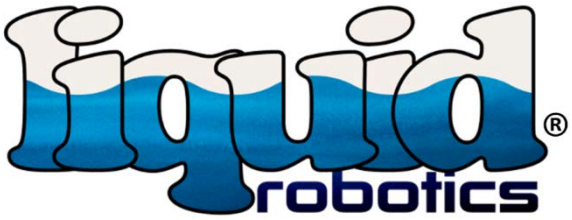

Developing for the SMC

User Manual

Version 0.03

26 September 2013

LIQUID ROBOTICS, INC.
1329 MOFFETT PARK DR.
SUNNYVALE, CA 94089
(408) 636-4200

LRI Part Number: 04340-03 ?

This Page Intentionally Left Blank

REVISION HISTORY

Rev	Changes	SMC	Author(s)
0.01	Pre-release version	2.0.0	MC

TRADEMARKS

PATENTS & INVENTIONS

The Wave Glider mechanism is patented under US Patent Number 7,371,136 and other patents pending. All aspects of the Wave Glider product have been developed at private expense by Liquid Robotics, Inc.

This Page Intentionally Left Blank

Table of Contents

Figures and Tables

Chapter 1 – Introduction

The Sensor Management Computer (SMC) is a Linux-based embedded ARM computer tailored to act as a Wave Glider™ payload and to implement algorithms and interfaces to sensors and other devices. This computer includes:

- Logic PD Torpedo System On Module featuring TI ARM Cortex A8 DM3730 or OMAP3530
- Eight (8) general-purpose serial ports
- Dedicated serial communications with the Wave Glider navigation computer (Float C&C)
- Serial and PPS interfaces to the system GPS
- Serial debug port
- Ethernet interface
- MicroSD interface (currently shipping with a 16GB microSD card)
- USB high-speed On-the-Go inteface

Linux was chosen as a platform because of its relative popularity and interoperability between desktop development environments and the embedded target. Liquid Robotics (**LRI**) recommends developing on Ubuntu desktop and supports developers in the Ubuntu environment.

The Wave Glider is powered entirely by solar panels. ARM Cortex A8 processors implement a variety of features that improve power efficiency when compared with other processors capable of supporting Linux. The SMC uses as little as 0.5W vs. 6W for the lowest Intel ATOM based processors.

This manual explains the resources available on the SMC

Chapter 2 – Setting Up the Development Environment

2.1 Operating System and Virtualization

The SMC software runs on an ARM-based Linux maintained by Timesys Corporation. We are currently recommending the Eclipse editor and the makepp build system documented below. LRI currently recommends Ubuntu 12.04 LTS (Precise Pangolin) 32-bit[1]

- Windows and MAC users are recommended to use Oracle VirtualBox[2] or VMWare. All examples will be using VirtualBox.
- Native Linux/MacOS users can also install Oracle VirtualBox[3] if not already using Ubuntu or running on an existing 64-bit Ubuntu installation.

Ubuntu can be downloaded for free from the site:

- Version 12.04 LTS is at http://releases.ubuntu.com/precise/

The standard download works well on top of Linux, Mac OS or Windows: be sure to choose the version that is compatible with your host operating system. (If you are installing on the bare metal, consult the Ubuntu documentation; if you are using a RAID disk on the bare metal, you'll need the alternate installer.)

Oracle VirtualBox and the Extension Pack can be downloaded free from the site at https://www.virtualbox.org/wiki/Downloads.

2.2 Recommended Hardware/Software Environment

- A PC or MAC with at least 8 GB of RAM and 100 GB free disk space
- (PC) Windows 7 or later or native Ubuntu installation
- Oracle VirtualBox (not VirtualBox OSE) for virtualization, if appropriate
- Available USB-based micro-SD card reader/writer

2.3 Configuring Ubuntu VM (using pre-existing image)

After installing VirtualBox:

1. From the VirtualBox File menu, select Install Appliance and select the appliance file (.ova); on the Appliance Settings screen, check the "Reinitialize the MAC address of all network cards" box. Click "Import."
2. When the import operation completes, start the virtual machine and log in with user name "dev" and password "1329moffet"

[1] Use the 32-bit version to avoid integer size incompatibilities, etc.
[2] Do not use the VirtualBox OSE distribution since it does not support USB devices well, and external card interfaces and ports are useful with SMC development.
[3] Do not use the VirtualBox OSE distribution since it does not support USB devices well, and external card interfaces and ports are useful with SMC development.

