



# User's Manual

**DG100**

Multi-service IoT Gateway

## Legal notice

Reproduction, transfer, distribution or storage of part or all of the contents in this document in any form without the prior written permission is prohibited. All rights reserved. All trademarks mentioned herein belong to their respective owners.

**Copyright © 2020 Decode**

## Disclaimer

Decode has used reasonable care in preparing the information included in this document, but does not warrant that such information is error free.

Decode, its associates, representatives, employees, and others acting on its behalf disclaim any and all liability for errors, inaccuracies, or incompleteness contained in any datasheet or in any other disclosure relating to any product.

In the interest of continuous product development, the Decode reserves the right to make improvements to this manual and the products described in it at any time and without prior notification or obligation.

The use of the product is at sole discretion of the user. Decode cannot be held responsible for any damages arising due to use of this product and makes no warranty, representation or guarantee regarding the suitability of the products for any particular purpose or the continuing production of any product.

Note: The specifications in this document are valid as of the listed versions of software and/or hardware. Revised versions of this manual, as well as software and driver updates are available in the download area of the Decode web site.

# Table of Contents

<b>1 Preface</b> .....	<b>5</b>
1.1 Symbols.....	5
1.2 Safety Instructions.....	5
1.3 Document versions.....	6
<b>2 Overview</b> .....	<b>7</b>
2.1 Ordering information.....	8
2.2 Accessories.....	8
<b>3 Device Description</b> .....	<b>9</b>
3.1 External connectors.....	10
3.1.1 Serial1 and Serial2.....	10
3.1.2 USB Debug connector.....	11
3.1.3 USB Device connector.....	11
3.1.4 USB Host connector.....	11
3.1.5 Ethernet connector.....	11
3.1.6 Power connector.....	12
3.2 Internal mikroBUS™ connector.....	12
3.2.1 RTC Battery.....	12
3.3 Indications and push buttons.....	13
3.3.1 Push buttons.....	13
3.3.2 LED indicators.....	13
3.4 Wall or plate mounting.....	14
<b>4 Operation description</b> .....	<b>15</b>
4.1 Command line interface.....	15
4.1.1 Access over USB Debug port.....	15
4.1.2 Access over SSH.....	17
4.2 Basic commands.....	18
4.2.1 Checking network parameters.....	19
4.2.2 Adjusting the Linux time.....	19
4.2.3 Save Linux time in RTC.....	19
4.2.4 Check RTC time.....	19
4.2.5 Check installed TTY ports.....	19
4.2.6 Stop DG100 program.....	19
4.2.7 Reboot.....	20

4.2.8 Changing network settings.....	20
<b>5 Web user interface.....</b>	<b>22</b>
5.1 Console.....	24
5.2 Bundle management.....	25
5.2.1 View bundles info.....	25
5.2.2 Starting, stopping, upgrading and uninstalling bundles.....	26
5.2.3 Installing new bundles.....	26
5.3 System Information.....	27
5.4 Settings.....	28
5.5 Sensors & Devices.....	29
5.6 Playground.....	30
5.6.1 Writing JavaScript programs.....	31
5.6.2 Saving JavaScript program.....	32
5.6.3 Exporting JavaScript program to bundle.....	32
<b>6 Technical specifications.....</b>	<b>33</b>
6.1 Processor board.....	33
6.2 Serial ports.....	33
6.3 Wireless.....	33
6.4 Common characteristics.....	33
6.5 Software.....	33
<b>7 Product label.....</b>	<b>34</b>
<b>8 Disposal and Recycling.....</b>	<b>34</b>
<b>9 Contact.....</b>	<b>34</b>

# 1 Preface

## 1.1 Symbols



**WARNING** – Safety notice, which must be followed, may have influence on the user's safety or the function of the device.



**IMPORTANT** – Notice, which must be followed to avoid possible problems, which can arise in specific cases.



**NOTE** – Notice, which contains useful advice.

## 1.2 Safety Instructions

Device must be used in compliance with any and all applicable international and national laws and in compliance with special restrictions regulating the utilization of the communications of the communication module in prescribed applications and environments.



**WARNING** – We suggest you to adhere to following recommendations so as to avoid any damage to person or property.

- **All the associated (interconnected) equipment, PC and power supply units (PSU) shell comply with requirements of standard IEC 60950- 1:2005+A1:2009+A2:2013.**
- **Power supply must have SELV output and for security reasons connection must include series 1A fuse protection.**
- **Installation and technical support of the device can be performed only by a qualified personnel or a person who has enough knowledge about this device and safety requirements.**
- **Unauthorized modifications or utilization of accessories that have not been approved may result in damage to the device and in a breach of applicable regulations, and result in the termination of the validity of the guarantee.**
- **Do not expose the device to extreme ambient conditions. Protect the device against dust, moisture and high temperature.**

## 1.3 Document versions

Document version	Date	Note
v1.0	16/10/2020	First release
v1.1	26/10/2020	Typo correction and correction of serial port names in 3.1.1 section

## 2 Overview

DECODE DG100 IoT Gateway is a compact and efficient Multi-service IoT Edge Gateway for home, office and industrial applications. Powered by NXP i.MX6 CPU at 400MHz, with 256MB of SDRAM memory and 4GB of eMMC, it offers great performance for great variety of next generation solutions. Robustness is guaranteed by wide range of power supply voltage with transient/surge/noise/reverse polarity protection and reliable hardware watchdog timer.



**Fig. 1: DG100 IoT Gateway**

Device provides powerful combination of interfaces and is ideally suited for M2M and IoT to connect sensors, actuators and other devices to cloud services. Two protected and isolated serial ports, both with RS-232 and RS-485 interfaces, offers connection to serial enabled devices such as: PLC's, pump controllers, HMI's or remote I/O devices. Embedded wireless connectivity with Wi-Fi and Bluetooth, three USB and one Ethernet port fulfill the needs for local and remote connectivity.

On board mikroBUS™ socket, placed inside enclosure, may be used for add-on boards and enable easy hardware expandability with a large number of standardized compact add-on boards, each one carrying a single sensor, transceiver, display, encoder, motor driver, connection port, or any other electronic module or integrated circuit.

DG100 comes with preinstalled Linux OS and macchina.io IoT edge device framework which takes the complexity out of integrating sensors, actuators, devices and cloud services, as well as device management. This framework is right solution when performance, low footprint, efficiency and security counts, supporting variety of field and IoT protocols and devices such as Modbus, MQTT, BtLE, GNSS or SensorTag. Built-in web server provides UI for device management and parameterization.

The DG100 enclosure is made of durable ABS plastic for desktop or wall mounting in home, office or industrial environment. The side and top panels features connectors for power supply, USB, Ethernet and serial communication, push buttons as well as LED indicators.

Using DG100 diverse sensors and devices can be connected to Cloud services. Typical applications are:

- industrial applications (PLCs, HMIs, SCADA,...)
- remote process monitoring (temperature, flow, pressure,...)
- real-time indoor location (hospitals, shopping centers,...)
- smart meter reading (heat meters, electricity, gas, water,...)
- vending machine monitoring
- smart city IoT applications (light, environment, traffic,...)
- agriculture sensing (wireless sensors, pump control,...)

## 2.1 Ordering information

Package includes DG100 in desktop plastic enclosure. For additional equipment see section *2.2 Accessories*.

Model	SKU	Description
DG100	10421	Multi-service IoT Edge Gateway

## 2.2 Accessories

Additional equipment is listed in the table below. More information about accessories can be found in the next chapter or at [www.decode.rs](http://www.decode.rs).

