

---

# **Freedomotic User Manual**

***Release 5.6.0***

**Freedomotic Team**

**Apr 06, 2022**

<b>1</b>	<b>What is Freedomotic?</b>	<b>2</b>
1.1	Vision . . . . .	2
1.2	Mission . . . . .	2
1.3	Current development stage . . . . .	3
<b>2</b>	<b>Project History</b>	<b>4</b>
<b>3</b>	<b>Team</b>	<b>5</b>
<b>4</b>	<b>Features</b>	<b>6</b>
<b>5</b>	<b>Press</b>	<b>8</b>
<b>6</b>	<b>Presentations</b>	<b>9</b>
<b>7</b>	<b>Academic papers &amp; thesis</b>	<b>10</b>
<b>8</b>	<b>What you can do with Freedomotic</b>	<b>12</b>
8.1	Security & Video surveillance . . . . .	12
<b>9</b>	<b>PAss Private Assisted House</b>	<b>13</b>
9.1	Resources . . . . .	13
9.2	Video . . . . .	14
<b>10</b>	<b>SED Special Electronic Design</b>	<b>15</b>
10.1	Software/Plugins . . . . .	15
10.2	Plugins used for the current system . . . . .	17
<b>11</b>	<b>Spoken House</b>	<b>18</b>
11.1	Features . . . . .	18
11.2	Requirements . . . . .	18
11.3	Voice Control . . . . .	20
11.4	Resources . . . . .	20
<b>12</b>	<b>Ubuntu Snappy</b>	<b>21</b>
12.1	Setting a Snappy development system . . . . .	21
12.2	Prepare your snap package . . . . .	21
12.3	Upload & install the snap to your snappy device . . . . .	21

12.4	Run Freedomotic . . . . .	22
12.5	Tested Boards . . . . .	22
<b>13</b>	<b>Architecture</b>	<b>25</b>
13.1	The Framework . . . . .	25
13.2	The Plugins . . . . .	26
13.3	Plugins, Objects and Automations interaction . . . . .	27
13.4	Explanation . . . . .	28
<b>14</b>	<b>Messaging system</b>	<b>29</b>
14.1	Events . . . . .	29
14.2	Triggers . . . . .	29
14.3	Commands . . . . .	29
14.4	Reactions (aka Automations) . . . . .	29
<b>15</b>	<b>Requirements</b>	<b>30</b>
<b>16</b>	<b>Download</b>	<b>31</b>
16.1	Latest stable release (recommended) . . . . .	31
16.2	Dailybuilds . . . . .	31
16.3	Old releases . . . . .	32
<b>17</b>	<b>Installation</b>	<b>33</b>
<b>18</b>	<b>Windows installation</b>	<b>34</b>
<b>19</b>	<b>Linux installation</b>	<b>35</b>
<b>20</b>	<b>Mac OSX</b>	<b>36</b>
<b>21</b>	<b>Raspberry Pi</b>	<b>37</b>
21.1	Get Raspbian OS and install it on an SD card . . . . .	37
21.2	Install Freedomotic automatically . . . . .	37
<b>22</b>	<b>Docker container</b>	<b>38</b>
22.1	Container health check . . . . .	38
22.2	Data persistence . . . . .	39
22.3	Docker Hub . . . . .	40
<b>23</b>	<b>Basic configuration</b>	<b>41</b>
23.1	Basic . . . . .	41
23.2	Internationalization . . . . .	41
23.3	Logging . . . . .	41
23.4	Plugins Marketplace . . . . .	42
23.5	Periodic data saving . . . . .	42
23.6	Persistence . . . . .	42
23.7	P2P . . . . .	42
23.8	Resources . . . . .	42
23.9	Security . . . . .	42
<b>24</b>	<b>Internationalization</b>	<b>43</b>
<b>25</b>	<b>What is a plugin?</b>	<b>44</b>
25.1	Plugin features . . . . .	44
25.2	Plugin manifest and configuration . . . . .	44

<b>26 Plugin installation</b>	<b>46</b>
26.1 Install manually . . . . .	46
<b>27 Plugin configuration</b>	<b>47</b>
<b>28 Devices plugins list</b>	<b>48</b>
<b>29 Arduino Remote Controller</b>	<b>51</b>
29.1 Overview . . . . .	51
29.2 Configuration . . . . .	51
29.3 Video . . . . .	53
29.4 Download . . . . .	53
29.5 Source code . . . . .	53
<b>30 Arduino Serial Communication</b>	<b>54</b>
30.1 Overview . . . . .	54
30.2 Configuration . . . . .	54
30.3 Arduino sketch . . . . .	55
30.4 Download . . . . .	55
30.5 Source code . . . . .	55
<b>31 Arduino WeatherShield</b>	<b>56</b>
31.1 Overview . . . . .	56
31.2 Board protocol . . . . .	56
31.3 How to read values from an object . . . . .	58
31.4 Download . . . . .	58
31.5 Source code . . . . .	58
<b>32 BT Speech Recognition</b>	<b>59</b>
32.1 Overview . . . . .	59
32.2 Configuration . . . . .	59
32.3 Video . . . . .	59
<b>33 Chat Plugin</b>	<b>60</b>
33.1 Overview . . . . .	60
33.2 Configuration . . . . .	60
<b>34 Google Calendar Events</b>	<b>61</b>
34.1 Overview . . . . .	61
34.2 Configuration . . . . .	61
<b>35 Harvester</b>	<b>62</b>
35.1 Overview . . . . .	62
35.2 Configuration . . . . .	62
35.3 Usage . . . . .	63
<b>36 HWg-STE Ethernet thermometer</b>	<b>64</b>
36.1 Overview . . . . .	64
36.2 Configuration . . . . .	64
36.3 How to read data from a sensor . . . . .	66
36.4 Download . . . . .	69
36.5 Source code . . . . .	69
<b>37 InfluxDB Persistence</b>	<b>70</b>
37.1 Overview . . . . .	70
37.2 Configuration . . . . .	70



37.3	Database structure . . . . .	71
37.4	How to save data . . . . .	71
37.5	Download . . . . .	72
37.6	Source code . . . . .	72
<b>38</b>	<b>IpCamera Motion Plugin</b>	<b>73</b>
38.1	Overview . . . . .	73
38.2	Configuration . . . . .	73
38.3	How to save a captured image . . . . .	73
<b>39</b>	<b>GCE Electronics IPX800</b>	<b>77</b>
39.1	Overview . . . . .	77
39.2	Configuration . . . . .	77
39.3	How to control an object with this board . . . . .	78
39.4	Download . . . . .	79
39.5	Source code . . . . .	79
<b>40</b>	<b>KMTronic Usb Relay</b>	<b>80</b>
40.1	Overview . . . . .	80
40.2	Configuration . . . . .	80
<b>41</b>	<b>Mailer</b>	<b>81</b>
41.1	Overview . . . . .	81
41.2	Configuration . . . . .	81
41.3	Create an automation example . . . . .	81
41.4	Command parameters . . . . .	83
41.5	Notes . . . . .	83
<b>42</b>	<b>MaryTTS Text to Speech</b>	<b>84</b>
42.1	Overview . . . . .	84
42.2	Configuration . . . . .	84
42.3	Automations examples . . . . .	84
42.4	How to create new custom commands . . . . .	85
42.5	How to install better and natural sounding voices . . . . .	85
42.6	Download . . . . .	85
42.7	Source code . . . . .	85
<b>43</b>	<b>Modbus</b>	<b>86</b>
43.1	Overview . . . . .	86
43.2	Configuration . . . . .	86
43.3	An XML example . . . . .	87
<b>44</b>	<b>MQTT Broker</b>	<b>89</b>
44.1	Overview . . . . .	89
44.2	Configuration . . . . .	89
44.3	How to manage a thing . . . . .	89
44.4	Multiple behaviors . . . . .	90
44.5	Video . . . . .	90
44.6	Download . . . . .	90
44.7	Source code . . . . .	90
<b>45</b>	<b>MQTT Client</b>	<b>91</b>
45.1	Overview . . . . .	91
45.2	Configuration . . . . .	91
45.3	Download . . . . .	91

45.4	Source code . . . . .	92
<b>46</b>	<b>MySensors</b>	<b>93</b>
46.1	Overview . . . . .	93
46.2	Configuration . . . . .	93
46.3	Supported objects . . . . .	93
<b>47</b>	<b>openPicus Flyport</b>	<b>94</b>
47.1	Overview . . . . .	94
47.2	Configuration . . . . .	94
<b>48</b>	<b>openPicus Grove System</b>	<b>95</b>
48.1	Overview . . . . .	95
48.2	Configuration . . . . .	95
48.3	The list of materials . . . . .	96
48.4	The build process . . . . .	97
48.5	Video . . . . .	97
<b>49</b>	<b>BTicino OpenWebNet</b>	<b>98</b>
49.1	Overview . . . . .	98
49.2	Configuration . . . . .	98
<b>50</b>	<b>Persistence</b>	<b>99</b>
50.1	Overview . . . . .	99
50.2	Configuration . . . . .	100
50.3	Requirements . . . . .	100
<b>51</b>	<b>ProgettiHw-Sw Ethernet Board v2</b>	<b>101</b>
51.1	Overview . . . . .	101
51.2	Configuration . . . . .	101
51.3	How to control a relay . . . . .	101
51.4	Use cases . . . . .	102
51.5	Download . . . . .	102
51.6	Source code . . . . .	102
<b>52</b>	<b>pURL</b>	<b>103</b>
52.1	Overview . . . . .	103
52.2	Configuration . . . . .	104
52.3	How to create custom triggers . . . . .	104
52.4	Upcoming features . . . . .	104
<b>53</b>	<b>Push Notifications</b>	<b>105</b>
53.1	Overview . . . . .	105
53.2	Features . . . . .	105
53.3	Supported providers . . . . .	105
53.4	Configuration . . . . .	106
53.5	Download . . . . .	107
53.6	Source code . . . . .	107
<b>54</b>	<b>RestAPI v3</b>	<b>108</b>
54.1	Overview . . . . .	108
54.2	Basic configuration . . . . .	108
54.3	Advanced configuration . . . . .	108
54.4	Usage (Documentation and tools) . . . . .	109
54.5	Try API with cURL . . . . .	109

<b>55</b>	<b>Sensors and tracking simulation</b>	<b>111</b>
55.1	Sensors simulator . . . . .	111
55.2	Tracking simulation . . . . .	111
<b>56</b>	<b>Teracom TCW122B-CM</b>	<b>117</b>
56.1	Overview . . . . .	117
56.2	Configuration . . . . .	117
<b>57</b>	<b>Telegram Bot</b>	<b>118</b>
57.1	Overview . . . . .	118
57.2	Configuration . . . . .	118
57.3	Create an automation example . . . . .	119
57.4	Download . . . . .	120
57.5	Source code . . . . .	120
<b>58</b>	<b>ThingSpeak</b>	<b>121</b>
58.1	Overview . . . . .	121
58.2	Configuration . . . . .	121
<b>59</b>	<b>Text To Speech</b>	<b>123</b>
59.1	Overview . . . . .	123
59.2	Configuration . . . . .	123
59.3	Automations examples . . . . .	123
59.4	How to create new custom commands . . . . .	124
59.5	How to install better, natural sounding voices . . . . .	124
59.6	Video . . . . .	124
<b>60</b>	<b>Twilight - Sunset and sunrise alerts</b>	<b>125</b>
60.1	Overview . . . . .	125
60.2	Configuration . . . . .	125
60.3	Source code . . . . .	126
<b>61</b>	<b>Twitter4Freedomotic</b>	<b>127</b>
61.1	Overview . . . . .	127
61.2	Configuration . . . . .	127
61.3	How to obtain the OAuth parameters . . . . .	127
61.4	Download . . . . .	128
61.5	Source code . . . . .	128
<b>62</b>	<b>Webserver</b>	<b>129</b>
62.1	Overview . . . . .	129
62.2	Configuration . . . . .	129
62.3	Download . . . . .	130
62.4	Source code . . . . .	130
<b>63</b>	<b>Zibase</b>	<b>131</b>
63.1	Overview . . . . .	131
63.2	Configuration . . . . .	131
<b>64</b>	<b>Z-Wave</b>	<b>132</b>
64.1	Overview . . . . .	132
64.2	Configuration . . . . .	132
64.3	Supported adapters . . . . .	132
64.4	Supported devices . . . . .	133
<b>65</b>	<b>Things (objects) plugins list</b>	<b>134</b>

<b>66</b>	<b>Base home automation</b>	<b>135</b>
66.1	Overview . . . . .	135
66.2	Things list . . . . .	135
<b>67</b>	<b>Jfrontend</b>	<b>138</b>
67.1	Environments . . . . .	138
67.2	Things . . . . .	141
67.3	Localization . . . . .	143
67.4	Automations . . . . .	143
67.5	Plugins . . . . .	144
67.6	Settings . . . . .	145
67.7	Help . . . . .	145
<b>68</b>	<b>Vue web client</b>	<b>147</b>
<b>69</b>	<b>Logging</b>	<b>148</b>
69.1	How to enable logging . . . . .	148
<b>70</b>	<b>Troubleshooting</b>	<b>150</b>
<b>71</b>	<b>FAQ</b>	<b>151</b>
71.1	What platforms are supported? . . . . .	151
71.2	Can I run Freedomotic on Raspberry Pi? . . . . .	151
71.3	What version of Java do I need to run Freedomotic? . . . . .	151
71.4	What technologies / tools are used? . . . . .	151
71.5	I need help. How? . . . . .	151
71.6	Can I use Freedomotic in commercial projects? . . . . .	152
71.7	How to run Freedomotic headless (server mode with no GUI) . . . . .	152
<b>72</b>	<b>Release Notes</b>	<b>153</b>
72.1	Version 5.5 (Bender) . . . . .	153
72.2	Version 5.6 RC3 (Commander) . . . . .	154
<b>73</b>	<b>How to contribute</b>	<b>156</b>

**Everything you need to know about Freedomotic.**

Contents:

---

## What is Freedomotic?

---

Freedomotic is an open source, flexible and secure Internet of Things (IoT) development framework. It can be used to build and manage modern smart spaces. It is targeted at individuals (home automation) as well as businesses (smart retail environments, ambient aware marketing, monitoring and analytics, etc). Freedomotic can interact with well-known automation protocols as well as with “do it yourself” solutions. It treats the web, social networks and branded frontends as first class components of the system.

It allows you to build smart spaces. Freedomotic can manage many spaces, ranging from small apartments to huge buildings, like museums, schools, corporate offices, malls and university campuses. For OEMs and software developers, Freedomotic is the solution to create building automation systems, smart retail environments, home automation managed services and innovative IoT ambient aware applications, drastically reducing development effort and time to market.

Freedomotic can be integrated with popular building automation technologies like BTicino OpenWebNet, Modbus RTU, Z-wave as well as custom automation projects using Arduino devices, do it yourself boards, third party graphical frontends, text to speech engines, motion detection using IP cameras stream, social networks, and much more... All this features can be delivered from a marketplace as downloadable plugins.

### 1.1 Vision

Bridging the gap between the physical and digital world; connecting people to things and value-added business services.

### 1.2 Mission

Developing an application framework, which reduces the effort and time to market required to produce solutions based on the Internet of Things concept. This means we are making the environment aware of the people and the things in it. Things can reach a new level of usefulness thanks to their new connected nature, allowing them to leverage the web and all of the information based services it provides.

## 1.3 Current development stage

The project is currently in an advanced beta stage. We are using the home automation segment to test and attract users but its range of application is much wider.

The final purpose of the project is to build a sort of **Content Management System (CMS)** for building automation. It will abstract and make easily available the common features required by building automation system in a way privates and companies can extend it to create custom context aware/environment aware services.

## CHAPTER 2

---

### Project History

---

The Department of Information Engineering and Computer Science (DISI) at the University of Trento had several research teams working on sensor networks for home monitoring and automation.

They needed a **framework** capable of easily integrating different projects developed in heterogeneous languages. This framework needed to make those projects work together, help simplify testing, and produce visual demos to show to research partners. Using a common framework, teams at DISI could focus on the core of their research instead of developing custom solutions for each individual project. The **Freedomotic** project was created to fulfill these needs.

The main goals and requirement of **Freedomotic** were: maintain a framework which was flexible and modular which could be easily integrated and adapted to different (and potentially unknown) needs, allowed for simple testing, and could produce visual demos.

These same goals and requirements continue to hold true today.



## CHAPTER 3

---

### Team

---

The Freedomotic team consists of a group of enthusiasts who work on the framework on a day to day basis.

Table 3.1: Main contributors

Name	From	Duties
Enrico Nicoletti	Italy	Project Founder, Project vision, Core Developer
Mauro Cicolella	Italy	Community Manager, Automation Protocols Integration, Marketing and end-users, Communication, Quality Assurance
Gabriel Pulido de Torres	Spain	APIs Engineer, Mobile Platform Developer, Product Strategist, Technical Research, Development Workflow Manager
Alberto Mengoli	Italy	.Net Frontend Development, Testing, Usability feedback
Matteo Mazzoni	Italy	Core Developer, APIs Engineer, Plugins Developer, Technical Research

Table 3.2: Past members

Name	From	Duties
Niko Zarzani	Italy	UI Development
Roberto Socrates	Spain	Core Developer, Software architect

Many other developers have contributed to the project. [Here](#) is the complete list.

## CHAPTER 4

---

### Features

---

**Identity:** All things have a persistent unique identifier. This identifier allows you to address it from all over the world, and it works no matter whichever automation protocol it uses. You are safe from the protocols out there.

**Services:** Freedomotic is different, automation is not the end of the story. The framework is centred around the concept of services for users, which may use automation to achieve a goal. It is the Internet of Things at a new level. Think Wider!

**Simulation:** Freedomotic allows you to fully run it without any sensor or actuator connected. You can configure and test your automation before buying the hardware. This is great when planning a system with your customers, give them a taste of the finished product.

**Realtime Marketing:** Freedomotic knows the environment's topology (ie: rooms, shapes and locations), the people and the things in it. This allows to track the users in the environment, profile them and create 1 to 1 realtime marketing campaigns. This feature is also great for disabled assistance and security focused systems.

**Cross language Restful API:** you can control any aspect of the system with our JSON based RESTful APIs, from listing and controlling the objects in an environment, to retrieve, install and manage plugins, all using familiar and developer friendly technologies. The entire system is event based. Components dialog together using text, events and commands, so it's easy to integrate your ERP, CRM or any legacy software you already run on your own premises. This is also great to build custom branded frontends for the web, mobile and desktop. In Freedomotic you can run concurrently as many frontends as you want, each one can be targeted to a specific audience.

**Distributed:** Freedomotic can be run as a decentralized peer to peer network with no single point of failure. It can be deployed on a network of embedded systems like Raspberry Pi or on standard PCs and servers. For businesses this means you can have an instance running in the cloud connected to different satellites. You can manage the configuration and provide unique compute intensive features in the cloud (eg: face recognition) for a monthly fee.

**Plugins:** The system features are not hardcoded. You can install new plugins at runtime enriching the features. If you are interested in plugins development take a look at our Developers - Getting Started tutorial. Any plugin can be uploaded to an online marketplace (our or your own) to allow 1-click installation.

**Auto discovery:** Wouldn't it be great if you can turn on a light and have it automatically configured on the virtual environment map? Freedomotic can autodiscover the objects (eg: home automation devices) deployed in your real environment. No more diving into complex configuration files.

**History aware:** it can track any status change in the environment and update them in a database for analysis. You can analyze consumption behaviors to implement real energy saving or learn more about how your customers interact with your business, for example their visit patterns.

**Secure, Multilanguage and Multiuser:** Freedomotic is built from the ground up with multilanguage, multiuser and security features in mind. All these features come free of cost for each new plugin you develop, sparing a lot of time and effort. You can focus on your core business and let Freedomotic do the heavy lifting.

## CHAPTER 5

---

### Press

---

- PC Professionale: Freedomotic, un maggiordomo digitale Open Source
- Domotica.it: Intervista a Mauro Cicoella - sviluppatore Freedomotic
- PC Professionale: FreeDomotic tiene sempre più sotto controllo tutta la casa
- Domotica.it: Freedomotic presentato al W3C Workshop di Berlino sull'Internet of Things
- ODROID Magazine: Sptember 2014
- AutomatedBuildings.com: Email Interview – Mauro Cicoella and Ken Sinclair
- Daivai.com: La Smart Home in versione Maker!

## CHAPTER 6

---

### Presentations

---

- SIAM domotica open source Fiera Elettronica 2014
- Smart Home Now Milano 12/05/2016

## Academic papers & thesis

- Nicoletti, E. (2009). “*Progetto e realizzazione di una piattaforma di programmazione per applicazioni domotiche*”, Master Thesis, Università di Trento (Italy).
- Bravi, M. (2014). “*Definizione e sviluppo di un sistema di configurazione per Private Assisted House*”, Master Thesis, Università di Camerino (Italy). [Thesis](#) - [Video](#)
- Campoli, F. (2015). “*Heima Off Grid Casa Auto-Suficiente Controlada*”, Master Thesis, Instituto Politecnico de Viana do Castelo (Portugal). [Thesis](#)
- da Silva Fidalgo, J.M. (2015). “*Study, design and development of an integration component with sensory features of objects through IoT middleware*”, Master Thesis, University of Coimbra (Portugal). [Thesis](#)
- Würth, M. (2015). “*Evaluación de Middlewares Sensibles al Contexto*”, Master Thesis, Universidad ORT (Uruguay). [Thesis](#)
- Shahzada, A. (2015). “*A comprehensive framework for the development of dynamic smart spaces*”, Doctoral Dissertation, Politecnico di Milano (Italy). [Thesis](#)
- Di Brino, M. (2015). “*SpokenHouse: applicazione mobile cross-platform di supporto ai non vedenti per il controllo domotico. Interfacciamento con un framework domotico*”, Master Thesis, Università del Sannio (Italy). [Thesis](#) - [Slides](#) - [Video](#)
- Guardabascio, D. (2015). “*SpokenHouse: applicazione mobile cross-platform di supporto ai non vedenti per il controllo domotico. Interazione con l’utente e usabilità*”, Master Thesis, Università del Sannio (Italy). [Thesis](#) - [Slides](#)
- Ristaino, G. (2015). “*Evoluzione di un’applicazione mobile cross platform per il supporto domotico ad utenti non udenti*”, Master Thesis, Università del Sannio (Italy). [Thesis](#) - [Slides](#) - [App Android](#)
- Jarraya A., Ramoly N., Bouzeghoub A., Arour K., Borgi A., Finance B. (2016). “*A fuzzy semantic CEP model for situation identification in smart homes*”, Conference Paper, European Conference on Artificial Intelligence - The Hague (Holland). [Paper](#)
- Trivellato, R. (2017). “*Sistemi domotici integrati per la gestione intelligente d’ambiente*”, Master Thesis, Università di Padova (Italy). [Thesis](#) - [Slides](#)
- Lombardi, M. (2017). “*Componentistica hardware e software coordinata da smartphone e destinata alla domotica per utenti con disabilità*”, Master Thesis, Università di Napoli “Federico II” (Italy). [Slides](#)

- Viscomi, S.E. (2017). “*Architettura hardware/software coordinata da smartphone e destinata alla domotica per utenti con disabilità*”, Master Thesis, Università di Napoli “Federico II” (Italy). [Slides](#)
- Ramoly, N. (2018). “*Contextual integration of heterogeneous data in an open and opportunistic smart environment : application to humanoid robots*”, Master Thesis, Université Paris-Saclay” (France). [Thesis](#)
- Di Federico, G. (2020). “*The application of process mining in a simulated smart environment to derive sensors placing* “, Master Thesis, Università di Camerino (Italy). [Thesis](#)

---

### What you can do with Freedomotic

---

#### 8.1 Security & Video surveillance

Here a quick solution to create an IP camera for video surveillance by “recycling” an old smartphone / tablet Android you no longer use.

You only need the app <https://play.google.com/store/apps/details?id=com.pas.webcam&hl=it>

After its installation go to settings and if you decide to leave the default values (no password set) just select the last option **Flow Start**. At this point you will see the video stream directly on the screen and the URL to open in your browser: something like `http://ip-smartphone:8080`.

Install the *IP Camera motion* plugin and open *cameras.xml* file to set the correct ip address.

For example:

```
<ipcam name="Tablet" url="http://192.168.0.101:8080/video" mode="push"/>
```

Now the stream captured by the DIY camera is redirected to the plugin so you can detect any motion and execute one or more commands e.g. sending a notification).



---

### PAss Private Assisted House

---

The PAss team is headed by dr. Francesco De Angelis, who is working together with INRCA and Regione Marche, on a complete platform that integrates and coordinates home automation, telemedicine solutions and smart objects in the same house.

A “smart house” meta-model is being developed as a result of a multidisciplinary collaboration team of the University of Camerino (architecture and computer science), the Rehabilitation Institute Santo Stefano (<http://www.sstefano.it/>) and many manufacturers located in Regione Marche, Italy(<https://www.google.pt/maps/place/Marche,+Italy>). These manufacturers produce hardware, mainly telemedicine devices and also everyday objects are being trasformed into “smart devices” with some form of integration with sensors/actuators.

Freedomotic was chosen as the platform for integration of domestic smart objects (sensors and actuators) in the home environment. These objects can communicate through a home gateway using a JSON payload carried on MQTT.

This architecture was designed for a dual purpose:

1. generating in local, house events to be fed to Freedomotic and create automations according to the rules if-then-else
2. carrying the data to an internet data center that integrates house-related information with telemedicine data. This data is then treated by ad-hoc algorithms - which knowing the disability/disease of those who live in the house- can trigger a remote control mechanism and lift alerts to a caregiver . I can be also used to fire in-house automations from outside.

As an example, one of our partners, is developing an app that uses Bayesian networks to control the “normal” patients’ behavior. This project aims to develop a mobile UI to be allocated to health care professionals and patients themselves.

This should simplify as much as possible the daily operations inside the house.

Warning : It is not intended, at the time, to let users configure their house environment and its devices.

## 9.1 Resources

- [Italian thesis](#)
- [PAss Private Assisted House Project](#)

- [PAss Forum Pa Challenge SlideShare](#)

## 9.2 Video

[SED](<http://www.sednet.com/>) Limited society environment management

1. Remote access
2. Automations aimed at limited consumption
3. Device automation override
4. Granular user privileges
5. Verification of consumption of variable temporal arc
6. Hierarchical control between buildings
7. Hardware/plant

The electrical system globally exploits the use of contactors in bistable relay for the control of devices that are not natively provided for a home automation system. The home automation control adapters have been added at a later stage and positioned in parallel to a pre-existing plant, in order to read the status of the devices and control the contactors. In this perspective, it wanted to show that it is possible to ‘extend’ a classic system with home automation controls, also to limit the planned set of activities and allow for retrofitting low costs. The Easybox cards are therefore used to add a classic on the system advanced features, including:

1. Timing the ignition of the fan coil, in order to ensure the homogeneous air conditioning of the rooms.
2. Timing the turning off of devices, in order to minimise the wastage from forgetfulness.
3. Collect data on the use of the devices, in order to estimate the fuel consumption and the resulting savings, compared to the scenario of a classic system.