3. Be sure to update the package via Update Manager before continuing.
4. To get a Timestorm license from Timesys, retrieve the MAC ID of the Ethernet adapter (eth0) as follows (shows as the HWaddr) and contact Timesys or your administrator to obtain a license:

```
dev@dev-VirtualBox:~ $ ifconfig ↵
eth0      Link encap:Ethernet  HWaddr 08:00:27:2b:d7:a1
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe2b:d7a1/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:29493 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7734 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:30816138 (30.8 MB)  TX bytes:469687 (469.6 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:88 errors:0 dropped:0 overruns:0 frame:0
          TX packets:88 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:7149 (7.1 KB)  TX bytes:7149 (7.1 KB)
```

The license file should be stored under ~/timestorm-4.0/licenses.

2.4 Configuring Ubuntu VM (manual process)

After installing VirtualBox:

1. Create a new Virtual Machine, selecting “Linux” as the type and “Ubuntu” as the version.
2. Allocate as much RAM as would be feasible. Ideally, you would have at least 4 GB allocated to Linux (you can change this configuration later if desired)
3. Choose the type of hard drive. The VirtualBox Disk Image (VDI) is recommended unless you need interoperability with another virtualization product.
4. Select fixed size (recommended if you have the free space) or dynamically allocated (if space is a concern.)
5. Choose a disk size (at least 50 GB is recommended, 100 GB would be better for long term use.)
6. It’s useful to enable bi-directional clipboard access (under Settings → General → Advanced)
7. Start the virtual machine and select the .ISO file downloaded from the Ubuntu site.
8. Follow the prompts to install Ubuntu download any updates during installation, using the entire virtual disk for Ubuntu.
9. Enter your name and user name and a strong password (the name “dev” and the password “1329moffet” are used for the development image)
10. Wait for installation process to complete and reboot when prompted.
11. From “System Settings” and “User Accounts”, create “smc” and “smc0” users, leaving the accounts disabled. These user names are needed when creating packages for the target (i.e. the SMC)
12. When the Update Manager prompts (usually with hundreds of updates to be performed), open it and download updates; restart Ubuntu when prompted.
13. At any point during the installation, you can pull down the VirtualBox Devices menu and click Install Guest Additions to address some issues with the size of the display. This will compile

some modules and then require a restart; the display size problems will be addressed after restart. Note that later installations may undo this, and it will have to be repeated; so it may be better to wait until most of the installations have been made.

14. Open a terminal session (Ctrl-Alt-T)
15. If desired, the password prompt for "sudo" (which allows a single command to be run as the root user) can be bypassed by running "sudo sudoedit /etc/sudoers" and adding the following at the end of the /etc/sudoers:
    dev ALL = (root) NOPASSWD: ALL
16. Install aptitude by running "sudo apt-get install aptitude"
17. If desired, restore classic gnome by running "sudo aptitude install gnome-session-fallback"; then log out. Pull down the menu presented in the login bar and select "Gnome Classic" or "Gnome Classic (no effects)", then continue the login.
18. Run "sudo usermod -a -G dialout dev" to allow user "dev" to use USB serial ports, whenever they are mounted.
19. If desired, install the Chromium web browser by running "sudo aptitude install chromium-browser"
20. Install typical support programs by running "sudo aptitude install g++ meld subversion kdiff3 rapidsvn libnss3-dev lua5.1 lua5.1-doc liblua5.1-dev screen"
21. For systems running on bare metal or where the system must be contactable via ssh, install the ssh server (sudo aptitude install openssh-server)
22. For systems running on bare metal or for additional security, enable the firewall, but allow ssh access:

```
sudo ufw enable ↵
sudo ufw allow ssh ↵
```

23. For systems running in a virtual machine or for enhanced security, install the "security-delete" package using Ubuntu Software Manager. Whenever a virtual machine appliance is exported, it is advisable to clear the free space to reduce the size of the resulting appliance image file. The security-delete package also contains utilities that write patterns of random data over files when they delete to reduce the risk of data recovery by a third party.
24. Download "makepp" source (makepp-2.0.tgz) from http://sourceforge.net/projects/makepp/files/2.0/ and expand it from the "dev" home directory (tar xvzf ~/Downloads/makepp-2.0.tgz) Then execute the following:

```
cd makepp-2.0 ↵
./configure --prefix=/usr/local ↵
sudo make install ↵
```

25. Install the precompiled boost libraries into well-known locations under /usr/local:

```
cd /usr/local ↵
sudo mkdir -p boost/1.46.1 ↵
sudo mkdir -p boost/1.50.0 ↵
cd boost/1.46.1 ↵
tar xvpf ~/Downloads/boost_1_46_1.tar.bz2 ↵
cd ../1.50.0 ↵
tar xvpf ~/Downloads/boost_1_50_0.tar.bz2 ↵
```

