



Cloud.Node APPLICATION NOTE

By SeeedStudio & DRAGINO TECHNOLOG CO LIMITED

VERSION: 1.0

2013.May.28

Index:

1.	Introduction	3
1.1	Overview of Dragino and Cloud.Node	3
1.2	Dragino in an IoT project	3
1.3	Features of the IoT firmware	4
2.	Interface to Daughter Board	5
2.1	Introduction	5
2.2	Commands	6
2.2.1	ADD a NODE	6
2.2.2	POST DATA	6
2.2.3	DELETE a NOTE	7
3.	Firmware of Daughter Board	8
3.1	Software Flowchart	8
3.2	Main Function	8
3.2.1	Add a Node Device	8
3.2.2	Post data to Yeelink	8
4.	Ways to access Dragino MS12	9
4.1	Access MS12 via WEB	9
4.2	Access MS12 via SSH	10
4.3	Get or Put files from/to Dragino	11
5.	WEB GUI manual	12
5.1	Sensor Settings	12
5.2	Wi-Fi Settings	14
5.3	Network Interfaces	15
5.4	System Status	16
5.5	DDNS settings	17
5.6	Upgrade Firmware	18
1	REFERENCE	19

1. Introduction

1.1 Overview of Dragino and Cloud.Node

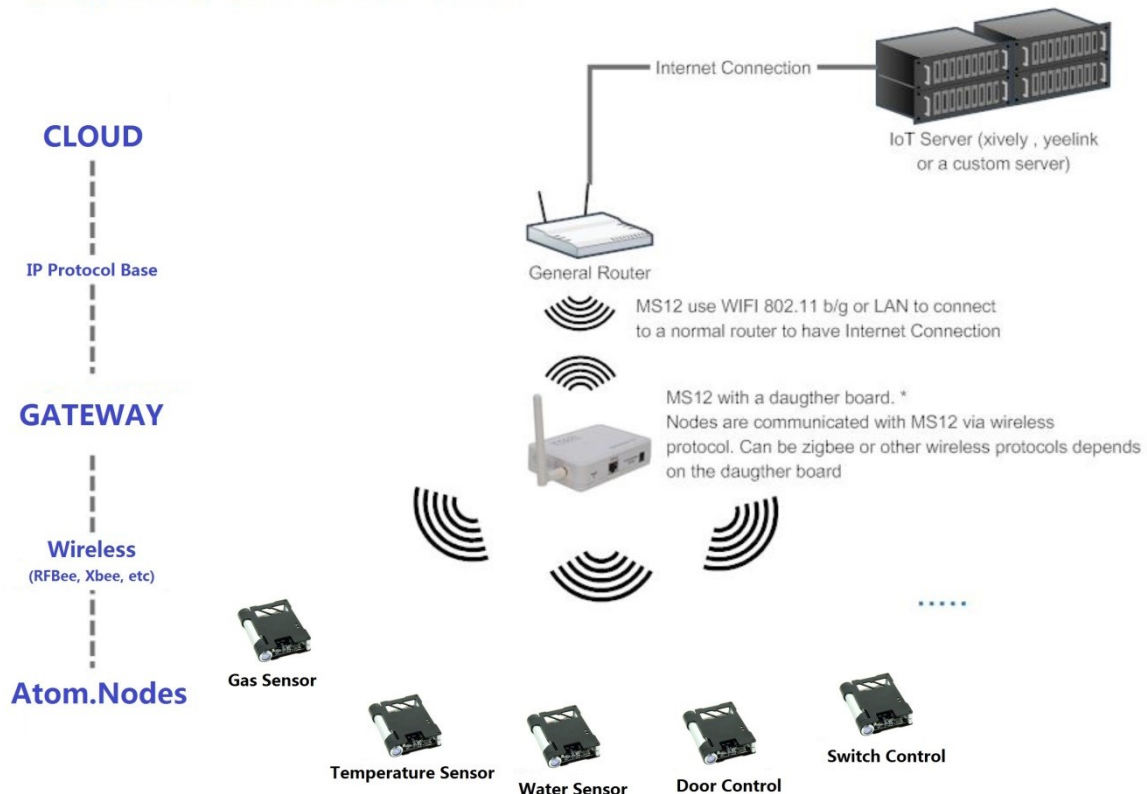
Node is a platform for IoT. It reads physical data from the Sensor and accomplishes certain actions via the Actuator. A Sensor is an electronic object that gets physical values from the environment. For instance, a Grove-Temperature Sensor reads temperature data and transports it to Beacon. An Actuator makes some certain actions. As an example, Grove-LED will blink when a HIGH level is sent to it.

