# RoboWare Viewer Manual

**Version：0.10.38**

**Date：2017-09-20**

## Contents

# 1 Introduction

## 1.1 Purpose

The purpose of this manual is to fully describe the function of RoboWare Viewer and its operating environment. It helps users understand the scope of RoboWare Viewer and how to use it, and provides the necessary information for the maintenance and updating of RoboWare Viewer software.

## 1.2 What is RoboWare Viewer?

• RoboWare Viewer is a development tool using JavaScript to develop ROS UI (User Interface). By dragging-and-dropping components. RoboWare Viewer makes developing ROS UI more intuitive, simpler, and easy to operate.

• Interface componentization: each component can be individually configured to adjust the display style, data content, and so on.

• Cross platform: support Linux, Windows, OS X and other platforms.

• Support custom components: you can encapsulate the interface components according to your own needs.

## 1.3 Features

**Cross-platform**

RoboWare Viewer can be used in Linux, Windows, Mac OS X and other operating systems. Download the installation file, double-click to install, and it works out-of-the-box.

**Zero Code**

RoboWare Viewer provides an elegant style, rich variety of ROS-related components. You can develop a beautiful ROS UI (User Interface) without writing any code.

**Drag-and-drop Development**

RoboWare Viewer encapsulates a large number of components. You only need to drag the required components to the desired location, adjust the components by changing parameters of color, size, location and so on. You can also change the data source by configuring the ROS master URI and topic names.

**WYSIWYG (What You See Is What You Get)**

RoboWare Viewer provides live preview, you can view the performance of your application during developing process, including the performance on PC and phone.

**Multi-platform package release**

RoboWare Viewer can package your application into Linux / Windows / Mac OSX as a desktop application, or into Android / IOS as a mobile app.

# 2 Installation

## 2.1 Preparation

1. Install NodeJS
   （1） Download NodeJS（https://nodejs.org/en/download/current/）
   （2） Double click to install for Windows and Mac users
   （3） For Linux users, you could download binaries to install, or use package manager (recommended): https://nodejs.org/en/download/package-manager
2. Preparation on machine where roscore runs
   （1） Install mongodb（If installed, you can skip this step）

   ```
   apt-get install mongodb
   ```

   （2） Install Roboware_Viewer_Bridge
   - Clone the project:

     ```
     git clone https://github.com/tonyrobotics/RoboWare-Viewer-Bridge.git
     ```

   - Enter the RoboWare-Viewer-Bridge directory:

     ```
     cd RoboWare-Viewer-Bridge
     ```

   - Install the dependencies

     ```
     npm install
     ```

   （3） Install rosbridge-suite node

   ```
   sudo apt-get install ros-kinetic-rosbridge-suite
   ```
3. Launch
   （1） Launch mongodb

   ```
   /etc/init.d/mongodb start
   ```

   （2） Launch roboware_viewer_bridge

   ```
   node index.js
   ```

   （3） Launch rosbridge_server rosbridge_websocket

   ```
   roslaunch rosbridge_server rosbridge_websocket.launch
   ```
4. Network

Make sure the network are properly configured that your local machine RoboWare Viewer runs on can communicate to ROS master. For a detailed tutorial, please refer to the following site, http://wiki.ros.org/ROS/Tutorials/MultipleMachines

## 2.2 Install RoboWare Viewer

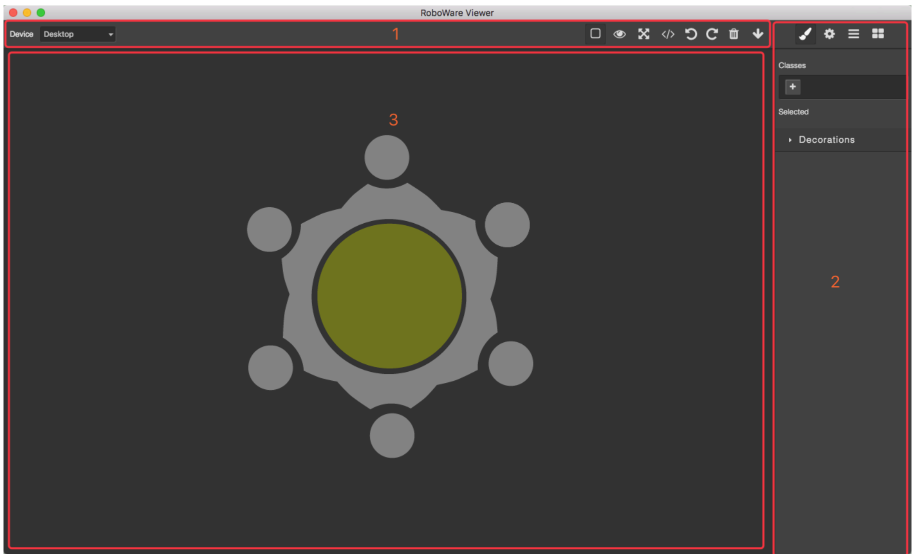
Download the latest version of RoboWare Viewer and install it easily by double click the deb file. For Linux users, you can also install it by the following commands in a terminal:

```
$ cd /path/to/deb/file/
$ sudo dpkg -i <roboware-viewer-.xxx>.deb
```

