the code: the SFLS library (File System for the CC3200 in order to retain parameters in a non-volatile memory <https://github.com/Ircam-R-IoT/SLFS>) and the BITalino basic library (a small wrapper to ease the communication with the BITalino (r)evolution <https://github.com/Ircam-R-IoT/bitalino-energia-library>). Please copy the folder '**bitalino1**', that has the newest version.

Those libraries must be placed in the `Documents\Energia\libraries` folder (PC or Mac).

## 9.5 Use Energia IDE

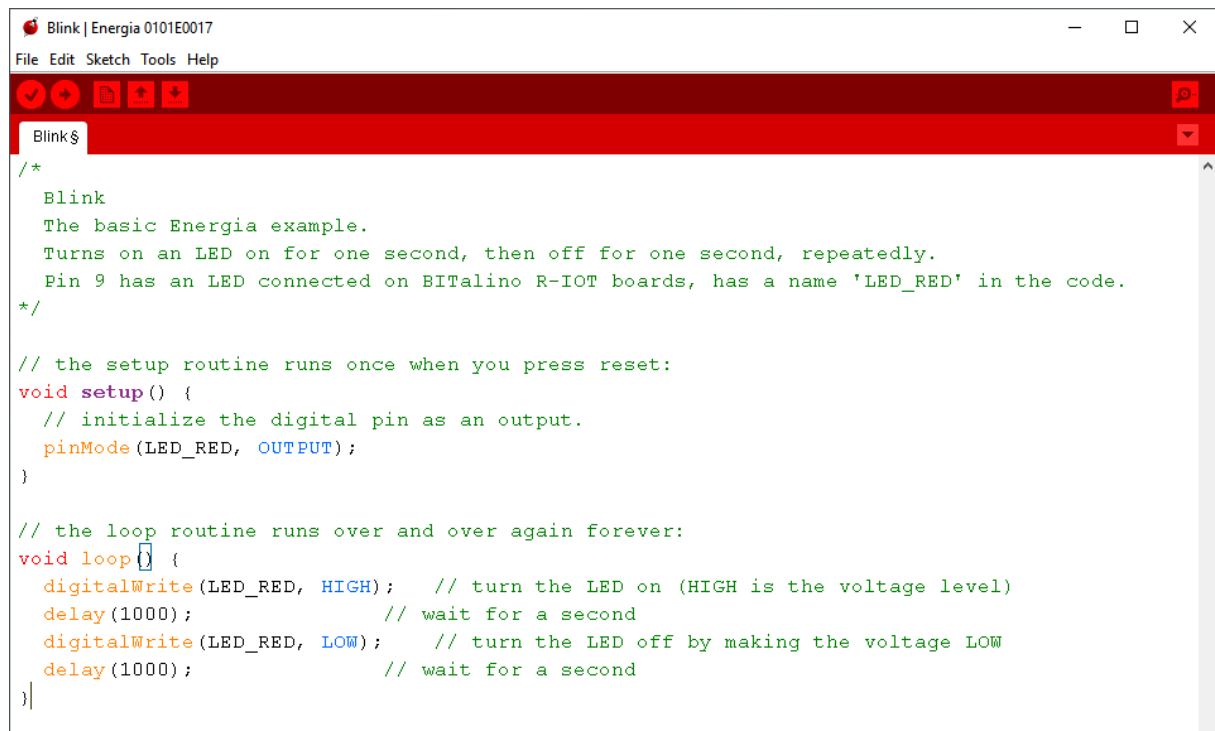
Launch Energia. A blank sketch appears. A sketch is composed of two essential functions, `setup()` and `loop()` which is equivalent to the main function in traditional C language. Energia, just like Arduino, uses C and C++, along with a basic API and set of classes to access the hardware in what became the Arduino standard.

<https://www.arduino.cc/en/Reference/HomePage>

The `setup` function, called prior the main loop, is used to configure the module hardware, default behaviors and everything that requires some initialization before executing the main program.

Once returning from the `setup` function, the `loop` function is called repetitively and is equivalent to any traditional endless program loop used on embedded platforms (a `while(1)` statement within the main function).