26. By convention, SMC source code is loaded into directories under /home/SMCDev/SMC, and some other files are placed into directories under /home/SMCDev. Run the following to create these directories:
```
cd /home ↵
sudo mkdir SMCDev ↵
sudo chown dev SMCDev ↵
sudo chgrp dev SMCDev ↵
```
27. Install omniORB support files:
```
mkdir /home/SMCDev/omniORB ↵
cd /home/SMCDev/omniORB ↵
tar xvf ~/Downloads/omniORB.tar.bz2 ↵
```
For the Ubuntu build, it may be necessary to rename omniORB/lib/python2.6 to match the current python build (for example, python2.7)
28. Install timesys toolchains for DM3730, both the 3.0 and the 6.0.0 kernels. Ensure that the .sh files are marked executable before starting:
```
mkdir -p /home/SMCDev/timesys/6.0.0 ↵
mkdir -p /home/SMCDev/timesys/3.0 ↵
echo /home/SMCDev/timesys/6.0.0 | ~/Downloads/dm3730_zoom_torpedo-development-
environment-6.0.sh ↵
echo /home/SMCDev/timesys/3.0 | ~/Downloads/dm3730_zoom_torpedo-development-environment-
3.0.sh ↵
```
29. Install the **ddd** front end to gdb for visual debugging:
```
sudo aptitude install ddd ↵
```
30. Install the **tofrodos** package (todos, fromdos utilities) to adjust how line ends are encoded:
```
sudo aptitude install tofrdos ↵
```
31. Install the **tmux** window manager if desired
```
sudo aptitude install tmux ↵
```
32. Install Eclipse IDE for C/C++ Developers. A necessary initial step is to install Java (`sudo aptitude install java7-runtime ↵`) The eclipse package can be found at www.eclipse.org/downloads. After downloading the tar ball, extract it from your login directory (it installs under a directory called 'eclipse')
33. Install the **zssh** Z-Modem over SSH utility if desired
```
sudo aptitude install tmux ↵
```
34. Install the source code under /home/SMCDev/SMC or load from Subversion:
```
mkdir –p /home/SMCDev/SMC ↵
svn checkout https://svn.liquidr.com/gliders/branches/SMC/sharable /home/SMCDev/SMC ↵
```
35. If using Eclipse, a few more steps are needed:
    a. The workspace is located at /home/SMCDev/SMC/eclipseWorkspaces (fill this path in when you start Eclipse)
    b. Add the following build variables under Window → Preferences → C/C++ → Build → Build Variables:

        makepp_build_location = /home/SMCDev/SMC/mpp2
        debug_path = common/Debug
        release_path = common/Release
        zoom_debug_path = common/ZoomDebug
        zoom_release_path = common/ZoomRelease
        z6debug_path = common/zzDebug6.0.0
        z6release_path = common/zzRelease6.0.0
        z6zoom_debug_path = common/zzZoomDebug6.0.0

z6zoom_release_path = common/zzZoomRelease6.0.0

c. Add the following variables under Window → Preferences → Run/Debug → String Substitution:

   debug_binary_path = /home/SMCDev/SMC/mpp2/common/Debug
   release_binary_path = /home/SMCDev/SMC/mpp2/common/Release
   z6debug_binary_path = /home/SMCDev/SMC/mpp2/common/zzDebug6.0.0
   z6release_binary_path = /home/SMCDev/SMC/mpp2/common/zzRelease6.0.0

d. Add the existing projects to Eclipse from File → Import…, then selecting General → Existing Projects Into Workspace, select the workspace directory (/home/SMCDev/SMC/eclipseWorkspaces) and then click Finish.

Chapter 3 – Working with the Development Environment

3.1 Software Development Directories

The following directories have special meanings for the build environment:

/home/SMCDev

This is the home directory for the SMC build environment. Timestorm has some dependencies on full paths, so it was decided to standardize on a single location to allow files to be version controlled.

/home/SMCDev/omniORB

This is the home directory for the omniORB development files. Two sub-directories (or soft links) are expected: "Ubuntu" is the home for the x86 Linux files used on the development machine, and "Zoom" is the home directory for the target files. omniORB files are installed from a provided archive.

/home/SMCDev/SMC

This is the home directory for the source code. Each project shown in Timestorm has a sub-directory that contains source code.

/home/SMCDev/SMC/eclipseWorkspaces

This is the home directory for Eclipse projects. Eclipse keeps some extra information in these directories that should not be checked in: the .cproject and .project files should be checked in.

/home/SMCDev/SMC/mpp2

This is the home directory for the *makepp* command-line build of the software.