### 10.1 Software/Plugins

The system is run on a workstation WIN 7 Intel Core i3 and 4GB RAM; The same machine also houses a SQLSERVER 2008 server.



Fig. 10.1: Special Electronic Design

## 10.2 Plugins used for the current system

1. Easybox boards control
2. Harvester: Storage of usage data on db.
3. Twilight: Events based on sunrise and sunset time.
4. Chat: Remote control, test control and automation.
5. Desktop Frontend

**Spoken House** is a mobile app for users with impaired vision and / or hearing developed in collaboration between Informatici senza Frontiere and the University of Sannio.

This cross platform mobile application allows people with visual impairments easier management of their home automation controlled devices.

Using the 4 corners of the device and electronic voice guidance allows easy use and an easy navigation within the application.

### 11.1 Features

- Interface divided into 4 equal parts, each corresponding to a corner of the device
- Increase or decrease the font size of text displayed
- Change the default theme with other combinations of colors (such as high contrast)
- Electronic voice to aid in navigation
- Encoding of messages to Morse Code through device vibration
- Encoding of messages marked to Italian
- View the video application help in sign language
- Voice control to change the state of objects in the house

### 11.2 Requirements

- Freedomotic 5.6.0-rc3

To connect the smartphone to Freedomotic, during the first launch of the application you are required to provide the following parameters:

## CASE HISTORY: SPOKENHOUSE

Creatori: Informatici Senza Frontiere Campania - Università degli Studi del Sannio  
Sviluppatori: Marco Di Brino, Daniela Guardabascio, Giuseppe Ristaino  
Relatori: Prof.ssa Lerina Aversano, Prof.ssa Maria Tortorella, Dr. Manuel Parrella  
Per ISF: Dr. Ing. Adiutrice Barretta

Applicazione di  
supporto a  
utenti con  
disabilità visive  
e/o uditive per il  
controllo della  
propria casa



Fig. 11.1: Spoken House

- IP Address of the machine running Freedomotic
- Username and Password used to access Freedomotic

## 11.3 Voice Control

The latest addition to Spoken House is the voice control! After activating this mode in settings (which is accessed by pressing any hardware button such as the volume control buttons), you will be sent a voice command application to more comfortably control objects in Freedomotic.

Currently the voice control mode only allows for controlling devices with “on” / “off” or by “opening” / “closing” of various objects.

The typical sentence structure recognized by the application requires a verb as the first word (eg “open” or “turn on”) and then the name of the object to be controlled followed by the name of the room and the floor where is located. A complete example would be: “Open the door of the living room of the first floor.”

## 11.4 Resources

- [Android App](#)
- Presentation on Prezi.com [http://prezi.com/7pkirffztvwb/?utm\\_campaign=share&utm\\_medium=copy](http://prezi.com/7pkirffztvwb/?utm_campaign=share&utm_medium=copy)
- *Slides and complete thesis*

**A great many thanks to the neo engineers Marco Di Brino, Daniela Guardabascio, Giuseppe Ristaino, and faculty Prof. Lerina Aversano and Prof. Maria Tortorella as well as the representatives of Informatici Senza Frontiere Dr. Manuel Parrella and Ing. Adia Barretta.**



Freedomotic porting on Ubuntu Snappy. Repository on GitHub

### 12.1 Setting a Snappy development system

1. Download Ubuntu 15.04
2. Install snappy cli with `sudo apt-get install ubuntu-snappy-cli`
3. Snappy checks your app configuration so you need to install review tools. For 15.04: [https://launchpad.net/~snappy-dev/+archive/ubuntu/tools/+files/click-reviewers-tools\\_0.26\\_all.deb](https://launchpad.net/~snappy-dev/+archive/ubuntu/tools/+files/click-reviewers-tools_0.26_all.deb)

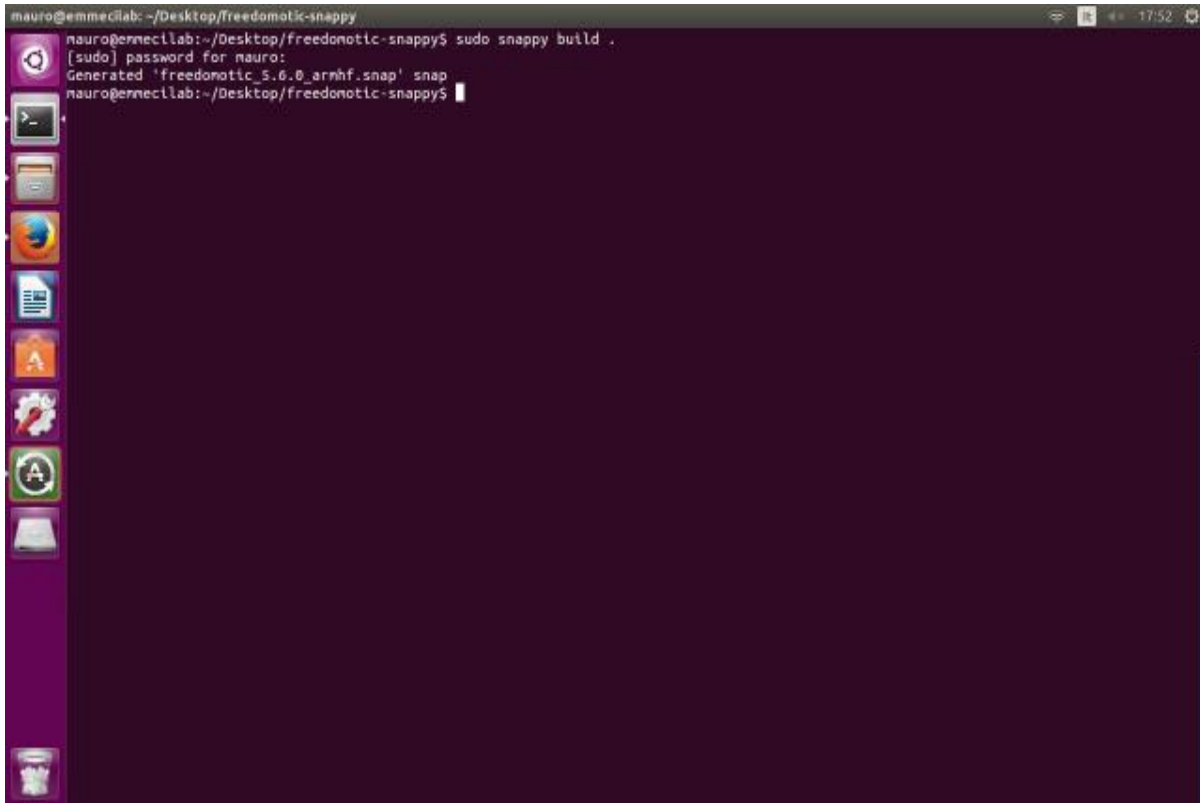
### 12.2 Prepare your snap package

1. `git clone https://github.com/freedomotic/fd-snappy.git`
2. `cd fd-snappy`
3. Download Oracle Java 7 for ARM from [http://archive.raspberrypi.org/debian/pool/main/o/oracle-java7-jdk/oracle-java7-jdk\\_1.7.0+update40\\_armhf.deb](http://archive.raspberrypi.org/debian/pool/main/o/oracle-java7-jdk/oracle-java7-jdk_1.7.0+update40_armhf.deb)
4. Extract the archive and copy the content of jre folder (bin & lib) into *fd-snappy/jre* folder
5. Download the last available Freedomotic build from here and extract it into *fd-snappy/freedomotic*
6. `sudo snappy build .`

### 12.3 Upload & install the snap to your snappy device

You have two options:

**from your development system**



Determine the ip address of your snappy device with ifconfig then digit

```

snappy-remote --url=ssh://<ip-of-your-snappy-core-device> install ./freedomotic_5.6.0_
↪armhf.snap

```

#### from local

Upload the app to your snappy device by ssh (using SCP)

then digit in console

```

sudo snappy install --allow-unauthenticated freedomotic_5.6.0_armhf.snap

```

## 12.4 Run Freedomotic

From command line digit

```

freedomotic.start

```

Point your browser to <http://ip-of-your-snappy-core-device:9111> and play with our API

## 12.5 Tested Boards

- Raspberry Pi2

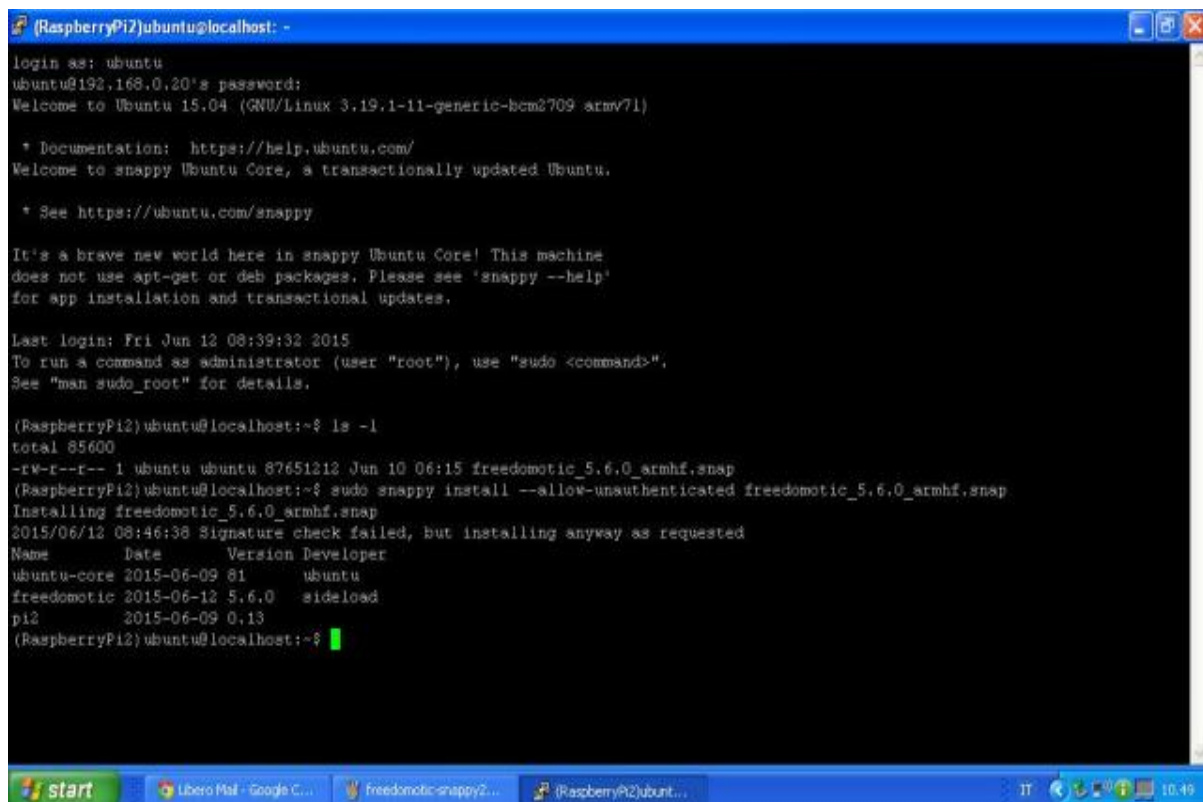
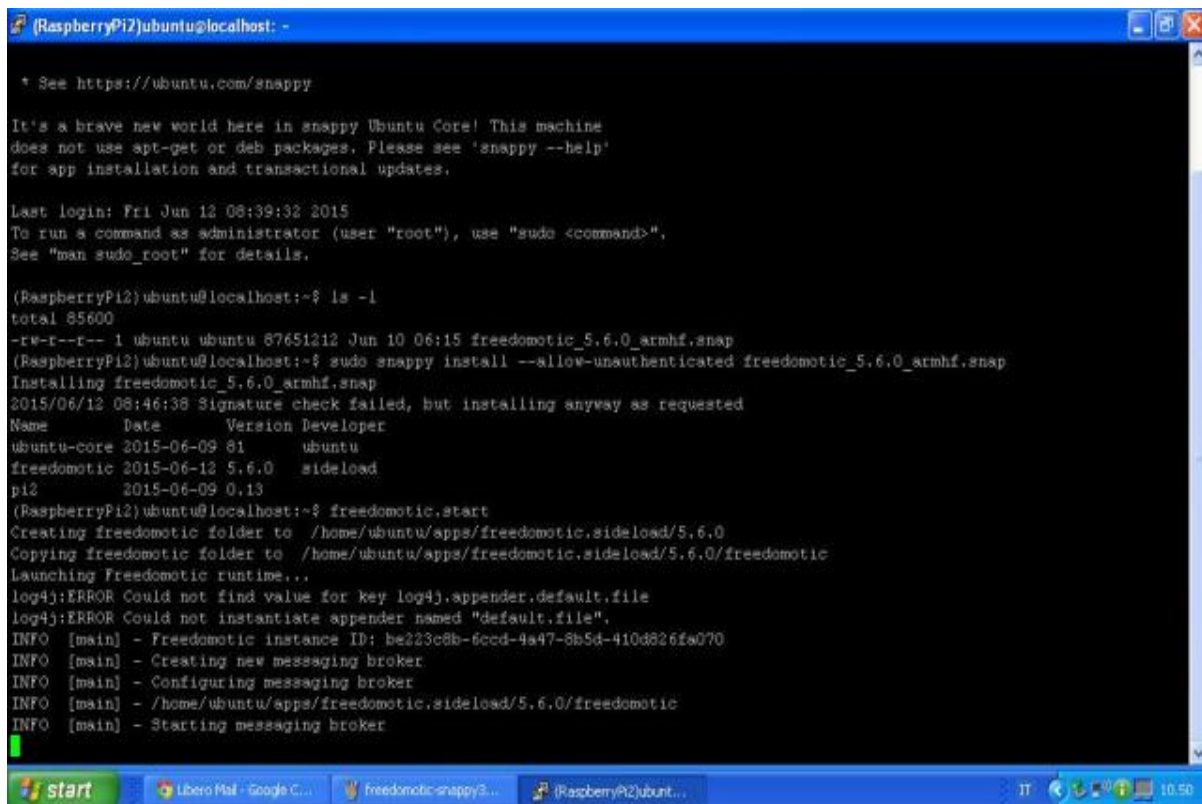


Fig. 12.1: Upload the app to your device by ssh (using SCP)



```

(RaspberryPi2)ubuntu@localhost: ~
* See https://ubuntu.com/snap

It's a brave new world here in snappy Ubuntu Core! This machine
does not use apt-get or deb packages. Please see 'snappy --help'
for app installation and transactional updates.

Last login: Fri Jun 12 08:39:32 2015
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

(RaspberryPi2)ubuntu@localhost:~$ ls -l
total 85600
-rw-r--r-- 1 ubuntu ubuntu 87651212 Jun 10 06:15 freedomotic_5.6.0_armhf.snap
(RaspberryPi2)ubuntu@localhost:~$ sudo snappy install --allow-unauthenticated freedomotic_5.6.0_armhf.snap
Installing freedomotic_5.6.0_armhf.snap
2015/06/12 08:46:38 Signature check failed, but installing anyway as requested
Name      Date      Version Developer
ubuntu-core 2015-06-09 81      ubuntu
freedomotic 2015-06-12 5.6.0  sideload
pi2       2015-06-09 0.13
(RaspberryPi2)ubuntu@localhost:~$ freedomotic.start
Creating freedomotic folder to /home/ubuntu/apps/freedomotic.sideload/5.6.0
Copying freedomotic folder to /home/ubuntu/apps/freedomotic.sideload/5.6.0/freedomotic
Launching Freedomotic runtime...
log4j:ERROR Could not find value for key log4j.appender.default.file
log4j:ERROR Could not instantiate appender named "default.file".
INFO [main] - Freedomotic instance ID: be223c8b-6cc0-4a47-8b5d-410d826fa070
INFO [main] - Creating new messaging broker
INFO [main] - Configuring messaging broker
INFO [main] - /home/ubuntu/apps/freedomotic.sideload/5.6.0/freedomotic
INFO [main] - Starting messaging broker

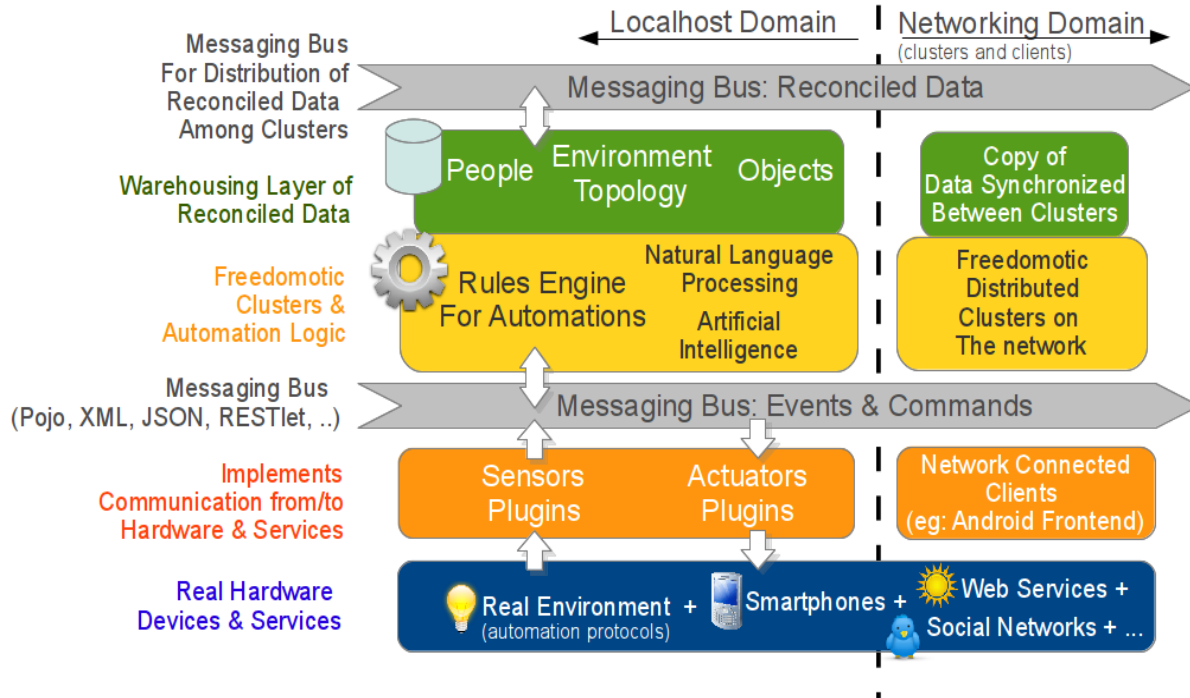
```

Freedomotic is composed by a core (the framework) plus some plugins.

### 13.1 The Framework

The **core part** is a framework that:

1. Implements a language independent messaging system based on Enterprise Integration Pattern. From this follows you can develop in your favorite language and just exchange messages with the other software components. The aim of the messaging system is to link all software modules together in a flexible and abstract way, relating them using the concept of channels (publish-subscribe to different levels of a topics hierarchy)
2. Maintains an internal data structure representing the environment (topology, rooms connections as a graph, ...), the objects in zones and their state (on, off, open, closed, 50% dimmed, ...)
3. Creates an abstraction layer, so users and external software modules can use a high level logic like “turn on kitchen light” instead of “send to COM1 port the string `##A01AON##`”. So a developer you can leverage the others plugins features at an high logical level, is just like the modules can see the same environment map as the user. All data component (environment, objects, triggers, commands) can be defined in XML and easily exchanged on the network between different nodes of the P2P Freedomotic network.
4. Provides a rules engine coupled with a natural language processing system to let the user writing automations in plain English like “if outside is dark turn on living-room light”. You can add, update and delete this automations at runtime using any human computer interface like GUIs, or even speak them.



## 13.2 The Plugins

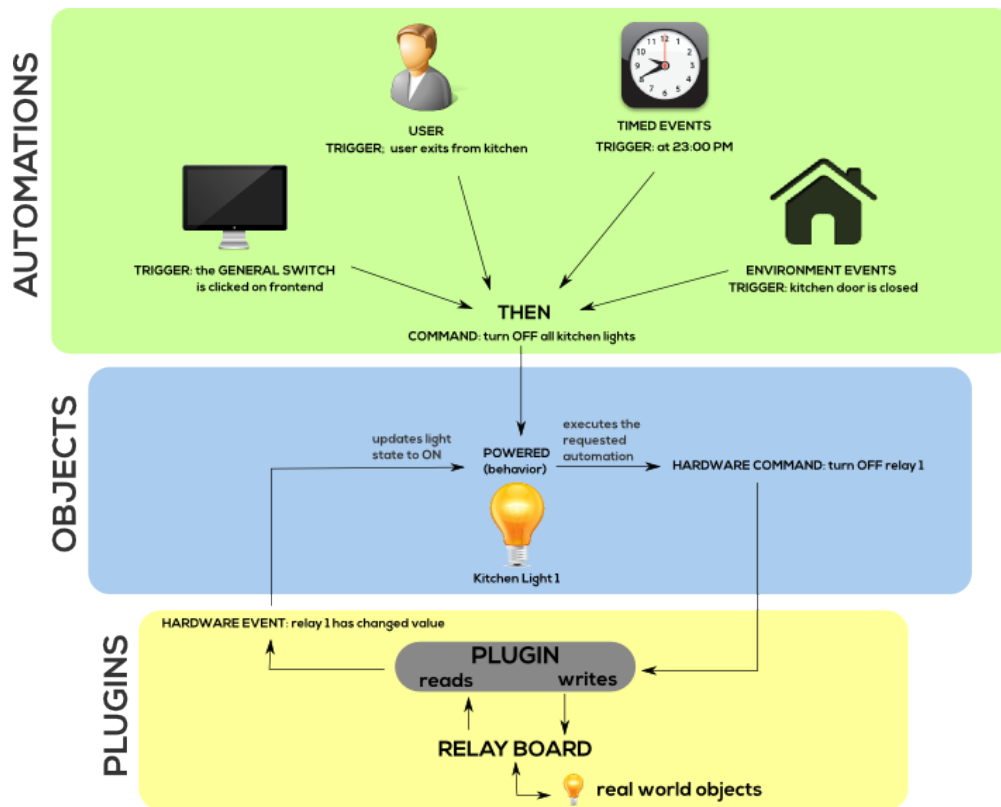
Freedomotic plugins can add more features to the framework and can be developed and distributed as completely independent packages on our marketplace.

### 13.2.1 Device plugins

They usually are developed to communicate with automation hardware like X10, KNX and so on, but also graphical frontends and “web service readers” are Freedomotic plugins just as any other source of info, like webcams, text to speech engines and SMS senders.

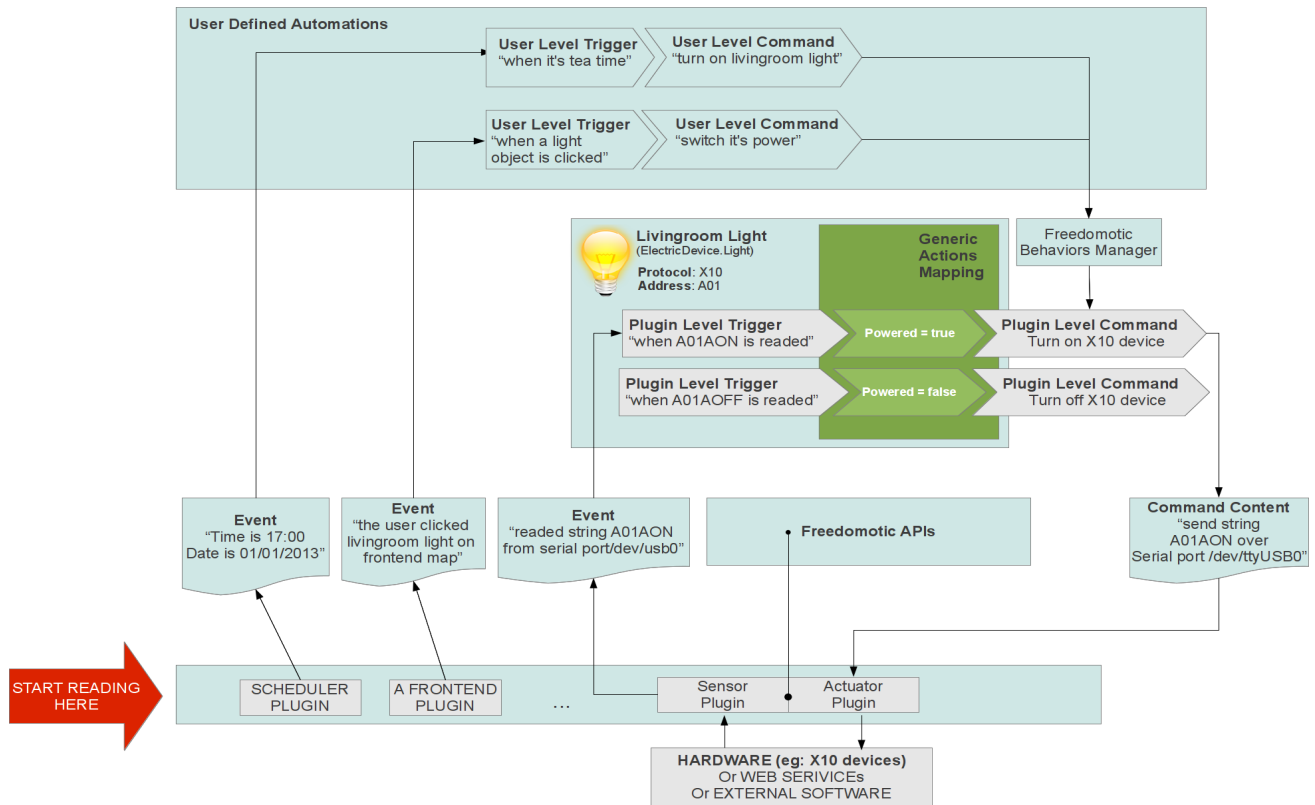
### 13.2.2 Object plugins

You can also develop object plugins which are pieces of software which models the behavior of objects like lamps, doors, etc... instructing the framework on how they behave. For example a lamp object plugin tells the framework that a lamp has a boolean behavior called **powered** and a **dimmed** behavior which is represented by an integer from 0 to 100. A lamp can turn on, turn off and dimm. If dimmed becomes 0% the lamp is **powered=false** and if dimmed > 0% the lamp is **powered=true**.



### 13.3 Plugins, Objects and Automations interaction

Here is a diagram explaining the interaction between plugins, events+triggers+commands and Freedomotic APIs



## 13.4 Explanation

The final goal is to define an automation which can **turn on** the livingroom light when it's tea time (17 o'Clock). The scheduler plugin notifies to Freedomotic the current time (17:00 PM). A trigger named *"it's tea time"* is configured to listen to all time based events. The it's tea time trigger carries a rule inside which is `event.time.hour == 17 AND event.time.minute == 0`. When the event is received by this trigger, the rule is evaluated. If the evaluation succeed then the trigger fires, indicating that now it's actually the time to take the tea. At this point all the corresponding automation IF (trigger: it's tea time) THEN (command: turn on livingroom light) is loaded by the system, and the command is executed forwarding the generic request "turn on livingroom light" to the plugin which can transform it to a protocol dependent command (eg: send string 'A01AON' on serial port /dev/ttyUSB0).