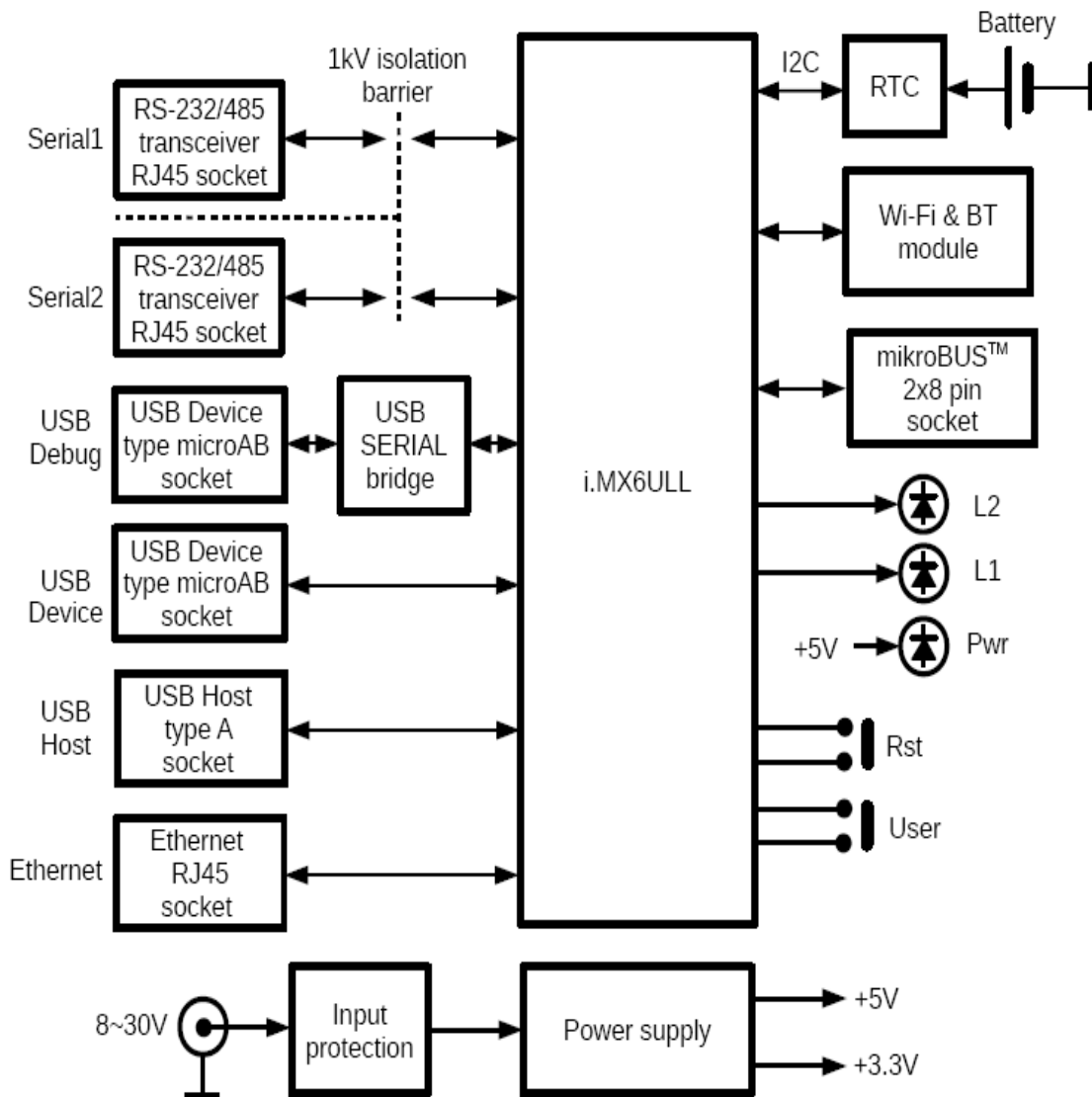
Model	SKU	Description
PS-1212-AD	10261	AC/DC adapter, 5.5x2.1mm, DC12V 12W, AC100-240V 50/60Hz
CB-ETH-0.5M	22133	Ethernet patch cable, RJ45 Male - RJ45 Male, 0.5m
CON-RJ45F-KL8	-	Adapter RJ45 Plug - 8-pin pluggable screw terminal



CON-RJ45F-DB9M	-	Adapter RJ45 Plug - DB9 Male
----------------	---	------------------------------

### 3 Device Description

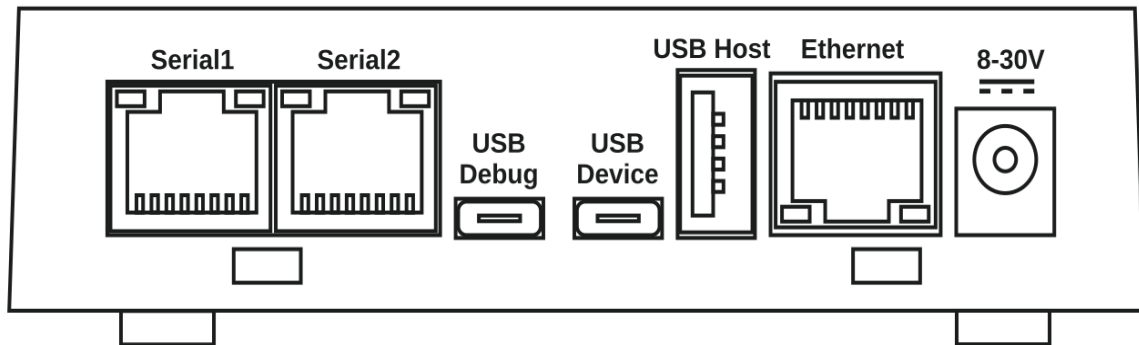
Block diagram depicts internal structure of a device.



**Fig. 1: Block diagram**

## 3.1 External connectors

On the front side panel there are connectors for a power supply and communication interfaces.



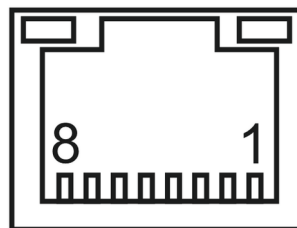
**Fig. 2: External connectors on front side panel**

### 3.1.1 Serial1 and Serial2

These two RJ45 connectors have same pinout and function. They serve as RS232 and RS485 interfaces for i.MX6ULL UARTs. Both ports are galvanically isolated from processor board and power supply. Serial1 is connected to **ttymxc3** and **Serial2** is connected to **ttymxc4**. Both interfaces, RS232 and RS485, are operational at the same time. Data is sent on both interfaces but receiving is possible only from one interface at the time, needing some protocol discipline.

The connector has light indications for transmit (left position - yellow color LED) and receive data (right position - red color LED).

Connector RJ45 pinout is shown on next picture.



**Fig. 3: Serial RJ45 female connector pinout**

Next table shows signals of Serial1 and Serial2 connector.

Pin number	Signal	Description
1	RTS	RS232 output
2	-	not connected
3	GND	Ground
4	TXD	RS232 Output
5	RXD	RS232 Input
6	A (+)	RS485 (+)
7	B (-)	RS485 (-)
8	CTS	RS232 Input

### 3.1.2 USB Debug connector

USB Debug is **microAB** connector type serving as device type. It is used as a Linux "command line" port on **ttymxc0** i.MX6ULL UART. It is based on FT231X USB to serial bridge controller. To connect Debug port to a computer, a VCP driver must be installed on a computer. Any serial terminal program can be used for debug with default communication parameters: 115200bps, 8N1 and no handshaking. Examples in this manual are referenced to Putty terminal. It is an open source software and can be downloaded, free of charge, at <https://www.putty.org/>.

Driver can be found at <http://www.ftdichip.com/Drivers/VCP.htm>.

### 3.1.3 USB Device connector

USB Device is **microAB** connector type serving as device type. It is used for Linux Bootloader recovery. For further information please contact Decode.

### 3.1.4 USB Host connector

USB Host is type A connector serving as host type. It is used as standard Linux Host USB with 500mA power supply capacity. It can handle USB Flash Drive sticks as well as USB serial adapters for RS232 and RS485

### 3.1.5 Ethernet connector

The Ethernet port connector RJ45 supports the 10/100TBase standard. The connector has light indications of activity (left LED) and network presence (right LED). This port allows

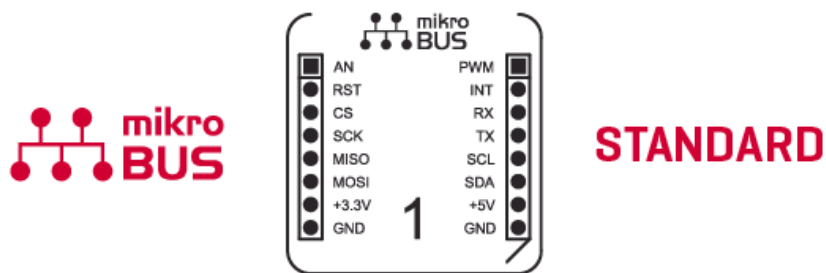
connection to local area networks (LANs), wireless network modems (WLANs), cable modems, GSM / GPRS routers, and similar communication devices.

### 3.1.6 Power connector

DC power source input is a standard DC power jack 5.5x2.1mm with (+) pole in center, compatible with most AC/DC adapters. Power input is from 8 to 30V DC. For power consumption see *Technical Specifications* section.

## 3.2 Internal mikroBUS™ connector

Inside the enclosure there is one Mikroelektronika **mikroBUS™** connector for future expansions. One i.MX6ULL UART **ttymxc1** is routed to mikroBUS.



**Fig. 4: mikroBUS™ connector standard**

Specifications of mikroBUS™ standard can be downloaded from following link:

<https://download.mikroe.com/documents/standards/mikrobus/mikrobus-standard-specification-v200.pdf>

This expansion have up to 5V/2A sourcing capability and can be used for GSM/GPRS/3G/LTE and BT/BLE click boards from Mikroelektronika. Also GPS/GNSS and sub -1GHz transceivers click boards with UART interface can be easily connected to DG100.

For compatibility and support please contact Decode.

### 3.2.1 RTC Battery

Battery for RTC has 10 years lifetime and can be changed only by opening the enclosure. Please contact Decode for further instruction.

## 3.3 Indications and push buttons

### 3.3.1 Push buttons





There are two push buttons on the back side panel. Short press on **Reset** button performs hard reset of device, equivalent to power cycling. **User** button is used for as Linux **gpio84** input found in `/sys/class/gpio/`. For further information please contact Decode.