Below an example of blinking endlessly the red LED from RGB\_LED of the module (GPIO I/O #9):

The image is a screenshot of the Energia IDE interface. The title bar reads 'Blink | Energia 0101E0017'. The menu bar includes 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. Below the menu bar is a toolbar with icons for opening, saving, and running. The main editor area has a red header bar with the text 'Blink\$' and a dropdown arrow. The code is as follows:

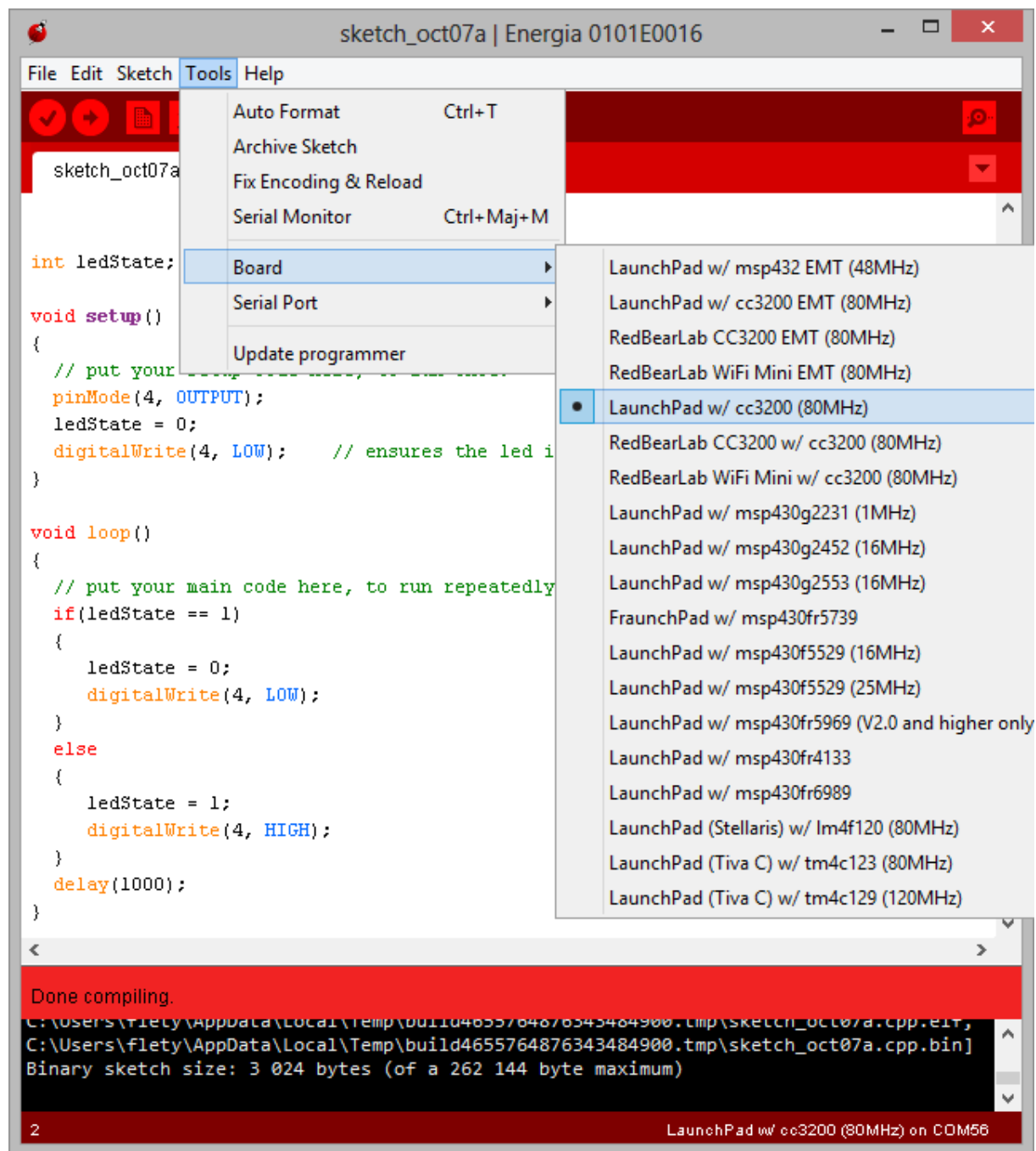
```
/*
  Blink
  The basic Energia example.
  Turns on an LED on for one second, then off for one second, repeatedly.
  Pin 9 has an LED connected on BITalino R-IOT boards, has a name 'LED_RED' in the code.
*/

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(LED_RED, OUTPUT);
}

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

**Figure 6-1 – Blink Code example.**

Before compiling, select the proper target in the Tools -> Board menu and select the launchPad w/ CC3200 (80 MHz).



*Figure 6-2 – Selection of the Board.*

To compile the program, simply click on the left-most icon (tick).

## Flashing the Firmware

To flash the code on the platform first plug the USB serial cable in a USB port then select the matching COM port in Energia Tool->Serial port menu.

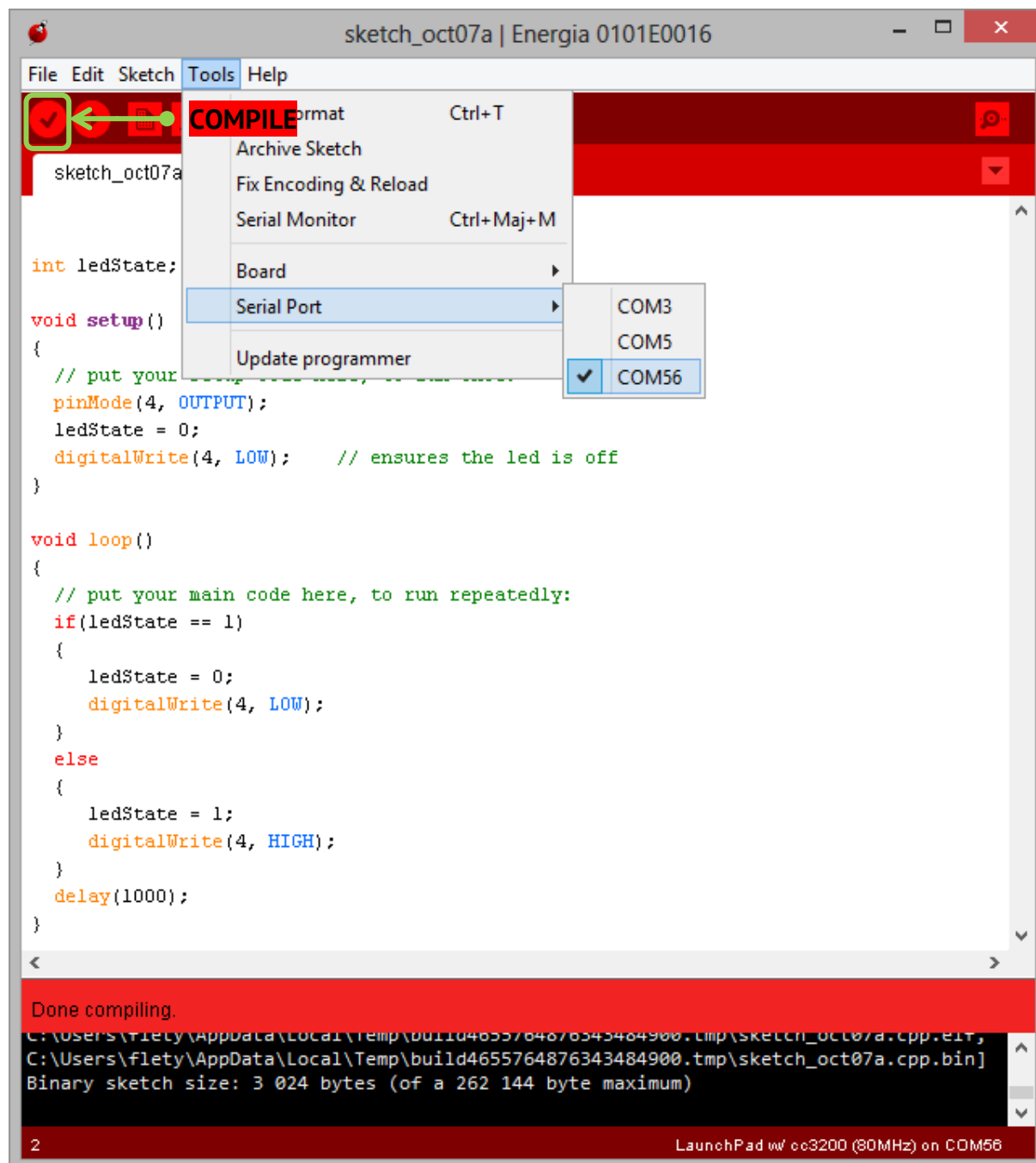


Figure 6-3 – Communication Port configuration.

With the module powered on, even already executing the code present in the chip, press both the flash and reset switches and click on the upload icon (right arrow). It will first compile the code. Once compiling is over, the "uploading" text will be printed above the console. Release the reset switch while keeping the flash switch pressed. After a while, you'll get a done uploading message with the following log in the console of the IDE.