Through messaging system travels Events, Triggers and Commands.

### 14.1 Events

Events are notification of facts like “the state of an hardware device is changed”, “the user have clicked an object on the GUI” or “an object changes its behavior from ON to OFF”. They can be published by any component of the system, a sensor, a frontend, the core itself...

### 14.2 Triggers

Triggers listen for events and filter they values (EVENT: “the user have clicked an object” -> TRIGGER: “if living-room light is clicked”)

### 14.3 Commands

Commands are instructions to do something like “turn on living-room light”.

### 14.4 Reactions (aka Automations)

Reactions bounds trigger and commands to create an automation: “if living-room light is clicked” THEN “turn on living-room light”

## CHAPTER 15

---

### Requirements

---

A Java virtual machine version 8+ (JRE 8+) for your OS.

Probably you have it already installed, if not you can download for free from Oracle website.

---

[Download](#)

---

## 16.1 Latest stable release (recommended)

This is the currently **RECOMMENDED** versions for most users. You can also try the latest dailybuild if you feel brave.

Release	Java Version	Size (MB)	Release Date	Notes
<a href="#">Freedomotic Commander RC4</a>	JRE 8	52	16 Aug 2017	
<a href="#">Freedomotic Commander RC3</a>	JRE 8	45	1 Jul 2016	
<a href="#">Freedomotic Commander RC2</a>	JRE 8	45	16 Nov 2015	
<a href="#">Freedomotic Commander RC1</a>	JRE 8	45	17 Oct 2015	
<a href="#">Freedomotic Bender 5.5.1</a>	JRE 6	44	11 Mar 2014	

[Downloads statistics on Sourceforge](#)

## 16.2 Dailybuilds

These releases are created daily by our [Continuous Integration System on JetBrains Teamcity](#).

Try one of them if you want to be on the bleeding edge of Freedomotic development.

**These releases are unstable** so don't use them in production environments.

[Download the latest available dailybuild \(JRE 8 required\)](#)

---

**Note:** The dailybuilds are created and stored on an external host ([teamcity.jetbrains.com](http://teamcity.jetbrains.com)). Sometimes it is down due to maintenance, we are sorry for the inconvenience, please be patient.

---

## **16.3 Old releases**

Do not use them. These are listed here just for historical reasons. No support will be provided for these versions.

# CHAPTER 17

---

## Installation

---

- *Windows*
- *Linux*
- *Mac OsX*
- *Raspberry Pi*
- *Docker container*

## CHAPTER 18

---

### Windows installation

---

- Download Freedomotic from [here](#)
- Unzip the downloaded file
- Double click on **freedomotic.exe** located into FREEDOMOTIC\_ROOT folder (the extracted folder)

## CHAPTER 19

---

### Linux installation

---

- *Download Freedomotic*
- Unzip the downloaded file
- Double click on **freedomotic.sh** or open the terminal and execute

```
./freedomotic.sh
```

## CHAPTER 20

---

### Mac OsX

---

- *Download Freedomotic*
- Unzip the downloaded file
- Open the terminal (/Applications/Utilities/ folder), go to Freedomotic root folder (the extracted folder) and type the following command

```
java -jar "freedomotic.jar"
```



### 21.1 Get Raspbian OS and install it on an SD card

You can follow the Quick Start guide on the official site.

If you already have an SD card with a working Raspbian OS skip this entirely and start from **Install Freedomotic automatically** section.

### 21.2 Install Freedomotic automatically

After you have a working Raspbian OS you can install Freedomotic on it.

Insert your SD card on Raspberry, boot your system and connect using SSH. On Linux write this on command line

```
ssh pi@YOUR_RASPBERRY_IP
```

Default password is **raspberry**.

Open a terminal and copy/paste this command to install and start Freedomotic automatically

```
sudo curl -s https://raw.githubusercontent.com/freedomotic/freedomotic/master/scripts/  
↪installation/freedomotic-raspberry-install | sh
```

Some things you would like to know:

- Java 8 is required and it is available on Raspbian OS by default.
- In this example Freedomotic runs with a graphical interface and an embedded web client. If you want to run it in *server mode* please remove the plugin **frontend-java** under *plugins/devices* folder.
- To try the web client open your browser and point to <http://ip-raspberry:8090>. As credentials use **admin/admin**.

## CHAPTER 22

---

### Docker container

---

Given you have Docker already installed on your machine you can start the container with the following instruction

```
docker run -d --name=freedomotic -p 9111:9111 -p 8090:8090 ghcr.io/freedomotic/  
↪freedomotic-5.6.0:dailybuild
```

After a few seconds RestAPI interface will be available on port 9111 and the web client on port 8090 of the host machine. So point your browser to <http://ip-docker-server:9111> or to <http://ip-docker-server:8090>.

---

**Note:** The latest version is mapped to **[dailybuild]** tag. If you want to try another version please specify the correct tag as in the following table.

---

Table 22.1: Docker images

Tag	Architecture	Hw platforms	Notes
latest	x86_64	Windows, Linux, Mac OS X	points to the last dailybuild
dailybuild	x86_64	Windows, Linux, Mac OS X	
arm32v7	ARMv7	Raspberry Pi v1-2	
arm64v8	ARMv8	Raspberry Pi v3, Odroid C2, Apple M1	

### 22.1 Container health check

Our images use Docker health check feature to ensure the application is running correctly. Every 5 minutes Docker verifies if the web client is up and stops the container with a code error if it doesn't receive a response by 3 seconds. The first check starts after 10 seconds.

## 22.2 Data persistence

Docker, by default, doesn't come with persistent storage so when the container is removed all data is lost. To fix the problem you need to use Docker volumes.

Our images expose two volumes: ["/srv/freedomotic/data"] and ["/srv/freedomotic/plugins"].

The first contains all data (commands, triggers, reactions, environments and things) and correspond to the "data" folder in the package release. The second contains all the plugins (executables and configuration files) and correspond to the same named folder in the package release

### 22.2.1 Bind mount

The most simple solution is to mount a folder on the host machine and map it to one of the previous volume.

For example if your local Linux folder is "/home/freedomotic-data" you have to copy all the files from "data" folder included in the package release and start the container in this way

```
docker run -d --name=freedomotic -p 9111:9111 -p 8090:8090 -v /home/freedomotic-data:/
↪srv/freedomotic/data freedomotic/freedomotic
```

So if you remove the container all data is saved on the host.

### 22.2.2 Data volumes

This solution is a little more complicated but very useful if you want to share data with other containers inside a cluster.

PD: the volume isn't cancelled if you remove the container.

Here a step by step guide:

- Create a new volume named **"FreedomoticVolume"**

```
docker volume create --name FreedomoticVolume
```

- Create a temporary container to copy the data. It's based on a minimal image (busybox) and mounts the volume under the path **"data"**

```
docker create -v FreedomoticVolume:/data --name temp busybox
```

- Download a Freedomotic package or reuse an existing installation. In every case you need to move to the **"data"** folder and copy its content inside the Docker volume. In order to do this we use **"temp"** container previously created

```
docker cp . temp:/data
```

- Remove **"temp"** container

```
docker rm temp
```

- Start Freedomotic container using **"FreedomoticVolume"**

```
docker run -d --name freedomotic -v FreedomoticVolume:/srv/freedomotic/data -p
↪9111:9111 -p 8090:8090 freedomotic/freedomotic
```

In case of problems you can take a look at the logs with

```
docker logs freedomotic
```

### 22.2.3 Backup

TODO

## 22.3 Docker Hub

All the official images are hosted on [Docker Hub](#).

On next days we'll try to test P2P feature in a Docker environment with (at least) two Freedomotic instances. If you think there may be more interesting usage scenarios for such containers, just share!

---

### Basic configuration

---

All the configuration parameters are included in the file *FREEDOMOTIC\_ROOT/config/config.xml*.

Default values are put in squared brackets.

#### 23.1 Basic

These properties are used to start the framework so **don't change the default values** if not strictly needed.

- **KEY\_BROKER\_URL** [*vm://localhost?persistent=false*]: ActiveMQ broker url
- **KEY\_MESSAGES\_TTL** [*5000*]: messages time to live
- **KEY\_DISCARD\_INVALID\_DESTINATIONS** [*true*]:
- **KEY\_INSTANCE\_ID** [*random uuid*]: uuid of Freedomotic instance

#### 23.2 Internationalization

- **KEY\_ENABLE\_I18N** [*auto*]: enable/disable multilanguage support (for more details [click here](#))

#### 23.3 Logging

- **KEY\_SAVE\_LOG\_TO\_FILE** [*OFF*]: enable/disable logging to file (for more details [click here](#))
- **KEY\_ADMIN\_SENDING\_ADDRESS** [*issue.reporter@freedomotic.com*]: sending email address of log reports (don't change it)
- **KEY\_ADMIN\_RECIPIENT\_ADDRESS** [*mauro@freedomotic.com*]: recipient address for log (don't change it if you want to send the log to the Freedomotic team)

## 23.4 Plugins Marketplace

- **KEY\_CACHE\_MARKETPLACE\_ON\_STARTUP** [**false**]: enable/disable plugins caching on startup
- **KEY\_ENABLE\_PLUGINS\_DOWNLOAD** [**true**]: enable/disable plugins download from marketplace

## 23.5 Periodic data saving

Basically all data are persisted when Freedomotic stopped. You can enable a periodic data saving at a fixed time interval.

- **KEY\_SAVE\_DATA\_PERIODICALLY** [**true**]: enable/disable periodically data saving
- **KEY\_DATA\_SAVING\_INTERVAL** [**15**]: data saving interval in minutes

## 23.6 Persistence

- **KEY\_OVERRIDE\_REACTIONS\_ON\_EXIT** [**true**]:
- **KEY\_OVERRIDE\_OBJECTS\_ON\_EXIT** [**true**]:

## 23.7 P2P

- **KEY\_P2P\_CLUSTER\_NAME** [**freedomotic-commander**]: cluster name used to identify the instance in a p2p network

## 23.8 Resources

- **KEY\_RESOURCES\_PATH** [/data/resources/]:
- **KEY\_ROOM\_XML\_PATH** [/df28cda0-a866-11e2-9e96-0800200c9a66/df28cda0-a866-11e2-9e96-0800200c9a66.xenv]:

## 23.9 Security

- **KEY\_ENABLE\_SSO** [**false**]: enable/disable single sign-on for authentication
- **KEY\_SECURITY\_ENABLE** [**true**]: enable/disable authentication

## CHAPTER 24

---

### Internationalization

---

---

### What is a plugin?

---

Freedomotic is an application extensible through plugins. Plugins are simple classes within a .jar java package. Each plugin is deployed in the FREEDOMOTIC\_ROOT/plugins/ folder and loaded and initialized automatically at Freedomotic startup. The communication between the plugin and Freedomotic is automatically managed via a Message Oriented Middleware. Plugins in addition to the 'Manager of the messages' have direct access to Freedomotic data structures. In a plugin, you can create, read, update, or delete data and use them to accomplish your goals.

### 25.1 Plugin features

1. plugin configuration management
2. user interface accessible by right click in plugins list
3. simplified access to freedomotic data structures
4. automatic management of plugin lifecycle (loaded, running, stopped,...)
5. access to the messaging system (read/write events and commands)
6. installation and upgrade of plugins from a marketplace
7. simplified programming implementing events (onCommand(), onRun(), onStart(), onStop(), ...).

### 25.2 Plugin manifest and configuration

Every Java plugin needs a XML manifest file to describe the plugin to Freedomotic. As multiple plugins can be in the same plugin package (the one you download from the marketplace) then more than one manifest file can be in the root folder of a plugin package.

This is an example of the most simple manifest file you can have:

```
<config>
  <properties>
    <property name="name" value="Hello World"/>
  </properties>
</config>
```



```

    <property name="description" value="A basic plugin that prints 'Hello World' on_
↪standard output"/>
    <property name="category" value="examples"/>
    <property name="short-name" value="hello-world"/>
  </properties>
</config>

```

Every plugin has a unique input **Messaging Channel** used for message exchange; it is addressed using the info you put in the manifest file. For plugins the Channel name is: app.actuators.CATEGORY.SHORT-NAME.in The plugin manifest is the ONLY place you should add configuration parameters for your plugin. You should not use external files; the manifest is all you need to include complex configuration options that can be changed at runtime using a Freedomotic frontend. You can add custom properties to this list.

### 25.2.1 Add configuration blocks to your plugin

If you have to configure multiple the same set of properties for different objects (eg: URL and port of a set of hardware boards) you can use tuples. You can add as many blocks as you need. The block may be added after on the same hierarchical level. Tuples are useful to have configuration data specific for you plugin to be loaded in Freedomotic at startup. Related configuration data can be stored in blocks called tuple.

```

<tuples>
  <tuple>
    <property name="Name" value="TemperatureZone1"/>
    <property name="SlaveId" value="1"/>
    <property name="RegisterRange" value="HOLDING_REGISTER"/>
    <property name="DataType" value="TWO_BYTE_INT_UNSIGNED"/>
    <property name="Offset" value="266"/>
    <property name="NumberOfRegisters" value="1"/>
    <property name="Multiplier" value="0.1d"/>
    <property name="Additive" value="0.0d"/>
    <property name="EventName" value="TemperatureZone1"/>
  </tuple>
  <tuple>
    <property name="Name" value="TemperatureZone2"/>
    <property name="SlaveId" value="1"/>
    <property name="RegisterRange" value="HOLDING_REGISTER"/>
    <property name="DataType" value="TWO_BYTE_INT_UNSIGNED"/>
    <property name="Offset" value="522"/>
    <property name="NumberOfRegisters" value="1"/>
    <property name="Multiplier" value="0.1d"/>
    <property name="Additive" value="0.0d"/>
    <property name="EventName" value="TemperatureZone2"/>
  </tuple>
</tuples>

```

This data is automatically available to your plugin. You can use free custom strings for the attribute name and the value. To read these tuples variables programmatically see the [\[Data Access Freedomotic Data page\]](#)

#### 26.1 Install manually

- Go to the [plugin marketplace](<https://github.com/freedomotic/freedomotic-plugins-marketplace>) and download the package file. The **.device** is a compressed archive so you can extract it simply by using any tool as winzip, winrar or unzip. There are two types of plugins: **devices** and **objects**.
- Extract “**devices**” to FREEDOMOTIC\_ROOT/plugins/devices folder.
- Extract “**objects**” to FREEDOMOTIC\_ROOT/plugins/objects folder.
- Restart Freedomotic.

## CHAPTER 27

---

### Plugin configuration

---



## CHAPTER 28

## Devices plugins list

Plugin	Description	Categories
<i>Arduino Remote Controller</i>	Receives IR signals from a remote controller and execute Freedomotic commands	Arduino, Automation Protocols, IoT
<i>Arduino Serial Communication</i>	A basic example to communicate with an Arduino board via serial connection	Arduino, Automation Protocols, IoT
<i>Arduino WeatherShield</i>	Interacts with Arduino WeatherShield by Ethermania.com	Arduino, HVAC, Weather
<i>BT Speech Recognition</i>	Speech recognition with Bluetooth & Android	Speech Recognition&TTS
<i>Chat</i>	A allows a user to interact with a Freedomotic server by 'chatting' with XMPP protocol	Frontend, Network&Communication
<i>Devantech Ltd Eth-Rly 16</i>	Controls the Ethernet RLY16 board developed powered by Devantech Ltd	Automation Protocols
<i>Google Calendar Events</i>	Allows to create time based automations using your Google calendar events	Network&Communication, Utilities
<i>Harvester</i>	Collects events in a db, for data analysis and much more	Utilities
<i>HWg-STE Ethernet thermometer</i>	Reads values from an ethernet thermometer powered by HW-group.com	HVAC
<i>InfluxDB Persistence</i>	Time series persistence on InfluxDB	Utilities
<i>IPCamera Motion</i>	Detects motion in MPEG streams	Media, Utilities
<i>GCE Electronics IPX800</i>	Enables communication with Ipx800 boards by gce-electronics.com	Automation Protocols
<i>Mailer</i>	Sends notification by email	Network&Communication, Utilities
<i>MaryTTS Text to Speech</i>	Text To Speech based on MaryTTS library	Speech Recognition&TTS
<i>Modbus</i>	Enables the communication with Modbus devices	Automation Protocols, IoT
<i>MQTT Broker</i>	A broker for MQTT protocol based on Moquette library	Automation Protocols, IoT
<i>MQTT Client</i>	A client for MQTT (MQ Telemetry Transport)	Automation Protocols, IoT
<i>MySensors</i>	Plugin for MySensors gateway	Automation Protocols, IoT
<i>openPicus Flyport</i>	Communicates with Flyport boards by openpicus.com	Automation Protocols, IoT

Plugin	Description	Categories
<i>Persistence</i>	Persists events and commands on a Cassandra database	Utilities
<i>Progetti-HwSw Ethernet Board v2</i>	Communicates with an ethernet relay board powered by ProgettiHw-Sw	Automation Protocols
<i>pUrl</i>	Reads URLs content like XML, HTML, JSON and sends it in a listenable event	Net-work&Communication, Utilities
<i>Push Notifications</i>	Sends custom push messages through many providers	Net-work&Communication, Utilities
Room based events	Sends events related to rooms' status and creates triggers accordingly	Utilities
<i>Sensors simulation and utilities</i>	A set of sensor simulators and utilities	Utilities
<i>Tcw1228-cm</i>	Controls a TCW122B-CM module powered by Teracom.cc	Automation Protocols
<i>Telegram Bot</i>	Controls your home via Telegram Bot	Access Control&Security, Social
<i>ThingSpeak</i>	Sends sends data to ThingSpeak.com platform	Utilities
<i>TTS Text to Speech</i>	Converts text to sound	Speech Recognition&TTS
<i>Twilight - Sunset and sunrise alerts</i>	Sends events related to Sunrise and Sunset time for the configured lat/long	Utilities, Weather
<i>Twitter4Freedomotic</i>	Makes your home post messages on Twitter social network	Social
<i>Webserver</i>	A modern browser based frontend for Freedomotic	Frontend
<i>Zibase</i>	Controls a Zibase board powered by zodianet.com	Automation Protocols
<i>Zwave</i>	Allows interfacing with Zwave-powered devices	Automation Protocols

---

## Arduino Remote Controller

---

**Description:** Use an Arduino board to receive IR signals from a remote controller and execute Freedomotic commands

**Type:** - **Categories:** Arduino, Automation Protocols, Internet of Things

**Development status:** Prototype

**Tested on:** All platforms

**Developer:** Mauro Cicoletta

### 29.1 Overview

This plugin uses an Arduino board with an ethernet shield to receive IR commands from a remote controller. Then it tells Freedomotic which button has been pressed and a specific command is executed (e.g. if you press the button 1 the kitchen light switches on/off).

The plugin consists of two parts a sketch to upload on your Arduino board and the java code. For this test we have used a Leadtek remote for the WinFast TV board. Most of the common remote controllers work well.

### 29.2 Configuration

First of all you must detect your remote codes using [this library](#). You can find a good tutorial [here](#).

Then you must substitute the hex code into the following sketch (included into the plugin package).

```
// Freedomotic Remote Controller Plugin
// by Mauro Cicoletta
// www.freedomotic.com

#include <IRremote.h>
#include <String.h>
#include <SPI.h>
```

```

#include <Ethernet.h>
#include <EthernetUdp.h>

int pinIRreceiver = 11;
IRrecv IRreceiver(pinIRreceiver);
decode_results receivedSignal;

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED }; // mac address
byte ip[] = { 192, 168, 0, 2 }; // ip arduino
byte subnet[] = { 255, 255, 255, 0 }; //subnet mask
byte gateway[] = { 192, 168, 0, 1 }; // ip gateway
EthernetServer server(80); // server web listening on port 80

/* UDP configuration */
unsigned int UDPport = 7878;
EthernetUDP Udp;
IPAddress ipServerUDP(192, 168, 0, 20);
unsigned int ServerUDPport = 8000;

void setup()
{
  Serial.begin(9600); // serial monitor
  IRreceiver.enableIRIn();
  Ethernet.begin(mac, ip, gateway, subnet);
  Udp.begin(UDPport);
}

void loop()
{
  if (IRreceiver.decode(&receivedSignal)) // received IR signal
  {
    IRreceiver.resume(); // ready to receive the next signal

    switch (receivedSignal.value) {
      // button 1
      case 0xC03FA05F:
        Serial.println("Pressed 1 button");
        sendUDPpacket('1');
        break;

      // button 2
      case 0xC03F609F:
        Serial.println("Pressed button 2");
        sendUDPpacket('2');
        break;

      // button 3
      case 0xC03FE01F:
        Serial.println("Pressed button 3");
        sendUDPpacket('3');
        break;

      // button 4
      case 0xC03F906F:
        Serial.println("Pressed button 4");
        sendUDPpacket('4');
        break;
    }
  }
}

```



```

        // button 5
        case 0xC03F50AF:
            Serial.println("Pressed button 5");
            sendUDPPacket('5');
            break;
    }
}

void sendUDPPacket(char button)
{
    Udp.beginPacket(ipServerUDP, ServerUDPport);
    Udp.write("AN1:");
    Udp.write(button);
    Udp.endPacket();
}

```

To avoid polling Arduino sends an udp packet to the udp server embedded into the plugin. The packet has the format **AN1:X** where X represents the pressed button. For example **AN1:1**.

The plugin starts an embedded udp server listening by default on address **192.168.0.20** and port **8000**. These values are specified into the manifest file *arduino-remote-controller-manifest.xml*. If you change them you must consequently change the sketch.

When a new udp packet arrives, the plugin extracts the encapsulated data and notifies Freedomotic an event with the pressed button.

In the plugin data folder there are some user level triggers called When button X is pressed on remote (where X represents the pressed button) filtering the events and their button.pressed property.

Now we can create automations from jfrontend in the form of when button X is pressed on remote THEN "ADD HERE ANY FREEDOMOTIC COMMAND".

We can make Freedomotic tweet it, turn off all lights, send a mail, speech some text, whatever...

## 29.3 Video

## 29.4 Download

[Download plugin latest version](#)

## 29.5 Source code

[GitHub repository](#)

---

## Arduino Serial Communication

---

**Description:** A basic example to communicate with an Arduino board via serial connection

**Type:** Driver - **Categories:** Arduino, Automation Protocols, IoT

**Development status:** Prototype

**Tested on:** All platforms

**Developer:** Mauro Cicoella

### 30.1 Overview

This example shows how to control an Arduino board connected via usb interface in order to switch on/off a led. The code is very simple but it's a base from which to create more complex plugin using serial protocol.

### 30.2 Configuration

To configure the plugin open *arduinousb-manifest.xml* in its folder and change the following properties. By default you need to change only the port name and set the one in use.

- **serial.port:** port name e.g. /dev/ttyACM0 for Linux or COMx for Windows
- **serial.baudrate:** baudrate of Arduino serial port (the same used in the sketch)
- **serial.databits:** databits of Arduino serial port
- **serial.parity:** parity bit of Arduino serial port
- **serial.stopbits:** stop bit of Arduino serial port
- **chunk.terminator:** read serial string terminator (by default "&#xA;" as "n")
- **chunk.size:** size of chunk to read from serial
- **delimiter:** fields delimiter in read string

## 30.3 Arduino sketch

The following sketch must be uploaded to your arduino board. It's very simple: when it receives a char 'a' from serial connection it switch on the led connected to pin 13. Viceversa with 'b' char the led is switched off. Every command writes a string on serial.

```
void setup() {
  Serial.begin(9600);
  //Set all the pins we need to output pins
  pinMode(13, OUTPUT);
}

void loop () {
  if (Serial.available()) {

    //read serial as a character
    char ser = Serial.read();

    //NOTE because the serial is read as "char" and not "int", the read value
    //must be compared to character numbers
    //hence the quotes around the numbers in the case statement
    switch (ser) {
      case 'a':
        pinON(13);
        Serial.println("13;on");
        break;
      case 'b':
        pinOFF(13);
        Serial.println("13;off");
        break;
    }
  }
}

void pinON(int pin) {
  digitalWrite(pin, HIGH);
}

void pinOFF(int pin) {
  digitalWrite(pin, LOW);
}
```

## 30.4 Download

[Download plugin latest version](#)

## 30.5 Source code

[GitHub repository](#)

---

## Arduino WeatherShield

---

**Description:** Communicate with Arduino WeatherShield by Ethermania.com

**Type: - Categories:**

**Development status:**

**Tested on:** All platforms

**Developer:** Mauro Cicoella

### 31.1 Overview

The WeatherShield, created by [Ethermania.com](<http://www.ethermania.com/>), is a unit that lets Arduino able to read Pressure, Temperature and Relative Humidity. The shield is equipped with three sensors and a PIC12F683 microcontroller programmed with a specific firmware that interfaces with Arduino through a two line bidirectional synchronous serial connection. The microcontroller is responsible for sampling and averaging the last 8 humidity, pressure and temperature values, providing it to the Arduino in an ASCII readable format expressed in Celsius, hPa and relative % units. More details can be found [here]([http://www.ethermania.com/shop/index.php?main\\_page=product\\_info&cPath=91\\_104&products\\_id=612&language=en](http://www.ethermania.com/shop/index.php?main_page=product_info&cPath=91_104&products_id=612&language=en)).

### 31.2 Board protocol

The proposed solution uses an ethernet shield to connect Arduino board to your lan. On Arduino you must upload the included sketch to interact with the WeatherShield to retrieve sensors values and shows them as a string with ":" char as delimiter. Also you must add to your Arduino IDE the WeatherShield lib (included into PLUGIN\_ROOT/resources/WeatherShield/WeatherShieldLibrary).

The default address is 192.168.0.150 on port 80. If you desire to change it you must edit the sketch, recompile and upload to Arduino board. To verify if it works open your browser and digit the board address "<http://192.168.0.150>". You would see a string reporting the sensors values.