Where <roboware-viewer-.xxx>.deb indicates the downloaded installation file.

# 3 Tutorial

Launch RoboWare Viewer:



There are THREE sections:

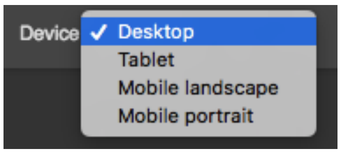
Section #1: Instruction Section

Section #2: Component Section

Section #3: Content Section

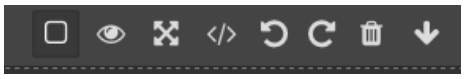
## 3.1 Instruction Section

Device list:



You could view the corresponding contents in “Content Section” according to these 4 different devices.

Others are as follows, from left to right: “View components”, “Preview”, “Full Screen”, “Show Codes”, “Undo”, “Redo”, “Clear All” and “Compile”.

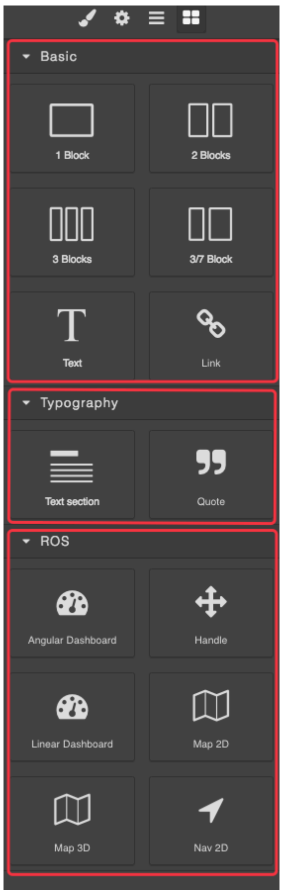


## 3.2 Component Section

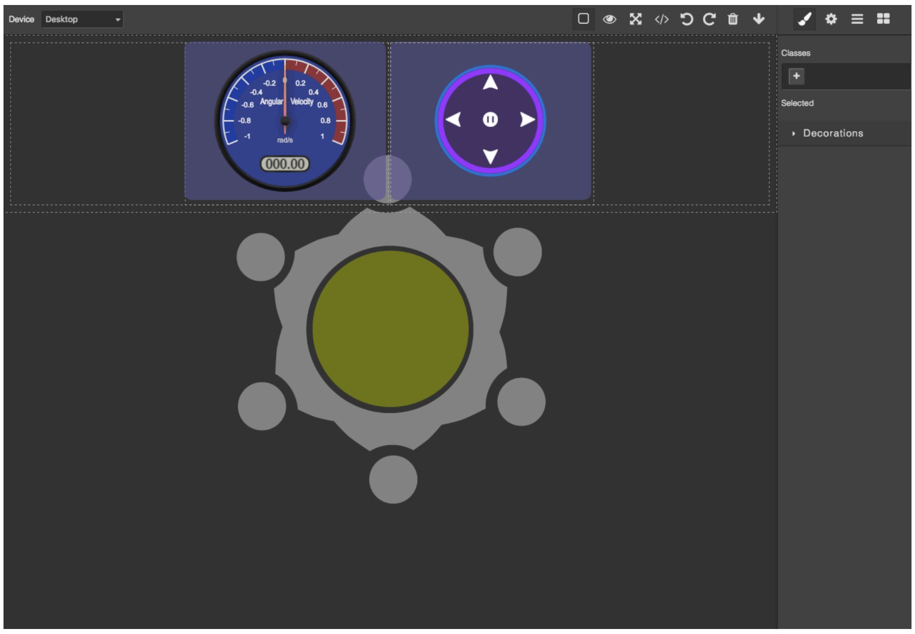
There are 4 kinds of managers in the “Component Section”, from left to right: "Style Manager”, “Settings Manager”, “Layout Manager” and “Component Manager”.