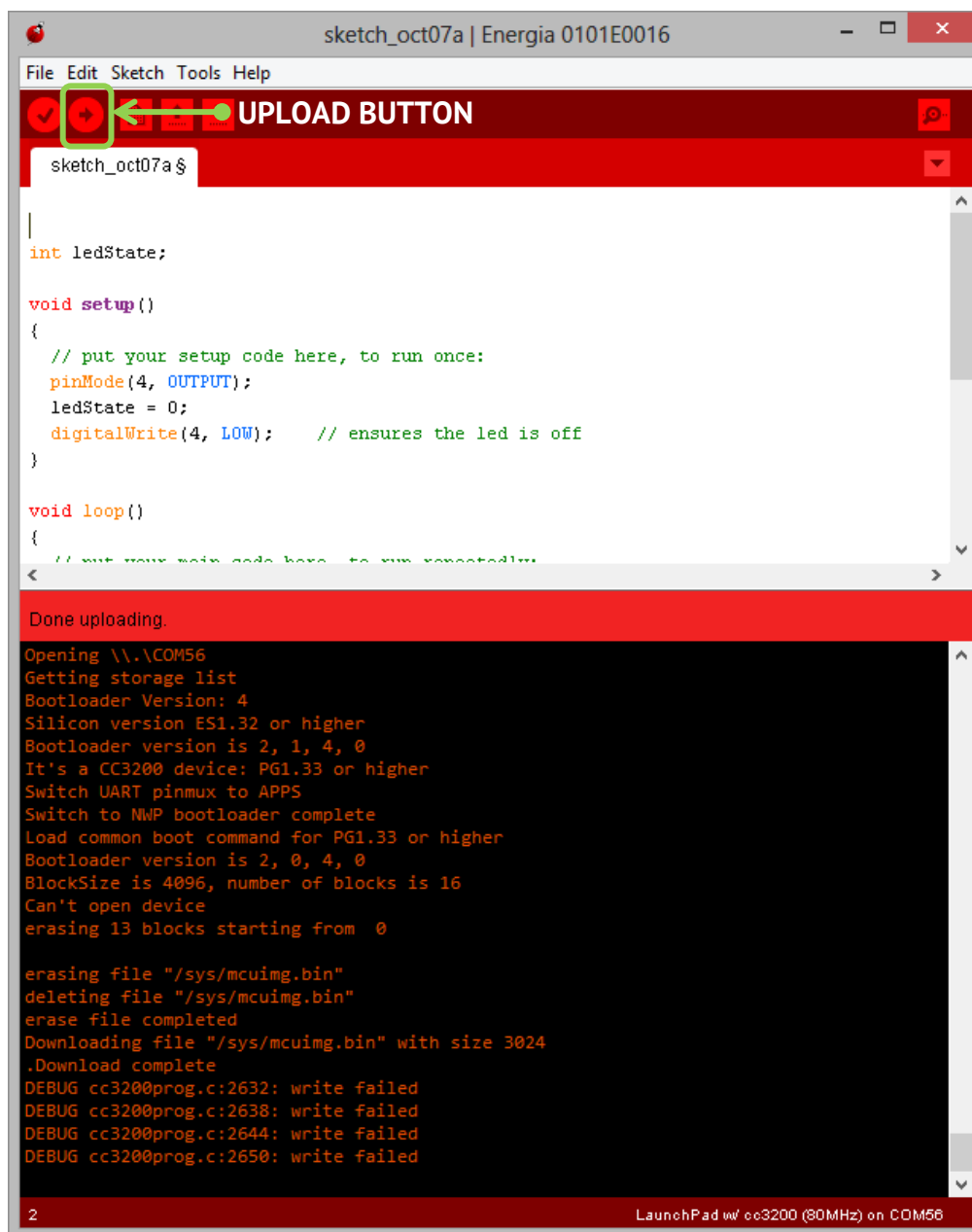


Figure 6-4 – Flashing firmware.

The DEBUG write fail messages are normal, as the bootloading program tries also to set the chip in debug mode and our platform doesn't have any JTAG debugger / port exported.

Once there, the module can be reset (cycle the on-off switch or press the reset onboard momentary switch) to leave flashing mode and execute the freshly uploaded code. With the above code, the blue LED should flash once per second.



## 10 Glossary

### Access Point

An access point is a device that allows other wireless devices to connect to it by the means of antennas.

### IMU

An inertial measurement unit (**IMU**) is an electronic device that measures and reports a body's specific force, angular rate, and sometimes the orientation of the body, using a combination of accelerometers, gyroscopes, and sometimes magnetometers.

For further information about our IMU sensor, please refer to the following datasheet: LSM9DS1.

### Max

Max, also known as Max/MSP/Jitter, is a visual programming language for music and multimedia developed and maintained by San Francisco-based software company Cycling '74. Over its more than thirty-year history, it has been used by composers, performers, software designers, researchers, and artists to create recordings, performances, and installations. [11]