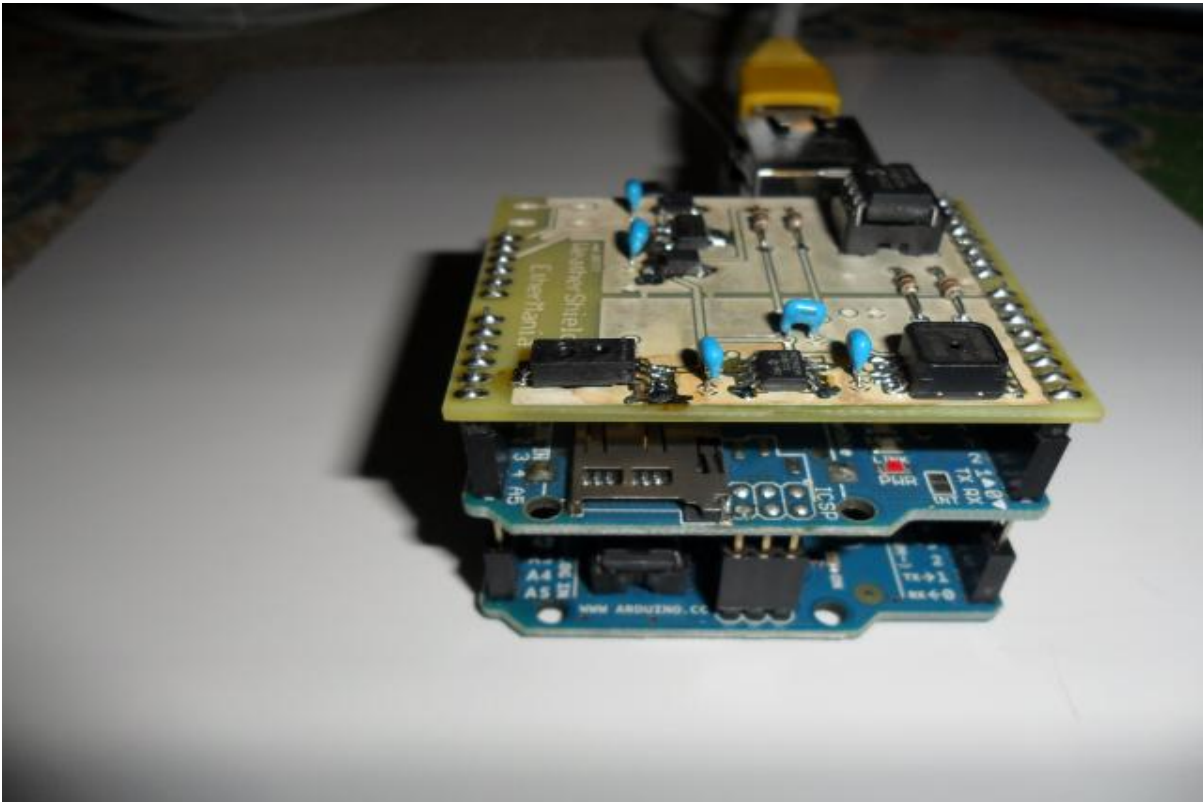


Fig. 31.1: Arduino WeatherShield

## 31.3 How to read values from an object

With this board you can read the temperature behavior of any thermometer device in your environment.

For this example we use a thermometer object:

- Right click on the thermometer object in the environment to show its configuration panel
- Change the property “protocol” to “ArduinoWeatherShield”
- Change the property “address” to a string composed of HTTP\_BOARD\_URL:HTTP\_PORT. For example “192.168.0.150:80:T” identifies the board listening on 192.168.0.150:80 for temperature values (use “P” for pressure and “H” for humidity).
- Under “temperature” (in Data Source Configuration) select the trigger called “Arduino WeatherShield reads temperature change”

The plugin is also able to read any humidity and pressure change.

From 5.6 version Freedomotic uses autodiscovering feature so 3 new things (thermometer, barometer and hygrometer) are added and configured automatically.

## 31.4 Download

[Download plugin latest version](#)

## 31.5 Source code

[GitHub repository](#)

---

### BT Speech Recognition

---

**Description:**

**Type: - Categories:**

**Development status:**

**Tested on:** All platforms

**Developer:** Mauro Cicoletta

### 32.1 Overview

With this plugin you can send vocal commands to Freedomotic using Google Speech Recognition service and blue-tooth to communicate.

You need a bluetooth dongle and an Android phone/tablet.

### 32.2 Configuration

Download and install the app [https://play.google.com/store/apps/details?id=robotspace.simplelabs.amr\\_voice](https://play.google.com/store/apps/details?id=robotspace.simplelabs.amr_voice)

Connect to the pc running Freedomotic.

Now you can say your command (e.g. “Turn on Kitchen Light”) and it’ll be executed.

The plugin finds the more similar command using NLP (<https://github.com/freedomotic/freedomotic/wiki/Natural-language-processing>)

### 32.3 Video

**Description:** This plugin allows a user to interact with a Freedomotic server, by just ‘chatting’ with it using the most popular protocol XMPP (the one that runs, for instance, Jabber.org )

**Type:** Driver - **Categories:** Frontends, Network & Communication

**Development status:** Beta version

**Tested on:** All platforms

**Developer:** Matteo Mazzoni

### 33.1 Overview

### 33.2 Configuration

In order to use the plugin you need to have 2 or more XMPP compatible accounts:

- the one to be used by the plugin (you can get a free account at Jabber.org)
- one for each user allowed to remote control Freedomotic

Post installation steps:

edit chat-manifest.xml, insert login data and set a password in order to make an unknown user to a ‘friend’. restart Freedomotic access your IM account add the Freedomotic-related user, send a first message containing the above password once accepted as a friend you can start remote controlling Freedomotic Usage Remote-Control capabilities:

run commands run commands on a certain condition (e.g. a reaction with a single run capability) add new reactions

NEED HELP ON USAGE? Just send Chat plugin a ‘HELP’ message.



---

### Google Calendar Events

---

**Description:**

**Type: - Categories:**

**Development status:**

**Tested on:** All platforms

**Developer:** Enrico Nicoletti

### 34.1 Overview

### 34.2 Configuration

**Description:** Collects events in a db, for data analysis and much more

**Type:** Driver - **Categories:** Utilities

**Development status:** Prototype

**Tested on:** All platforms

**Developer:** Matteo Mazzoni

## 35.1 Overview

If you need to collect events, this plugin is what you are looking for.

Features:

- Filter events based on Trigger mechanism
- Sample configuration for major dbms, including Mysql, MSQL, H2, Sqlite
- Easily support almost any dbms

## 35.2 Configuration

---

**Note:** Users running JAVA1.7 and up, must add a vm option when running Freedomotic in order to workaround a known issue with OpenJPA and ClassLoader. Option is: -XX:+AlwaysLockClassLoader

---

Post installation steps:

- edit harvester-manifest.xml, to select your db type
- edit \_YOUR\_DB\_TYPE\_.xml to properly configure it

- restart Freedomotic

## 35.3 Usage

In order to save an event on db you have to ‘intercept’ it. This is easy with the built-in Reaction mechanism in Freedomotic.

A basic Reaction is provided alongside the plugin, and it is made up by:

- A trigger, “When a object changes its status”
- A command “Save event on Harvester”
- A reaction that links the elements above

You could made your custom trigger, and link it to the “Save event on Harvester” command in order to put on db other kinds of data.

---

### HWg-STE Ethernet thermometer

---

**Description:** This plugin reads values from an ethernet thermometer by HW-group.com

**Type:** Driver - **Categories:** HVAC

**Development status:** Stable Release

**Tested on:** All platforms

**Developer:** Mauro Cicoletta

### 36.1 Overview

This plugin reads values from an ethernet thermometer by HW-group.com. You can connect up to two sensors of temperature and humidity. Indoor/outdoor sensors are available. Even temperature sensor with flat cable for temperature monitoring in the fridges. More details at [http://www.hw-group.com/products/HWg-STE/STE\\_ip\\_temperature\\_sensor\\_en.html](http://www.hw-group.com/products/HWg-STE/STE_ip_temperature_sensor_en.html)

It works sending periodically **SNMP** (Simple Network Management Protocol) request to the device using the **OID** (Object Identifier) specified by the manufacturer ([http://ste.hwgroup.cz/HWg-STE\\_OID.txt](http://ste.hwgroup.cz/HWg-STE_OID.txt)).

### 36.2 Configuration

All the configuration parameters are included in the *hwgste-manifest.xml* file. You can use it without any changes.

Each thermometer data are included in a `<tuple></tuple>` block where you need to specify:

- an 'alias' to identify the device
- ip address of the device
- port of the device
- number of sensors (2 by default)

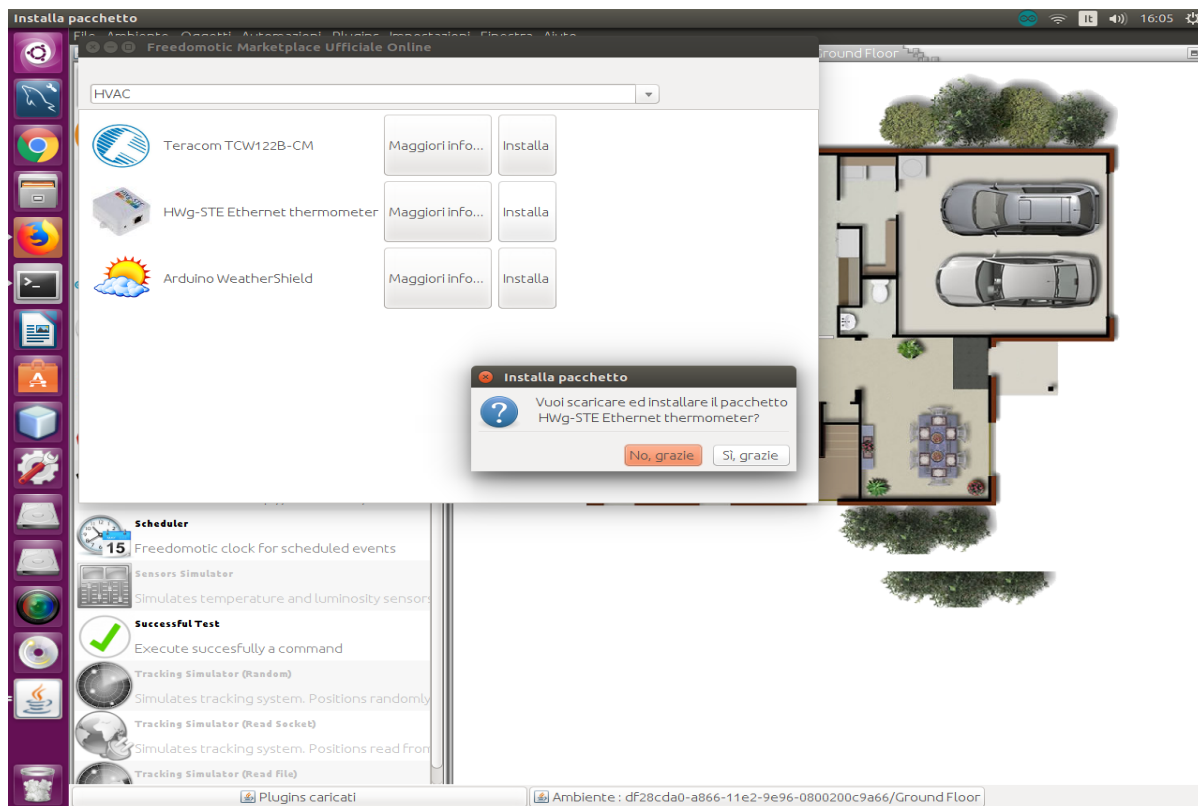


Fig. 36.1: HWg-STE plugin download

## 36.3 How to read data from a sensor

### 36.3.1 Automatic setting

By default the auto configuration feature is enabled (`<property name="enable-autoconfiguration" value="true"/>`) so the plugin is able to add a configured thing on the map.

The type of thing is based on the unit read from the sensor: Thermometer for 'C'. 'F' or 'K'; Barometer for '%'.

### 36.3.2 Manual setting

First of all you have to set this property `<property name="enable-autoconfiguration" value="false"/>` to disable auto configuration.

Add a **Thermometer** thing to your Freedomotic environment and then:

- Right click on the **Thermometer** in the environment to show its configuration panel
- Change the property **protocol** to **hwgste**
- Change the property **address** to a string composed of **ALIAS:SENSOR-ID** where **ALIAS** is the alias used to identify your device and **SENSOR-ID** is its id. An example address can be **board1:215**.
- Under **temperature** (in **Data Sources**) select from the list `HWg-STE reads temperature`.

The same procedure is required for **humidity**. In this case you have to add a **Barometer** thing to the map, set **address** and **protocol** as described previously and assign the trigger `HWg-STE reads humidity` to **humidity** behavior.

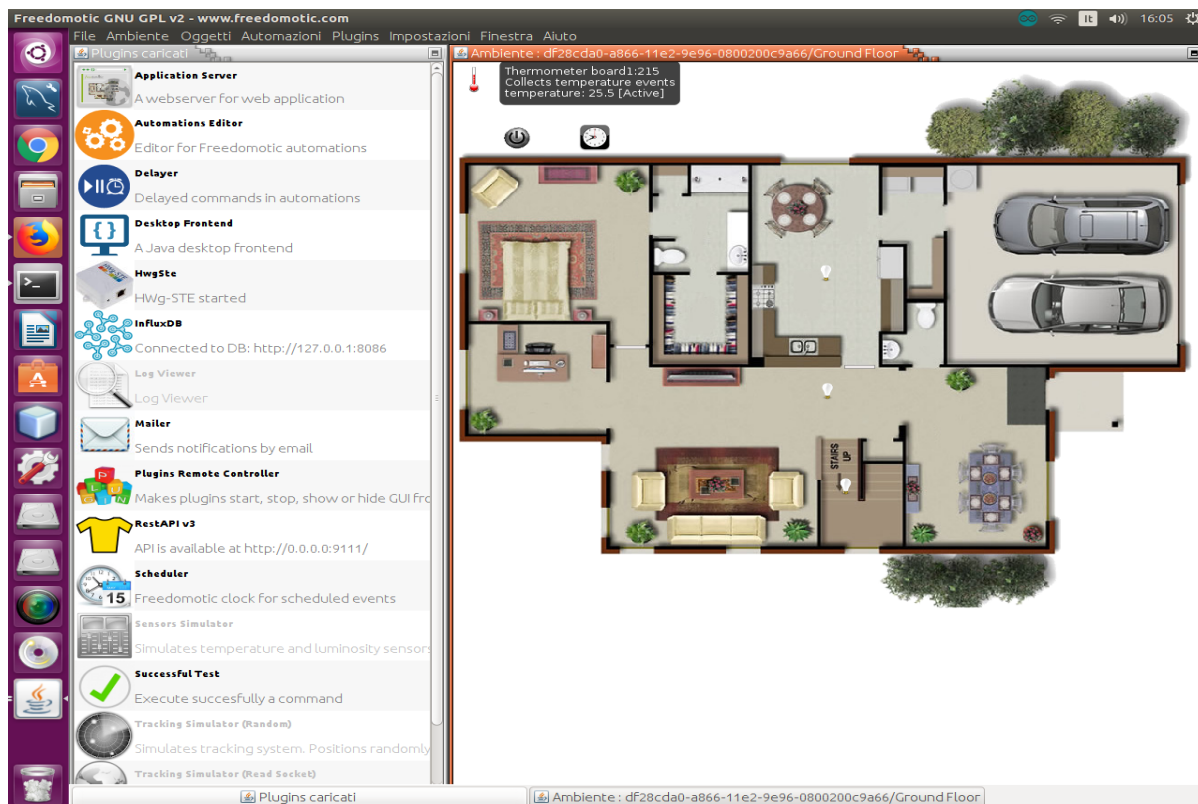


Fig. 36.2: Thermometer reads data from a sensor



Fig. 36.3: HWg-STE online demo

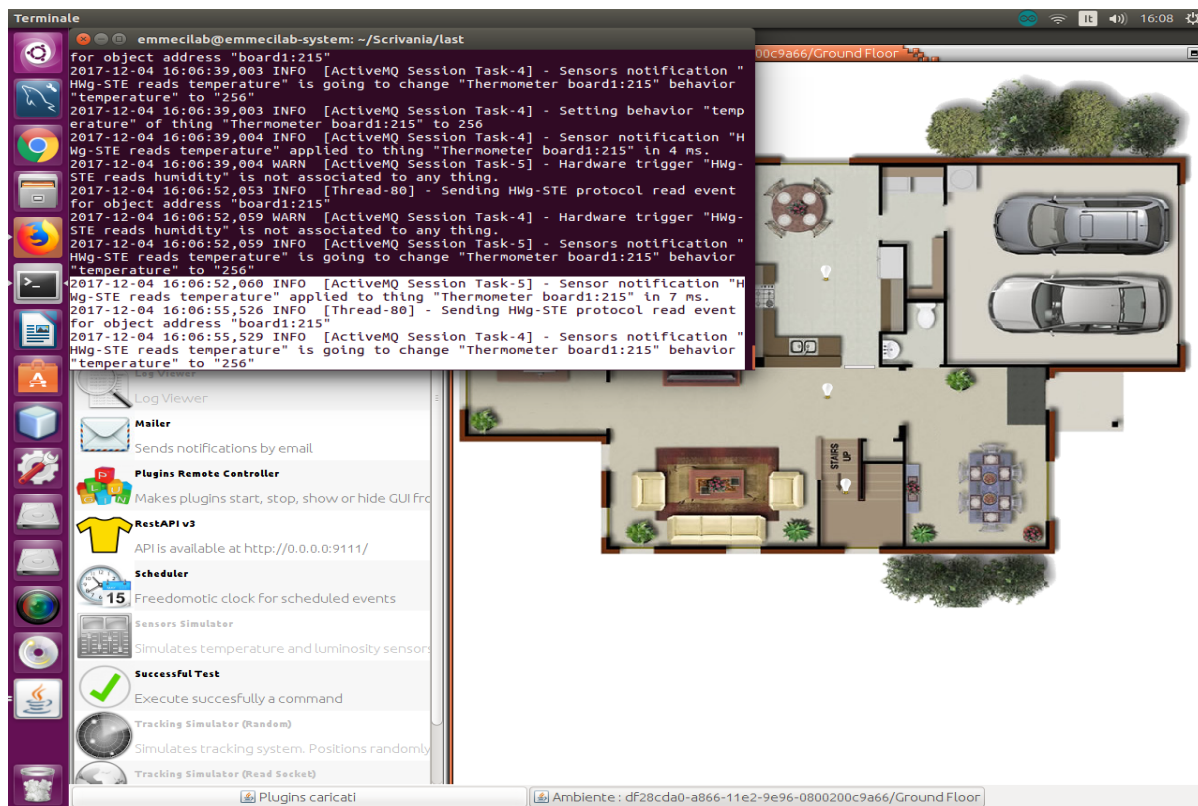


Fig. 36.4: HWg-STE plugin data log



## 36.4 Download

[Download plugin latest version](#)

## 36.5 Source code

[GitHub repository](#)

---

InfluxDB Persistence

---

**Description:** Time series persistence on InfluxDB

**Type:** Driver - **Categories:** Utilities

**Development status:** Prototype

**Tested on:** All platforms

**Developer:** Mauro Cicoletta

## 37.1 Overview

This plugin is intended to store data into an InfluxDB database. It collects all things behaviors and values in the form of time series so data can be analyzed and used to create charts.

## 37.2 Configuration

First of all you need a running instance of InfluxDB and you have to set the connection parameters into the manifest file. If the authentication is enabled username and password are required. in the *influxdb-persistence-manifest.xml* file. If the authentication is enabled on the database, username and password are required.

By default the database name is “freedomotic” but you can change it.

Parameter	Description	Default value
db-url	database url	<a href="http://127.0.0.1:8086/">http://127.0.0.1:8086/</a>
db-name	database name	freedomotic
username	username to access the database	root
password	password to access the database	root
retention-policy	data retention policy	autogen

## 37.3 Database structure

This plugin saves data about Freedomotic things' behaviors "only" when the measurement changes. For example you could have a single stored datapoint if the temperature is constant for an hour or multiple datapoints if it changes.

The database has the following structure:

Name	Description	Type
time	timestamp of measurement - autogenerated	timestamp
freedomotic_instance_uuid	UUID of Freedomotic instance	field
object_name	name of the object	field
object_protocol	protocol of the object	field
object_address	address of the object	field
object_uuid	uuid of the object	field
value	measurement value	measurement

```

emmecilab@emmecilab-system: ~/Scrivan/last
emmecilab@emmecilab-system:~/Scrivan/last$ influx
Connected to http://localhost:8086 version 1.3.4
InfluxDB shell version: 1.3.4
> use freedomotic
Using database freedomotic
> select * from powered
name: powered
time                freedomotic_instance_uuid  object_address  object_name  object_protocol
object_uuid          value
-----
1504185858556000000  5bfeb800-1a6e-4f4b-9f2d-f456ef766f02  unknown        Livingroom light  unknown
76fd619c-617b-4fb7-9870-8f08164c4926  1
1504186033460000000  5bfeb800-1a6e-4f4b-9f2d-f456ef766f02  unknown        Kitchen Light     unknown
21969e70-7e00-4a0d-a13f-a4728240d9d0  0
1504186035025000000  5bfeb800-1a6e-4f4b-9f2d-f456ef766f02  unknown        Livingroom light  unknown
76fd619c-617b-4fb7-9870-8f08164c4926  0
1504186036375000000  5bfeb800-1a6e-4f4b-9f2d-f456ef766f02  unknown        Kitchen Light-145 unknown
24b01c5b-ea44-4a6e-8d89-f3f468765840  1
1504186041264000000  5bfeb800-1a6e-4f4b-9f2d-f456ef766f02  unknown        Kitchen Light     unknown
21969e70-7e00-4a0d-a13f-a4728240d9d0  1
1504186044807000000  5bfeb800-1a6e-4f4b-9f2d-f456ef766f02  unknown        Kitchen Light     unknown
21969e70-7e00-4a0d-a13f-a4728240d9d0  1
1504186047648000000  5bfeb800-1a6e-4f4b-9f2d-f456ef766f02  unknown        Kitchen Light-145 unknown
24b01c5b-ea44-4a6e-8d89-f3f468765840  1
1504186048298000000  5bfeb800-1a6e-4f4b-9f2d-f456ef766f02  unknown        Kitchen Light-145 unknown
24b01c5b-ea44-4a6e-8d89-f3f468765840  0
> select * from powered where object_name = 'Livingroom light'
> select * from powered where object_name = 'Livingroom light'
name: powered
time                freedomotic_instance_uuid  object_address  object_name  object_protocol
object_uuid          value
-----
1504185858556000000  5bfeb800-1a6e-4f4b-9f2d-f456ef766f02  unknown        Livingroom light  unknown
76fd619c-617b-4fb7-9870-8f08164c4926  1
1504186035025000000  5bfeb800-1a6e-4f4b-9f2d-f456ef766f02  unknown        Livingroom light  unknown
76fd619c-617b-4fb7-9870-8f08164c4926  0
>

```

Fig. 37.1: InfluxDB data

You can use these fields to create complex queries.

## 37.4 How to save data

In order to save data you need to create a specific automation. In Jfrontend go to Automations -> Manage Automations and select one of the following triggers:

- When a temperature has changed (detects any change about temperature)

- When an electric device behavior has changed (detects any change - on/off - about electric devices)
- When a generic sensor behavior has changed (detects any change about sensors)

and write in the Command text field `Save data on InfluxDB`. Confirm and press OK button.

## 37.5 Download

[Download plugin latest version](#)

## 37.6 Source code

[GitHub repository](#)

---

### IpCamera Motion Plugin

---

**Description:** This plugin detects motion in MPEG streams

**Type:** Driver - **Categories:** Media, Utilities

**Development status:** Prototype

**Tested on:** All platforms

**Developer:** Mauro Cicoletta

### 38.1 Overview

### 38.2 Configuration

- Open the *camera.xml* file and set a name for the camera and the url to access to the stream in your browser (take a look at the examples).
- Double click on the plugin icon to start it. Then right click on the same icon and on **Configure IPCamera plugin** to open the GUI and see all the images from the cameras.

### 38.3 How to save a captured image

It's possible to save an image when motion is detected by a camera. The default folder is *FREEDOMOTIC\_ROOT/plugins/devices/ipcamera-motion/data/captured-images/*.

- Open **Manage Automations** (F7 key) and search the trigger IpCamera motion detected. In the command field write *Capture image from an IpCamera*.

Press the **Confirm** button and then **OK** at the bottom of the window.

Now for every notified event a message will appear on the map and a snapshot of the camera will be saved in a file named as the following format *[name-webcam]/[date]/[hour].jpg*.