Node is found by two parts, Atom and Cloud. You can use a Cloud to send data to Yeelink, Xively etc.

Dragino MS12 is an open source, Wi-Fi/Linux enable appliance for MCU project. The goal of the Dragino is to solve the connectivity problem and greatly enhance microcontroller products such as the Arduino. Dragino MS12 is motherboard base and it normally need to work with different kind of plug-in daughter boards for different projects.

1.2 Dragino in an IoT project

Dragino MS12 in an IoT cloud



Note: The daughter is a separate part from MS12, a reference can be found on [dragrove](http://dragrove.com)

Above is a structure to show how to use Dragino MS12 to develop IOT products. In this structure, there are three main parts:

- **IoT Server:** they are servers which store data from the sensor and process these data in its manner (for example plotting the data for easy reading). IoT servers are not subject in monitoring; they may also send commands to control the sensors.
A public server example is www.xively.com (formally pachube or cosm)
- **Gateway:** Sensors normally can't communicate to IoT servers directly. They need a gateway. Here MS12 plus a daughter board acts as a gateway, the daughter board get data from sensors and send it to MS12, MS12 then send the data to IoT server via TCP/IP protocol.
- **Nodes:** sensors or controllers

Dragino has developed a firmware for IoT (Internet of Things) applications. The firmware can be found on this link: http://wiki.dragino.com/index.php?title=Release_Note

1.3 Features of the IoT firmware

1. Completely Open Source.
2. RESTful server compatibility
 - Compatible with Yeelink, xively RESTful server
 - Automatically create nodes in RESTful server
 - Automatically update sensor data to RESTful server
 - Automatically delete nodes from RESTful server
3. Support custom commands to process sensor data
4. Support record data to local server
5. Support internet connection via Wi-Fi or LAN port.
6. Support DDNS (Dynamic DNS) service
7. Support Firmware Upgrade via Web GUI

2. Interface to Daughter Board

2.1 Introduction

In the IoT firmware, the IoT daemon runs in background. It will get all data from the UART port on the MS12 2x8 connector and parse these raw data as commands. To differentiate the commands, it requires the daughter board sends the commands in certain format to its UART port.

The general format for a command string is:



START: "ss"

DATATYPE: 1: ADD NODE; 2: POST DATA; 3: DELETE NODE

SPACE: a space between DATATYPE and DATA

DATA: Valid Data

END: "gg"

For the hardware connection detail, please refer [the hardware connection detail from the REFERENCE](#).

2.2 Commands

2.2.1 ADD a NODE

Format: “ss1 NODE_ID, SENSOR_ID, ACTUATOR_IDgg”