*Fig. 5: Push buttons on back side plate*

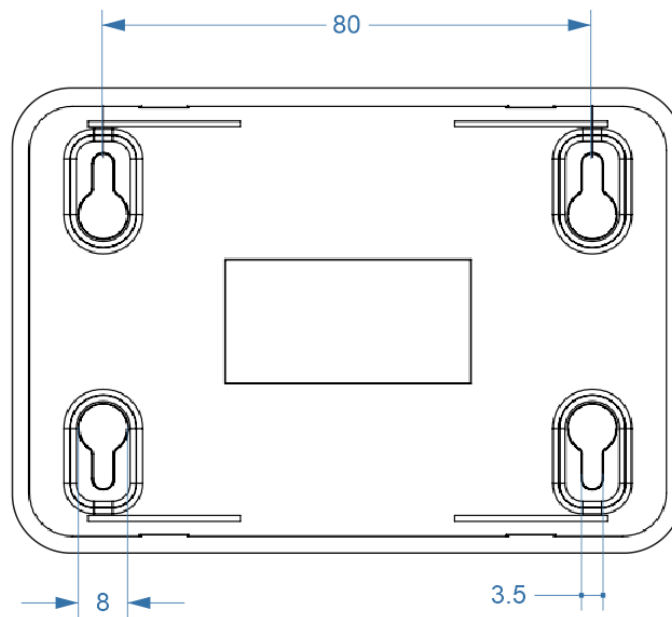
### 3.3.2 LED indicators

Three LEDs on the top panel indicate the presence of the power supply voltage and activity of the device. LED L2 used as Linux **LED2** found in `/sys/class/leds/`. Following table describes device status depending on LED presentation:

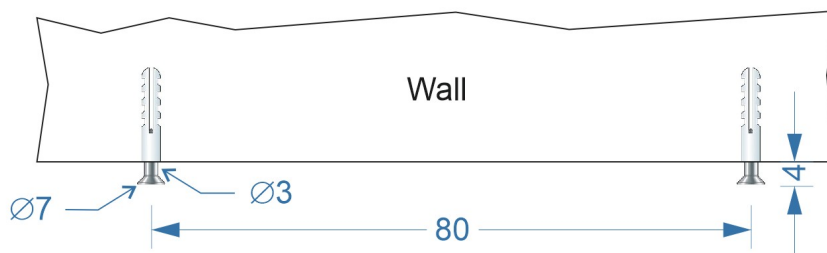
Name	Device state	Presentation	Description
Pwr (green)	Device is OFF		Continuously OFF
	Device power is ON		Continuously ON
L1 (green)	Device is OFF or is not initialized		Continuously OFF
	Device is initialized and operating		Two ON pulses every second Linux heartbeat

### 3.4 Wall or plate mounting

Wall or plate mounting can be realized with two 3mm screws separated by 80mm. Screw head must be approx. 7mm in diameter, and when fastened, head must be 4mm away from the wall or plate.



**Fig. 6: Enclosure mounting dimensions**



**Fig. 7: Wall mounting screws position**

## 4 Operation description

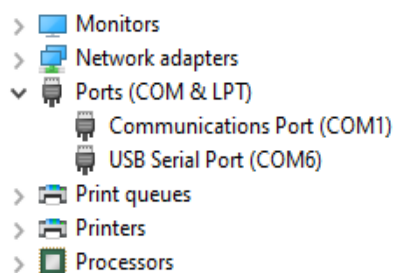
DG100 gateway may be accessed over several interfaces, but in common use two of them are typical: "Command line" on USB Debug COM port or using SSH protocol over TCP network and web user interface using HTTP access over TCP network. To use web user interface connect DG100 to power supply and LAN network with RJ45 patch cable and go directly to *Web interface* section.

### 4.1 Command line interface

For "Command line" communication with DG100 use terminal program such as "PuTTY". It is an open source software and can be downloaded, free of charge, at <https://www.putty.org/>. It may be used for both COM port and SSH access.

#### 4.1.1 Access over USB Debug port

Install VCP drivers on computer (already explained in 3.1.2 *USB Debug connector* section) and connect DG100 USB Debug port to computer USB port using appropriate USB cable. To find COM or TTY port installed open Device Manager on Windows OS or use **dmesg | grep tty** command on Linux OS. In Device Manager open Ports (COM & LPT), find USB Serial Port and memorize COM port number. In following example COM6 is the correct port. Close Device Manager.



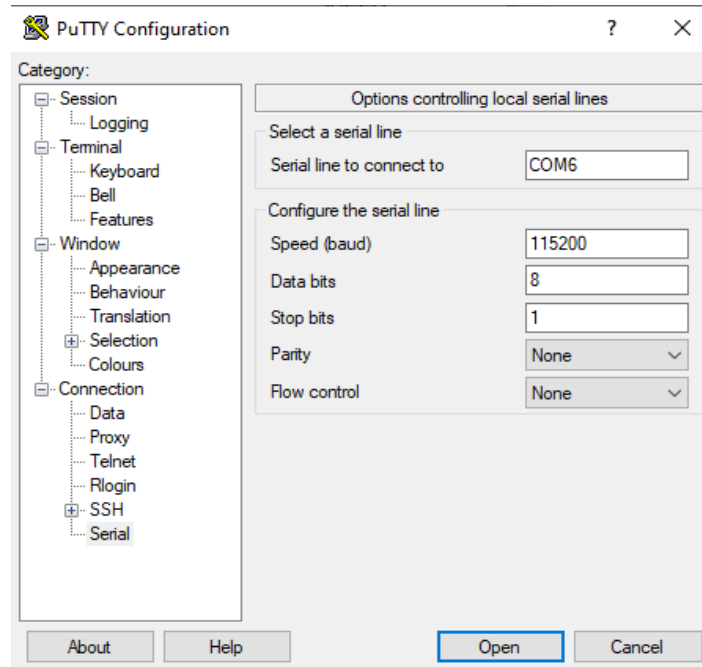
**Fig. 8: List of COM ports**

Start the Putty.



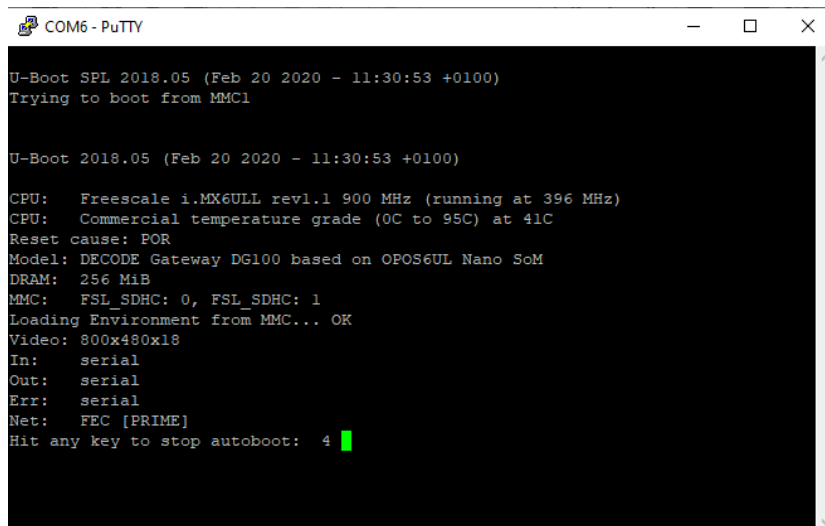
**Fig. 9: PuTTY icon**

Select Serial radio button and from Category menu Connection-Serial. Enter the memorized COM port and set serial port parameters 115200bps, 8bits, no parity bit, 1 stop bit and no RTS/CTS control.



**Fig. 9: Settings serial parameters**

Power on device by connecting power supply.



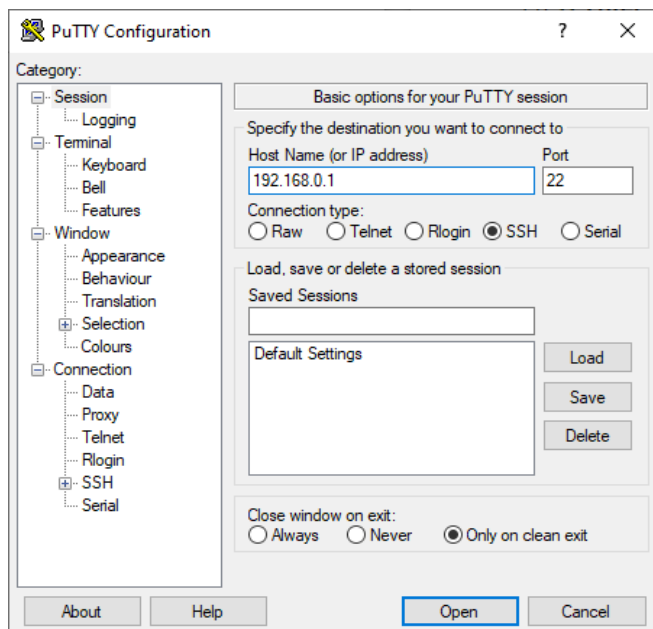
**Fig. 10: DG100 is booting**

Device starts booting and after approximately 30 seconds DG100 is operational. User may see booting process in terminal window. To use web user interface connect DG100 to LAN network with RJ45 patch cable and go directly to 5 *Web interface* section.