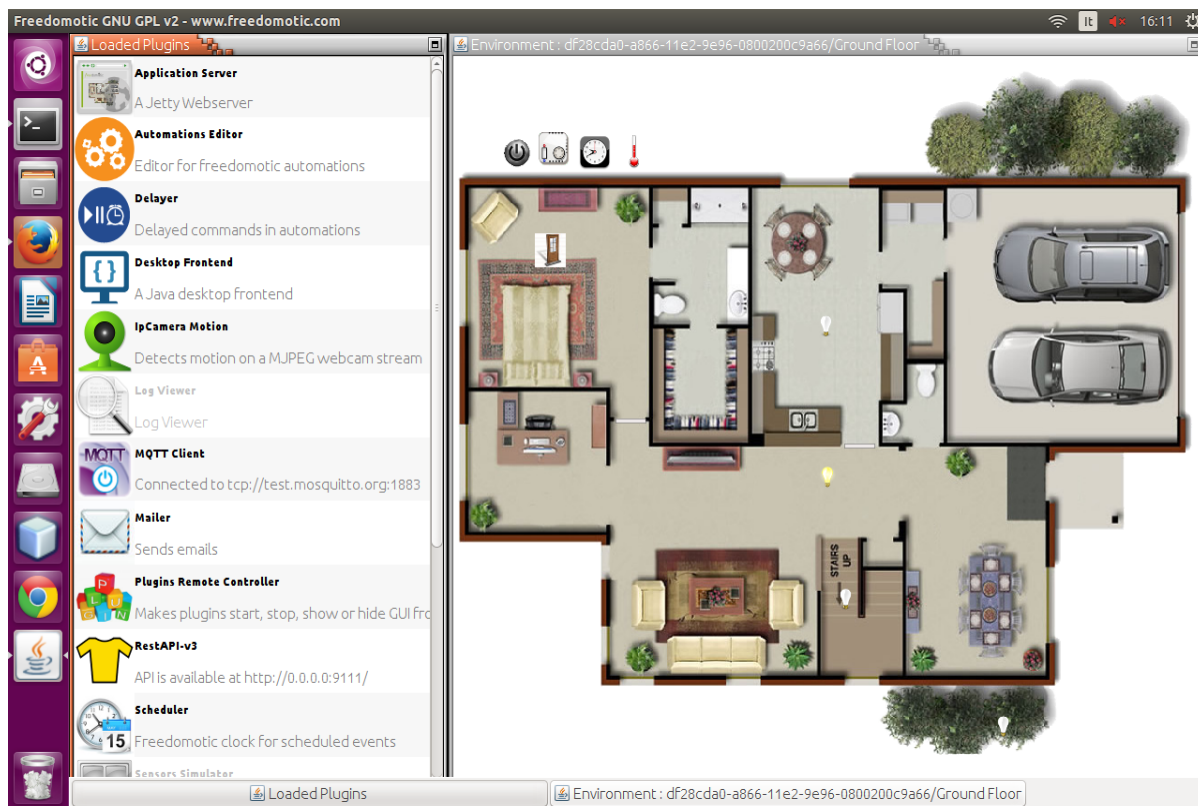


Fig. 38.1: IPCamera Motion Plugin

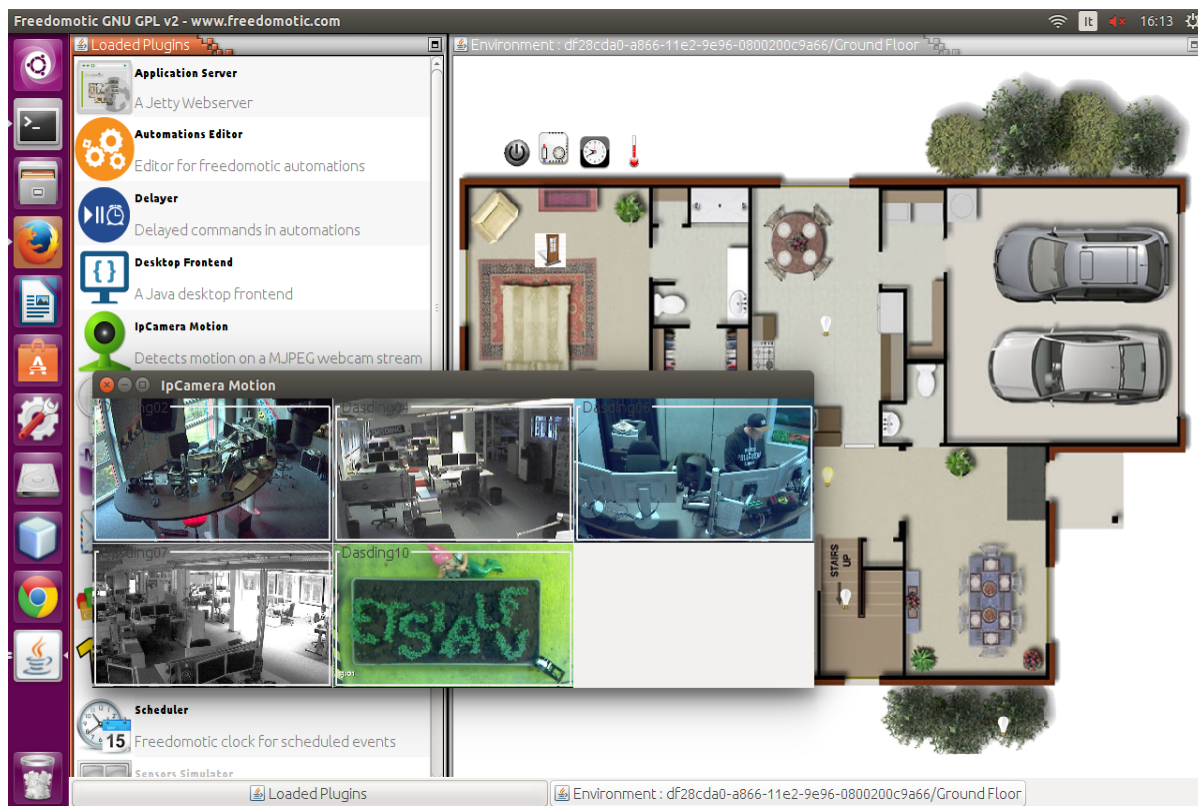


Fig. 38.2: IPCamera Motion Plugin GUI

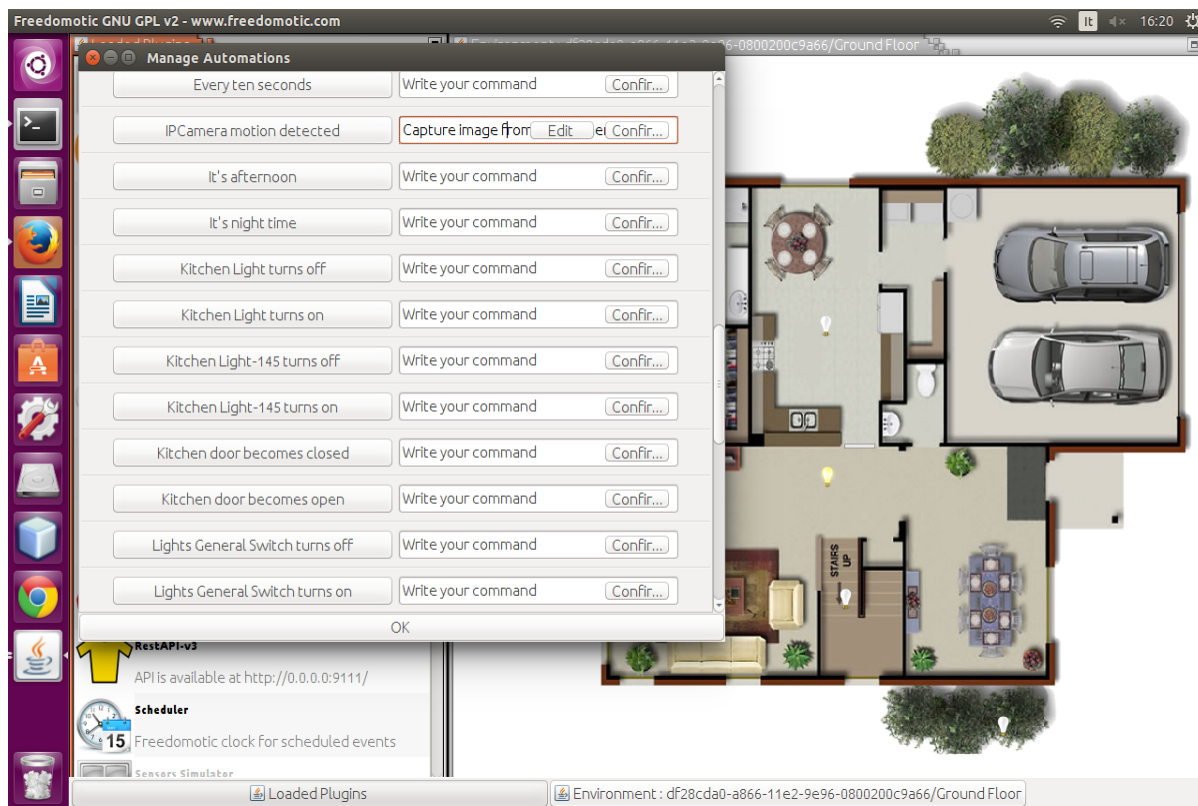


Fig. 38.3: IPCamera Motion Plugin Automations



**Description:** This plugin enables communication between Freedomotic and Ipx800 boards by gce-electronics.com

**Type:** Driver - **Categories:** Automation Protocols

**Development status:** Beta version

**Tested on:** All platforms

**Developer:** Mauro Cicoletta

## 39.1 Overview

## 39.2 Configuration

The *ipx800-manifest.xml* file contains all the plugin configuration parameters. It's possible to control more than one board setting its parameters into a specific `<tuple></tuple>` block.

Here an example of configuration file:

```
<config>
  <properties>
    <property name="description"      value="Communicates with ipx800 board"/>
    <property name="protocol.name"    value="ipx800"/>
    <property name="category"         value="protocol"/>
    <property name="short-name"       value="ipx800"/>
    <property name="address-delimiter" value=":"/>
    <property name="startup-time"     value="on load"/>
    <property name="change-state-relay-url" value="leds.cgi?led="/>
    <property name="send-pulse-relay-url" value="rlyfs.cgi?rlyf="/>
    <property name="get-status-url"   value="status.xml"/>
  </properties>
  <tuples>
    <tuple>
```

```

<property name="alias" value="default"/>
<property name="ip-to-query" value="ip-board"/>
<property name="port-to-query" value="port-board"/>
<property name="relay-number" value="8"/>
<property name="analog-input-number" value="2"/>
<property name="digital-input-number" value="4"/>
<property name="starting-relay" value="0"/>
<property name="led-tag" value="led"/>
<property name="analog-input-tag" value="analog"/>
<property name="digital-input-tag" value="btn"/>
</tuple>
</tuples>
</config>

```

Parameter	Re-quired	Values	Effect	Note
address-delimiter	yes	: or something else	delimiter for address components	
change-state-relay-url	yes	leds.cgi?led=	integrated webserver url for relay changing state	
send-pulse-relay-url	yes	rlyfs.cgi?rlyf=	integrated webserver url for relay sending pulse	
get-status-url	yes	status.xml	integrated webserver url for retrieving state	

For each connected board the parameters are the following

Parameter	Re-quired	Values	Effect	Note
alias	yes	string	identifies the board used in the object address	
ip-to-query	yes	ip-address	sets the board ip address	
port-to-query	yes	any port number e.g. 80	sets the port number	
relay-number	yes		number of relays	
analog-input-number	yes		number of analog inputs	
digital-input-number	yes		number of digital inputs	
starting-relay	yes	0 or 1	starting number in status.xml	
led-tag	yes	led	status.xml tag for compatibility with previous versions	
analog-input-tag	yes	analog or an	status.xml tag for compatibility with previous versions	
digital-input-tag	yes	btn	status.xml tag for compatibility with previous versions	

### 39.3 How to control an object with this board

With this board you can control the powered behavior of any electric device in your environment, meaning you can turn on/off electric devices.

For this example we use a light thing:

- Open the file ipx800.xml into *freedomotic/plugins/devices/ipx800* folder and set ip-address and port number of your board

- Right click on the light object in the environment to show its configuration panel
- Change the property **protocol** to **ipx800**
- Change the property “address” to a string composed of **ALIAS:RELAY\_NUMBER:led** where ALIAS is the string used to identify the board, RELAY\_NUMBER is the relay on which your object is connected (from 1 to 8 the maximum number of supported relays) and led is used to retrieve the relay state from status.xml file. An example address can be “default:1:led” this identifies the first relay on a board listening on port 80 of the http URL 192.168.201.10
- Under **turn on** (in **Actions**) select the command called Turn on Relay on Ipx800 board
- Under **turn off** (in **Actions**) select the command called Turn off Relay on Ipx800 board. Now the plugin is able to read any relay status change and update the object one.
- Under **powered** (in **Data Sources**) select the command called IPX800 board reads a state change

## 39.4 Download

[Download plugin latest version](#)

## 39.5 Source code

[GitHub repository](#)

## CHAPTER 40

---

### KMTronic Usb Relay

---

**Description:** A plugin for usb relay devices by KMtronic

**Type:** Driver - **Categories:** Automation Protocols

**Development status:** Beta version

**Tested on:** All platforms

**Developer:** Mauro Cicoella

### 40.1 Overview

A general purpose USB Relay controller for connection to a PC's USB port using VCP (Virtual COM port). Control devices using your PC. USB Relay controller allows a PC to control a single external device using simple RS232 commands. Relay is fully powered from the USB bus.

The plugin allows to control this device.

### 40.2 Configuration

**Description:** This plugin sends notification by email

**Type:** Driver - **Categories:** Network & Communication, Utilities

**Development status:** Stable version

**Tested on:** All platforms

**Developer:** Enrico Nicoletti

---

**Note:** From 5.6.0 version it's included in the package distribution

---

## 41.1 Overview

## 41.2 Configuration

- You need a Gmail account
- Set your Google username and password in the manifest file or if you are using **Jfrontend** click on menu **Plugins** -> **Configure** -> **Mailer**. Change only 'username' and 'password'.
- Save the changes

## 41.3 Create an automation example

- Right click on a thing
- Switch to **Automations** tab in **Jfrontend**
- In the command field corresponding to the trigger when `OBJECT_NAME` object is clicked write  
Notify this event by mail

- Save changes by pressing **Confirm** and **OK** button.

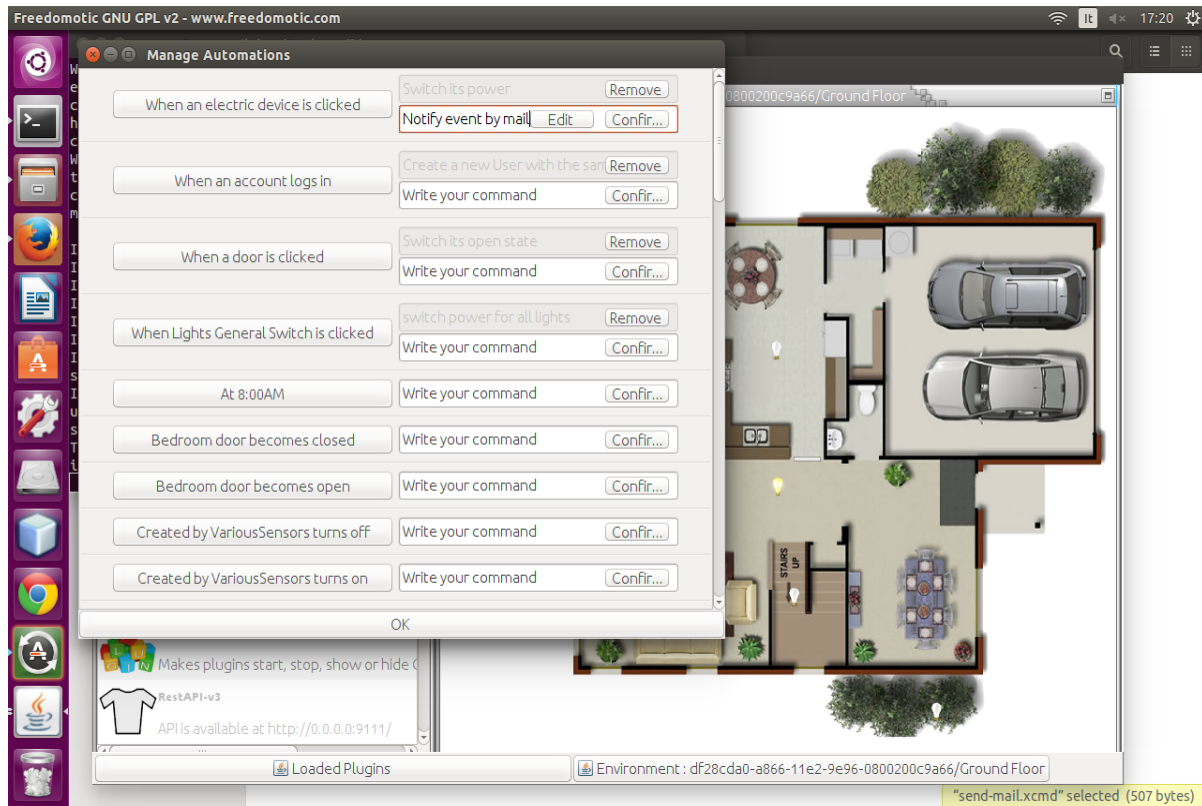


Fig. 41.1: Mailer automations

When you click on this thing you should receive an email with the description of what happened.

Others plugins can use the mailer to send custom messages.

### 41.3.1 Send a mail notification with no attachment

```
<command>
  <name>Notify event by mail</name>
  <receiver>app.actuators.messaging.mail.in</receiver>
  <description>send a mail</description>
  <hardwareLevel>false</hardwareLevel>
  <delay>0</delay>
  <timeout>0</timeout>
  <editable>false</editable>
  <properties>
    <properties>
      <property name="subject" value="A notification from your home"/>
      <property name="message" value="Event notified: thing @current.object.name_
↩clicked"/>
      <property name="attachment" value="no-attachment"/>
    </properties>
  </properties>
</command>
```

### 41.3.2 Send a mail notification with attachment

```
<command>
  <name>Notify event by mail with attachment</name>
  <receiver>app.actuators.messaging.mail.in</receiver>
  <description>send a mail</description>
  <hardwareLevel>false</hardwareLevel>
  <delay>0</delay>
  <timeout>0</timeout>
  <editable>false</editable>
  <properties>
    <properties>
      <property name="subject" value="A notification from your home"/>
      <property name="message" value="Event notified: thing @event.description"/>
      <property name="attachment" value="/home/mauro/Desktop/fd/plugins/devices/
↵mailer/data/cmd/index.txt"/>
    </properties>
  </properties>
</command>
```

## 41.4 Command parameters

Property	Description	Values
name	command name	any string
subject	mail subject	any string
message	mail text	any string
attachment	absolute path of attachment	path string (can't be empty)

## 41.5 Notes

Be aware that your firewall can block the sending of the email. Check the firewall settings.

Also, if needed, go to <https://www.google.com/settings/security/lesssecureapps> and enable “**the less secure apps**” option.

---

## MaryTTS Text to Speech

---

**Description:** Text To Speech plugin based on MaryTTS library

**Type:** - **Categories:** Speech Recognition & TTS

**Development status:** Beta version

**Tested on:** All platforms

**Developer:** Mauro Cicoella

### 42.1 Overview

This plugin can be used to speak any text when an event occurs. It is completely programmable using the automations system (IF this THEN say THAT).

It's based on **MaryTTS** project (<http://mary.dfki.de/project-summary.html>).

### 42.2 Configuration

### 42.3 Automations examples

Text to speech plugin comes with these predefined commands which can be attached to a trigger to create an automation (if you don't know how to do read this chapter of the tutorial):

- say current time
- say current temperature of the thermometer
- say electric device status
- say door/window status
- say readed motion level



which can be attached to any trigger to create **automations** like:

- if an object is clicked -> say electric device status
- a door becomes open -> say door/window status

...and so on

## 42.4 How to create new custom commands

- Right click on an object and switch to **Automations** table
- In command box write `say current time` or any other command starting with “say” so you can use it as a template for yours
- Click the edit button. In the next window change the say property according to your needs (remembered you can use token substitution to insert variables like `@event.object.name`)
- Save as new command

Now your command will be available to be used in automations.

## 42.5 How to install better and natural sounding voices

By default the plugin uses an English female voice. You can customize it by downloading a new voice, copying the jar file into *FREEDOMOTIC\_ROOT/plugins/devices/marytts/lib* folder and modifying the *marytts-manifest.xml* file as reported in the following table.

Table 42.1: International voices

Language	Voice	Link	Config parameters
English	Female	Already installed	<code>&lt;property name='voice-jar-file' value='voice-cmu-slt-hsmm-5.1.1.jar'/&gt; &lt;property name='voice' value='cmu-slt-hsmm'/&gt;</code>
German	Male	<a href="http://mary.dfki.de/download/5.1/voice-bits3-hsmm-5.1.zip">http://mary.dfki.de/download/5.1/voice-bits3-hsmm-5.1.zip</a>	<code>&lt;property name='voice-jar-file' value='voice-bits3-hsmm-5.1.jar'/&gt; &lt;property name='voice' value='bits3-hsmm'/&gt;</code>
Italian	Female	<a href="http://mary.dfki.de/download/5.1/voice-istc-lucia-hsmm-5.1.zip">http://mary.dfki.de/download/5.1/voice-istc-lucia-hsmm-5.1.zip</a>	<code>&lt;property name='voice-jar-file' value='voice-istc-lucia-hsmm-5.1.jar'/&gt; &lt;property name='voice' value='istc-lucia-hsmm'/&gt;</code>

## 42.6 Download

Download plugin latest version

## 42.7 Source code

GitHub repository

**Description:**

**Type: - Categories:**

**Development status:**

**Tested on:** All platforms

**Developer:** Gabriel Pulido de Torres

### 43.1 Overview

This plugin allows Freedomotic to be configured as a Master device in the Modbus network to read values from the slaves of the system.

The read values are then published as events in the system.

NOTE: from 5.6 version TCP and RTU over TCP are supported.

### 43.2 Configuration

The sensor usual parameters should be configured. See the Sensor page. Also the following parameters must be defined in the xml manifest file.

Table 43.1: Generic properties

Parameter	Required	Values	Effect	Note
modbus-protocol	yes	'RTU' or 'TCP'	Configuration for RTU or TCP	
port	yes	'COM1' or '/dev/ttyUSB0'	Name of the Serial port used	Only needed if modbus-protocol is RTU
baudrate	yes		Serial Port Parameter	Only needed if modbus-protocol is RTU
data-bits	yes		Serial Port Parameter	Only needed if modbus-protocol is RTU
parity	yes	0 1 2	Serial Port Parameter	Only needed if modbus-protocol is RTU
stop-bits	yes		Serial Port Parameter	Only needed if modbus-protocol is RTU
host	yes	IP address	TCP host address	Only needed if modbus-protocol is TCP
tcp-port	yes	Port	TCP host port	Only needed if modbus-protocol is TCP
encapsulated	yes		RTU over TCP flag	Only needed if modbus-protocol is TCP and you want to enable RTU over TCP
timeout	yes		Time in milliseconds	Modbus Parameter
retries	yes		Number of retries to obtain a value	Modbus Parameter
Num-Registers	yes	0l..ln	Number of registers that are going to be configured	This value should match the total of tuples defined

For every register that is going to be read from the network a tuple should be configured with the correct parameters to locate and transform from the modbus system to Freedomotic system as one Event. For every tuple, the sensor sends an event on the Freedomotic system.

Tuples properties  
 Parameter Required Values Effect Note  
 Name yes String with the name of this value in the system  
 Must be unique in the sensor SlaveId yes 0l..ln Number of the slave from which the value is read  
 RegisterRange yes See table of Register Ranges allowed  
 DataType yes See table of Data Type allowed  
 Bit no 0l..ln Sets the bit to be read from the Binary register  
 It is only used when the DataType is set to Binary  
 Offset yes 0l..ln Address of the register to be read in the slave  
 NumberOfRegisters no 0l..ln Number of consecutive registers to be read  
 Not used at this moment  
 Multiplier no double value Used with the Additive parameter to transform the value from the modbus scale system to the Freedomotic scale system  
 $Mx+A$  where M= Multiplier x= value readed A= Additive  
 Additive no double value Used with the Additive parameter to transform the value from the modbus scale system to the Freedomotic scale system  
 $Mx+A$  where M= Multiplier x= value readed A= Additive  
 EventName yes String Name of the parameter in the event that is going to be sent with the value  
 This value configures a GenericEvent with a parameter with name="EventName" value="Mx+A"

Register Ranges: Name Note COIL\_STATUS INPUT\_STATUS HOLDING\_REGISTER INPUT\_REGISTER

Data Types Name Note  
 BINARY TWO\_BYTE\_INT\_UNSIGNED TWO\_BYTE\_INT\_SIGNED  
 FOUR\_BYTE\_INT\_UNSIGNED FOUR\_BYTE\_INT\_SIGNED FOUR\_BYTE\_INT\_UNSIGNED\_SWAPPED  
 FOUR\_BYTE\_INT\_SIGNED\_SWAPPED FOUR\_BYTE\_FLOAT FOUR\_BYTE\_FLOAT\_SWAPPED  
 EIGHT\_BYTE\_INT\_UNSIGNED EIGHT\_BYTE\_INT\_SIGNED EIGHT\_BYTE\_INT\_UNSIGNED\_SWAPPED  
 EIGHT\_BYTE\_INT\_SIGNED\_SWAPPED EIGHT\_BYTE\_FLOAT EIGHT\_BYTE\_FLOAT\_SWAPPED  
 TWO\_BYTE\_BCD FOUR\_BYTE\_BCD

### 43.3 An XML example

This xml file configures 2 registers.

The first register reads a temperature value from the slave and, as the Modbus in this example stores the value multiplied by 10, the sensor transforms that value multiplying it by 0.1

The second register reads the bit value from position 6 from the register.

```
<config>
  <properties>
    <property name="description" value="Plugin for Modbus protocol"/>
    <property name="category" value="protocol"/>
    <property name="short-name" value="modbus"/>
    <property name="polling-time" value="1000"/>
    <property name="NumRegisters" value="2"/>
    <property name="port" value="/dev/ttyUSB10"/>
    <property name="baudrate" value="19200"/>
    <property name="data-bits" value="8"/>
    <property name="parity" value="2"/>
    <property name="stop-bits" value="1"/>
    <property name="timeout" value="10000"/>
    <property name="retries" value="2"/>
  </properties>
  <tuples>
    <tuple>
      <property name="Name" value="TemperatureZone1"/>
      <property name="SlaveId" value="1"/>
      <property name="RegisterRange" value="HOLDING_REGISTER"/>
      <property name="DataType" value="TWO_BYTE_INT_UNSIGNED"/>
      <property name="Offset" value="266"/>
      <property name="NumberOfRegisters" value="1"/>
      <property name="Multiplier" value="0.1d"/>
      <property name="Additive" value="0.0d"/>
      <property name="EventName" value="TemperatureZone1"/>
    </tuple>
    <tuple>
      <property name="Name" value="BitTest"/>
      <property name="SlaveId" value="1"/>
      <property name="RegisterRange" value="HOLDING_REGISTER"/>
      <property name="DataType" value="BINARY"/>
      <property name="Bit" value="6"/>
      <property name="Multiplier" value="1"/>
      <property name="Additive" value="0"/>
      <property name="Offset" value="256"/>
      <property name="NumberOfRegisters" value="1"/>
      <property name="EventName" value="BitTest"/>
    </tuple>
  </tuples>
</config>
```

**Description:** A broker for MQTT protocol based on Moquette library

**Type:** Driver - **Categories:** Automation Protocols, IoT

**Development status:** Prototype

**Tested on:** All platforms

**Developer:** Mauro Cicoella

### 44.1 Overview

This plugin adds a MQTT broker to Freedomotic so you can manage this protocol without using an external broker. It uses an embedded instance of [moquette](#), a Java small MQTT broker implementation.

### 44.2 Configuration

By default it uses 0.0.0.0 as server ip and 1883 as port number.

If you want to customize them please edit the previous properties in *mqtt-broker-manifest.xml* file located in the plugin folder.

### 44.3 How to manage a thing

- In **Jfrontend** add a thing on the map. Right click on its icon, go to **Control Panel** and set **mqtt-broker** in the protocol field and the mqtt topic in the **address** field.
- In **Data Source** panel select **MQTT reads a value** in the trigger list.

## 44.4 Multiple behaviors

## 44.5 Video

## 44.6 Download

[Download plugin latest version](#)

## 44.7 Source code

[GitHub repository](#)

**Description:** A client for MQTT (MQ Telemetry Transport). It can be consider as a template for plugins based on this protocol

**Type:** Driver - **Categories:** Automation Protocols, IoT

**Development status:** Prototype

**Tested on:** All platforms

**Developer:** Mauro Cicoella

## 45.1 Overview

This plugin is a client for MQTT protocol.

## 45.2 Configuration

You can customize the plugin according to your needs.

Follow these steps:

1. open mqtt-client-manifest.xml and change at least the properties
  - (a) **broker-url:** ip address of MQTT broker
  - (b) **topic(s):** the topic(s) to subscribe in the <tuples></tuples> section
  - (c) other properties can be modified if needed (for example authentication with userid and password)

## 45.3 Download

Download [plugin latest version](#)

## 45.4 Source code

GitHub repository



**Description:** This plugin supports MySensors gateways

**Type:** Driver - **Categories:** Automation Protocols, IoT

**Development status:** Prototype

**Tested on:** All platforms

**Developer:** Mauro Cicoella

## 46.1 Overview

## 46.2 Configuration

## 46.3 Supported objects

- Light
- Thermostat
- Light sensor

## CHAPTER 47

---

### openPicus Flyport

---

**Description:**

**Type: - Categories:**

**Development status:**

**Tested on:** All platforms

**Developer:** Mauro Cicoella

### 47.1 Overview

### 47.2 Configuration

# CHAPTER 48

---

## openPicus Grove System

---

**Description:** This plugin reads data from a Grove Nest sensors board

**Type:** Driver - **Categories:** Automation Protocols

**Development status:** Prototype

**Tested on:** All platforms

**Developer:** Mauro Cicoletta

### 48.1 Overview

The plugin acts as a simple udp server listening for packets from Grove Nest boards. When it receives a packet, it extracts the data (Grove Nest ip address, sensor connector, sensor value). Then it notifies the event on the Freedomotic messages channel and updated the specific object.