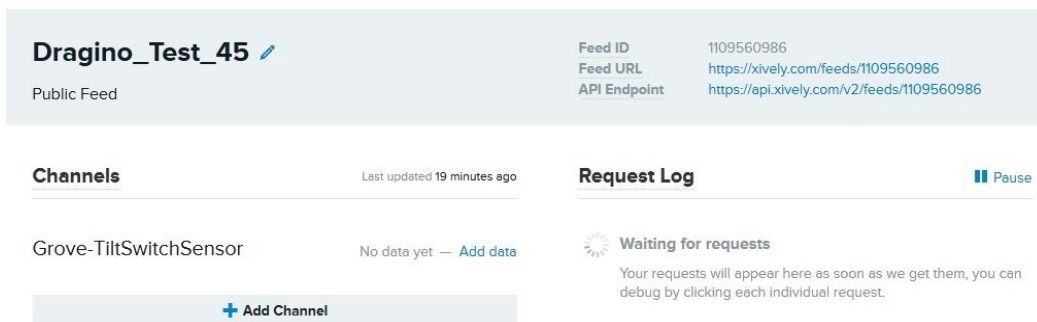
Function: Create a Node with Specify SENSOR_ID and ACTUATOR_ID

Example: “ss 1 45,2,201gg”

Create a new Node, with node_id 45, sensor id 2, and actuator_id 201.

Once MS12 get this command from its UART port, MS12 will:

1. Get the sensor info from local sensor_table in /usr/lib/sensor/
2. Get the actuator info from local actuator_table in /usr/lib/sensor/
3. Create a node in the IoT server, we use xively here so the node will be created in xively server as below:



The screenshot shows the Xively IoT server interface. At the top, there's a header for the feed 'Dragino_Test_45' with a public feed status. Below this, there are fields for Feed ID (1109560986), Feed URL (https://xively.com/feeds/1109560986), and API Endpoint (https://api.xively.com/v2/feeds/1109560986). The main content area is divided into two sections: 'Channels' and 'Request Log'. The 'Channels' section shows a single channel 'Grove-TiltSwitchSensor' with a status of 'No data yet' and an 'Add data' button. There is also an 'Add Channel' button. The 'Request Log' section shows a status of 'Waiting for requests' with a message: 'Your requests will appear here as soon as we get them, you can debug by clicking each individual request.'

1. Create the node info locally as below:



The screenshot shows the 'IoT Server Configure' form. It has two main sections: 'IoT Server Configure' and 'Devices List'. The 'IoT Server Configure' section has fields for 'Enable IoT Service' (checked), 'IoT Service' (xively), 'Internet Connection to xively.com' (Up), 'API Key' (NkYHeIFJaC4DPkGDx20ZiW2Jg7AJc36EY0oHTR), and 'Title' (Dragino_Test). The 'Devices List' section has a title 'Devices List' and a subtitle 'Below devices are connected to Dragino'. It shows a 'Device ID: 45' and a table with columns 'Sensor Name' and 'Actuator Name'. The table has two rows: 'Grove - Tilt Switch Sensor' and 'Grove - OLED 96x96'. Below the table, there is a 'POST URL' field with the value 'https://api.xively.com/v2/feeds/1109560986/datastreams/Grove-TiltSwitchSensor/datapoints'.

Note:

- 1) Node id is unique in every MS12.
 - 2) Each node maximum support one sensor id and one actuator id.
 - 3) Sensor id or actuator id can be set to 0 if there is no sensor/actuator in the node
- Developer can add customized sensor/actuator info in sensor_table or actuator_table in /usr/lib/sensor. Format is specified in the files.

2.2.2 POST DATA

Format: “ss2 DEVICE_ID,VALUEgg”

Function: Post a sensor data to the IoT server

Example: “ss2 45,30gg”

This command will ask the MS12 to POST a value (36) for Device 45 to the IoT server.

Dragino_Test_45

Public Feed

Feed ID
1109560986


Feed URL
<https://xively.com/feeds/1109560986>

API Endpoint
<https://api.xively.com/v2/feeds/1109560986>

Channels


Last updated 3 hours ago


Request Log

 Pause

Grove-TiltSwitchSensor

30.00
boolean

 Add Channel

 Waiting for requests

Your requests will appear here as soon as we get them, you can debug by clicking each individual request.

Note: Value Type only support the general type (only one parameter) at the moment.

2.2.3 DELETE a NOTE

Format: "ss3 DEVICE_IDgg"

Function: Delete a node from the IoT server and MS12.