The top level directory has instructions and scripts that are used commonly throughout the build.

There are sub-directories for each project, and sub-directories under those for each build type (Debug, Release, ZoomDebug and ZoomRelease.) Each project directory has a *rules* file that explains how to build the object files (.o) for the source.

The *common* directory contains the rules to build all of the executables and libraries. The full set of executables from the build is accumulated under *common/build-type*.

/home/SMCDev/SMC/runtime

This directory contains scripts and configurations that gather up executable files and make them available for installation.

/home/SMCDev/SMC/scripts

This directory contains useful scripts, notably the *makecard* script, which installs a root filesystem onto an SD card and optionally encrypts a partition.

/home/SMCDev/timesys

This is the home directory for installed toolchain and root filesystems. For example, at the time of this writing, there are three shipping toolchains: 6.0.0-3530, 6.0.0-3730 and 3.0. They are circulated as self-extracting shell archives. The latest toolchain is the 3.0 version, which is a glibc-

based 3730 image; the 6.0.0-3730 toolchain can be used by selecting the correct build configuration and is provided for backward compatibility. Although a 3530-based toolchain is provided for the 6.0.0 ucLibc-based Linux kernel and RFS, the 3530 is no longer a target for development.

/usr/local

As part of the installation process, some files are installed under /usr/local, including:

- Ubuntu build Boost library files (the version that matches what is installed on the SMC)
- makepp built from source

3.2 Software Version Control

The software is currently archived using Subversion at the following locations:

https://svn.liquidr.com/gliders/trunk/Embedded%20Systems/SMAC

This is the development trunk where engineers check in files on a regular basis. It is not guaranteed to be functional and it contains some files that may not be sharable with developers outside of LRI.

https://svn.liquidr.com/gliders/branches/SMC/sharable

This is a branch of the SMC source code that explicitly is sharable with developers outside of LRI.

https://svn.liquidr.com/gliders/tags/SMC

This is the home of shipping released source code. For example, https://svn.liquidr.com/gliders/tags/SMC/SMC%201.2.0%202012-07-19 is the location is the location of the source for the 1.2.0 build. The tag name is composed as "SMC *version yyyy-mm-dd*"

Note that Subversion drops hidden directories named as ".svn" in every directory that it controls. These directories store information about the files that were checked out. Subversion's global settings are stored under ~/.subversion.

3.3 Building the Software using Eclipse

The software can be compiled using one of four build recommended profiles:

- **Debug**: un-optimized, includes symbols, best for running with the debugger on Ubuntu (Boost 1.50.0 and omniORB 4.1.7)
- **Release**: optimized, best for execution time and size on Ubuntu (Boost 1.50.0 and omniORB 4.1.7)
- **ZoomDebug**: un-optimized, includes symbols, best for running with the debugger on the target (Boost 1.50.0, omniORB 4.1.7 and 3.0 toolchain)
- **ZoomRelease**: optimized, best for execution time and size on the target; this is the build that is sent to customers (Boost 1.50.0, omniORB 4.1.7 and 3.0 toolchain)

An additional 4 build configurations (zzDebug6.0.0, zzRelease6.0.0, zzZoomDebug6.0.0 and zzZoomRelease6.0.0) are provided for compatibility with the 6.0.0 kernel (Boost 1.46.1, omniORB 4.1.5 and 6.0.0 3730 toolchain)

To compile, select all projects on the left pane (Project Explorer), and right click on a project, select Build Configurations → Set Active from the popup menu and choose one of the build profiles. Then select a single project, right click on it, and select Build Project from the popup menu. Eclipse has been configured to use the **makepp** build system, so it infers build rules from the source code as well as from its configuration files.

3.4 Building the Software using "makepp" (command line)

makepp is a kind of make utility that performs source file scanning to understand dependencies and which does not require as much specification in order to perform a build. Note that makepp drops hidden directories named as ".makepp" in source code subdirectories where it stores the results of its dependency scans.

3.4.1 Maintaining the makepp build

The *rules* file in each project directory (i.e. mpp2/*project*) contains a list of modules to be included. The advantage of this is that C or C++ files in the source directory (i.e. /home/SMCDev/SMC/*project*) can exist, but are not included in the build; but it also means that the list of files must be maintained. When a new module files are added to the project, or when modules are removed, update the list; for example, the highlighted list should be updated (mpp2/GPSListenerLib/rules):

```
SRCDIR := $(SRCPARENT)/GPSListenerLib
DEP0 := SMACFramework EventLib
DEP := SMACIdl $(DEP0)
CXXFLAGS := -I$(SRCDIR) -I$(SRCPARENT)/$(DEP0) $(IDLCFLAGS) -
I$(CORBA_INSTALL)/include -I$(CORBA_INSTALL)/include/COS $(CXXFLAGSSTD) -c

MODULES := \
GPSEventSource \
GPSListenerPersistentSettings \
GPSListenerProgram \
GPSListenerSettings \
GPSListenerTestProgram \
NMEAParser

$(phony all):   $(MODULES).o

$(phony clean):
        rm -f $(MODULES).o

$(phony cleanall): $(TOP)/$(DEP)/$(BUILDDIR)/cleanall clean

%.o : $(SRCDIR)/%.cpp
    $(CXX) $(CXXFLAGS) $(input) -o $(output)
```