### 48.2 Configuration

The plugin acts as a simple udp server listening for packets from Grove Nest boards. When it receives a packet, it extracts the data (Grove Nest ip address, sensor connector, sensor value). Then it notifies the event on the Freedomotic messages channel and updated the specific object.

The firmware code is included in the taskFlyport.c. Let's analyze it step by step.

First of all the libraries required for Grove Nest and analog temperature sensor are included.

```
#include "taskFlyport.h"  
#include "grovelib.h"  
#include "analog_temp.h"
```

Now let's create an integer variable for udp socket reference, a char array for sending messages and udp server ip address and port number. The last parameters are used by Freedomotic plugin. If you change these you must change also the flyport variables.

```
int clientUDPsocket;
char send[120];
void FlyportTask()
{
    BYTE Socket=0;    //UDP Socket
    char *UDPPort="7331"; //UDP port
    char *UDPAddress="192.168.1.100";
```

The next step is defining variables for grove nest and used sensors and attaching them to the board. The float variables are used to store the new and the old sensors values.

The application core is included into the infinite loop. It detects the new sensor value, compares it with the stored one and if there is a change sends a new udp packet to the Freedomotic server with a prefixed message format: connector:sensorType:value. The char ":" is the string delimiter. The allowed sensor types are (in this example) temperature and luminosity. You can't change them without changing the Freedomotic plugin code because it makes a string parsing to extract the data. After the socket is closed.

```
while(1)
{
    vTaskDelay(500);
    //Socket=UDPCClientOpen(UDPAddress,UDPPort);
    // Get the new temperature value
    tempVal = get(temp_sensor);
    if(tempVal!=tempValStored) {
        tempValStored=tempVal;
        sprintf(send,"AN1:temperature:%3.1f", (double)tempVal);
        Socket=UDPCClientOpen(UDPAddress,UDPPort);
        UDPWrite(Socket,send,strlen(send)); //Send the data through the UDP
        UDPCClientClose(Socket);
    }
    // Get the new luminosity value using the get() function
    lightVal = get(light_sensor);
    if(lightVal!=lightValStored) {
        lightValStored=lightVal;
        sprintf(send,"AN3:luminosity:%3.0f", (double)lightVal);
        Socket=UDPCClientOpen(UDPAddress,UDPPort);
        UDPWrite(Socket,send,strlen(send)); //Send the data through the UDP
        UDPCClientClose(Socket);
    }
}
}
```

No packets are sent if there is no sensor data change. This reduces the network traffic.

## 48.3 The list of materials

- A Flyport\_WiFi
- A Grove\_NEST
- A **GROVE**\_Light\_Sensor
- A **GROVE**\_Temperature\_Sensor\_Analog

## 48.4 The build process

### Software

- Download the Grove System project package and extract it into a new folder.
- Open the openPicus IDE, and from the environment open the project folder.
- Click the “Compile” button, and when the operation is completed click “Download” so that the new firmware is downloaded in the Flyport’s microcontroller.

### Hardware

Let’s connect the analog sensors to the Grove Nest: temperature sensor to AN1 and light sensor to AN3.

Then let’s connect the flyport wifi module and the power supply.

Start Freedomotic by double click on its icon or by command line with `java -jar freedomotic.jar`. The pc running Freedomotic must have 192.168.1.100 as ip address. In “Plugins” menu choose “Install from marketplace”. After the list is updated (it can take up to a minute) you will see all available plugin categories for your current Freedomotic version. Select Automation Protocols, then Openpicus Grove System icon and click on Install button following the video instructions.

## 48.5 Video

**Description:**

**Type: - Categories:**

**Development status:**

**Tested on:** All platforms

**Developer:** Mauro Cicoella

### 49.1 Overview

### 49.2 Configuration

**Description:** Save events and commands on a db Cassandra

**Type:** Driver - **Categories:** Utilities

**Development status:** Proof of Concept

**Tested on:** All platforms

**Developer:** P3trur0

## 50.1 Overview

This plugin allows to persist on Cassandra server both `commands` and `events` occurred on Freedomotic instances. Using this plugin, for each automation available on the platform, the user can run a **Persist data on Cassandra** command that will perform the operation to save data on the database.

Once started, this plugin tries to connect to the Cassandra server defined in its configuration properties (see below). All the data are saved in a table named `freedomotic_data`. It is not required to create it manually since this plugin, once started, automatically detects the existence of `freedomotic_data` table creating it if does not exist.

This table contains the following information:

- `id`, it is the identifier of each persisted row
- `datatype`, it is either “command” or “event”
- `data`, it is a binary serialization of persisted event/command
- `avro_schema`, it is the Avro schema used for serialize/deserialize the persisted data
- `persistence_timestamp`, it is the persistence timestamp
- `freedomoticInstance`, it is the identifier of the currently used Freedomotic instance

These data could be later used for any kind of data analysis you would like to perform on Freedomotic usage. Basically it is required to deserialize the binary serialization data using the Avro schema provided to analyze the original data values properly.

## 50.2 Configuration

Within the plugin manifest, are defined all the properties available to configure the communication of the plugin with Cassandra instance. Moreover, you can also specify both the replication factor and strategy of the server instance.

Here follows an example of manifest.

```
<config>
  <properties>
    <property name="description"
      value="Cassandra database for storing Freedomotic events_
↳and commands" />
    <property name="name" value="persistence" />
    <property name="category" value="protocol" />
    <property name="short-name" value="persistence" />
    <property name="protocol.name" value="persistence" />
    <property name="startup-time" value="on load" />
    <!-- Cassandra configuration -->

    <property name="cassandra.host" value='127.0.0.1' />
    <property name="cassandra.port" value='7000' />
    <property name="cassandra.keyspace" value='freedomotic' />
    <property name="cassandra.replicationFactor" value='1' />
    <property name="cassandra.strategy" value='SimpleStrategy' />
    <property name="cassandra.user" value='user' />
    <property name="cassandra.password" value='password' />

    <!-- end of cassandra configuration -->
  </properties>
</config>
```

The properties for the Cassandra instance configuration are:

- **cassandra.host**, it represents the Cassandra IP address
- **cassandra.port**, it represents the Cassandra Port
- **cassandra.keyspace**, it is the name of Cassandra keyspace used by freedomotic
- **cassandra.replicationFactor**, it is the replication factor of the data
- **cassandra.strategy**, it is the Cassandra replication strategy
- **cassandra.user**, Cassandra username
- **cassandra.password**, Cassandra password

## 50.3 Requirements

**Persistence** has been tested using Cassandra server (ver. 3.9).



---

## ProgettiHw-Sw Ethernet Board v2

---

**Description:** Communicates with an ethernet relay board by ProgettiHw-Sw.

**Type:** Driver - **Categories:** Automation Protocols

**Development status:** Stable Release

**Tested on:** All platforms

**Developer:** Mauro Cicoletta

### 51.1 Overview

With this plugin Freedomotic can communicate with an ethernet relay board by ProgettiHw-Sw. These boards have from 8 up to 16 relays, an ethernet port and an integrated web server

### 51.2 Configuration

With this type of board you can control the powered behavior of any electric device in your environment, meaning you can turn on/off electric devices.

### 51.3 How to control a relay

- Right click on the light object in the environment to show its configuration panel
- Change the property **protocol** to **phwswethv2**
- Change the property **address** to a string composed of **ALIAS:RELAY\_NUMBER:TAG** where **ALIAS** is the string set in the configuration file (phwswethv2-manifest.xml) to identify the board. **RELAY\_NUMBER** is the relay used to control your device (from 1 to 16). **TAG** is one of the following string: **led** (for relays), **pot** (for

analog inputs), **btn** (for digital inputs). For example **default:1:led** identifies the first relay on a board called **default**.

- Under **turn on** (in **Actions**) select the command called **Turn ON Relay on ProgettiHwSw Eth board**
- Under **turn off** (in **Actions**) select the command called **Turn OFF Relay on ProgettiHwSw Eth board**

The plugin is able to read any relay status change and update the object one.

- Under **powered** (in **Data Sources**) select the trigger called **ProgettiHwSwEthV2 reads a state change**.

## 51.4 Use cases

- <https://www.emmecilab.net/gestione-di-un-appartamento-con-freedomotic/> (in Italian)

## 51.5 Download

[Download plugin latest version](#)

## 51.6 Source code

[GitHub repository](#)

**Description:** This reads URLs content like XML, HTML, JSON and sends it in a listenable event

**Type:** Driver - **Categories:** Network & Communication, Utilities

**Development status:** Prototype

**Tested on:** All platforms

**Developer:** Enrico Nicoletti

## 52.1 Overview

This plugin takes a set of urls (web services or standard web pages) from its manifest file, periodically gets their content (XML, JSON, HTML) and notifies it as a Freedomotic event. This event can be listened by the triggers shipped with this plugin (read temperature from URL, read weather from URL, ...) or you can create your own.

The triggers can be used as a data source for objects, like setting the temperature of a thermostat, or to create high level automations like

WHEN <your trigger here> THEN <say something using text to speech> WHEN <your trigger here> THEN <send me an email> WHEN <your trigger here> THEN <send a twitt> WHEN <your trigger here> THEN <turn on a light> let us know about your own triggers

Q: What about the plugin name?

A: The name is inspired by cURL command line linux program. <http://en.wikipedia.org/wiki/CURL>

Q: Where I can find example triggers?

A: As usual, they are in the data/trg folder in the plugin installation directory (freedomotic/plugins/devices/pURL/data/trg)

Q: How to reference the result of the XPATH query performed in the trigger?

A: The xpath query result is stored in property @event.url.content.xpathresult that you can use in commands. For example if you are parsing an XML content using an XPATH query to retrieve weather forecast, you can have a text to speech command like “say= Tomorrow will be @event.url.content.xpathresult” to hear “Tomorrow will be sunny”

## 52.2 Configuration

Actually you can just monitor a single URL changing it in the plugin manifest (menu Plugins -> Configure to edit). Detailed explanation coming when the plugin will be completed.

## 52.3 How to create custom triggers

Create a trigger (hardware level or not) which listen to messaging channel “app.event.sensor.url”. This plugins notify the following properties in the event:

url: the source of data (eg: <http://www.website.com/?q=aQuery>) url.content: the XML, JSON or HTML content of the web page url.content.length: the number of characters readed from the url (always more than zero otherwise the event is not sent)

## 52.4 Upcoming features

multi url monitoring custom polling interval for each url username/password authentication for protected urls XPATH feature available in freedomotic-core to do xml queries directly in your trigger (Freedomotic v5.6)

---

### Push Notifications

---

**Description:** This plugin can send custom push messages through many providers everytime you want

**Type:** Driver - **Categories:** Network & Communication, Utilities

**Development status:** Beta version

**Tested on:** All platforms

**Developer:** Matteo Mazzoni

### 53.1 Overview

### 53.2 Features

- Global and per-user notification params
- Global notification: send a common notification through a globally configured provider
- Per-user-state notification: notify user using a different provider given his/her state
- Group notification: notify multiple users at once (with per-user-state rule)
- Customizable message text with variable substitution

### 53.3 Supported providers

This plugin could support almost every provider that allows sending notifications with GET or POST http requests.

Currently the plugin ships with predefined configuration blocks for the following providers:

- Prowl - [www.prowlapp.com](http://www.prowlapp.com)
- PushOver - [www.pushover.net](http://www.pushover.net)

- Notify My Android - [www.notifymyandroid.com](http://www.notifymyandroid.com)
- LiveNotifier - [www.livenotifier.net](http://www.livenotifier.net)
- Trendoo (SMS Provider) - [www.trendoo.it](http://www.trendoo.it)
- Pushsafer - [www.pushsafer.com](http://www.pushsafer.com)

If you need to use a different provider, please ask me to add proper configuration, or, if you feel brave, have a look at your provider's API and write a configuration block in the plugin manifest file.

## 53.4 Configuration

First of all, register on the provider and request needed api keys and/or application keys.

Go to plugin manifest file (or open plugin configuration from Freedomotic Frontend), find the proper configuration block and add missing apikey params. In the end, activate the provider changing the `<property name="active" value="TRUE" />`.

Templating When customizing message string, you can insert variable placeholders that will be substituted with event-related data everytime a notification is sent. Look at the first configuration block for hints.

```
<property name="param.message" value="Looks like ${event.object.name} power state changed to ${event.object.behavior.powered}" />
```

Usage Global notification The easiest way of using Push Notification is creating a Reaction, and linking a Trigger with the predefined Command "Send push notification". Please note, such Command has to be modified in order to use your desired provider and/or customize params.

This usage allows definition of a common way of notifying many events.

Per user notification Want to provide different notification scenarios for every user? e.g. notify Tom through SMS only when he's not at home.

What to do:

As for global notification, define a reaction, but use the Command "Notify users". e.g. "When a electric device is clicked, notify users" Add a Person object to the environment, set his name to Tom (or whatever name) Close Freedomotic, find Tom's object file (under `/data/furn/YOUR_ENVIRONMENT/obj/`), open it with an editor and find the block

```
<com.freedomotic.model.object.PropertiesBehavior>
  <name>properties</name>
  <description></description>
  <active>true</active>
  <priority>-1</priority>
  <readOnly>false</readOnly>
  <properties>
    <property name="birthdate" value="01/01/1900"/>
    <property name="email" value="info@freedomotic.com"/>
  </properties>
</com.freedomotic.model.object.PropertiesBehavior>
```

If not present, add a line `<property name="mobile" value="" />` after `<property name="email" .... />` and set its value with Tom's mobile phone number (international format) Save and close

Next time you run Freedomotic, Tom's ready to receive SMS notifications!

## 53.5 Download

[Download plugin latest version](#)

## 53.6 Source code

[GitHub repository](#)

**Description:** RestAPI v3 for managing your Freedomotic instance

**Type:** - **Categories:** API, Frontends, Networking & Communication

**Development status:** Beta version

**Tested on:** All platforms

**Developer:** Matteo Mazzoni

---

**Note:** This plugin is included in the package distribution

---

## 54.1 Overview

## 54.2 Basic configuration

Useful configuration parameters you can insert into your plugin manifest:

- **listen-address** [0.0.0.0] : the address to listen to (0.0.0.0 means every address)
- **enable-ssl** [true] : enables support for Secure Socket Layer
- **enable-cors** [true] : enables support for Cross Origin Resource Sharing

## 54.3 Advanced configuration

### Ports

- **http-port** [9111] : port for plain http
- **https-port** [9113] : port for secure http



### Debugging options

- **debug** [**false**] : enables debug information in console and in http headers
- **debug-entity** [**true**] : when debugging, traces message payload too

### SSL options

- **KEYSTORE\_SERVER\_FILE** [**keystore\_server**] : path (relative to plugin data directory) of keystore file that contains server certificates
- **KEYSTORE\_SERVER\_PWD** [**freedomotic**] : password for certificate keystore

### Cross Origin Resource Sharing options (change them very carefully!)

- **Access-Control-Allow-Headers** [**Accept, Accept-Version, Authorization, Content-Length, Content-MD5, Content-Type, Date, Origin, X-Access-Token, X-API-Version, X-CSRF-Token, X-File-Name, X-Requested-With**] : lists allowed headers, when accessing API through XHR
- **Access-Control-Allow-Origin** [**\***] : lists allowed origins for accessing API through XHR (“\*” means every origin)
- **Access-Control-Allow-Methods** [**GET, POST, PUT, DELETE, HEAD**] : lists allowed HTTP methods, when accessing API through XHR

### Extra options

- **serve-static** [**swagger**] : RestAPI plugin can serve static files in its root path. The default choice is serving SWAGGER, a API doc and tooling system

## 54.4 Usage (Documentation and tools)

You can learn the API and test it, simply accessing this URL: `http(s)://{listen_address}:{port}/`

e.g. if using default plugin configuration, go to <https://localhost:9111/>

WebSocket endpoints documentation

- **/v3/ws/objectchange**

When subscribed to this channel, clients will get an updated thing (aka **EnvObject**) everytime any of them is modified. This is handy to keep a client updated without polling data.

- **/v3/ws/zonechange**

When subscribed to this channel, clients will get a updated Zone when its boundaries are changed

- **/v3/ws/pluginchange**

When subscribed to this channel, clients will get a updated Plugin data when it is started/stopped or its configuration changes.

## 54.5 Try API with cURL

You can test API using the commandline tool cURL

Command format is

```
curl --user <username>:<password> http://<listen_address>:<port>/v3/<API>
```

For example if you want to retrieve the users' list on our online demo authenticated as admin please digit

```
curl --user admin:admin http://api.freedomotic.com/9111/v3/users
```

---

### Sensors and tracking simulation

---

**Description:** A must have set of basic plugins like a clock to enable timed automations, automations editing and a set of sensors and tracking simulators, performance trackers, log viewer for both developers and users

**Type:** Driver - **Categories:** Utilities

**Development status:** Stable Release

**Tested on:** All platforms

**Developer:** Enrico Nicoletti

---

**Note:** This plugin is included in the package distribution

---

#### 55.1 Sensors simulator

With this plugin you can

- simulate luminosity or temperature changing
- ask user something
- add a new object using JoinDevice command
- execute free-text commands (e.g. write “turn on all lights”)

#### 55.2 Tracking simulation

You can simulate user’s movements inside your environment. First of all add an “user” thing to the map.

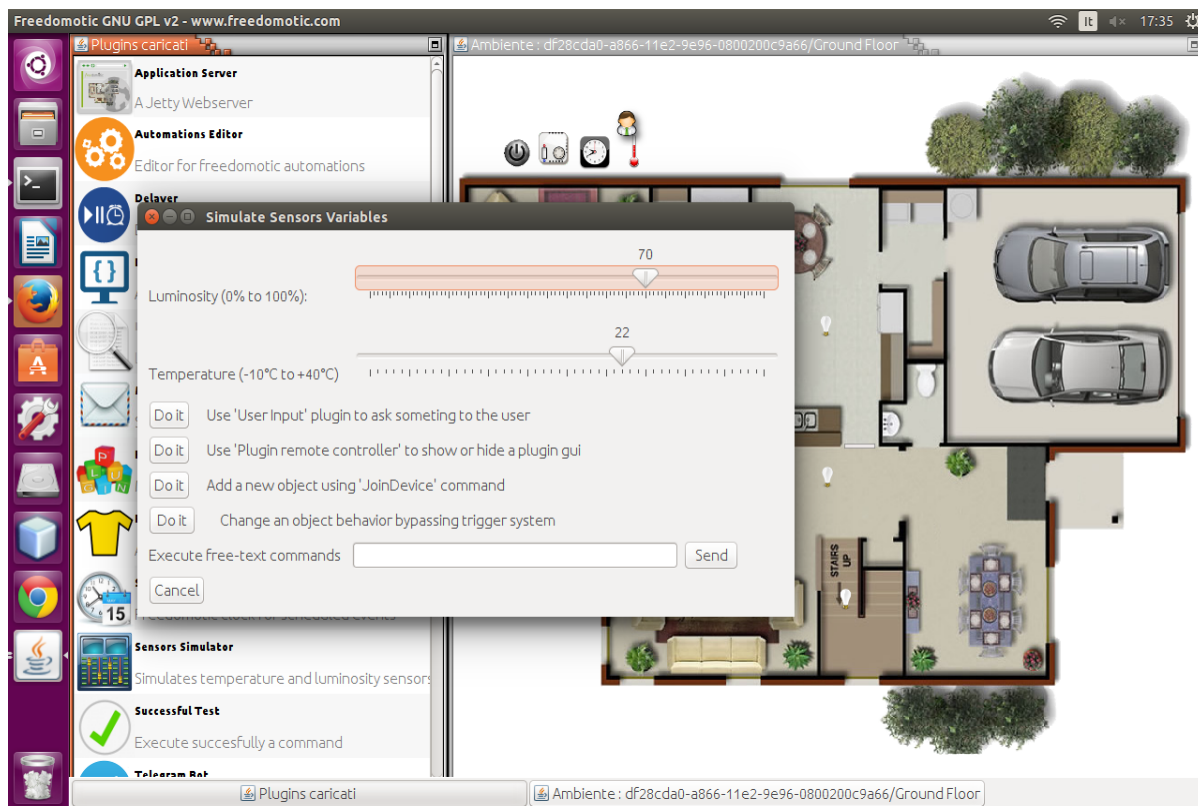


Fig. 55.1: Sensors simulation

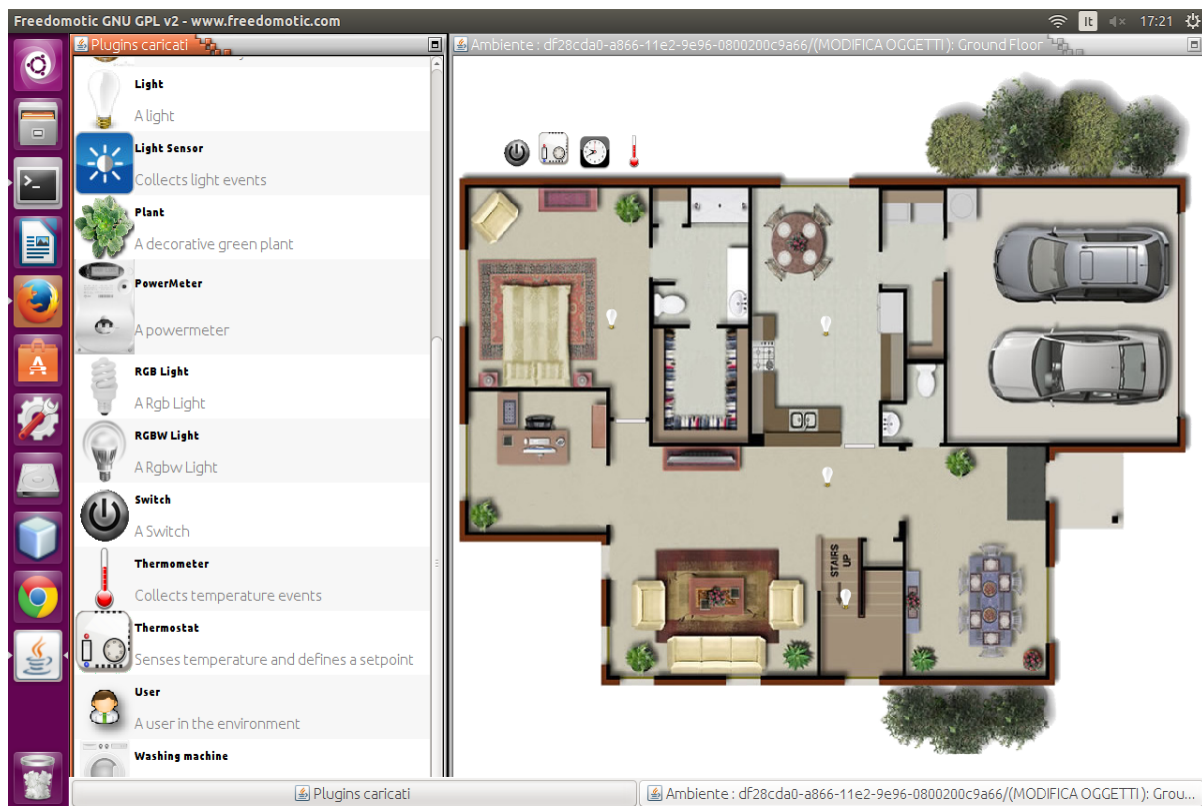


Fig. 55.2: Add an user to the map

### 55.2.1 Tracking simulator (Random)

Each user thing on the map is moved to a random position with a time interval specified by the `sleep-time` property.

Table 55.1: Configuration parameters

Property	Description	Values
<code>sleep-time</code>		2000

### 55.2.2 Tracking simulator (Read file)

This plugin reads user's positions from a file.

This file, located under `FREEDOMOTIC_ROOT/plugins/devices/simulation/data/motes`, has `.mote` extension and its name must reflect the user thing one (e.g. if the user is named *admin* the tracking file must be *admin.mote*).

We have two options for the positions: **coordinates** or **room/zone** names.

In the first case the `.mote` file has the following data format [x coord, y coord, time in ms].

```
300,250,2000
450,390,5000
```

Alternatively you can use the format [room name, time in ms].

```
Kitchen,2000
Bedroom,5000
House,3000
```

In every case each row represents a different movement.

In the previous example the user thing is moved to the Kitchen where it stays for 2 seconds then goes to the Bedroom and after 5 seconds to the House. The last zone House doesn't exist so it's skipped.

Table 55.2: Configuration parameters

Property	Description	Values
<code>data-type</code>	data format in the <code>.mote</code> file	coordinates   rooms
<code>iterations</code>	how many times the movements sequence must be executed	any integer > 0

### 55.2.3 Tracking simulator (Read socket)

Table 55.3: Configuration parameters

Property	Description	Values
<code>socket-server-port</code>		7777
<code>sleep-time</code>		1000
<code>max-connections</code>		any integer > 0 or -1 no limits
<code>stop-connection-char</code>		.