Example: "ss3 45gg"

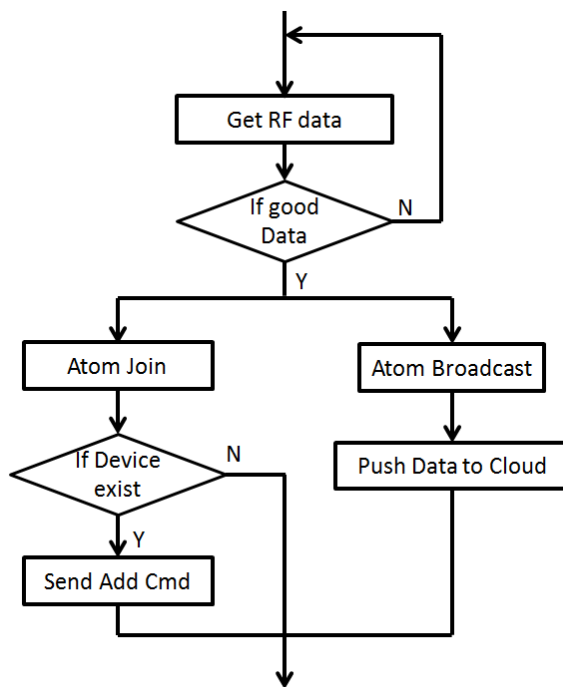
Delete Node 45

3. Firmware of Daughter Board

Firmware in daughter completes the following features mainly:

- Receive Add Frame from Atom, and add a device to Yeelink and dragino device list
- Receive sensor data from Atom, and post it to Yeelink

3.1 Software Flowchart



3.2 Main Function

3.2.1 Add a Node Device

```

/*****
** Function name:      postDta to yeelink
** Descriptions:      return 1: ok 0: nok
*****/
void NodeManage::yeelinkAdd(unsigned char idNode, unsigned char idSensor, unsigned char idActuator);
  
```

3.2.2 Post data to Yeelink

```

/*****
** Function name:      postDta to yeelink
** Descriptions:      return 1: ok 0: nok
*****/
void NodeManage::yeelinkPost(unsigned char idNode, unsigned int psDta);
  
```


4. Ways to access Dragino MS12

4.1 Access MS12 via WEB

Dragino has a default IP address 172.31.255.254 in its LAN port. To access Dragino, you can simply set your computer to

ip address: 172.31.255.253

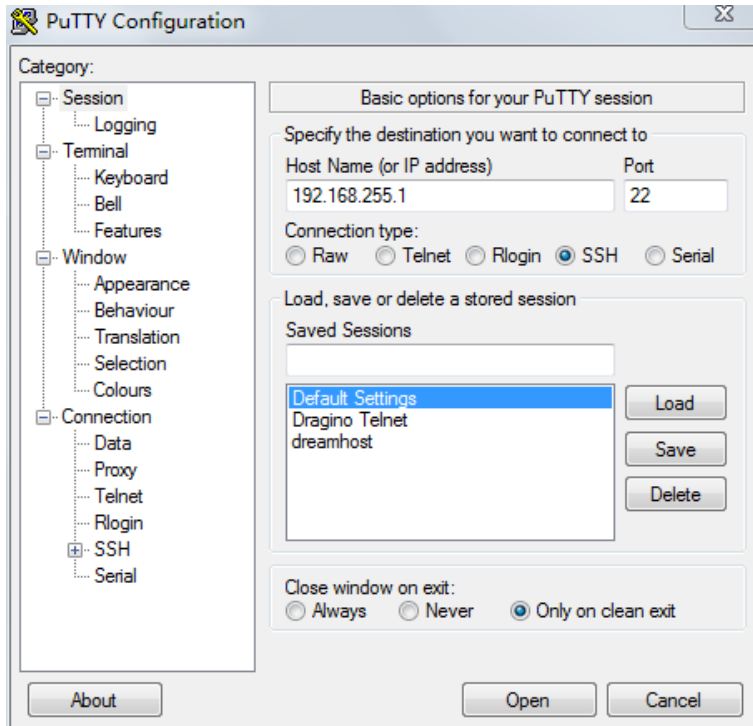
netmask: 255.255.255.252

Then connect an Ethernet cable between your computer and Dragino and type 192.168.255.1 in your browser and you will see Dragino set up page.



