



Globalcode

JAVA IOT SURFING API



FILES FOR THIS CLASS

[HTTPS://PORTALALUNO.TOOLSCLOUD.NET/REDMINE/PROJECTS/IOTSURFBOARD/FILES](https://portalaluno.toolscloud.net/redmine/projects/iotsurfboard/files)

□ PRESENTATION: IOT_SURFING_CLASS_9_EN.PDF

JAVA + IOT

□ JAVA WAS BORN FOR THINGS:



JAVA PLATFORM

- **JAVA ME**: VERY GOOD SHAPE IN VERSION 8 RUNNING ON K64F FREESCALE
- **JAVA SE**: NICE PERFORMANCE ON RASPBERRY PI & ARM IN GENERAL
- **JAVA EE**: NOW CAN RUN ON DIFFERENT SINGLE-BOARD COMPUTER!
- **JAVA 9**: MODULARITY, PROFILES, ETC. = **JAVA ON DEVICES!**

JAVA IOT SURFING API

- ALLOWS YOU TO CONTROL YOUR IOT SURFBOARD USING JAVA:



USB CABLE
Serial Communication RXTX



JAVA IOT SURFING API: SAMPLE CODE

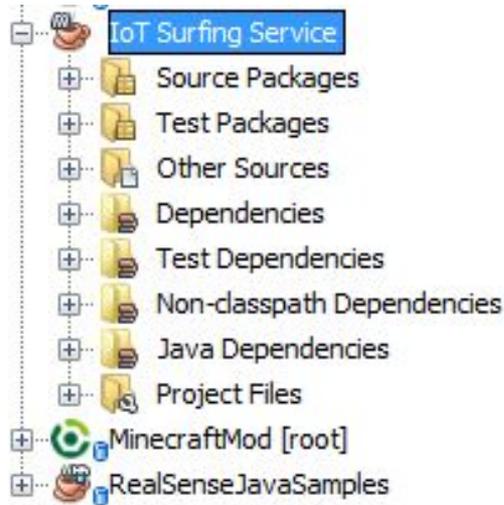
```
board = new IoTSurfboard("COM3", 9600);  
System.out.println("Alcohol      : " + board.alcohol());  
System.out.println("Temperature  : " + board.temperature());  
System.out.println("Humidity     : " + board.humidity());  
System.out.println("Light        : " + board.light());  
System.out.println("Potentiometer: " + board.potentiometer());  
System.out.println("Light        : " + board.light());  
board.red(255);  
Kernel.delay(1000);
```

RXTX NATIVE CODE

- WE NEED TO USE RXTX LIBRARY TO ESTABLISH A SERIAL COMMUNICATION
- DOWNLOAD & INSTALL: [HTTP://RXTX.QBANG.ORG/WIKI/INDEX.PHP/DOWNLOAD](http://rxtx.qbang.org/wiki/index.php/download)

STARTING WITH JAVA IOT SURFING API + NETBEANS

- CLONE OUR REPOSITORY: [HTTPS://GITHUB.COM/SURFBOARD/SERVICE](https://github.com/surfboard/service)



FIX THE WORKING DIRECTORY:

Project Properties - IoT Surfing Service

Categories:

- General
- Sources
- Configurations
- JavaScript Libraries
- Build
 - Compile
 - Run**
 - Actions
- JavaScript Frameworks
 - RequireJs
- License Headers
- Formatting
 - CheckStyle Formatting
- Hints

Configuration: <default config>

Main Class: org.surfing.kernel.Kernel

Arguments: -classpath %classpath org.surfthing.Kernel

Working Directory: ::\Users\vinic_000\Google Drive\IoT Surfboard\apps\IoT Surfing Service

VM Options: -Dsurfing.config="./config/etc/surfing"
-Djava.library.path="./etc/win64" Wrap text

Main Class: org.surfing.kernel.Kernel

Arguments: -classpath %classpath org.surfthing.Kernel

Working Directory: ::\Users\vinic_000\Google Drive\IoT Surfboard\apps\IoT Surfing Service

VM Options: -Dsurfing.config="./config/etc/surfing"
-Djava.library.path="./etc/win64" Wrap text