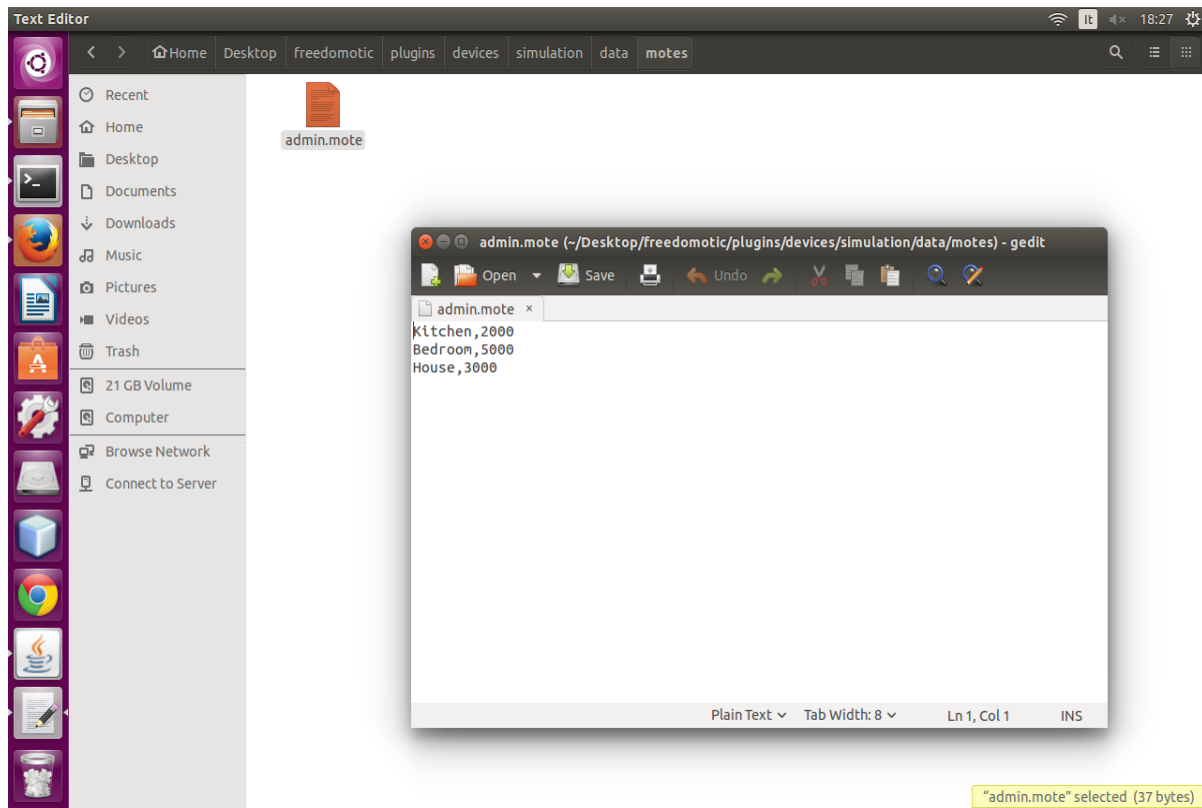


Fig. 55.3: Mote file example (room/zone format)

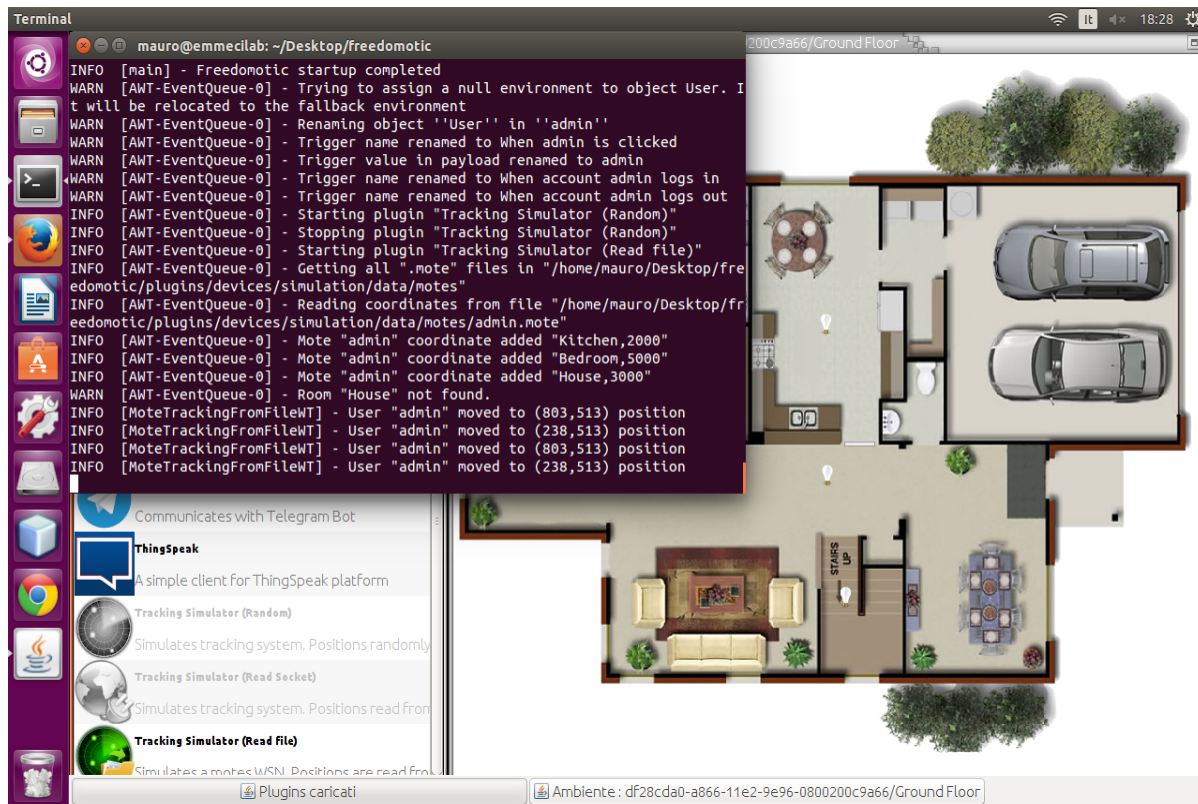


Fig. 55.4: Tracking log



---

### Teracom TCW122B-CM

---

**Description:** This plugin controls a TCW122B-CM module by Teracom.cc.

**Type: - Categories:** Automation Protocols, HVAC

**Development status:** Prototype

**Tested on:** All platforms

**Developer:** Mauro Cicoella

### 56.1 Overview

TCW122B-CM is a multifunctional device for remote monitoring and control, designed to work in IP based networks. It can be managed by WEB interface, SNMP programs and user applications. The controller can be used as standalone device or as a part of control and monitoring systems. The device utilizes special schematic for long 1-Wire interface support, up to 2 temperature or temperature/humidity sensors can be connected. Along with ambient parameters, two analog voltages and status of two digital inputs can be monitored. For every parameter can be sent e-mail and/or SNMP trap, if it goes outside of previously set-up range. The both relays can be activated locally - from status of monitored parameters or remotely - by WEB interface, SNMP managers etc.

### 56.2 Configuration

**Description:** This plugin interacts with a Telegram Bot

**Type:** Driver - **Categories:** Access Control & Security, Social

**Development status:** Prototype

**Tested on:** All platforms

**Developer:** Mauro Cicoletta

## 57.1 Overview

## 57.2 Configuration

### 57.2.1 Create a bot

You have to talk to another Bot, called **The BotFather**. You can start a conversation with the BotFather using this [link](#).

Once you've followed the prompt to start interacting with the bot, send the `/newbot` command to begin the interactive bot-creation process.

First of all the BotFather asks a name for your bot. This is the friendly name displayed when the bot speaks to you. You can choose anything you like.

Then you need to pick a username. Telegram forces all bots to have a username that ends with 'bot'. This is how you'll add the bot to Telegram to begin interacting with it. You might, for example, name your bot "MyName's Bot" and select "MyNameBot" as username.

Once you've picked a name and a username, the BotFather will reply with your bot's token.

That's all! Your bot is ready to work.

## 57.2.2 Start a chat and find its ID

To send a message through the Telegram bot API, you have to provide the ID of the chat with your bot.

So start a conversation with your bot with `/start`. Then open in your browser the url <https://api.telegram.org/bot\protect\T1\textdollarTOKEN/getUpdates> where **\$TOKEN** is the key provided by the BotFather in previous step.

You'd see a JSON response like the following (if needed go to page source in your browser):

```
{
  "ok": true,
  "result": [
    {
      "update_id": 123123123,
      "message": {
        "message_id": 12,
        "from": {
          "id": 12345,
          "first_name": "Bob",
          "last_name": "Jones",
          "username": "bjones"
        },
        "chat": {
          "id": 12345,
          "first_name": "Bob",
          "last_name": "Jones",
          "username": "bjones",
          "type": "private"
        },
        "date": 1452933785,
        "text": "Hi there, bot!"
      }
    }
  ]
}
```

The chatID is 12345 in this example.

## 57.2.3 Plugin configuration

Open the *telegram-bot-manifest.xml* and set your bot's **token**, **username** and **chatID**.

Then start the plugin.

## 57.3 Create an automation example

- Right click on an object
- Switch to Automations tab
- In the field when OBJECT\_NAME object is clicked write this command Notify event by Telegram
- Save changes
- When you click on this object you should receive a notification on your Telegram client with the following predefined message "Event notified: @event.description"

Soon we'll add more examples and info about messages customization.

---

**Note:** The plugin is not completed. If you want to follow the development and contribute or submit your comments please go to <https://github.com/freedomotic/freedomotic/issues/224>

---

## 57.4 Download

[Download plugin latest version](#)

## 57.5 Source code

[GitHub repository](#)

**Description:** A plugin to publish data on ThingSpeak.com platform

**Type:** Driver - **Categories:** Utilities

**Development status:** Beta version

**Tested on:** All platforms

**Developer:** Mauro Cicoletta

## 58.1 Overview

This plugin allows to publish data on [ThingSpeak.com](https://thingspeak.com) platform so you can persist things data to create charts and make custom analysis.

## 58.2 Configuration

Open the file `thingspeak-manifest.xml` and set your ThingSpeak API key in `<property name="api-key" value="api-key-to-set"/>`.

For each thing you want to persist add a `<tuple></tuple>` block as the following

```
<tuple>
  <property name="thing-name" value="Thermometer"/>
  <property name="thing-behavior" value="temperature"/>
  <property name="thingspeak-channel" value="XXXXXX"/>
  <property name="thingspeak-field" value="1"/>
</tuple>
```

Table 58.1: Configuration parameters

Parameter	Description	Values
thing-name	Freedomotic thing name	e.g. Kitchen Thermometer
thing-behavior	the thing behavior to persist	as in the ‘Things behaviors’ table
thingspeak-channel	ThingSpeak channel	the channel ID
thingspeak-field	ThingSpeak channel field	1-8

Table 58.2: Things behaviors

Thing class	Behavior
Barometer	pressure
Hygrometer	humidity
Light sensor	luminosity
Thermometer	temperature
Thermostat	temperature

---

**Note:** Of course you can specify any valid behavior of your things. Please check it is correct and compatible with ThingSpeak field values.

---

**Description:** A TTS plugin to convert text to sound based on the eSpeak Speech Synthesizer

**Type:** Driver - **Categories:** Speech Recognition & TTS

**Development status:** Beta version

**Tested on:** All platforms

**Developer:** Enrico Nicoletti

## 59.1 Overview

This plugin can be used to speak any text after an event occurs. It is completely programmable using the automations system (if this then say that).

## 59.2 Configuration

## 59.3 Automations examples

Text to speech plugin comes with these predefined **commands** as

- say current time
- say current temperature of the thermometer
- say electric device status
- say door/window status
- say readed motion level

which can be attached to any trigger to creates **automations** like:

- if an object is clicked-> say electric device status
  - a door becomes open-> say door/window status
- ... and so on

## 59.4 How to create new custom commands

- Right click on an object and switch to **Automations tab**
- In command box write `say current time` or any other say something command, so you can use it as a template for yours
- Click the edit button. In the next window change the say property according to your needs (remembered you can use token substitution to insert variables like `@event.object.name`)
- Save as new command

Now your command will be available to be used in automations.

## 59.5 How to install better, natural sounding voices

Default voices have a very metallic sound. To have more natural sounding voices

- go to the mbrola project website and download the **mbrola** executable for your operative system (under “getting the mbrola binary”). For Linux download from [here](#). For Windows download from [here](#)
- unzip the downloaded archive
- rename the mbrola-SOMETHING executable in “mbrola” (for Linux pc the file is named mbrola-linux-i386)
- put the renamed executable (or mbrola.exe for Windows) into *YOUR\_FREEDOMOTIC\_FOLDER/plugins/devices/freetts/data/voices*
- put the voice folders (e.g. us1, us2, etc) into *YOUR\_FREEDOMOTIC\_FOLDER/plugins/devices/freetts/data/voices*
- open the plugin manifest file from **Plugins -> Configure** and set the `<mbrola-voice>` key to the voice you want to use (e.g. mbrola\_us1, mbrola\_us2 etc)
- start Freedomotic

---

**Note:** Not compatible with openjdk - Oracle jdk required

---

## 59.6 Video



---

## Twilight - Sunset and sunrise alerts

---

**Description:** This plugin sends events related to Sunrise and Sunset time for the configured latitude/longitude.

**Type: - Categories:** Utilities, Weather

**Development status:** Beta version

**Tested on:** All platforms

**Developer:** Matteo Mazzoni

### 60.1 Overview

This plugin sends events related to Sunrise and Sunset time for the configured latitude/longitude. It's based on OpenWeatherMap and EarthTools providers.

### 60.2 Configuration

In the manifest file you have to set the following properties:

Table 60.1: Parameters

Parameter	Meaning
lat	latitude
log	longitude

By default the plugin uses OpenWeatherMap provider. If you want to use EarthTools you have to set it in `<property name="provider" value="earthtools"/>`.

Events are populated with the following properties

Table 60.2: Event properties

Property	Meaning	Values
isSunrise	sunrise happened a few seconds ago	true or false
isSunset	sunset happened a few seconds ago	true or false
beforeSunset	minutes before the sun sets	integer
afterSunset	minutes after the sun sets	integer
beforeSunrise	minutes before the sun rises	integer
afterSunrise	minutes after the sun rose	integer
Download		

## 60.3 Source code

[GitHub repository](#)

**Description:** Makes your home post messages on Twitter social network

**Type:** - **Categories:** Social

**Development status:** Stable release

**Tested on:** All platforms

**Developer:** Gabriel Pulido de Torres

## 61.1 Overview

This plugin allows Freedomotic to publish messages, as status updates, on Twitter. A Twitter account is required to be configured to allow an external application status post messages.

## 61.2 Configuration

This plugin post messages on a twitter account so it needs to know the OAuth parameters to connect to your twitter account.

The following parameters have to be defined in the xml that you can find under *FREEDOMOTIC\_ROOT\_FOLDER/plugin/devices/twitter/twitter-actuator.xml*.

## 61.3 How to obtain the OAuth parameters

- Create an account on Twitter that it is going to publish Freedom messages. It's recommended **NOT TO USE** your own but another as "house account". Maintain it private, and only accept users by invitation (if not, all persons could see your Freedomotic messages).

Now we have to obtain the **ConsumerKey** and **ConsumerSecret** for this application. In Twitter, login with the new user, and go to <https://dev.twitter.com/apps>. Register the app that is going to be used. (configure it as read&write in the Settings page) login to <https://dev.twitter.com/apps> with your new twitter “house account”

- Click on “create an application”. Fill the form with Name: Freedomotic ; Description: Twitter plugin for freedom building automation; Website: <http://freedomotic.com>; Application Type: Read&write; the rest can be left empty. Accept twitter rules, fill the captcha, click on “Create your twitter application”
- Now click on your application name and switch to tab “OAuth Tool”: we have obtained the ConsumerKey and the ConsumerSecret values.
- Now we have to obtain the AccessToken and the AccessTokenSecret. In the plugin jar is included a java executable: OAuthSetup
- Execute it in a terminal, paste the ConsumerKey and the ConsumerSecret. In output you obtain an url.
- Paste the url in a browser to obtain a Code.
- Copy and paste that code in the commandline where OAuthSetup is being executing. After OAuthSetup gives you the AccessToken? and AccessTokenSecret.
- Fill the twitter-actuator.xml with this 4 values (substitute them in the included example of xml) and your twitter plugin is configured.

```
<config>
  <properties>
    <property name="description"           value="Actuator for Twitter"/>
    <property name="version"               value="40"/>
    <property name="required"              value="40"/>
    <property name="category"              value="Twitter4Freedom"/>
    <property name="short-name"            value="TwitterActuator"/>
    <property name="OAuthConsumerKey"      value="**PASTE OAuthConsumerKey HERE**
↪"/>
    <property name="OAuthConsumerSecret"   value="**PASTE OAuthConsumerSecret_
↪HERE**"/>
    <property name="OAuthAccessToken"      value="**PASTE OAuthAccessToken HERE**
↪"/>
    <property name="OAuthAccessTokenSecret" value="**PASTE OAuthAccessTokenSecret_
↪HERE**"/>
  </properties>
</config>
```

## 61.4 Download

[Download plugin latest version](#)

## 61.5 Source code

[GitHub repository](#)

**Description:** A modern browser based frontend for Freedomotic

**Type: - Categories:** Frontend

**Development status:** Stable release

**Tested on:** All platforms

**Developer:** Gabriel Pulido de Torres

## 62.1 Overview

This plugin serves a modern HTML5 frontend.

## 62.2 Configuration

- Download the last available 5.6 dailybuild
- Open config.xml under FREEDOMOTIC\_ROOT/config and set KEY\_SECURITY\_ENABLE to false
- Start Freedomotic

This plugin wraps a Jetty embedded web server.

Once the plugin is running the Jetty server address is [http://freedomotic\\_instance\\_ip:8090](http://freedomotic_instance_ip:8090)

This server is used to be the holder of the web application client for Freedomotic.

This web application could be used to control the Freedomotic instance using any browser.

As it uses websockets, the browser needs to be a Chrome, Firefox>=10 or IEExplorer>=10 (or any browser that implements the websockets feature)

## 62.3 Download

The plugin is included inside the release package

## 62.4 Source code

[GitHub repository](#)

**Description:**

**Type: - Categories:**

**Development status:**

**Tested on:** All platforms

**Developer:** Mauro Cicoella

### 63.1 Overview

### 63.2 Configuration

**Description:** This plugin allows interfacing with zwave-powered devices

**Type:** Driver - **Categories:** Automation protocols

**Development status:** Proof of Concept

**Tested on:** All platforms

**Developer:** Matteo Mazzoni

## 64.1 Overview

This plugin uses a usb serial-to-zwave adapter (or every serial to zwave port) and an ad-hoc library (Zwave4j) to interface with devices. Requires Java 8 or above.

## 64.2 Configuration

## 64.3 Supported adapters

Not all this devices are tested, but they are supposed to work:

- Tricklestar (USB)
- ACT ZCS101 (Serial port)
- Z-troller (Serial port)
- Aeon ZStick (USB)
- Seluxit ViaSens 100 (USB)
- Z-Wave.Me Z-Stick (USB)



## 64.4 Supported devices

At a first stage will be supported Lights and Thermostats devices. Many other devices will be added during the beta phase. The following list reports all tested devices

### **SENSORS**

- Aeon Labs Multisensor DSB05-ZWEU

### **ACTUATORS**

## CHAPTER 65

---

### Things (objects) plugins list

---

Plugin	Description	Categories
<i>Base Home Automation</i>		
Harvester		
TV		
Weather		

---

## Base home automation

---

**Description:** The base things for home automation like on/off electric devices, lights, doors, sensors and toggle buttons

**Type: - Categories:** Access Control & Security, Lighting, Localization & Positioning

**Development status:** Stable Release

**Tested on:** All platforms

**Developer:** Enrico Nicoletti

---

**Note:** This plugin is included in the package distribution

---

### 66.1 Overview

### 66.2 Things list

#### 66.2.1 Air conditioner

**Description:**

Table 66.1: Behaviors

Name	Description	Type	Value
powered	power status	Boolean	on/off

Table 66.2: Triggers

Name	Description
powered	reads power status

Table 66.3: Commands

Name	Description
turn on	turns on the thing

### 66.2.2 Barometer

Description:

### 66.2.3 Clock

Description:

### 66.2.4 Decoration

Description:

### 66.2.5 Door

Description:

### 66.2.6 Fridge

Description:

### 66.2.7 Grocery list

Description:

### 66.2.8 Hygrometer

Description:

### 66.2.9 Light

Description:

### 66.2.10 Light sensor

Description:

### 66.2.11 Power meter

Description:

### **66.2.12 RGB light**

Description:

### **66.2.13 RGBW light**

Description:

### **66.2.14 Switch**

Description:

### **66.2.15 Thermometer**

Description:

### **66.2.16 Thermostat**

Description:

### **66.2.17 User**

Description:

### **66.2.18 Whashing machine**

Description:

## 67.1 Environments

### 67.1.1 Change renderer

This frontend supports different renderers.

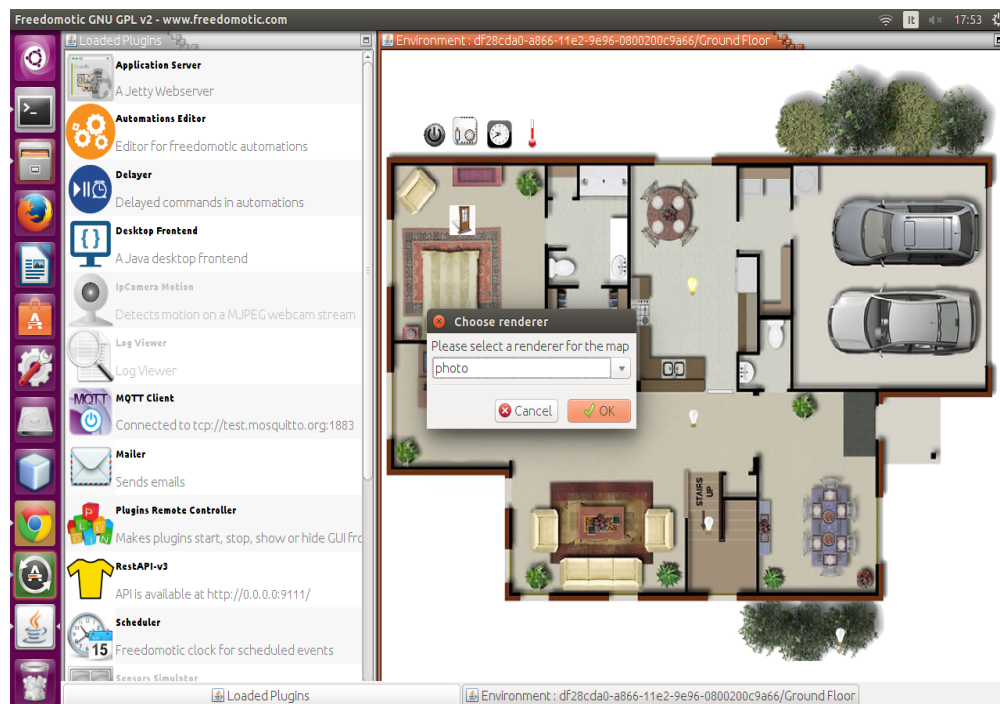


Fig. 67.1: Change renderer

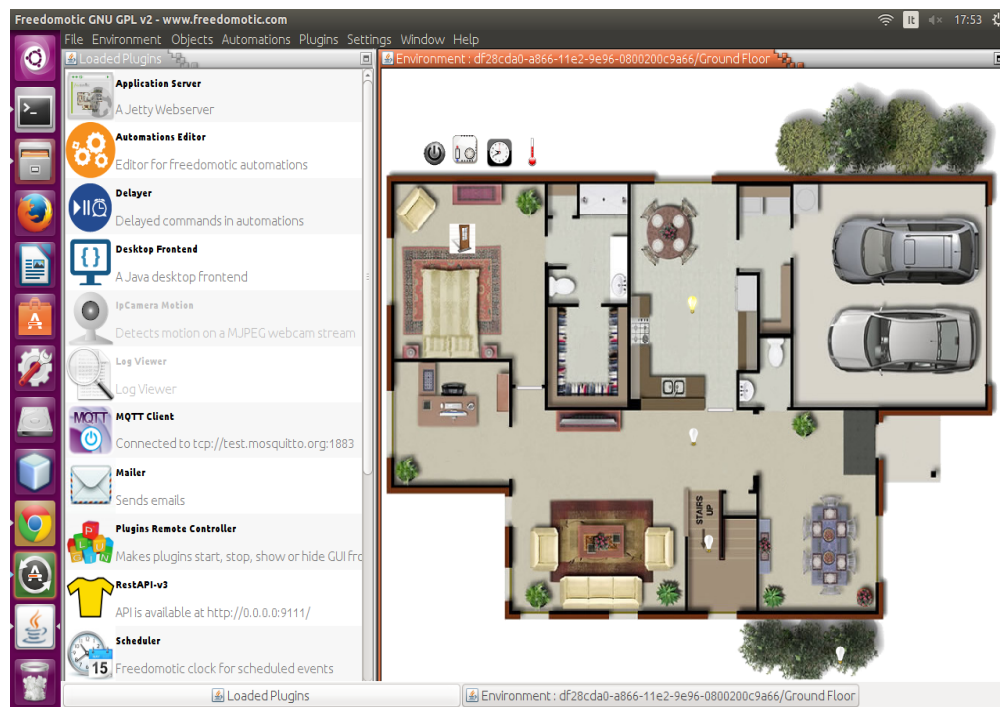


Fig. 67.2: Photo renderer

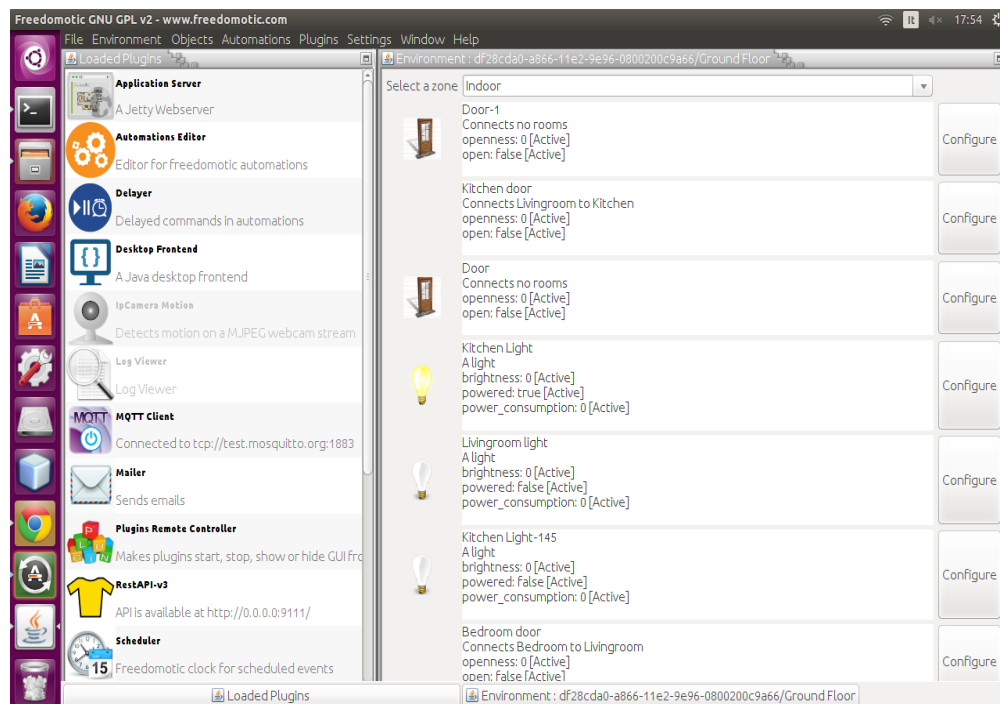


Fig. 67.3: List renderer

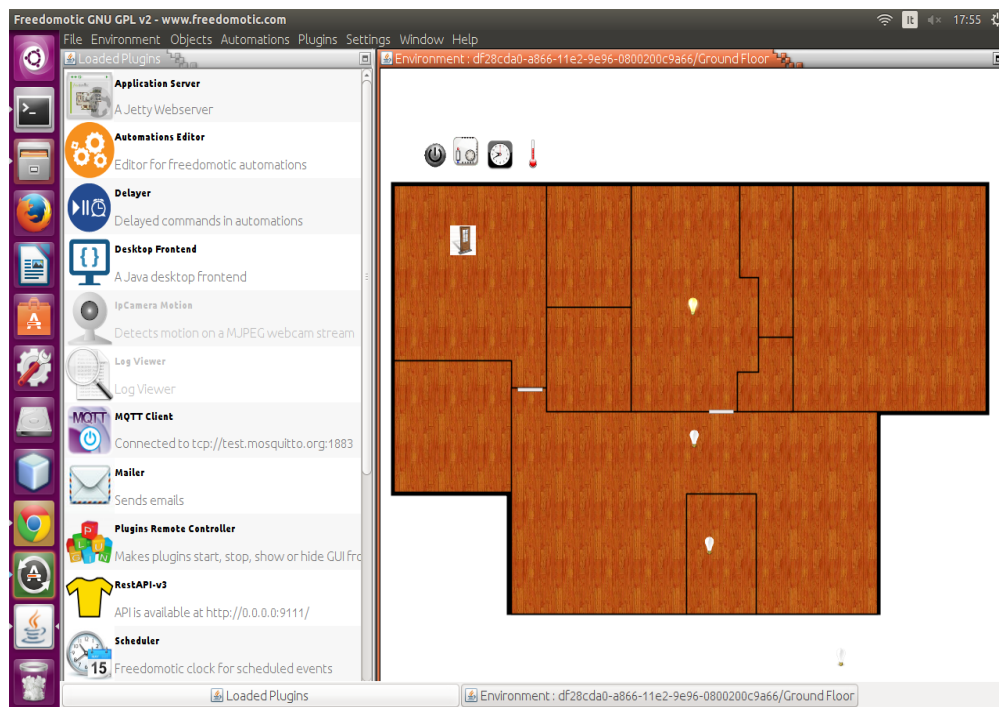


Fig. 67.4: Image renderer

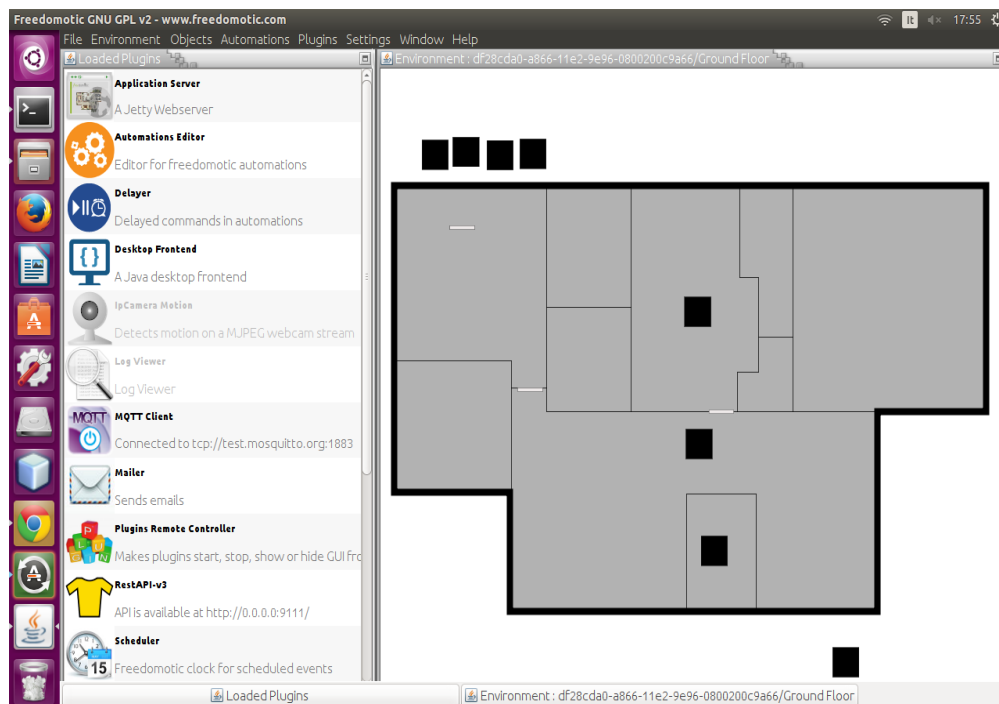


Fig. 67.5: Plain renderer



## 67.1.2 Change background images

Click on **Edit -> Environment -> Change Background** and select a PNG or a JPG file.