4.2 Access MS12 via SSH

Dragino MS12 is running embedded Linux system: OpenWrt. You can access it via SSH and customized the system for different application.



The SSH access for Dragino is:

IP address: 172.31.255.254 on LAN port.

Username: root

Password: root

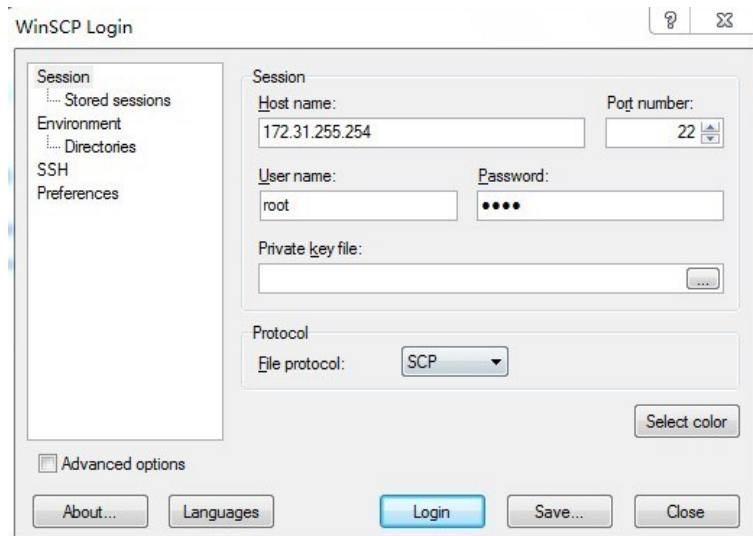
SSH access password can be changed by type below commands:

```
root@dragino-751aff:~# passwd
Changing password for root
New password:
Retype password:
Password for root changed by root
```

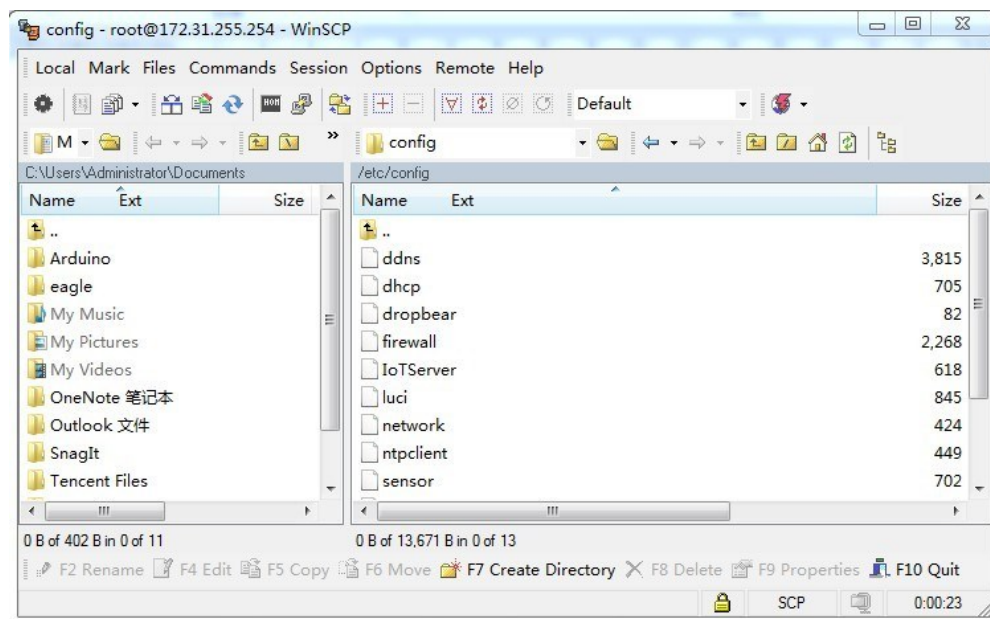
Notice: for security reason, it is recommend that you change the SSH access password after first log in.