CHOOSE MAIN CLASS: TESTUSBO

Main Class:	<input type="text" value="TestUSBO"/>	<input type="button" value="Browse..."/>
Arguments:	<input type="text" value="-classpath %classpath org.surfthing.Kernel"/>	
Working Directory:	<input type="text" value="::\Users\winic_000\Google Drive\IoT Surfboard\apps\IoT Surfing Service"/>	<input type="button" value="Browse..."/>
VM Options:	<input .="" config="" etc="" surfing\"<br="" type="text" value="-Dsurfing.config=\"/> -Djava.library.path=\"./etc/win64\" "/>	<input type="button" value="Customize..."/> <input checked="" type="checkbox"/> Wrap text

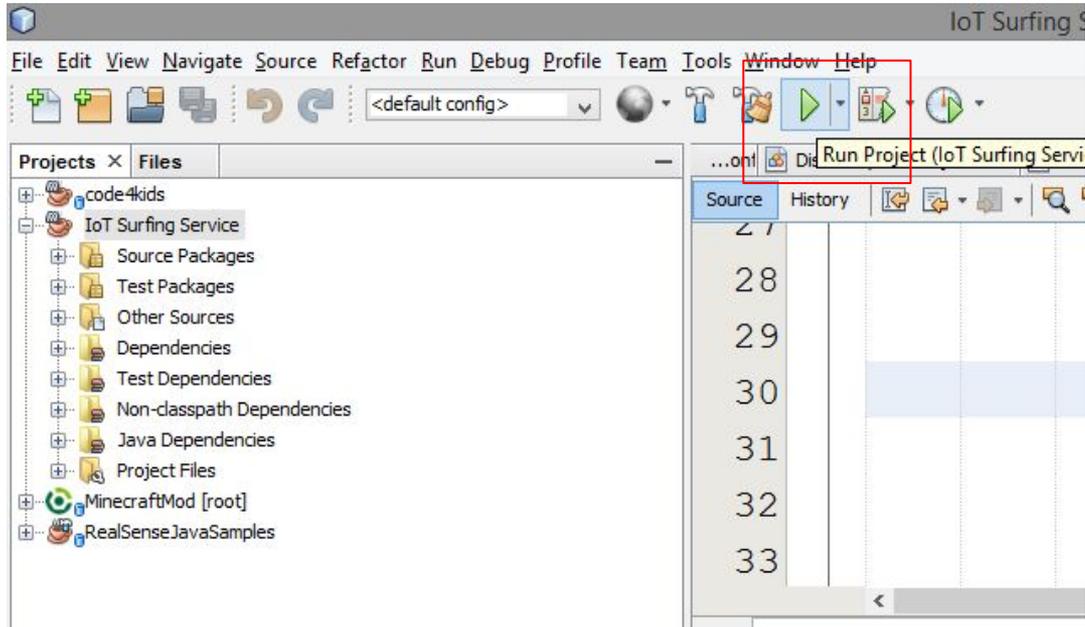
FINALLY SETUP RXTX NATIVE DIRECTORY

Main Class:	<input type="text" value="TestUSB0"/>	<input type="button" value="Browse..."/>
Arguments:	<input type="text" value="-classpath %classpath org.surfthing.Kernel"/>	
Working Directory:	<input type="text" value="::\Users\winic_000\Google Drive\IoT Surfboard\apps\IoT Surfing Service"/>	<input type="button" value="Browse..."/>
VM Options:	<input .="" config="" etc="" surfing\"<br="" type="text" value="-Dsurfing.config=\"/> -Djava.library.path=\"./etc/win64\""/>	<input type="button" value="Customize..."/>
		<input checked="" type="checkbox"/> Wrap text

CHOOSE THE RIGHT SERIAL / COM PORT

```
board = new IoTSurfboard("COM3", 9600);  
System.out.println("Alcohol      : " + board.alcohol());  
System.out.println("Temperature  : " + board.temperature());  
System.out.println("Humidity     : " + board.humidity());  
System.out.println("Light        : " + board.light());  
System.out.println("Potentiometer: " + board.potentiometer());  
System.out.println("Light        : " + board.light());  
board.red(255);  
Kernel.delay(1000);
```

RUN THE PROJECT



IT MAY FAIL THE FIRST TIME!