Presently, the same list appears in the section related to the artifact under mpp2/common/rules, and must be maintained:

```
############################################################
################ GPSListenerLib Library ################
############################################################

GPSLISTENERLIBMODULES := \
GPSEventSource \
GPSListenerPersistentSettings \
GPSListenerProgram \
GPSListenerSettings \
GPSListenerTestProgram \
NMEAParser
```

```
GPSLISTENERLIBDEP := SMACIdl SMACFramework EventLib

libGPSListenerLib.so:
$(TOP)/GPSListenerLib/$(BUILDDIR)/$(GPSLISTENERLIBMODULES).o
lib$(GPSLISTENERLIBDEP).so
        $(LINKER) $(LFLAGSLIB) -o $(output)
$(TOP)/GPSListenerLib/$(BUILDDIR)/$(GPSLISTENERLIBMODULES).o -L. -
l$(GPSLISTENERLIBDEP) $(BOOSTLFLAGS)
```

3.4.2 Building Everything

`cd /home/SMCDev/SMC/mpp2`

`makepp common/build-type/all` (for example, `makepp common/ZoomRelease/all`)

3.4.3 Cleaning Everything (removes top level build artifacts)

`cd /home/SMCDev/SMC/mpp2`

`makepp common/build-type/clean` (for example, `makepp common/Debug/clean`)

3.4.4 Cleaning Everything Recursively (removes all build artifacts)

`cd /home/SMCDev/SMC/mpp2`

`makepp common/build-type/cleanall` (for example, `makepp common/Release/cleanall`)

3.4.5 Force Rebuild of Everything

`cd /home/SMCDev/SMC/mpp2`

`makepp common/build-type/cleanall` (for example, `makepp common/Release/cleanall`)

`makepp common/build-type/all` (for example, `makepp common/Release/all`)

3.4.6 Build a Single Project

`cd /home/SMCDev/SMC/mpp2`

`makepp common/build-type/artifact` (for example, `makepp common/ZoomDebug/libSMACFramework.so`)

3.4.7 Force Rebuild of a Single Project

`cd /home/SMCDev/SMC/mpp2`

`rm -f common/build-type/artifact` (for example, `rm -f common/Debug/libSMACFramework.so`)

`makepp project/artifact/clean` (for example, `makepp SMACFramework/Debug/clean`)

`makepp common/build-type/artifact` (for example, `makepp common/Debug/libSMACFramework.so`)

Appendix A: Recommended Reading

C++

Boost Template Library for C++ Development: http://www.boost.org/

Standard Template Library for C++ Development: http://www.cplusplus.com/reference/

CORBA

omniORB 4.1 Reference: http://omniorb.sourceforge.net/omni41/omniORB/

Eclipse

Eclipse web site: http://www.eclipse.org

LogicPD

LogicPD makes "system on modules" for the ARM OMAP3530 (legacy) and the DM3730.

DM3730 module spec sheet: http://www.logicpd.com/products/system-on-modules/texas-instruments-dm3730-am3703-torpedo-som/

OMAP3530 module spec sheet: http://www.logicpd.com/products/system-on-modules/omap35x-torpedo-som/

Lua

Lua Reference Manual (Lua 5.1): http://www.lua.org/manual/5.1/

Programming in Lua: http://www.lua.org/pil/

Timesys

LRI's board vendor (LogicPD) supports embedded Linux though Timesys, who provide a custom build package for the kernel and the root filesystem.

Timesys web site: http://www.timesys.com/

Ubuntu

Download link for Ubuntu Version 10.04 LTS (legacy): http://releases.ubuntu.com/lucid/

Download link for Ubuntu Version 12.04 LTS (recommended): http://releases.ubuntu.com/precise/

Official Ubuntu Documentation: https://help.ubuntu.com/

Ubuntu Community Documentation: https://help.ubuntu.com/community/

Other links ...

makepp home page: http://makepp.sourceforge.net/

VirtualBox downloads: https://www.virtualbox.org/wiki/Downloads