### 4.1.2 Access over SSH

To access DG100 device using "Command line", in local network or remotely, SSH protocol is used.



**Fig. 11: SSH connection**

Enter the default IP address and default port 22 and click **Open** button. Login parameters are **root** for username and **root** for password.

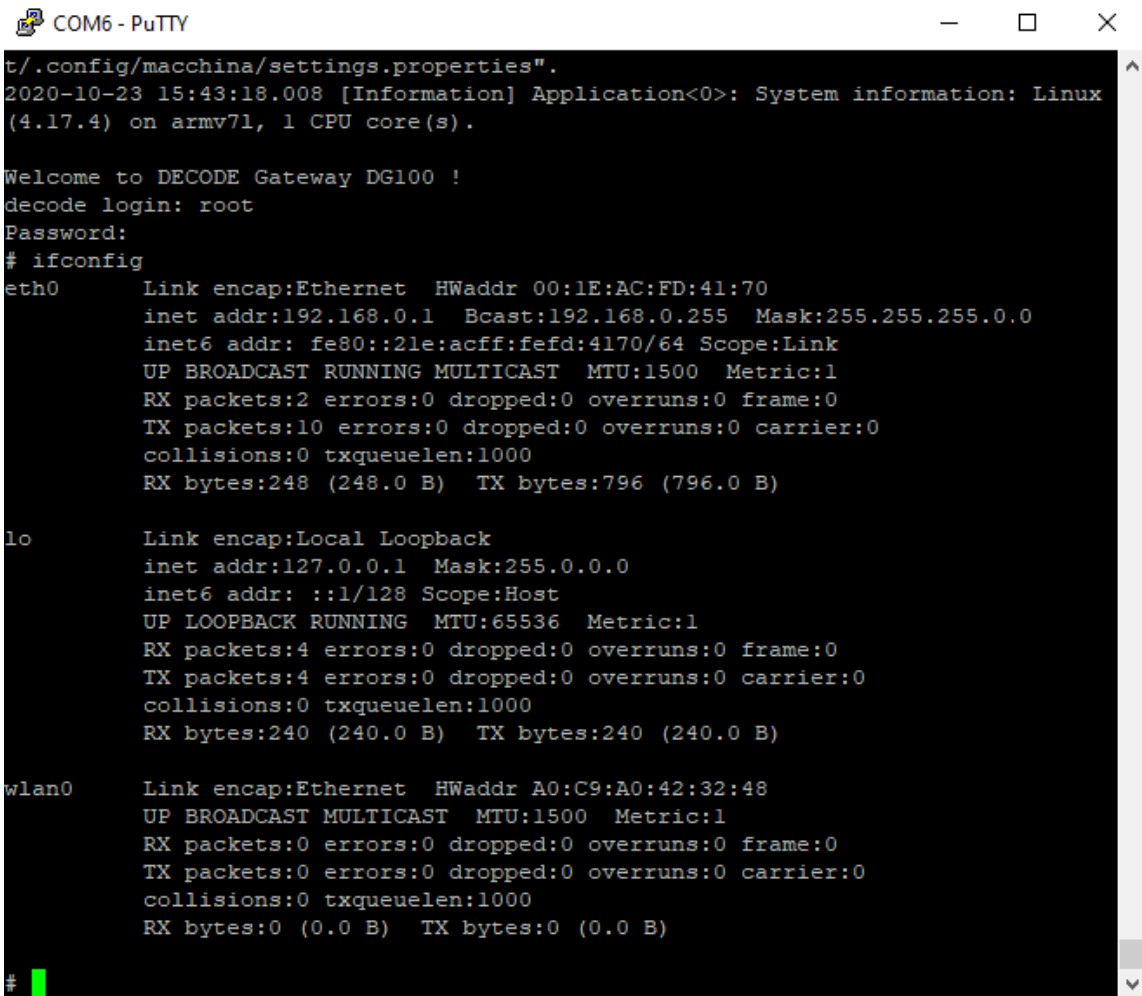


**Fig. 12: SSH login**

## 4.2 Basic commands

Press **Enter** on a computer keyboard and type **root** for username and **root** for password when asked. Press **Enter** to confirm each entry.

```
Welcome to DECODE Gateway DG100 !
decode login: root
Password: root
```



```
COM6 - PuTTY
t/.config/macchina/settings.properties".
2020-10-23 15:43:18.008 [Information] Application<0>: System information: Linux
(4.17.4) on armv7l, 1 CPU core(s).

Welcome to DECODE Gateway DG100 !
decode login: root
Password:
# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:1E:AC:FD:41:70
          inet addr:192.168.0.1  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::21e:acff:fe4d:4170/64  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2 errors:0 dropped:0 overruns:0 frame:0
          TX packets:10 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:248 (248.0 B)  TX bytes:796 (796.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128  Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:4 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:240 (240.0 B)  TX bytes:240 (240.0 B)

wlan0     Link encap:Ethernet  HWaddr A0:C9:A0:42:32:48
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

#
```

**Fig. 13: Network parameters**

## 4.2.1 Checking network parameters

```
# ifconfig
```

## 4.2.2 Adjusting the Linux time

```
# date -s 2020.01.29-12:00
```

## 4.2.3 Save Linux time in RTC

```
# hwclock -w
```

## 4.2.4 Check RTC time

```
# hwclock -r
```

## 4.2.5 Check installed TTY ports

```
# dmesg | grep tty
```

Response contains all installed and uninstalled tty devices.

```
[ 0.000000] Kernel command line: console=ttymx0,115200 root=/dev/mmcblk0p2 ro rootfstype=ext4
rootwait
[ 0.666273] 2020000.serial: ttymxc0 at MMIO 0x2020000 (irq = 21, base_baud = 5000000) is a IMX
[ 1.374970] console [ttymxc0] enabled
[ 1.383594] 21e8000.serial: ttymxc1 at MMIO 0x21e8000 (irq = 51, base_baud = 5000000) is a IMX
[ 1.396235] 21ec000.serial: ttymxc2 at MMIO 0x21ec000 (irq = 52, base_baud = 5000000) is a IMX
[ 1.408944] 21f0000.serial: ttymxc3 at MMIO 0x21f0000 (irq = 53, base_baud = 5000000) is a IMX
[ 1.421721] 21f4000.serial: ttymxc4 at MMIO 0x21f4000 (irq = 54, base_baud = 5000000) is a IMX
#
```

## 4.2.6 Stop DG100 program

DG100 program is started as a process. To stop it, first of all process number must be found.

```
# ps
```

Response contains a list of all processes. Find lines with two DG100 processes and memorize its numbers.

```
196 root {DG100AppRestart} /bin/sh /root/DG100AppRestart.sh
205 root /root/macchina/bin/macchina -config=/root/macchina/etc/macchina.
```

Stop the processes with memorized numbers by issuing following commands:

```
# kill 196
# kill 205
```

After second command, shutdown process is seen, ending with **Shutdown complete** message.

```
2020-09-03 12:22:21.153 [Information] Application<0>: Shutdown complete.
```

## 4.2.7 Reboot

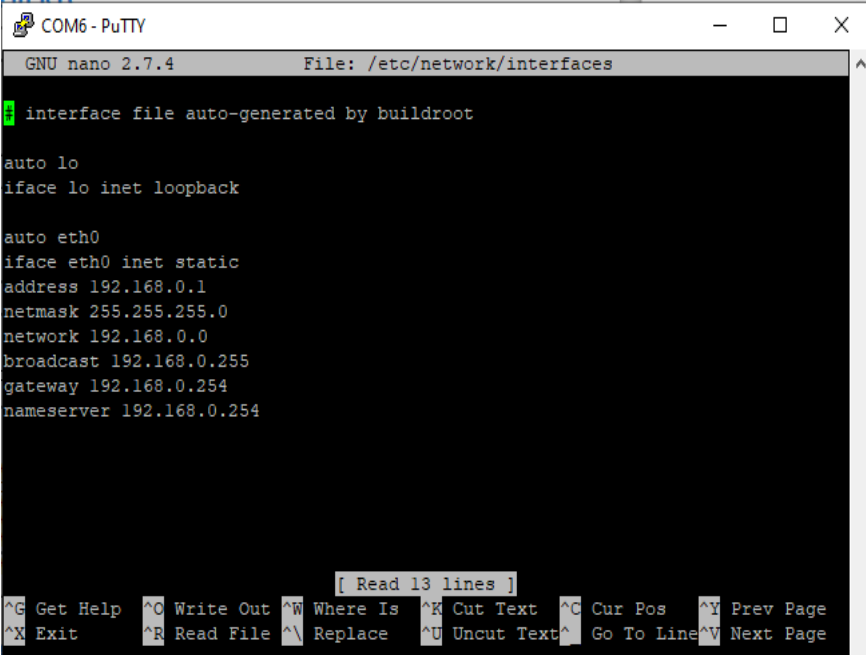
```
# reboot
```

Reboot can be done also by power cycling or pressing **Reset** button, but this method is recommended and more secure.