RUN THE PROJECT

INFO: Connection Stabilished with //./COM3

Alcohol :0

Temperature :26.0

Humidity :21.0

Light :22

Potentiometer:872

Light :22

Feb 08, 2016 9:54:42 AM org.surfing.device.SerialDevice close

INFO: Closing device on //./COM3

Closing FINAL //./COM3 port

--

LIVE DEMO

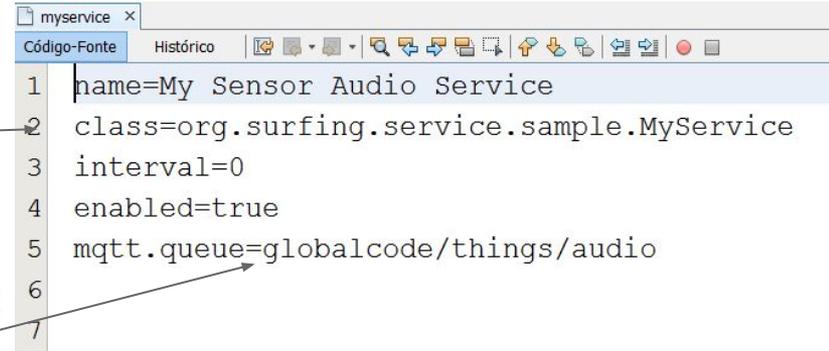


IOT SURFING SERVICE

- JAVA **MICROKERNEL** FOR IOT
- PROVIDES **ACCESS** TO YOUR **IOT SURFBOARD** VIA **REST** AND **MQTT**
- MANY SERVICES: **SPEECH**, CAMERA, **FTP**, SERIAL, **PERSISTENCE**, DEVICE DISCOVERY
- EACH SERVICE HAS A **SERVICE** OR **MQTT** SUB-CLASS + **CONFIG FILE**
- PERFORM GREAT ON RASPBERRY PI!

SERVICE EXAMPLE

```
@Path("/myservice")
public class MyService extends MQTTController {
    @GET
    @Produces("text/html")
    @Path("/s1/{name}")
    public String execute(@PathParam("name") String name) {
        System.out.println("Name " + name);
        return "";
    }
    public void processMessage(String msg) {
        for (Device device : Kernel.getInstance().getDevices()) {
            Thing t = device.getThings().get(msg);
            if(t.getName().equals(msg)) {
                AudioTTS.speak(msg + " value is " + t.getLastValue(), true);
            }
        }
    }
}
```



The screenshot shows an IDE window titled 'myservice' with a 'Código-Fonte' (Source Code) tab. The window displays a list of configuration parameters for an MQTT service, numbered 1 through 7. The parameters are: 1. name=My Sensor Audio Service, 2. class=org.surfing.service.sample.MyService, 3. interval=0, 4. enabled=true, 5. mqtt.queue=globalcode/things/audio, 6. (empty line), and 7. (empty line). Two arrows originate from the code on the left: one points from the 'MyService' class name to line 2, and another points from the 'processMessage' method to line 5.

```
1 name=My Sensor Audio Service
2 class=org.surfing.service.sample.MyService
3 interval=0
4 enabled=true
5 mqtt.queue=globalcode/things/audio
6
7
```

1. CHOOSE CLASS KERNEL TO RUN

Configuration: <default config> ▼

Main Class: Browse...