4.3 Get or Put files from/to Dragino

MS12 support SCP protocol, developer can use WinSCP to do file transferring with MS12. Below is the setting page on WinSCP:



After successful log in, you can see the file transferring window. Just drag file as a FTP service.



5. WEB GUI manual

5.1 Sensor Settings



DRAGINO
Technology Changes Life

sensor Wifi Interfaces System Syslog DDNS Upgrade

Sensor Settings

Configure Sensors for Dragino

General Settings

Save Sensor Data to Local File?	<input checked="" type="checkbox"/>
Sensor Data Location	/var/log/sensor.log
Debug	<input checked="" type="checkbox"/>
Enable Custom Code	<input checked="" type="checkbox"/>
Custom Code	<input type="text"/>

IoT Server Configure

Enable IoT Service	<input checked="" type="checkbox"/>
IoT Service	xively
Internet Connection to xively.com	Up

General Settings:

- **Save Sensor Data to Local File:** Enable/Disable local logger for sensor data from UART port.
- **Sensor Data Location:** File to store the sensor data. Can be changed in /etc/config/sensor
- **Debug:** Enable/Disable Debug. Debug info will show on Syslog page
- **Enable Custom Code:** Enable/Disable Custom Code.
- **Custom Code:** use Linux commands to process sensor data. The Custom commands will be executed once there is new incoming string from UART port. Developer can use macro **[RAW_DATA]** to get the incoming string

Example:

```
echo [RAW_DATA] >> tem.log; atftp -p -r tem.log 192.168.1.2; rm tem.log
```

This command will forward the RAW_DATA to tftp server with file name tem.log

Note: Due to security reason, Enable Custom Code and Custom Code options are set to invisible by default. They can be set to visible by set the option "**show_custom_code**" to "1" in file /etc/config/sensor

IoT Server Configure

Enable IoT Service	<input checked="" type="checkbox"/>
IoT Service	xively
Internet Connection to xively.com	Up
API Key	NkYHeIFJaC4DPkGDX20ZiW2Jg7AJc36EY0oHTR
Title	Dragino_Test

Devices List

Below devices are connected to Dragino

Device ID: 45

Sensor Name	Grove - Tilt Switch Sensor
Actuator Name	Grove - OLED 96x96
POST URL	https://api.xively.com/v2/feeds/1109560986/datastreams/Grove-TiltSwitchSensor/datapoints

Reset Save

IoT Server Configure:

- **Enable IoT Service:** Disable/Enable IoT Service
- **IoT Service:** Choose Service Provider
- **Internet Connection:** Show the connection to the service provider
- **API Key:** API key from service provider
- **Title:** Default title prefix when create nodes on the IoT server. If leave this blank, the default tile will be set to host name of the MS12.

Devices List:

Shows the Node Connected to Dragino MS12.

5.2 Wi-Fi Settings

DRAGINO
Technology Changes Life

sensor Wifi Interfaces System Syslog DDNS Upgrade

Wifi
Here you can configure installed wifi devices.

Network Name (ESSID)	ChinaNet-edwin
Encryption	WPA-PSK/WPA2-PSK Mixed Mode
Key	ABCDABCDABCDABCD

40bit/104bit WEP is autodetected based on key length. Use either 5/13 ASCII or 10/26 HEX characters as WEP key. A valid ASCII-based key will be translated into a HEX-based one. WPA(2)-PSK keys should be 64 HEX characters.