## 4.2.8 Changing network settings

To change network settings, first stop DG100 program and start **nano** editor to edit configuration file.

```
# nano /etc/network/interfaces
```



```
COM6 - PuTTY
GNU nano 2.7.4 File: /etc/network/interfaces
interface file auto-generated by buildroot

auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.0.1
netmask 255.255.255.0
network 192.168.0.0
broadcast 192.168.0.255
gateway 192.168.0.254
nameserver 192.168.0.254

[ Read 13 lines ]
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^C Cur Pos  ^Y Prev Page
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text ^_ Go To Line ^V Next Page
```

**Fig. 14: Edit network settings**

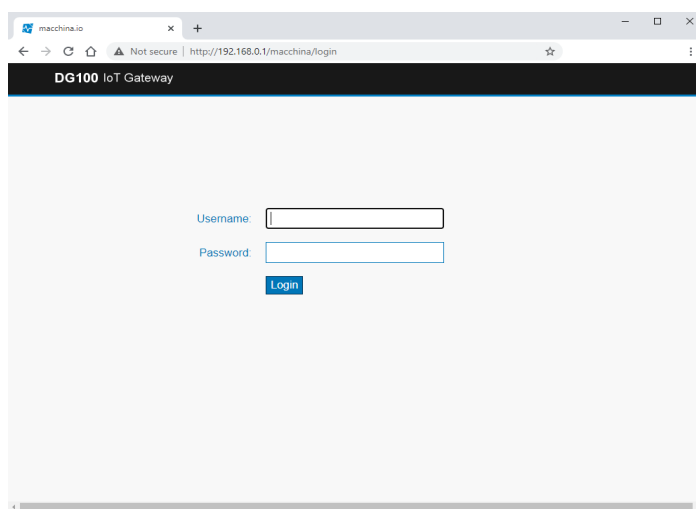
To exit editor click **Ctrl+X** buttons on keyboard and then restart LAN by issuing following commands:

```
# ifdown eth0  
# ifup eth0
```

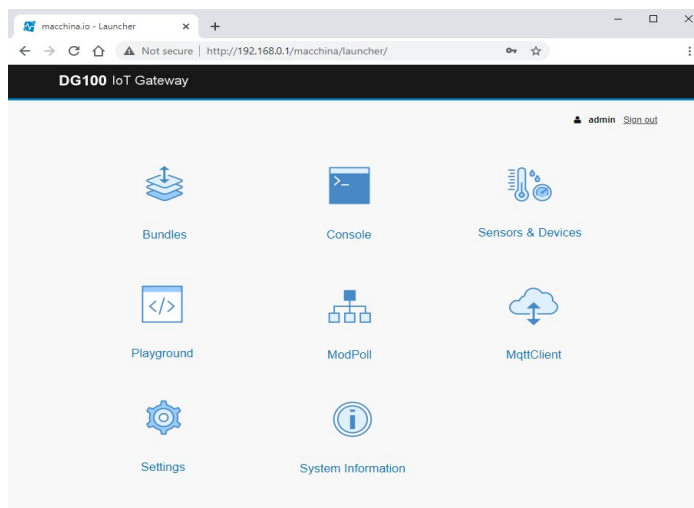
Rebooting device also makes changes active. Be aware that when changing the network settings always remember last working parameter settings.

## 5 Web user interface

Web user interface is implemented using built-in web server. It is accessed via a web browser by entering the device IP address in browser address bar. Default IP address is **192.168.0.1** and server default port is **80**. On the starting page, enter username and password and click on Login button. Default username is **admin** and default password is **admin**.



**Fig. 15: Login screen**

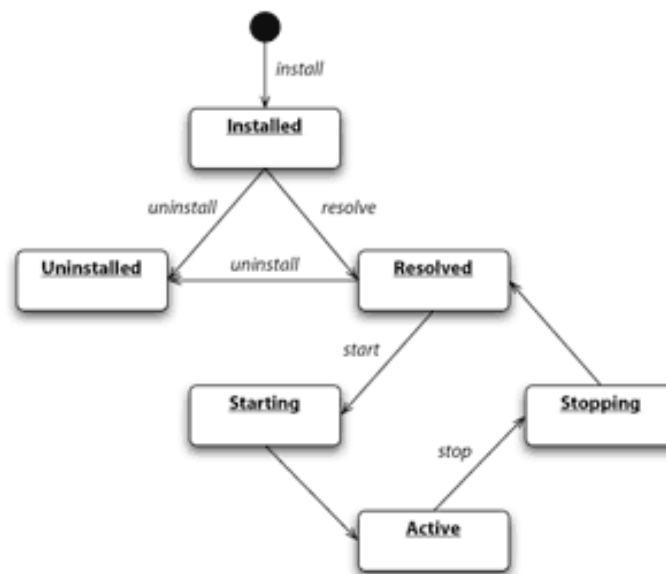


**Fig. 16: Application icons**

After successful logging in, the user sees a grid of app icons. Clicking an icon will "launch" the respective app.

Installed apps are deployed on DG100 in a form of packed directory called bundle. New apps can be added simply by installing new bundles in "Bundles" app.

DG100 is based on macchina.io framework with OSP (Open Service Platform) architecture providing Bundle Life Cycle Management which ensures that every bundle in the system follows this life cycle, which is outlined in the following.



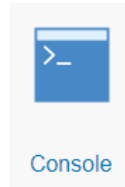
**Fig. 17: Bundle Life Cycle Management**

For every bundle that is found on DG100 the initial state is installed. Once all bundles have been found, OSP tries to resolve each bundle. Resolving a bundle means checking whether all required bundles of a bundle are available. Every bundle that has been successfully resolved enters the resolved state. A resolved bundle will eventually be started. A bundle that is started is first put into starting state. Then, all required bundles are started as well.

Once all required bundles have entered active state, the bundle's activator is invoked. When the activator completes, the bundle is finally put into active state. Eventually, at least at shutdown time, the bundle will be stopped. Stopping a bundle means putting the bundle into stopping state, invoking the bundle's activator, then putting the bundle into resolved state. At shutdown, OSP ensures that all bundles are shutdown in the correct order. A bundle in resolved or installed state can be uninstalled, which means it is completely removed from the system.

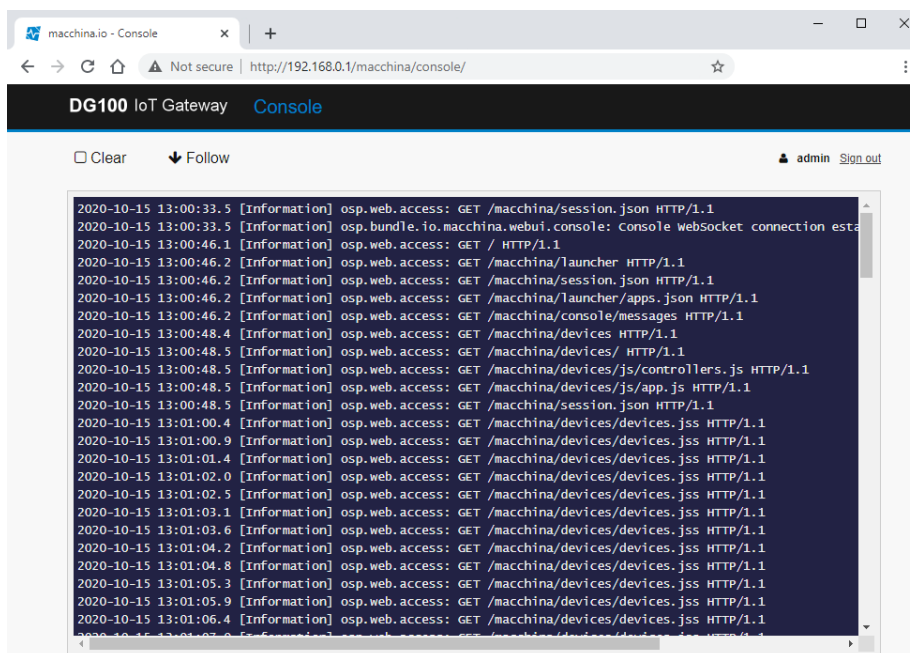
## 5.1 Console

By clicking on **Console** icon "Console" program is started.



**Fig. 18: Console App icon**

"Console" application provides a way for the applications to output text-based messages to the user. Every line in console starts with timestamp and holds other useful informations such as message type and application name.



**Fig. 19: Console**

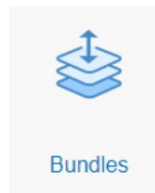
To clear "Console" window click on **Clear** icon, and to move to last line of scrolled window click on **Follow** icon.

Several separate "Console" windows can be opened at the same time.