The Max program is modular, with most routines existing as shared libraries. An application programming interface (API) allows third-party development of new routines (named external objects). Thus, Max has a large user base of programmers unaffiliated with Cycling '74 who enhance the software with commercial and non-commercial extensions to the program. Because of this extensible design, which simultaneously represents both the program's structure and its graphical user interface (GUI). [12]

For further information please visit <https://cycling74.com/>.

## OSC

Open Sound Control (OSC) is a protocol for communication among computers, sound synthesizers, and other multimedia devices that is optimized for modern networking technology. Bringing the benefits of modern networking technology to the world of electronic musical instruments, OSC's advantages include interoperability, accuracy, flexibility, and enhanced organization and documentation.

This simple yet powerful protocol provides everything needed for real-time control of sound and another media processing while remaining flexible and easy to implement.

Features:

- Open-ended, dynamic, URL-style symbolic naming scheme
- Symbolic and high-resolution numeric argument data
- Pattern matching language to specify multiple recipients of a single message
- High resolution time tags
- "Bundles" of messages whose effects must occur simultaneously
- Query system to dynamically find out the capabilities of an OSC server and get documentation

For further information please visit <http://opensoundcontrol.org/>.

## 11 Bibliography

- [1] STMicroelectronics., “LSM9DS1 iNEMO inertial module: 3D accelerometer, 3D gyroscope, 3D magnetometer,” *STMicroelectronics.Ginebra, Switz.*, 2014.