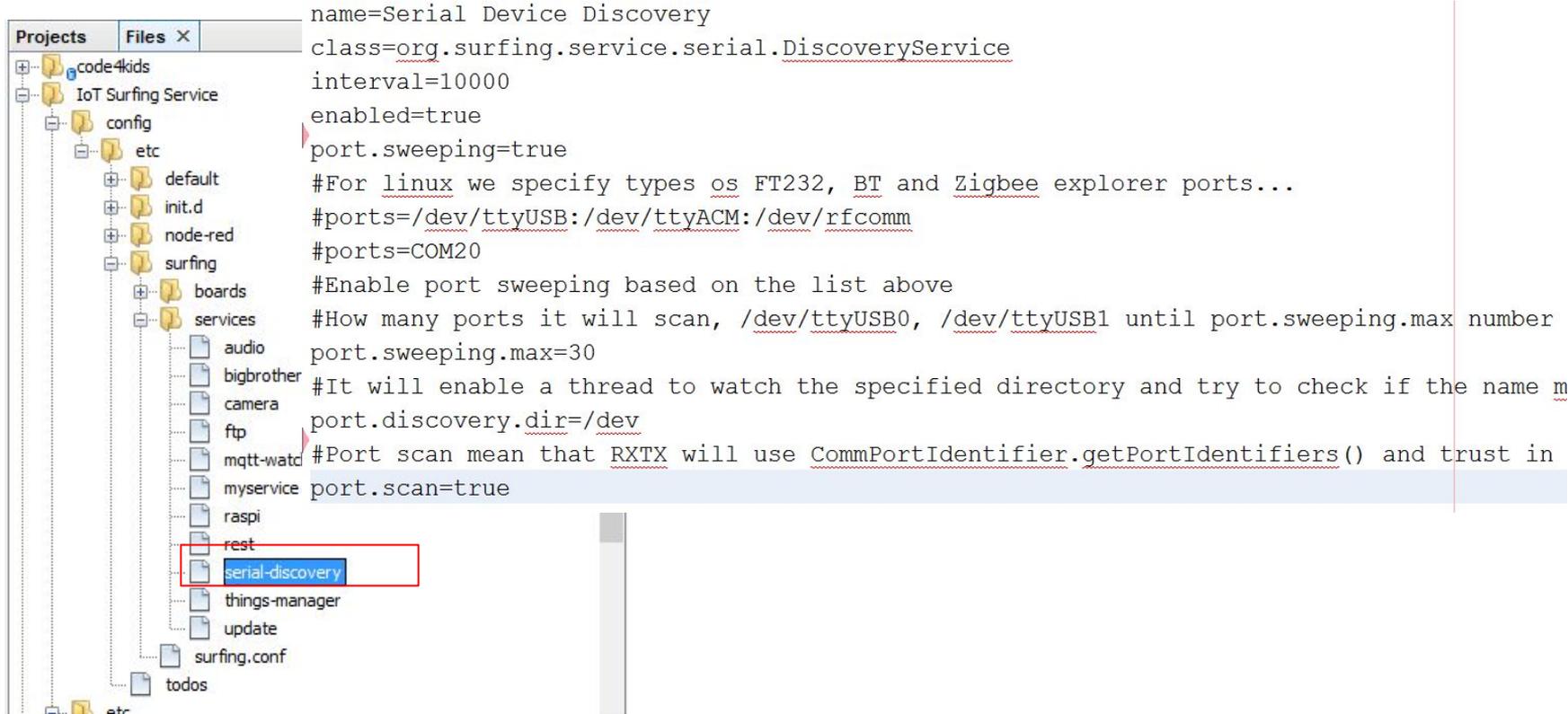
Arguments:

Working Directory: Browse...