## 5.2 Bundle management

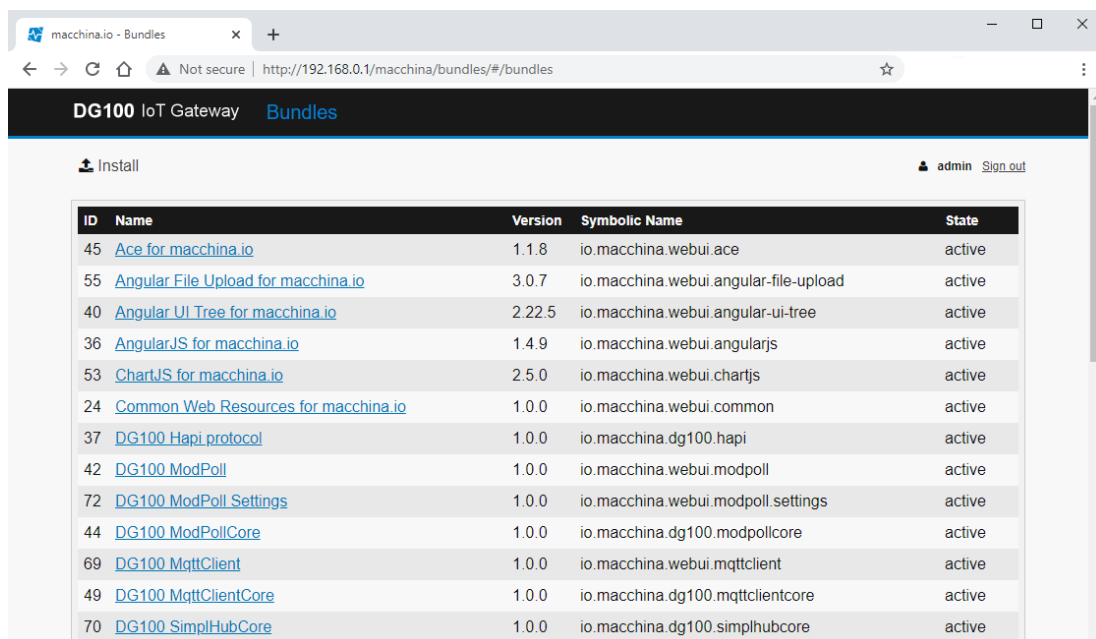
By clicking on **Bundles** icon, "Bundles" program is started. Several operation on bundles may be performed using this program: starting, stoping, installation, uninstalling, upgrading and viewing information.



**Fig. 20: Bundles App icon**

### 5.2.1 View bundles info


When "Bundles" program is started, list view is shown with short info of installed bundles: ID, name, version, symbolic name and state. By clicking on column name, sorting is performed on rows in ascending/descending order.

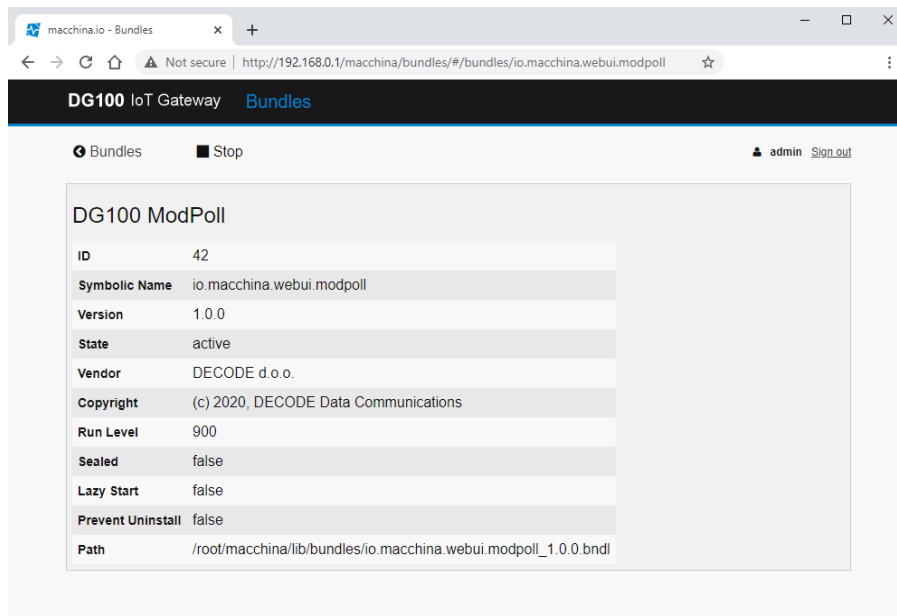
A screenshot of a web browser displaying the Bundles application. The browser address bar shows "http://192.168.0.1/macchina/bundles/#/bundles". The page header includes "DG100 IoT Gateway" and "Bundles". Below the header, there is a table with columns: ID, Name, Version, Symbolic Name, and State. The table lists 15 bundles, all with a state of "active".

ID	Name	Version	Symbolic Name	State
45	<a href="#">Ace for macchina.io</a>	1.1.8	io.macchina.webui.ace	active
55	<a href="#">Angular File Upload for macchina.io</a>	3.0.7	io.macchina.webui.angular-file-upload	active
40	<a href="#">Angular UI Tree for macchina.io</a>	2.22.5	io.macchina.webui.angular-ui-tree	active
36	<a href="#">AngularJS for macchina.io</a>	1.4.9	io.macchina.webui.angularjs	active
53	<a href="#">ChartJS for macchina.io</a>	2.5.0	io.macchina.webui.chartjs	active
24	<a href="#">Common Web Resources for macchina.io</a>	1.0.0	io.macchina.webui.common	active
37	<a href="#">DG100 Hapi protocol</a>	1.0.0	io.macchina.dg100.hapi	active
42	<a href="#">DG100 ModPoll</a>	1.0.0	io.macchina.webui.modpoll	active
72	<a href="#">DG100 ModPoll Settings</a>	1.0.0	io.macchina.webui.modpoll.settings	active
44	<a href="#">DG100 ModPollCore</a>	1.0.0	io.macchina.dg100.modpollcore	active
69	<a href="#">DG100 MqttClient</a>	1.0.0	io.macchina.webui.mqttclient	active
49	<a href="#">DG100 MqttClientCore</a>	1.0.0	io.macchina.dg100.mqttclientcore	active
70	<a href="#">DG100 SimplHubCore</a>	1.0.0	io.macchina.dg100.simplhubcore	active

**Fig. 21: Bundles info**

## 5.2.2 Starting, stopping, upgrading and uninstalling bundles

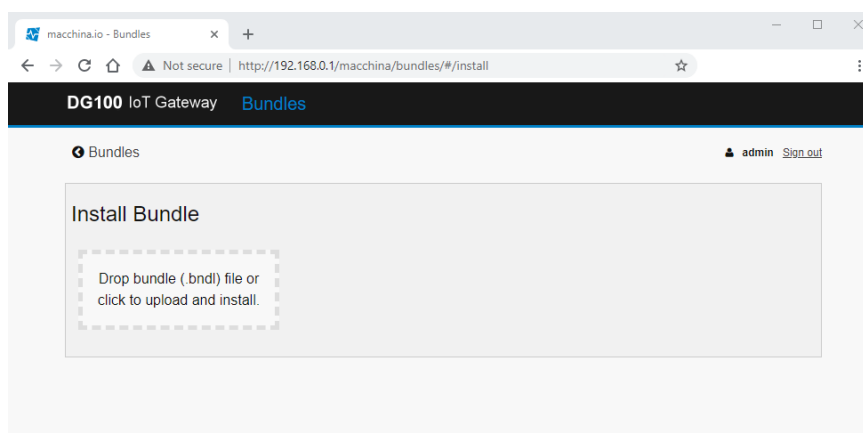
Click on specific bundle name opens more detailed bundle information with additional controls. Active bundles can be stopped, resolved bundles can be started, upgraded or uninstalled by clicking to corresponding icons. Click on  **Bundles** to return to list view.



**Fig. 22: Stopping active bundle**

## 5.2.3 Installing new bundles

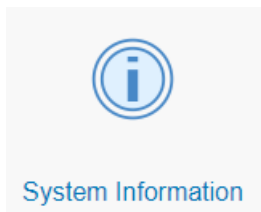
Clicking on Install icon from bundles list view opens the **Install** bundle view. New bundle can be uploaded and installed by dragging and dropping bundle file in framed rectangle or by clicking framed rectangle, to open file search dialog window. Only bundle files with **.bndl** extension can be uploaded and installed.



**Fig. 22: Installing new bundles**

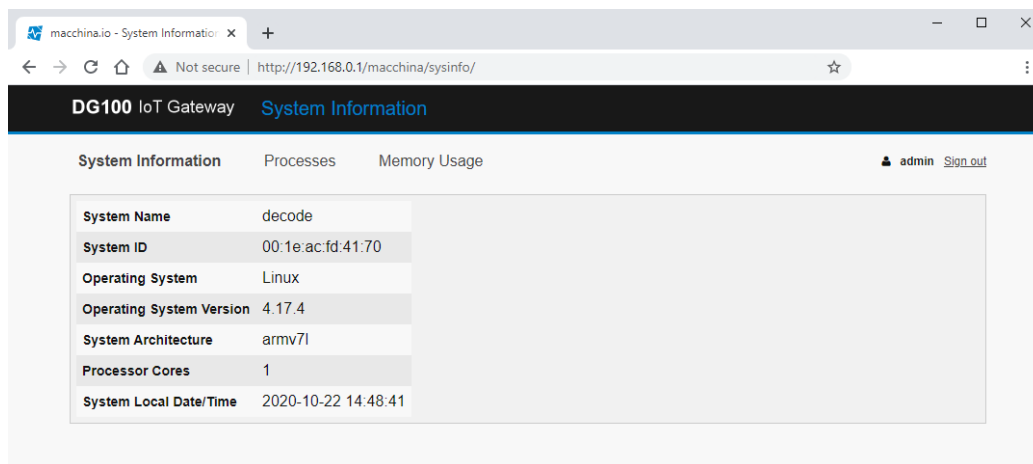
## 5.3 System Information

By clicking on **System Information** icon "System Information" program is started.