Reset Save

Wi-Fi Settings:

- **Network Name(ESSID):** Input your wifi router SSID
- **Encryption:** Encryption method used by your router
- **Key:** Encryption Key of your wifi network

5.3 Network Interfaces

Network
Manage Dragino Network Connections

Internet Connection

WAN Interface: WiFi Interface

Protocol: DHCP

Clamp Segment Size: ☐ Fixes problems with unreachable websites, submitting forms or other unexpected behaviour for some ISPs.

Local Network

LAN Interface: RJ45 Interface

IPv4-Address: 192.168.255.1

IPv4-Netmask: 255.255.255.0

IPv4-Gateway (optional):

DNS-Server (optional):

Reset Save

Internet Connection

WAN Interface: Interface for internet access, either WIFI or RJ45 port

Protocol: Way to get IP: Manual, DHCP, PPPoE

Clamp Segment Size: Normally don't use this

Local Network – Network distributed by MS12

LAN interface: Interface for local network, either WIFI or RJ45 port








IPv4-Address: IP address for the interface used for local network.

IPv4-Netmask: Net mask for this local network

IPv4-Gateway: Gateway for this local network, default is the IP address set up above.

DNS-Server: DNS for this local network.

5.4 System Status

 sensor
  Wifi
  Interfaces
  System
  Syslog
  DDNS
  Upgrade

System

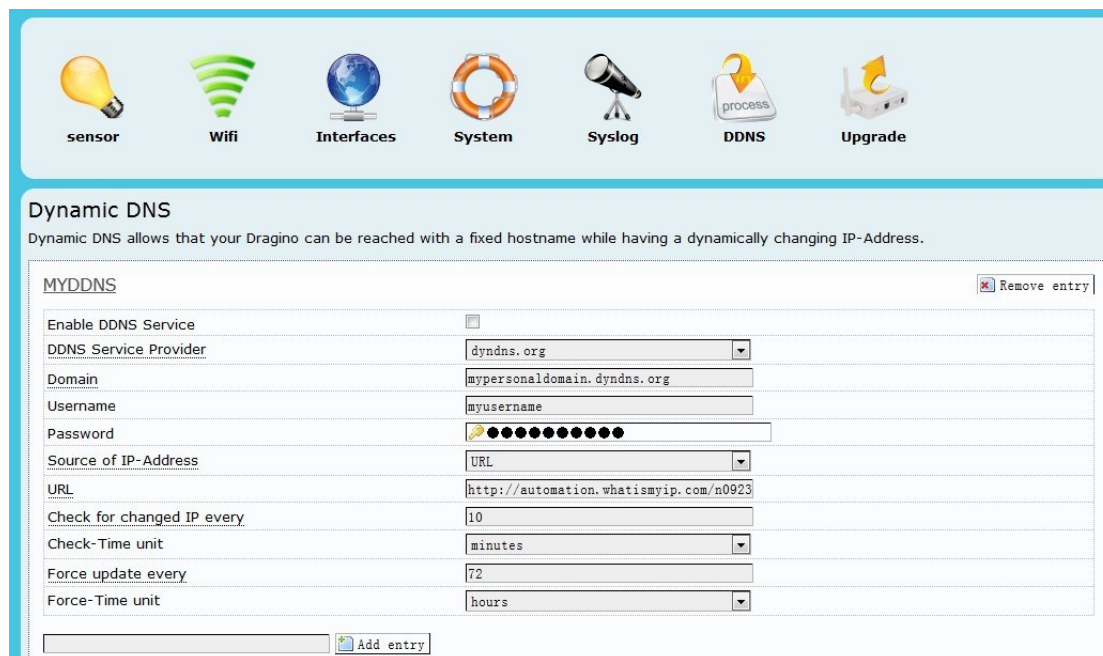
General

Dragino Firmware Version	Dragino MS12 2.2-IoT-1, Build Mon May 20 09:40:43 CST 2013
Shield Model	MT-DS
Shield Application Info	Demo Sketch to show how to use SPI
Shield Sketch Version	0.1
Shield Hardware Version	0.1
Load	0.00, 0.00, 0.00
Memory	29.10 MB (24% cached, 7% buffered, 36% free)
Timezone	UTC
Language	English
System Time	Mon May 20 08:54:21 2013
Uptime	07h 05min 43s
Hostname	dragino

System status page.