- [2] Texas Instruments, “CC3200 SimpleLink Wi-Fi® and Internet-of-Things solution, a Single-Chip Wireless MCU | TI.com.” [Online]. Available: <https://www.ti.com/product/CC3200>. [Accessed: 06-Feb-2020].
- [3] Energia, “Energia - Home.” [Online]. Available: <https://energia.nu/>. [Accessed: 06-Feb-2020].
- [4] Arduino, “Arduino - Home.” [Online]. Available: <https://www.arduino.cc/>. [Accessed: 06-Feb-2020].
- [5] Wiring, “Wiring.” [Online]. Available: <http://wiring.org.co/>. [Accessed: 06-Feb-2020].
- [6] Texas Instruments, “Hardware Kits & Boards | Design Resources | TI.com.” [Online]. Available: <http://www.ti.com/design-resources/embedded-development/hardware-kits-boards.html?keyMatch=LAUNCHPAD&tisearch=Search-EN-everything>. [Accessed: 06-Feb-2020].
- [7] Open Sound Control, “Introduction to OSC | opensoundcontrol.org.” [Online]. Available: <http://opensoundcontrol.org/introduction-osc>. [Accessed: 07-Feb-2020].
- [8] X. Technologies, “Open source IMU and AHRS algorithms,” 2012. [Online]. Available: <https://x-io.co.uk/open-source-imu-and-ahrs-algorithms/>.
- [9] K. Kamperman, “OSC Data Monitor ★ Kasper Kamperman.” [Online]. Available: <https://www.kasperkamperman.com/blog/processing-code/osc-datamonitor/>. [Accessed: 06-Feb-2020].
- [10] FTDI, “FT232R USB IC Datasheet,” in *Technology*, 2008.
- [11] M. Sheffield, “Max/MSP for average music junkies,” *Hope&Fears*. [Online]. Available: <http://www.hopesandfears.com/hopes/culture/music/168579-max-msp-primer>. [Accessed: 16-Sep-2018].
- [12] T. Place and T. Lossius, “Jamoma: A modular standard for structuring patches in max,” in *International Computer Music Conference, ICMC 2006*, 2006.