## 67.1.3 Add a new room

To add a new room, click on **Edit -> Room Edit Mode (F5)** and then **Edit -> Environment -> Add Room**. A new square polygon will be created on the top left corner of the environment. The blue handles may be dragged to position the room correctly on the map.

## 67.1.4 Rename a room

To rename a room, click on **Edit -> Environment -> Rename Room** and enter the new name into the dialog box.

## 67.1.5 Delete a room

To delete a room, select it by clicking on it, then click on **Edit -> Environment -> Remove Room**.

## 67.1.6 Edit room shape

To edit the map, simply click on **Edit -> Room Edit Mode (F5)**. Now you can drag around the blue handles, to edit the room's shape. The measures are in centimeters. When you are satisfied with your changes, disable the edit mode in menu **Edit -> Room Edit Mode (F5)**.

To add new handles, double click on a pre-existent one. To remove any handle, right-click on it.

# 67.2 Things

## 67.2.1 Move things

By clicking **Edit -> Objects Edit Mode (F6)** furniture may be dragged and dropped around the environment.

## 67.2.2 Configure things

Right clicking on a thing will display its configuration dialog.

## 67.2.3 How to add things to the environment map

There are two ways to add things to the map:

1. Add it from the toolbar: click on **Edit -> Objects Edit Mode** or press **F6**. Now you can see the list of things you can add into your environment. Right click on the thing to be added and choose **Add to environment**. Press **F6** again to return to the viewing mode.
2. Clone an already existing thing from the map: select the thing, right click on it, go to **Properties** and press **Create a Copy** button.

A new thing of the same type will be created and placed in the top left corner of the environment. To learn how to rename it and move it, read the next paragraph.

## 67.2.4 How to customize the things icons

Thing icons are stored in *FREEDOMOTIC\_FOLDER/data/resources*. To edit a thing icon (or create a completely new one) first edit (or create) the PNG image used to represent the thing and put it in the previously mentioned location.

Next, the icon must be associated with the thing. To do so right click on a thing and select the **Representation** tab. Click on **Change Image** and select the custom image that was placed into the *FREEDOMOTIC\_FOLDER/data/resources* folder.

Every thing behavior (on, off, etc.) can have a different icon to represent itself, so you have to repeat the operation for every representation you want to change. To change the current behavior of a thing, go to the **Control** tab and use the controls to switch it, then you can change its icon for this behavior, as explained above.

## 67.2.5 Connect things to real devices

In Freedomotic things are independent from the hardware used to drive them. For example: a light object is the same in Freedomotic whether it is being controlled by OpenWebNet, Arduino, z-wave, or something else.

First, ensure that the right driver plugin is installed. If not, install it by following the plugin-specific instructions on its marketplace page.

To bind any thing with a specific protocol, you have to right click on the thing to open its configuration dialog. Now go to the **Commands** tab and bind the thing to generic actions like `turn on` and `turn off` with the specific hardware command to execute it. For example a light `turn on` action can be bound with `turn on OpenWebNet (OWN) light` command selected from the list on the right.

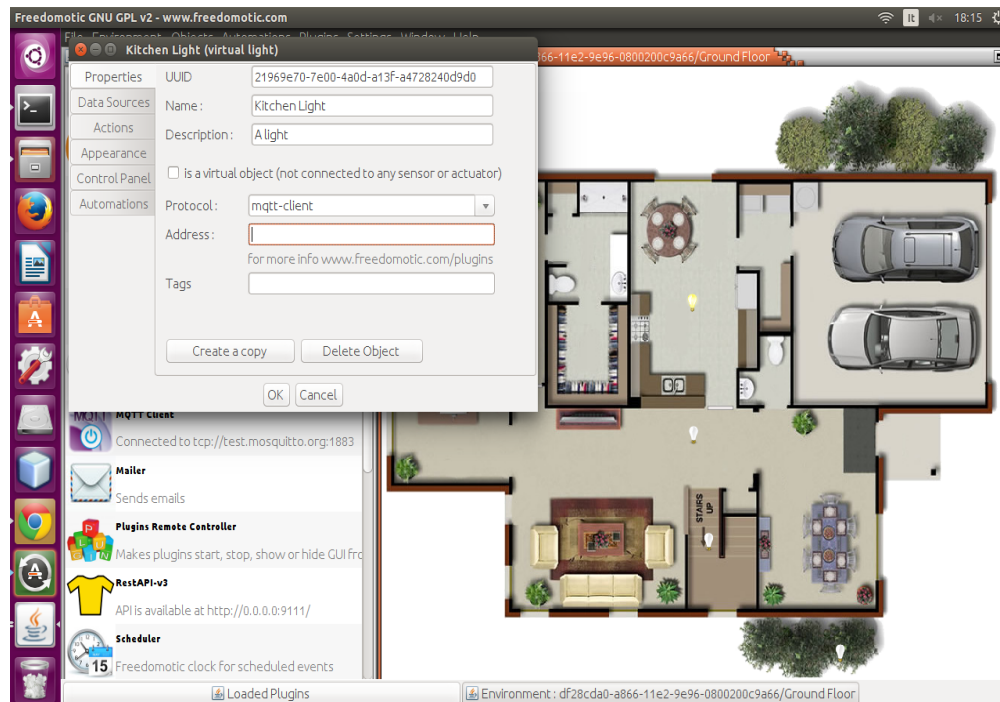


Fig. 67.6: Connect a thing to a real device

## 67.3 Localization

Freedomotic can detect your PC configuration and set the correct language.

If it's not available or you have chosen the default value **auto**, then the software uses **English**.

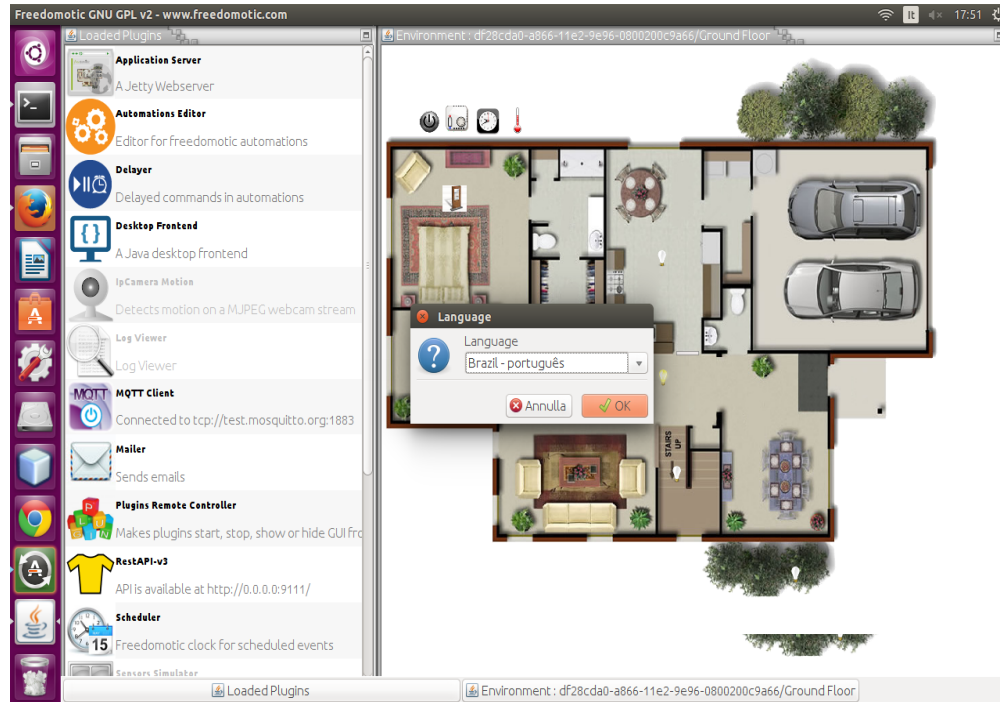


Fig. 67.7: Multilanguage Support

## 67.4 Automations

In Freedomotic, automations are more powerful than simple, timed turn on/off of objects. They can be created in nearly any natural language (we are working on it) in the form *if this happens, then do that*.

### 67.4.1 Event Driven Automations

To create a new automation, right click on the related thing, switch to **Automations Tab** and start to write your command into the input box related to the trigger you desire for your command.

For example:

1. Right click on a light.
2. Select the **Automations** tab.
3. Type `switch its power in the input box corresponding to the trigger if XXX is clicked`.
4. Click **OK**.

The light will now turn on and off when it is clicked.

## 67.4.2 Time Driven Automations

For timed automations such as `Do something every minute`, a **Clock** object is needed. If one is not already on the map, add it by pressing F6 and double clicking on the **Clock** thing on the list on the left side of the screen.

Right click on the new object, select the automations tab and create the automation in the same manner as the Event Driven Automations (explained above). For example: Switch power for all lights every 5 seconds.

Right click on the clock object and select the **Automations** tab, or use the automations editor, which lists all available triggers. Find the trigger `every 5 seconds` and link it to the command `switch power for all lights`.

## 67.5 Plugins

### 67.5.1 Download new features from the marketplace

In the **Plugins** menu click **Install from marketplace**. After the list is updated (it can take up to a minute) the list of plugins in the marketplace that are available for your current Freedomotic version will be displayed. To install one, double click on it and follow the instructions.

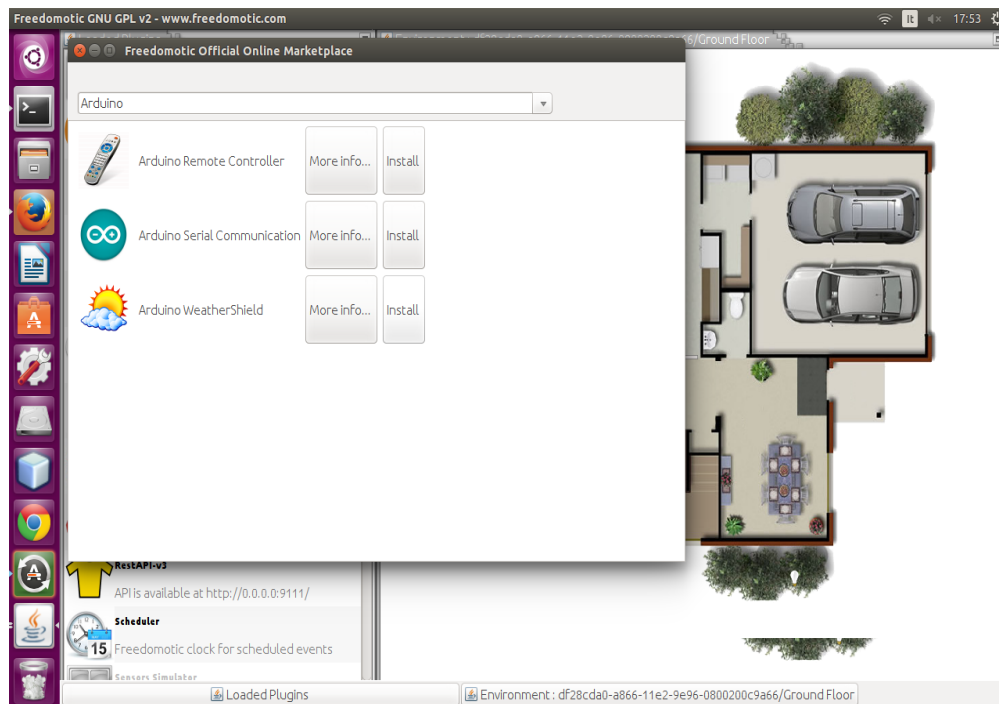


Fig. 67.8: Install plugins from the marketplace

### 67.5.2 Start and stop plugins

Loaded plugins are visible in the list on the left of the environment map. Running plugins are represented by a coloured icon. To start a plugin (or to stop an already running plugin), double click on its name. When a plugin is running, the feature it provides is available to the system. For example the OpenWebNet plugin enables communication with BTicino OpenWebNet (OWN) devices. This means the plugin doesn't provide automations to drive OWN devices

itself but only “translates” the Freedomotic user commands as turn on kitchen light into hardware-level specific commands. This allows you to forget about hardware and internal communication details. Simply say turn on kitchen light, and Freedomotic takes care of the rest.

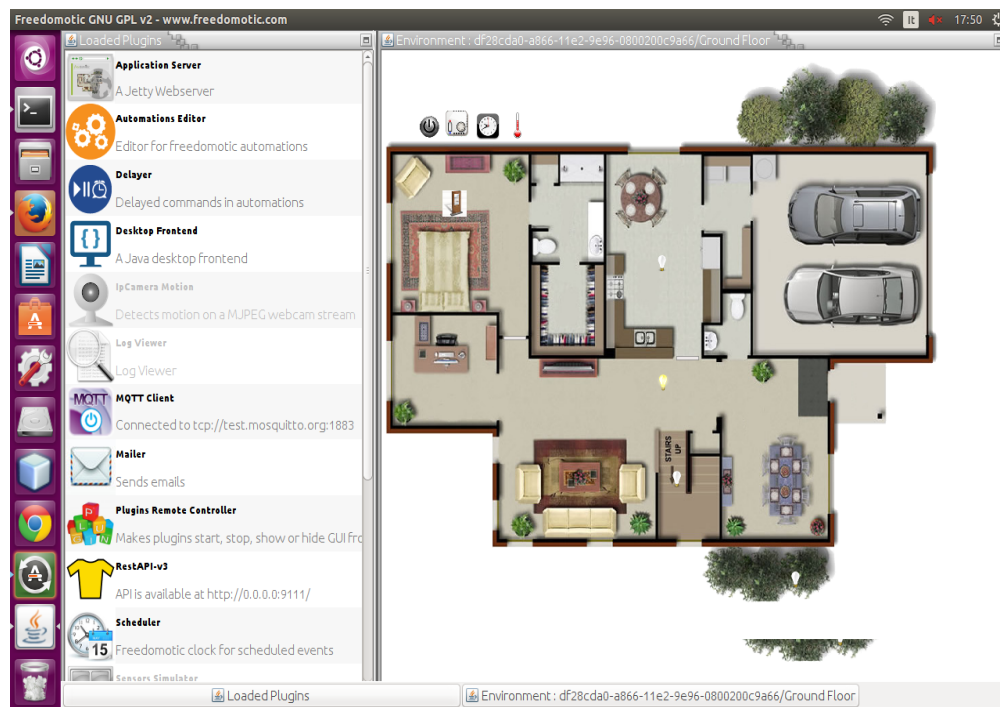


Fig. 67.9: Plugins list

### 67.5.3 How to configure a plugin

Some plugins offer a configuration dialog to interact with its features. To view it, right click on the plugin name. For an example, try this on the “**Sensors Simulator**” plugin, which is a development tool to simulate a temperature and luminosity sensor. You can make this fake sensor notify a temperature change to Freedomotic, by moving the **temperature** slider.

**Note:** Not all plugins have a configuration dialog, so if you right click and nothing shows, it is because the plugin have no configuration options.

## 67.6 Settings

## 67.7 Help

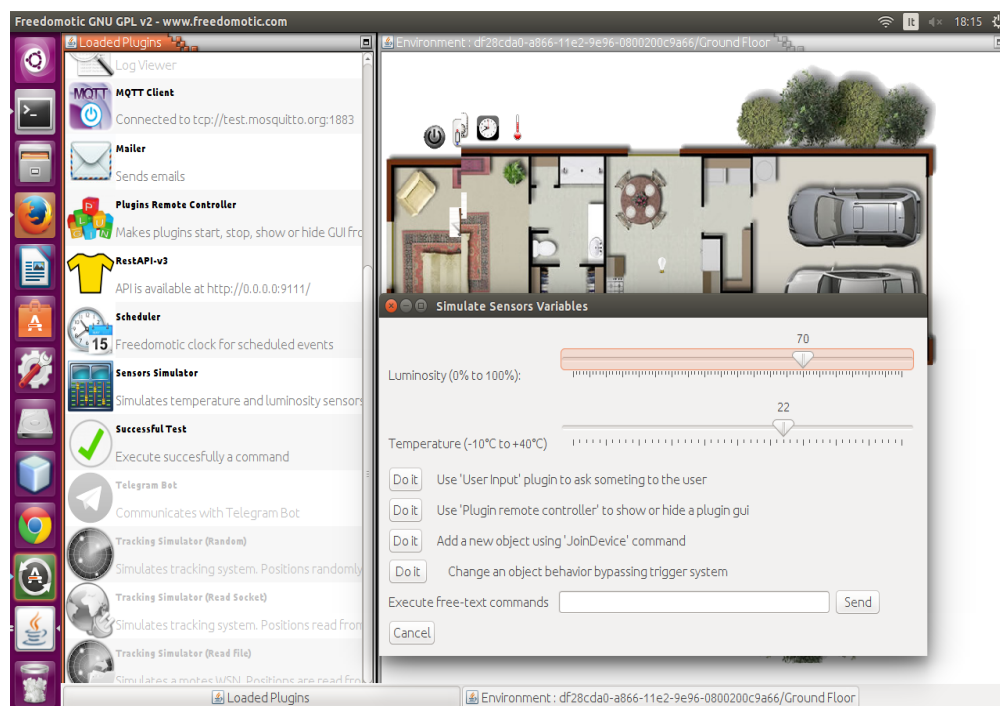


Fig. 67.10: Simulator Plugin

## CHAPTER 68

---

### Vue web client

---

TODO add description

<https://github.com/freedomotic/fd-vue-webapp>

If enabled Freedomotic logs all messages in a file called *freedomotic.log* located in *FREEDOMOTIC\_ROOT/log* folder. It is configured to roll if the size is more than 500Kb.

## 69.1 How to enable logging

Change the property `KEY_SAVE_LOG_TO_FILE` in *FREEDOMOTIC\_ROOT/config/config.xml* by choosing one of the following values:

- **OFF**: no logging
- **ALL**: all levels
- **LOG LEVEL (INFO | DEBUG | WARN | ERROR)**: specify a log level

For example

```
KEY_SAVE_LOG_TO_FILE = ALL
```

or

```
KEY_SAVE_LOG_TO_FILE = INFO
```

Here a log example

```
date LEVEL [thread] (Class:row) Message
2017-03-27 17:34:10,695 INFO [main] (BusBroker.java:60) Creating new messaging broker
2017-03-27 17:34:10,960 INFO [main] (BusBroker.java:63) Configuring messaging broker
2017-03-27 17:34:11,314 INFO [main] (BusBroker.java:66) Starting messaging broker
2017-03-27 17:34:12,211 INFO [main] (BusConnection.java:96) Creating connection_
↪factory
```



---

**Note:** **INFO** is the best choice in many cases

---

## CHAPTER 70

---

### Troubleshooting

---

### 71.1 What platforms are supported?

Freedomotic can run on any OS with Java support (Linux, Windows, Mac, ...).

### 71.2 Can I run Freedomotic on Raspberry Pi?

Sure, follow *these instructions*

### 71.3 What version of Java do I need to run Freedomotic?

Freedomotic can run on JDK version 8+.

### 71.4 What technologies / tools are used?

Freedomotic adopts open source technologies for development and project management.

### 71.5 I need help. How?

First of all consult this documentation. If you are a developer you can subscribe our technical mailing list (<https://groups.google.com/forum/#!forum/freedom-domotics>). Italian developers can join <https://groups.google.com/forum/#!forum/freedomotic-it>

## 71.6 Can I use Freedomotic in commercial projects?

Yes you can, and we encourage it!

Freedomotic is released under GNU GPL 2 license, both for core and for java plugins so any right/restriction of this license is applied. You can even sell Freedomotic copies or get paid to develop plugins for third parties. You can also fork Freedomotic, modify it and sell it. About this last case if you have some use cases that are not currently covered, before forking the project we suggest to start a discussion and let us know your needs, this way we can plan the implementation of new features into freedomotic-core and maintain the development effort on a single direction. You can participate in first person to change the freedomotic-core if you want it, we are glad to have new core developers. However the plugins system allows you to extend Freedomotic in a very flexible/powerful way, also using not Java languages, so rarely core changes are needed, even for complex extensions.

## 71.7 How to run Freedomotic headless (server mode with no GUI)

In Freedomotic GUIs are plugins. Freedomotic is shipped with a basic desktop GUI plugin installed into *FREEDOMOTIC\_ROOT/plugins/devices/frontend-java* folder. Just move the **frontend-java** folder away from there and restart Freedomotic without any GUI.

### 72.1 Version 5.5 (Bender)

#### **Project management**

- Mavenization
- Guice Dependency Injection
- Code refactoring
- Debian packages available
- First steps acquiring an Agile Development thanks to the use of Youtrack as issue tracker system
- Migration to TeamCity as integration server

#### **Framework**

- Authentication/Authorization (based on username/password and privileges list)
- Internationalization/Localization (about 20 languages supported)
- Enhanced Automation management
- Objects group commands support

#### **GUI Frontend**

- Multienvironment support
- Users and Privileges Management

#### **Docs**

- Developers Wiki
- Video tutorials on Youtube Channel

## Marketplace

Freedomotic main goal is to create a great ecosystem of plugins to control almost any aspect of an automated life. We are very proud of our efforts made to make the development of plugins easy to the developers that want to contribute. These include:

- Complete mavenization of the system and of the plugins build system
- Improved the development documentation. This is a permanent task on our workflow
- Active forums

## 72.2 Version 5.6 RC3 (Commander)

### Project management

- Enforced dependency injection
- Enforced adoption of internal API for core and plugins
- Enabled coherent access methods to repositories
- Moved issues from Youtrack to GitHub

### Framework

- Revamped P2P
- RestAPI v3
- Library version updates
- Enhanced overall stability
- Switched from JUL to SLF4j (logging)
- Extended extra conditions in Reactions (aka Automations)
- Updated javadoc
- Added periodic saving for commands/triggers/reactions

### Extra packaging

- Snappy App
- Docker images

### GUI Frontend

- Added option for sending logs via email

### Docs

- Moved the documentation to ReadTheDocs platform

## Marketplace

New plugins:

- MQTT (broker and client)
- MySensors
- Persistence (under development)
- Push notifications

- Telegram Bot (under development)
- ThingSpeak

New Things:

- Air Conditioner
- Thermostat

### **Internationalization**

- New languages added

---

### How to contribute

---

We are looking for people interested in our project!

You can contribute in many ways:

- **Start a personal project** based on Freedomotic or integrate one you have already coded
- **Join our development team**
- **Create plugins** to do remote control, media streaming, mobile development, fuzzy logic, artificial intelligence, or device control (such as X-10, KNX/EIB, OpenWebNet, Arduino, Z-Wave, Zigbee, EnOcean, Insteon PLC, etc)
- **Join our translation team** on [Transifex](#)
- Help us to **improve our documentation**
- **Publish tutorials/videos** about something you can do with Freedomotic
- **Use and test** the Freedomotic software and plugins
- **Design** user interfaces, icon packs, and project branding
- **Promote** us by talking about Freedomotic on your blog or social networks
- **Suggest use cases and projects** where Freedomotic can be used
- **Sponsor** your personal project joining as an **Affiliate Project**
- If you are an ITC Company, please consider our **Partnerships & Supporters** program

Please read our [development manual](#) to get started