All available components in “Component Manager” are managed by the other 3 manager : "Style Manager”, “Settings Manager” and “Hierarchy Manager”.

## 1 Component Manager

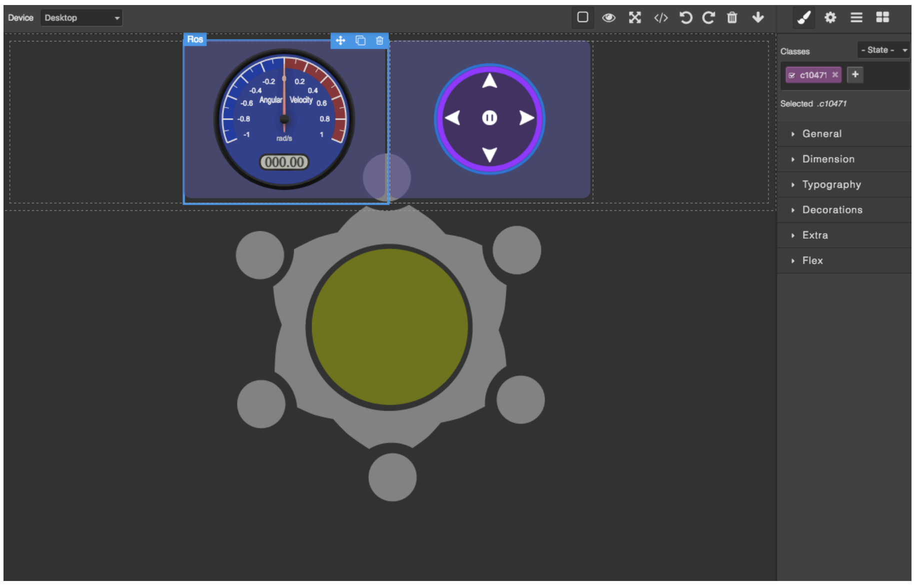


As shown in the figure above, there are 3 kinds of components in Component Manager: basic components, layout components and ROS components. You can drag-and-drop them to the Content Section. In this case, we drag-and-drop 2 components to Content Section: “Angular Dashboard” and “handle”, as follows:

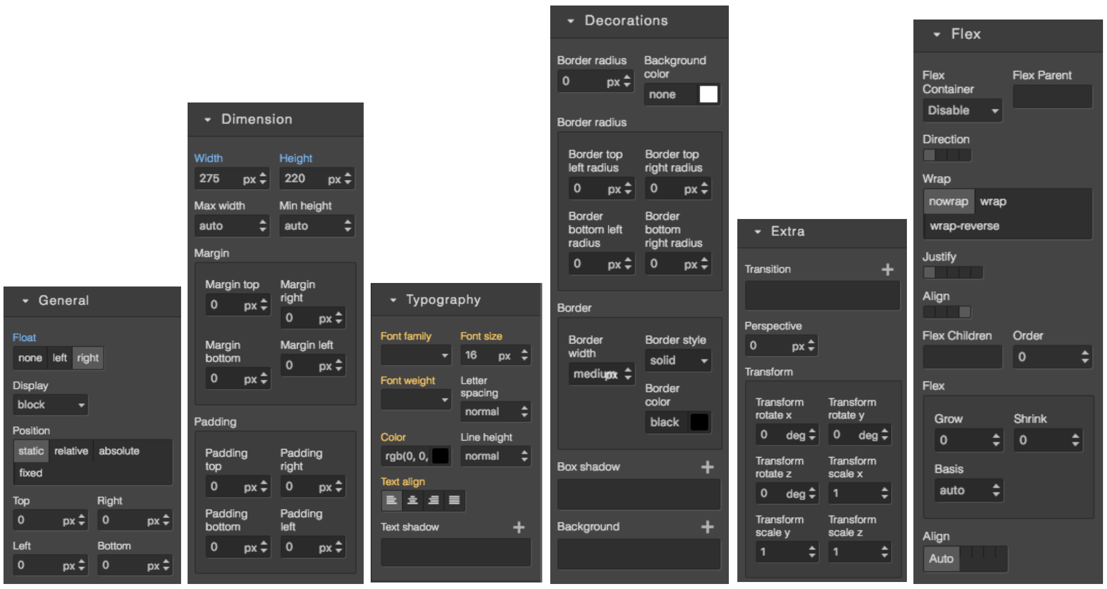


## 2 Style Manager

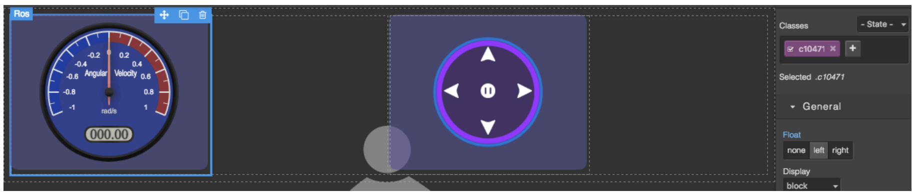
You can use Style Manager to configure any style of each component conveniently. When you select one of the components, the contents of Style Manager change, as shown below.