**Fig. 23: System Information App icon**

This application provides many useful informations about DG100 device, operating system and memory resources.

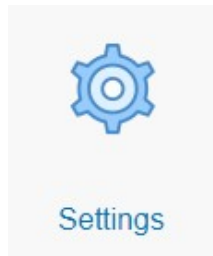


**Fig. 24: System Information**

Navigation is possible using tabs **System Information**, **Processes** and **Memory Usage**.

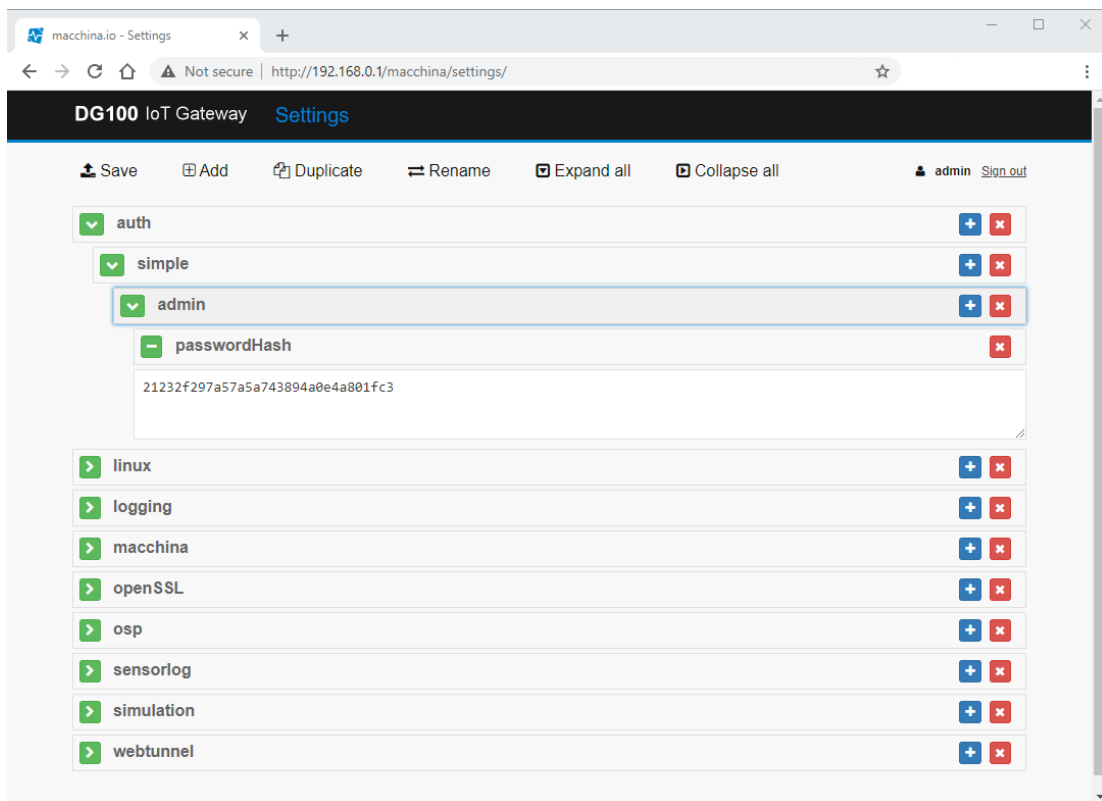
## 5.4 Settings

By clicking on **Settings** icon, "Settings" program is started.



**Fig. 25: Settings App icon**

It provides a user interface for macchina.io properties configuration in a form of editable tree view structure.

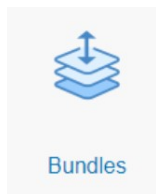


**Fig. 26: Settings**

By clicking on icons user can save configuration, add, duplicate and rename items and expand and collapse the tree view. Clicking on green icon at the beginning of each row tree item can be expanded and collapsed.

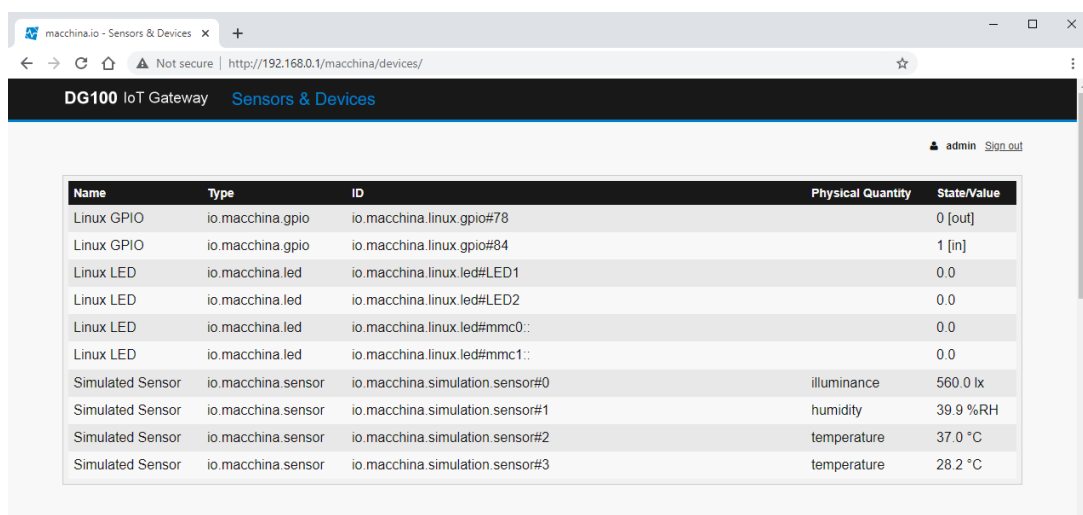
## 5.5 Sensors & Devices

By clicking on **Sensors & Devices** icon "Sensors & Devices" program is started.



**Fig. 27: Sensors & Devices App icon**

It provides a overview of all sensors and devices registered in OSP Service Registry with physical quantities automatically updated at one second interval.

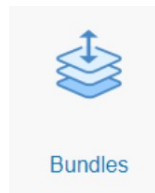
A screenshot of a web browser displaying the "Sensors & Devices" page of the DG100 IoT Gateway. The browser address bar shows "http://192.168.0.1/macchina/devices/". The page header includes "DG100 IoT Gateway" and "Sensors & Devices". A user profile "admin" with a "Sign out" link is visible in the top right. The main content is a table with the following data:

Name	Type	ID	Physical Quantity	State/Value
Linux GPIO	io.macchina.gpio	io.macchina.linux.gpio#78		0 [out]
Linux GPIO	io.macchina.gpio	io.macchina.linux.gpio#84		1 [in]
Linux LED	io.macchina.led	io.macchina.linux.led#LED1		0.0
Linux LED	io.macchina.led	io.macchina.linux.led#LED2		0.0
Linux LED	io.macchina.led	io.macchina.linux.led#mmc0::		0.0
Linux LED	io.macchina.led	io.macchina.linux.led#mmc1::		0.0
Simulated Sensor	io.macchina.sensor	io.macchina.simulation.sensor#0	illuminance	560.0 lx
Simulated Sensor	io.macchina.sensor	io.macchina.simulation.sensor#1	humidity	39.9 %RH
Simulated Sensor	io.macchina.sensor	io.macchina.simulation.sensor#2	temperature	37.0 °C
Simulated Sensor	io.macchina.sensor	io.macchina.simulation.sensor#3	temperature	28.2 °C