VM Options: -Djava.library.path=\",./etc/win64\""/>

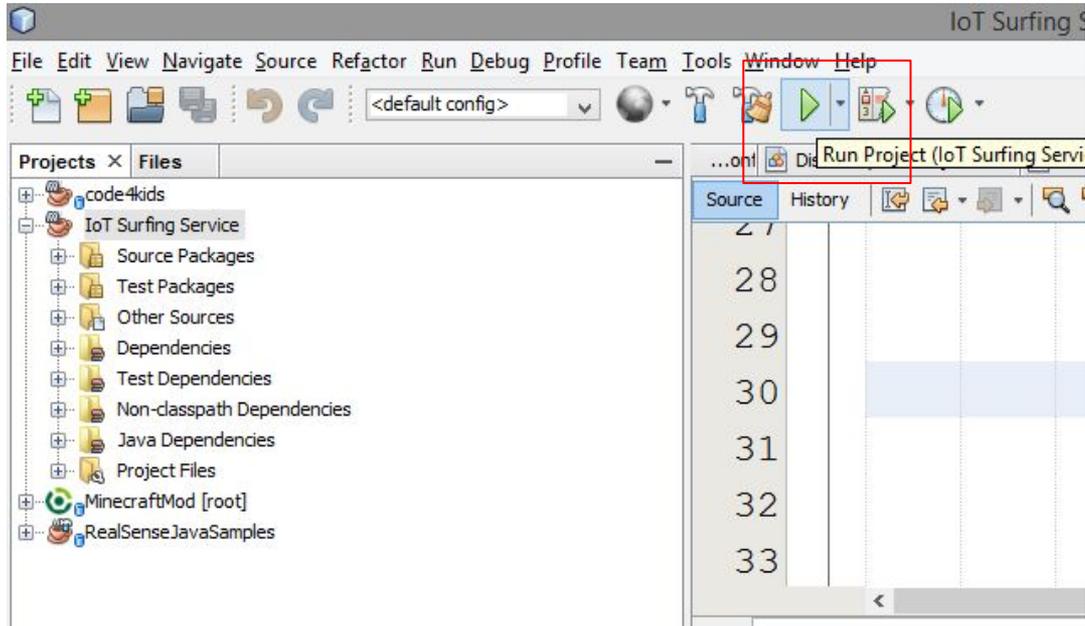
Customize...
 Wrap text

2. CHECK SERIAL-DISCOVERY SERVICE CONFIG



```
name=Serial Device Discovery
class=org.surfing.service.serial.DiscoveryService
interval=10000
enabled=true
port.sweeping=true
#For linux we specify types os FT232, BT and Zigbee explorer ports...
#ports=/dev/ttyUSB:/dev/ttyACM:/dev/rfcomm
#ports=COM20
#Enable port sweeping based on the list above
#How many ports it will scan, /dev/ttyUSB0, /dev/ttyUSB1 until port.sweeping.max number
port.sweeping.max=30
#It will enable a thread to watch the specified directory and try to check if the name m
port.discovery.dir=/dev
#Port scan mean that RXTX will use CommPortIdentifier.getPortIdentifiers() and trust in
port.scan=true
```

3. RUN THE PROJECT



4. PLUG YOUR IOT SURFBOARD AND CHECK THE OUTPUT

```
Feb 08, 2016 10:25:17 AM org.surfing.service.serial.DiscoveryService scanP
INFO: Scanning port: COM3
Feb 08, 2016 10:25:17 AM org.surfing.service.serial.DiscoveryService scanP
INFO: Serial Device Port found COM3. Trying to discovery this device.
Feb 08, 2016 10:25:17 AM org.surfing.device.SerialDevice open
INFO: Connection Stabilished with //./COM3
Cleaning initial data #1 null
Cleaning initial data #1 null
Cleaning initial data #1 null
Feb 08, 2016 10:25:19 AM org.surfing.service.mqtt.MQTTBaseService fixConne
INFO: Connection Stablshed!
Feb 08, 2016 10:25:19 AM org.surfing.service.mqtt.MQTTController fixConnec
INFO: MQTT Receiver Subscribing globalcode/things
Feb 08, 2016 10:25:20 AM org.surfing.kernel.Kernel initServices
```

5. MANAGE YOUR SURFBOARD VIA MQTT

- BY DEFAULT **MQTT SERVICE** WILL USE THE ECLIPSE SANDBOX: **IOT.ECLIPSE.ORG**
- TWO DIFFERENT QUEUES:
 - SENSORS: GLOBALCODE/THINGS/SURFBOARDX (X = YOUR SURFBOARD NUMBER)
 - CONTROL: GLOBALCODE/THINGS
- YOU CAN **SUBSCRIBE GLOBALCODE/THINGS/SURFBOARDX** TO START **RECEIVING** YOUR SENSORS VALUE!