You can see that the currently selected component classes are "c10471" and the classes of the other component "c10931", which have 6 attributes: General, Dimension, Typography, Decorations, Extra, Flex. Detailed properties are:



At this point, change the Float property of the selected component General property to “left”.



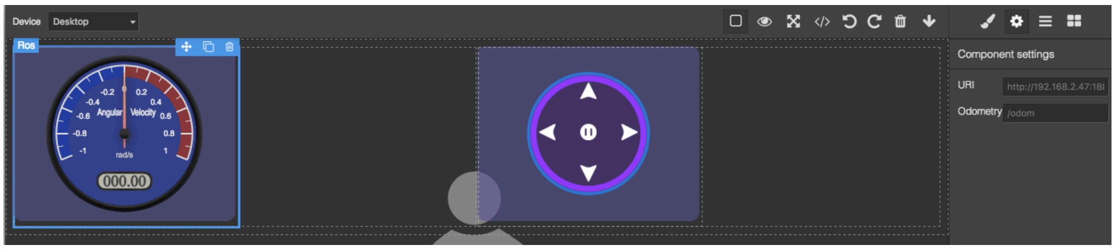
This component is shown on the left. But there is no response from the other component.

At this point, change the classes of both components to "c10471", then select any component, change the “General” - “Float” property, you would find that the two components have the same response.

Therefore, “Classes” control the styles of the components, and the Style Manager modifies “Classes”, thus affecting the components.

## 3 Settings Manager

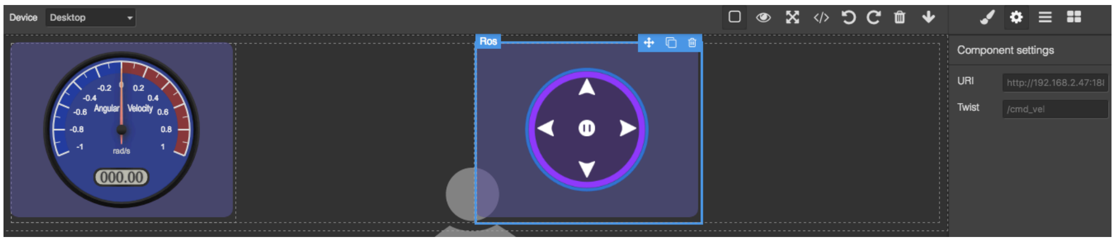
The Settings manager can modify the internal parameters of the component, and different components are shown with different parameters. Unlike the Style Manager, the parameters set are only applied to the currently selected component, as follows:



Component settings

URI http://192.168.2.47:18

Odometry /odom

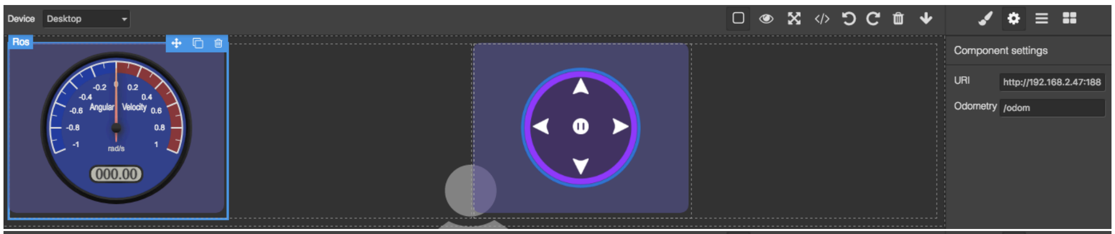


Component settings

URI http://192.168.2.47:18

Twist /cmd_vel

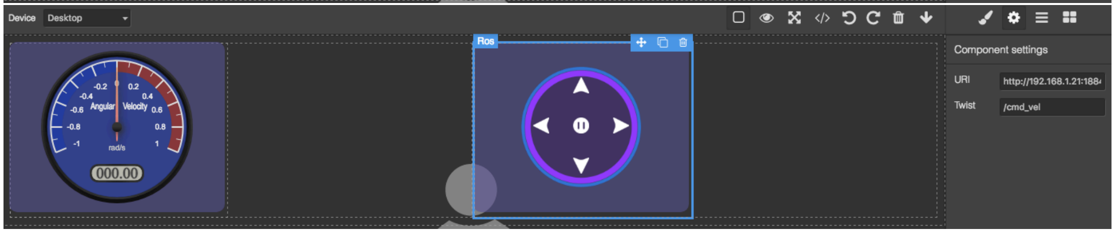
Select the two components separately, and you will find their parameters are different. Set parameters respectively:



Component settings

URI http://192.168.2.47:188

Odometry /odom



Component settings

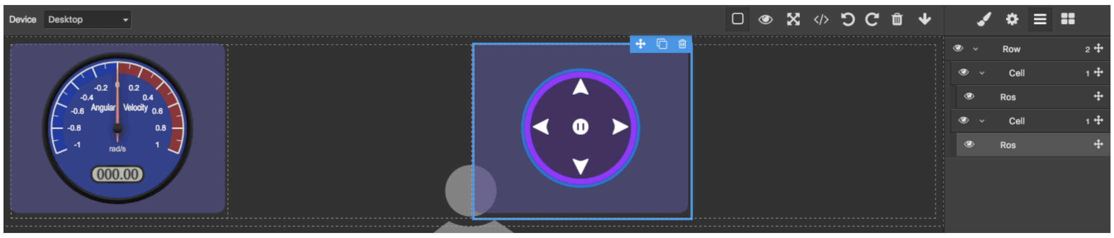
URI http://192.168.1.21:188

Twist /cmd_vel

According to the URL, parameters do not affect each other.

**4 Hierarchical Manager**

When using components, you can nest components as much as you like. The Hierarchical Manager can easily manage and align your layers.

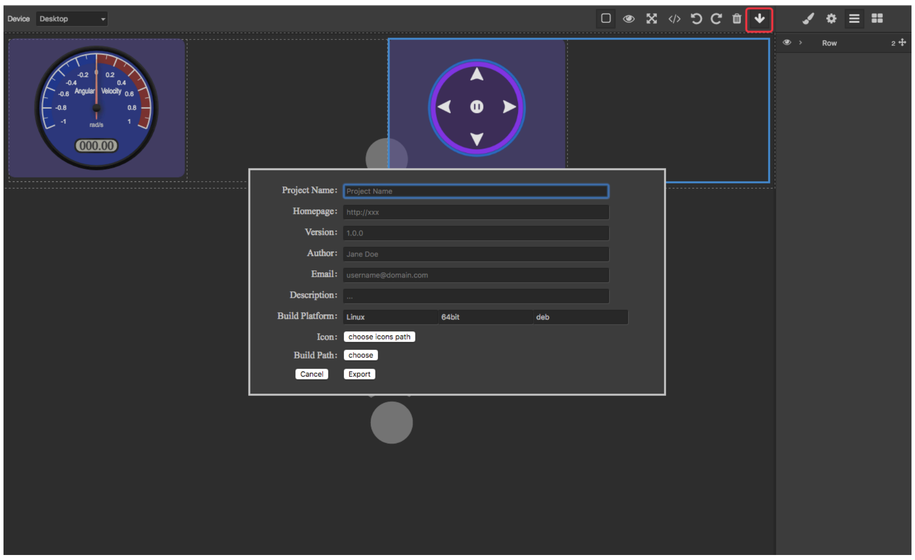


Row 2
Cell 1
Ros
Cell 1
Ros

## 4 Build standalone application

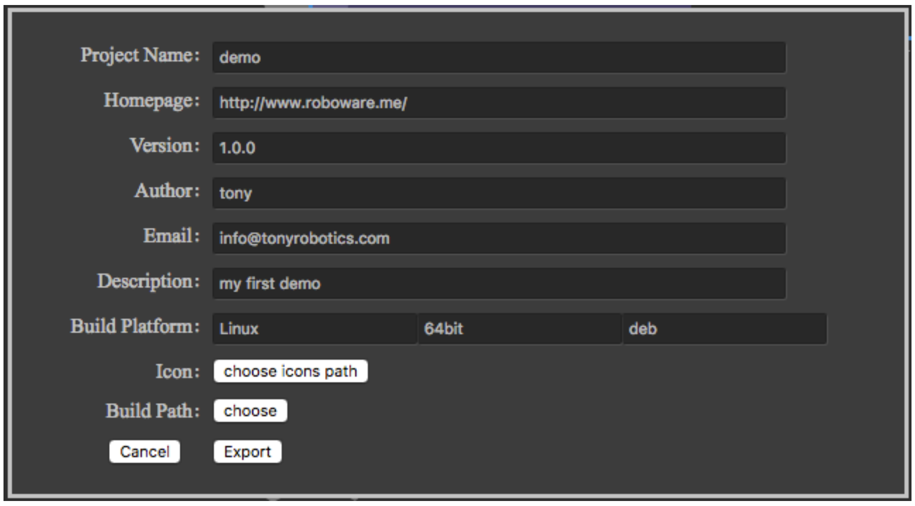
Once you have created your satisfied ROS interface, you can package it as a standalone application under Linux, Windows, and MacOS. Steps to generate standalone application:

Step 1: Select the compiler button in the Instruction Section, and the following interface will pop up.



Project Name: Project Name
Homepage: http://xxx
Version: 1.0.0
Author: Jane Doe
Email: username@domain.com
Description: ...
Build Platform: Linux 64bit deb
Icon: choose icons path
Build Path: choose
Cancel Export

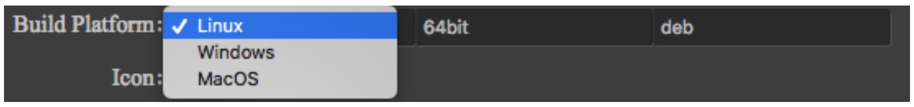
Step 2: Fill in the content according to the prompt.



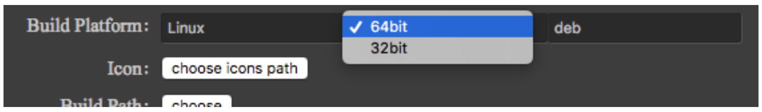
|  |  |
|---|---|
| Project Name: | demo |
| Homepage: | http://www.roboware.me/ |
| Version: | 1.0.0 |
| Author: | tony |
| Email: | info@tonyrobotics.com |
| Description: | my first demo |
| Build Platform: | Linux 64bit deb |
| Icon: | choose icons path |
| Build Path: | choose |

Cancel Export

The optional Build Platforms are Linux, Windows or MacOS platforms.

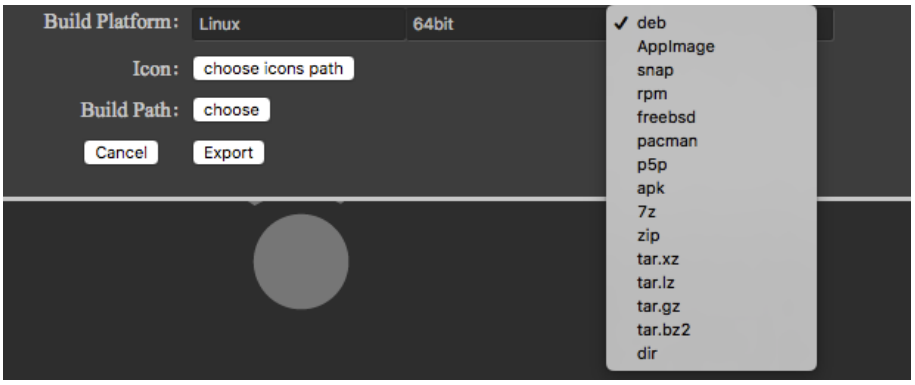


Each platform has 64/32 bit architecture.

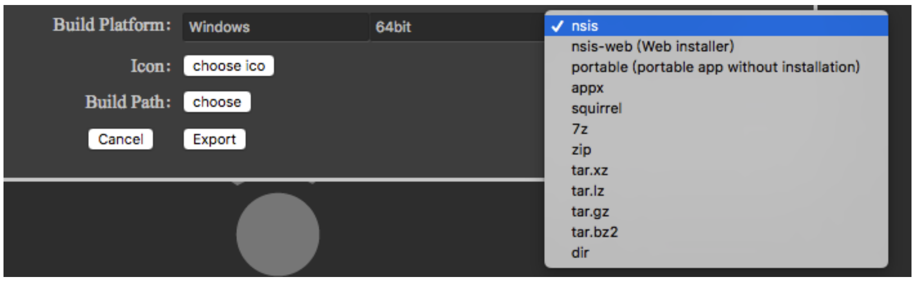


The compiled target depends on the platform:

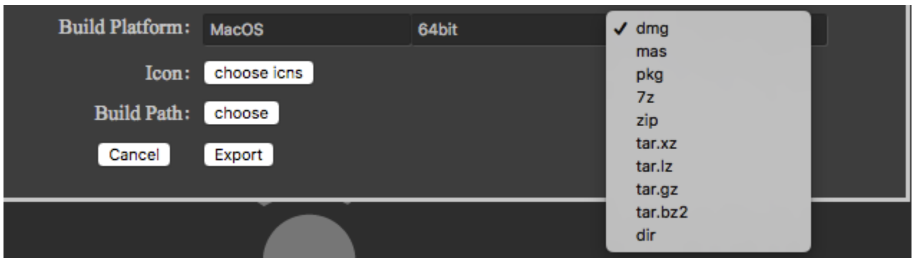
15 different targets can be compiled on Linux platform: deb, AppImage, snap, rpm, freebsd, pacman, p5p, apk, 7z, zip, tar.xz, tar.lz, tar.gz, tar.bz2, dir.



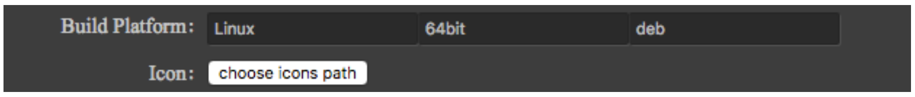
12 different targets can be compiled on windows platform: nsis(exe), nsis-web, portable, appx, squirrel, 7z, zip, tar.xz, tar.lz, tar.gz, tar.bz2, dir.



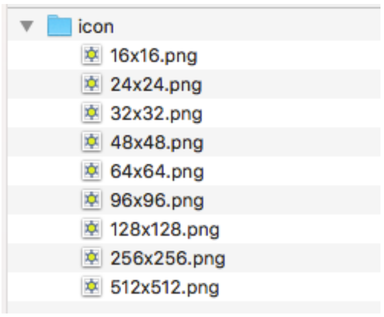
10 different targets can be compiled on MacOS platform: dmg, mas, pkg, 7z,zip, tar.xz, tar.lz, tar.gz, tar.bz2, dir.



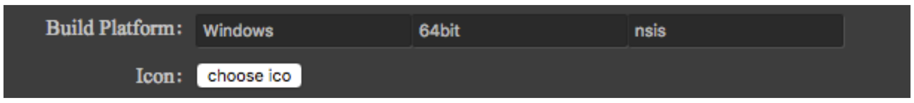
In this case, the Icon's choice is: when the Build Platform is Linux, you must select a folder for the Icon.



In the folder, there must be PNG images for different sizes:

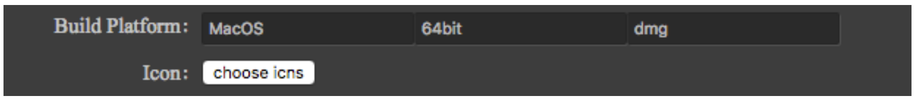


When the Build Platform is Windows, you must select a *.ico file for the Icon.


Build Platform: Windows 64bit nsis

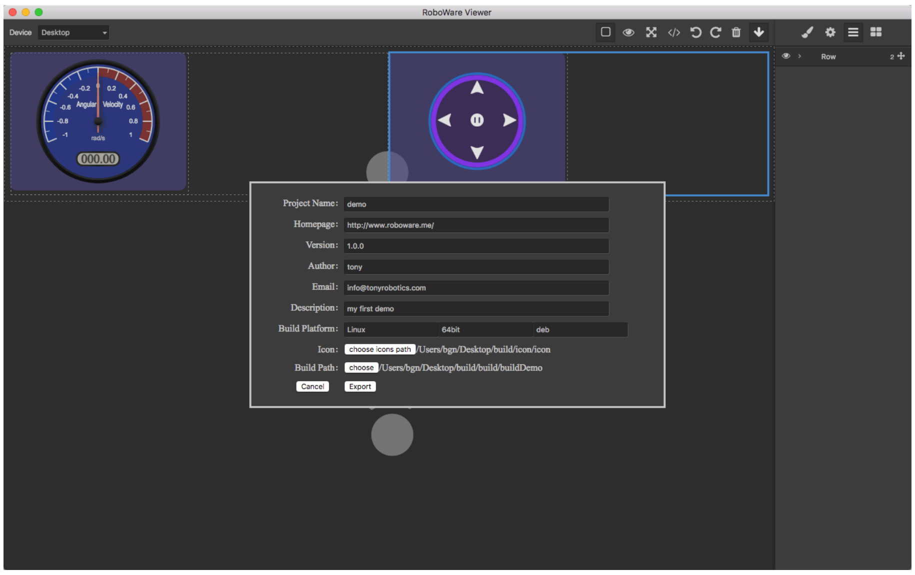
Icon: choose ico

When the Build Platform is MacOS, you must select a *.icns file for the Icon.


