# Document on MSC with AR-Drone 2

Zhangyuan Wang

05/18/2016

This document introduces how to set up the environment to run MSC on AR-Drone 2, both on Windows and Linux. In addition, the general way to configure node-drone project is presented. And the work to be done in the future is discussed.

## Set up on Windows

1. **Set up Map Seeking Circuit**

This part is mostly done by the Visual Studio configuration. All the compiled opencv file are included in the github repository.

1. Install Visual Studio 2015 Community.
2. Git clone from <https://github.com/nnnrpq/msc>
3. Add ***MSCROOT/msc/opencv\_2411/bin/Debug*** and ***MSCROOT/msc/opencv\_2411/bin/Release***to system environment.
4. Open msc.sln. It should be ready to use in Visual Studio, both in debug mode and release mode. F5 to build and start.
5. **Set up node-ar-drone**

We need to manually setup this part, since the packages needs to be installed in the correct place.

1. Install nodejs
2. cd to ***MSCROOT/msc***
3. Install [ar-drone](https://github.com/felixge/node-ar-drone) through nodejs:

***npm install git://github.com/felixge/node-ar-drone.git***

1. Install [node-arraybuffer](https://github.com/vmolsa/node-arraybuffer) and [nan](https://github.com/nodejs/nan), to convert nodejs object to void\* in C.

***npm install node-arraybuffer –save***

***npm install nan –save***

1. Install [node-gyp](https://github.com/nodejs/node-gyp), native addon build tool for nodejs. Do as the github says:
2. Install ***npm install -g node-gyp***
3. Install python 2.7 and run ***npm config set python python2.7***
4. Launch cmd, ***npm config set msvs\_version 2015***
5. Create (binding) addon project for VS

Since we need the packages in nodejs and still want to configure the whole project in VS2015, we first create a “hello” project with node-gyp and change the files and settings of the created project.

In ***MSCROOT/msc,*** there’s already a binding.gyp and hello.cc. Run ***node-gyp configure install*** and node-gyp will create a VS project in the ***MSCROOT/msc/build*** directory.

1. Add files and configure the project

Open the binding.sln or MSC\_drone.vcxproj. Change to Release mode. Add the source files from ***MSCROOT/msc/src*** as in the previous MSC project. Delete **hello.cc** from projectand **jsmsc.cc** instead**.**

Edit the properties of project. Open View-Others-Property Manager and in the Release part, add existing property sheet **opencv2411r.props.** This should load the existing property page.

I’m not sure why this happens, but if it cannot find **Username\.node-gyp\4.4.4\Release,** copy the folder ***MSCROOT*\.node-gyp\4.4.2\Release** to the place. Also, you might need to add the folder of **..\ pthreads-w32-2-9-1-release\lib** to **Project-Properties-linker-general-additional library directories**.

Build the solution, which produces in a .node file for nodejs.

1. Call the node from nodejs and fly the drone

Get the drone ready. Connect to its wifi from PC.

Go to ***MSCROOT/msc/js.*** We need to use binary library ffmpeg to get the image stream.

Open **ff-prompt.bat.** Go back to the upper directory (cd ..). Type **node jstest.js**, which will run the js script in node.

Demo done!

## Set up on Linux

## Work summary