5.5 DDNS settings

Dynamic DNS allows you access/control the Dragino & Dragrove from other location even you don't have a fix IP Address.



Enable DDNS Service: Enable/Disable DDNS service

DDNS Service Provider: choose your service provide here

Domain: the hostname provide by your DDNS service provider.

Username: Username of your DDNS service

Password: Password of your DDNS service

Source of IP-Address: Where to look for your external IP address. You can choose:

Network: Set external IP according to your network interface info, e.g. wan, lan

Interface: Set external IP according to your hardware network interface info: eth0 , ath0.

URL: Set external IP according to URL info, for example, you can set it to <http://www.whatismyip.com/automation/n09230945.asp> so the Dragino will connect to this url and get its external IP. It is used when Dragino have a private IP address in its wan port.

Check for changed IP: how often to check if it needs to update its IP to ddns service provider.

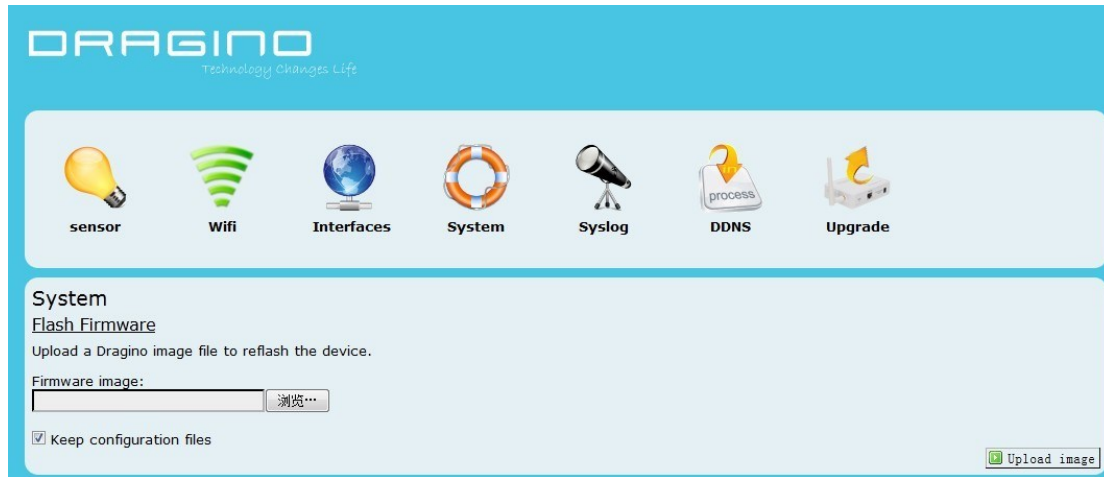
Force update: how often to force an update to DDNS service provider

5.6 Upgrade Firmware

The latest firmware of Dragino can be found in below link:

<http://www.dragino.com/downloads/index.php?dir=MS12/firmware/>

Valid upgradable firmware via GUI has a suffix combined.img. For IoT application, choose the firmware in under IoT directory.



1 REFERENCE

- www.seeedstudio.com : Dragrove vendor, more info about Dragrove and its development kit can be found here.
- www.openwrt.org: Embedded linux used in Dragino.
- wiki.dragino.com: General software/hardware design info for Dragino MS12
- www.xively.com: A public IoT RESTful server.
- www.yeelink.com: A public IoT RESTful server used in China.
- https://github.com/seeedstudio/Cloud_Dragrove: Daughter board firmware code
- https://github.com/seeedstudio/Cloud_Dragino_Firmware: Dragino Firmware