Build Platform: MacOS 64bit dmg

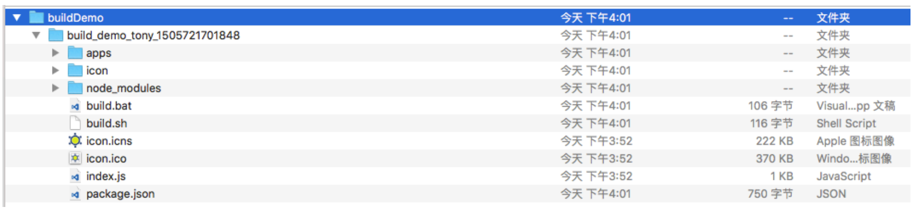
Icon: choose icns

"Build Path" specifies the folder you want to compile to. Don't use a folder whose name contains Chinese characters. The current user must have read and write permissions. The final information is as follows:



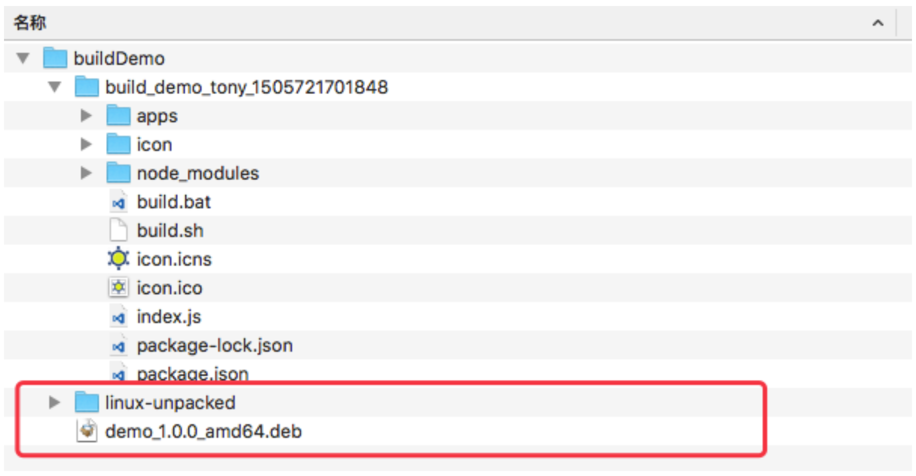
|  |  |
|---|---|
| Project Name: | demo |
| Homepage: | http://www.roboware.me/ |
| Version: | 1.0.0 |
| Author: | tony |
| Email: | info@tonyrobotics.com |
| Description: | my first demo |
| Build Platform: | Linux 64bit deb |
| Icon: | choose icons path /Users/bgn/Desktop/build/icon/icon |
| Build Path: | choose /Users/bgn/Desktop/build/build/buildDemo |

Click the Export button, a folder named "build_${Project Name}_${Author}_ timestamp" and necessary build files will be generated in "Build Path".



|  |  |  |  |
|---|---|---|---|
| ▼ buildDemo | 今天 下午4:01 | -- | 文件夹 |
| ▼ build_demo_tony_1505721701848 | 今天 下午4:01 | -- | 文件夹 |
| ▶ apps | 今天 下午4:01 | -- | 文件夹 |
| ▶ icon | 今天 下午4:01 | -- | 文件夹 |
| ▶ node_modules | 今天 下午4:01 | -- | 文件夹 |
| build.bat | 今天 下午4:01 | 106 字节 | Visual...pp 文稿 |
| build.sh | 今天 下午4:01 | 116 字节 | Shell Script |
| icon.icns | 今天 下午3:52 | 222 KB | Apple 图标图像 |
| icon.ico | 今天 下午3:52 | 370 KB | Windo...标图像 |
| index.js | 今天 下午3:52 | 1 KB | JavaScript |
| package.json | 今天 下午4:01 | 750 字节 | JSON |

A standalone application will be generated in the current folder by executing a build.sh file for Mac and Linux users and build.bat for Windows users. As follows:



名称
buildDemo
build_demo_tony_1505721701848
apps
icon
node_modules
build.bat
build.sh
icon.icns
icon.ico
index.js
package-lock.json
package.json
linux-unpacked
demo_1.0.0_amd64.deb