**Fig. 28: Sensors & Devices**

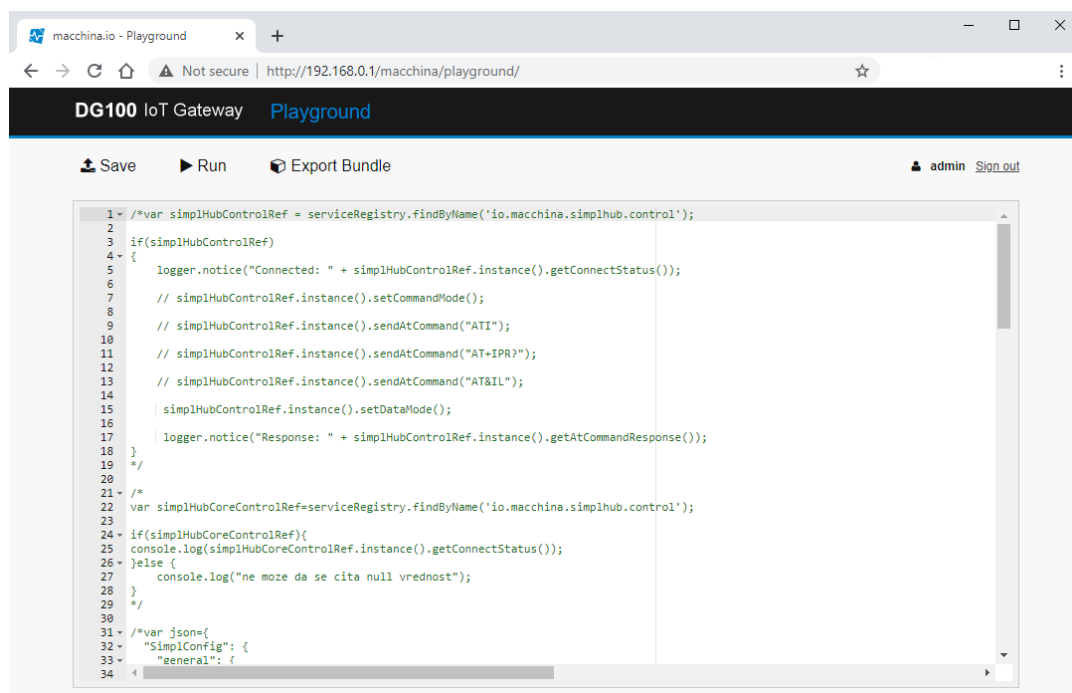
## 5.6 Playground

In general, all JavaScript code executed within DG100 must be contained in bundles. There are different ways to create a bundle and deploy it into a DG100 server. The easiest way is to use the Playground app in the web user interface by clicking on **Playground** icon.



**Fig. 29: Playground App icon**

It provides a browser-based editor that allows you to create and edit JavaScript scripts, as well as to execute them in DG100. This is great for trying out things very quickly.



**Fig. 30: Playground**

Although JavaScript is intuitive and beginner-friendly, some basic JavaScript language concepts and knowledge must be learned for effective programming of DG100.

To better understand JavaScript programming on DG100 using macchina.io framework use following link:

<https://macchina.io/docs/00200-MacchinaJSProgramming.html>

For a functional description and description of programming interface of Decode bundles installed on DG100 please use specific bundle programming reference document which could be obtained from Decode support. For two standard DG100 Decode bundles ModPoll and MqttClient functional description and programming reference are given in following chapters.

### 5.6.1 Writing JavaScript programs

JavaScript program can be typed in Playground editor or can be pasted to web editor from another page or document. Note that when working with the Playground app, it's a good idea to have a nearby console window with macchina.io's log output open to see script output or error messages. You can also open the "Console" app in a separate browser window.

The best way to start programming in Playground is to write simple "Hello World!" script that outputs the [Information] message "Hello, World!" to web Console window. Following script is version which additionally outputs the [Error] message "No Error!" to show possibility to output different types of messages in different colors using logger function.

```
//  
// Playground Hello World Script  
//  
// This script will print Hello World! message in Console window  
// in white color and No Error! message in red color  
//  
  
var message = "Hello World!";  
  
logger.information(message);  
logger.error('No Error!');
```

Clicking on Run icon script is saved and started. After several seconds, console output should print following messages:

```
2020-10-22 12:35:57.7 [Information] osp.bundle.io.macchina.webui.playground.sandbox: Hello world!  
2020-10-22 12:35:57.7 [Error] osp.bundle.io.macchina.webui.playground.sandbox: No Error!
```

Already started bundle can be Restarted or Stopped by clicking to respective icon.

DG100 is based on OSP (Open Service Platform) and Service Registry enabling user to write JavaScript applications that use a services provided by other bundles. A JavaScript program can discover a certain service by using a simple query language to query the Service Registry for a service with certain properties. Since bundles can appear and disappear in an

application at any time, the Service Registry also provides notification mechanisms so that a bundle can be informed when another bundles it uses disappears from the system.

More information about OSP and interfacing with Service Registry may be found on following link:

<https://macchina.io/docs/00100-OSPOverview.html>

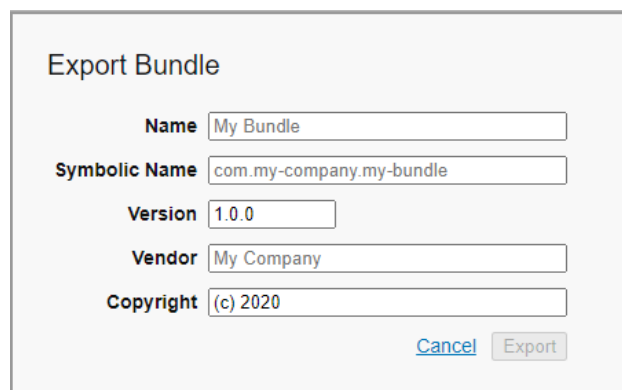
## 5.6.2 Saving JavaScript program

To Save the content of JavaScript editor on the filesystem, click on Save. Saved content will not be lost after DG100 power down.

## 5.6.3 Exporting JavaScript program to bundle

The Playground can also be used to pack JavaScript program to bundle and download a complete bundle containing the script. This bundle can then be deployed to a device, either by placing it on the filesystem (in one of the directories configured as bundle repositories), or by installing them using the web based Bundle utility.

Begin with a click on Export Bundle. Modal window is opened.



The image shows a modal window titled "Export Bundle". It contains five input fields for metadata: "Name" (My Bundle), "Symbolic Name" (com.my-company.my-bundle), "Version" (1.0.0), "Vendor" (My Company), and "Copyright" ((c) 2020). At the bottom right, there are two buttons: "Cancel" and "Export".

**Fig. 31: Export bundle modal window**

User must enter **Name**, **Symbolic name** and **Vendor information**. Version and Copyright information is set by default to **1.0.0** and **(c) 2020** and can be changed.

After clicking to Export button, JavaScript program is packed and downloaded as one bundle file with **.bndl** extension. Bundle can be installed using "Bundles" app.



## 6 Technical specifications

### 6.1 Processor board

Central processor	32-Bit MCU Single Core, ARM A7, i.MX6ULL, 400MHz (max. 900MHz)
Memory	256MB DDR3 SDRAM (up to 512MB factory option)
Storage	eMMC 4GB Flash (up to 32GB factory option)
USB	1 x USB2.0 Host type A, 1 x USB2.0 Device type microAB
USB Debug	1 x USB2.0 device type microAB, FT231X USB to serial bridge controller
Ethernet	RJ45, 10/100TBase
Real Time Clock (RTC)	Implemented (powered by a CR1220 battery)

### 6.2 Serial ports

Serial ports	2 x RS-232C (TD, RD, RTS, CTS signals) and RS-485 (A+ & B-) Galvanically isolated to 1kV with send and receive LED indicators
--------------	--

### 6.3 Wireless

WiFi	Single-band 2.4 GHz IEEE 802.11b/g/n
Bluetooth	BT 4.1 (Low Energy compatible)

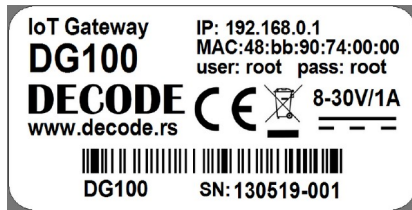
### 6.4 Common characteristics

Power supply	8~30V DC,
Power consumption	1W Typical, 15W Max
Enclosure protection	IP40
Temperature range	from 0°C to +55°C and from 0 to 95% RH (without condensation) (industry option available from -20°C to +65°C)
Dimensions	110 x 76 x 27mm
Mounting	Desktop, Wallmount

### 6.5 Software

OS	Linux Buildroot 4.17.4 or Yocto 4.9.88
SDK	Buldroot-based Eclipse Tooling
IoT Framework	macchina.io C++ and JavaScript IoT Edge Device SDK

## 7 Product label



**Fig. 31: Product label**

The label fixed on the right side of enclosure comprises information listed in next table.

<b>Line 1</b>	Product name		Additional information about product	
<b>Line 2</b>	Product model			
<b>Line 3</b>	Manufacturer	CE sign	Waste Disposal	Supply voltage
<b>Line 4</b>	Manufacturer address			Maximum current
<b>Line 5</b>	Bar code with Product ID and Serial number			
<b>Line 6</b>	Product ID		Serial number	

## 8 Disposal and Recycling



You must dispose of this product properly according to local laws and regulations. Because this product contains electronic components, it must be disposed of separately from household waste. When this product reaches its end of life, contact local authorities to learn about disposal and recycling options, or simply drop it off at your local Decode office or return it to Decode.

## 9 Contact

Please contact a Decode office if you have any questions regarding the information contained in this manual or Decode products, or if you have any other inquiries.

### **Decode d.o.o.**

30A Nikole Tesle Blvd,

11080 Belgrade, Serbia

Tel./Fax. +(381 11) 311 00 27

Email: office@decode.rs

Web: www.decode.rs