6. SUBSCRIBE SENSORS QUEUE USING MQTT.FX

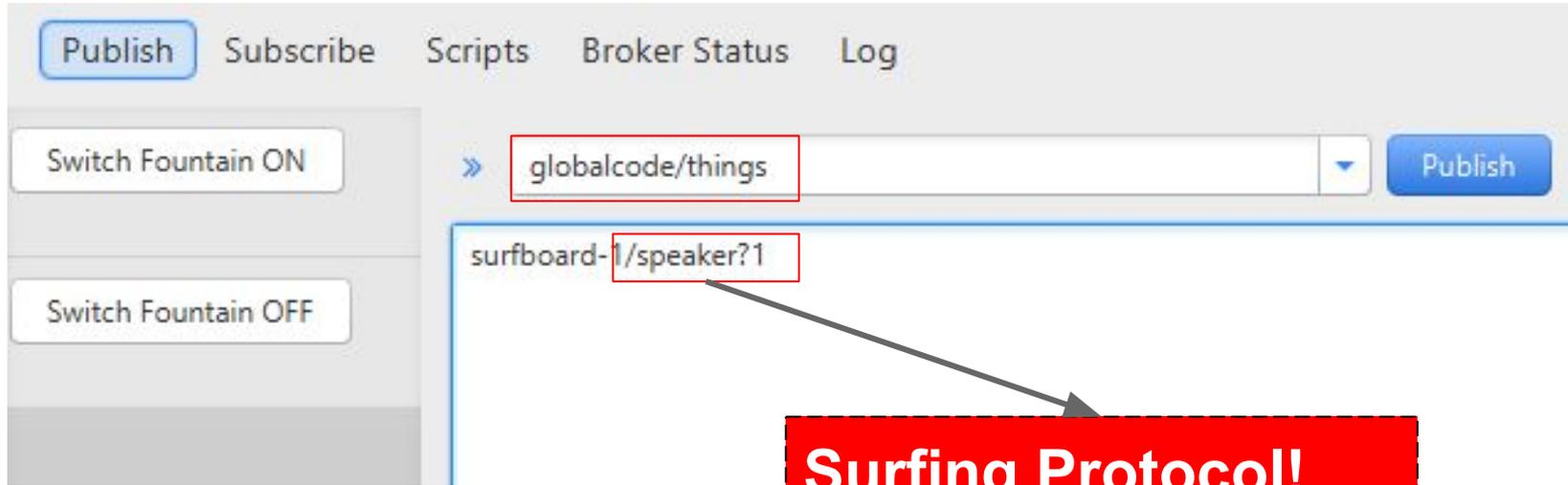
The screenshot displays the MQTT.fx 1.0.0 application window. The 'Connect' button is highlighted with a red box. The 'Subscribe' button is also highlighted. The subscription topic is 'globalcode/things/surfboard-1'. The received message is a JSON object:

```
globalcode/things/surfboard-1  
1  
08-02-2016 10:30:36.37836215  
Copy Payload  
{"name": "surfboard-1", "firmware" ...  
{"name": "surfboard-1", "firmware": "2", "serial": "-1", "key": "-1", "components": [{"name": "alcohol", "value": "0"}, {"name": "pot", "value": "672"}, {"name": "light", "value": "109"}, {"name": "distance", "value": "0"}, {"name": "clock", "value": "2/8/2016 09:30:11"}, {"name": "temp", "value": "27.00"}, {"name": "humidity", "value": "22.00"}]}
```

An inset window shows the subscription controls for 'globalcode/things/surfboard-1' with 'Subscribe' and 'Unsubscribe' buttons. Below it, a message list shows '2 Messages' for the same topic.

**PS. your surfboard name should be surfboard159, surfboard180, etc.
This example uses a testing Surfboard which has number -1**

7. CONTROL ACTUATORS USING MQTT.FX



The screenshot shows the MQTT.fx web interface. At the top, there are navigation tabs: "Publish" (highlighted), "Subscribe", "Scripts", "Broker Status", and "Log". On the left side, there are two buttons: "Switch Fountain ON" and "Switch Fountain OFF". In the center, there is a text input field containing "globalcode/things" with a red box around it, and a "Publish" button to its right. Below this, a message box displays "surfboard-1/speaker?1" with a red box around it. An arrow points from this message box to a red callout box containing the text "Surfing Protocol!".

Try this:
***/speaker?1**

Surfing Protocol!

8. CONTROL YOUR SURFBOARD VIA REST

□ `HTTP://LOCALHOST:8888/THINGS/DATA/SURFBOARDX/[SENSOR]`

`HTTP://LOCALHOST:8888/THINGS/DATA/SURFBOARD-1/TEMP`

□ `HTTP://LOCALHOST:8888/THINGS/SURFBOARDX/[ACTUATOR]/[VALUE]`

`HTTP://LOCALHOST:8888/THINGS/SURFBOARD-1/SPEAKER/1`

`HTTP://LOCALHOST:8888/THINGS/SURFBOARD-1/BLUE/100`

NOW IS ABOUT HAVING FUN!!!!

LIVE DEMO



SUMMARY

- ❑ IOT SURFING API PROVIDES ACCESS TO YOUR BOARD VIA SERIAL COMMUNICATION;
- ❑ IOT SURFING SERVICE IS A SERIAL - TCP/IP GATEWAY
- ❑ MQTT + REST = NICE COMBO!
- ❑ NOW YOU CAN CONTROL YOUR SURFBOARD USING ANY PLATFORM!

IOT SURFBOARD + ARDUINO + JAVA =
LOVE!

