Building verdex-based Framework for iRobot Create Using QuaRC

Chinpei Tang*, Pratik R. Maheshwari and Mark W. Spong

August 4, 2009

Abstract

The major purpose of this document is to show step-by-step the installation and setup required to build a verdex-based control system for iRobot Create. The major software required is MATLAB/Simulink-based equipped with a commercial software QuaRC. Since verdex is running under Open Embedded, a Linux operating system, some level of Linux operation details are also provided.

1 Introduction

Figure 1 shows the complete implementation framework that will be described in detailed in this document. The system consists of a host personal computer (PC) installed with MATLAB/Simulink and QuaRC, which can interact wirelessly with multiple iRobot Create through a very small form factor embedded processor called gumstix. While this guide attempts to illustrate the steps as detailed as possible, it is advised to fully understand the procedure before getting hands dirty since some of the steps might be outdated pretty quickly in the near future.

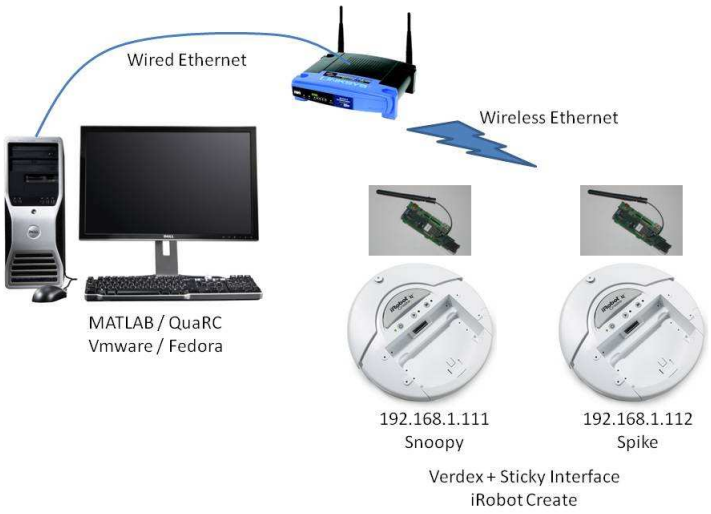


Figure 1: Complete Framework

This guide is organized in the following manner: we first briefly introduce gumstix in Section 2. It lists all the required components to work with the system as well. Section 3 then introduce how to "install" a Fedora Linux system on a Window machine to work with gumstix. Section 4 starts configuring Fedora for the gumstix. We will then briefly illustrate how QuaRC is installed

on the host PC in Section 5. Once we are done configuring the host PC, we will look into the hardware of the gumstix in Section 6 and its configuration in Section 7. We then configure the router to establish our networking system in Section 8. Finally, QuaRC will be implemented on iRobot Create in Section 9.

Throughout the document, we will follow these conventions as closely as possible: (a) the texts in `verbatim` are either commands in Linux or selections in Windows; and (b) the texts in sans are hardware's name.

2 gumstix

gumstix is a very small form factor embedded processor. The motherboard that this guide is based on is the verdex pro XL6P, which is based on the Marvell PXA270 XScale processor running at 600 MHz. It also comes with 128 MB RAM and 32 MB flash memory. The Input/Output (I/O) is provided by the expansion boards depending on the required functionality. The netpro-vx is to provide Ethernet wired and Wifi wireless networking. The console-vx is to provide serial communications with other peripherals, such as the host computer to perform cross-compiling. However, in our case, we got the Sticky Interface that can interface directly with iRobot Create through the DB-25 Cargo Bay Connector.

The major issue when working with gumstix is the lack of available documentation. There is no standard guide on how to purchase the required components to get started. Some of the components, for instance the serial null modem cable with DB-9 connector that is required is hard to get from other vendors but not included as a standard package. Note that the USB cable (mini-B to standard-A) does not allow the serial communication between the host computer and gumstix. However, there are two major web resources that most users relied on: (a) the gumstix developer site at `http://www.gumstix.net`, and (b) the gumstix forum at `http://www.nabble.com/Gumstix-f22543.html`.

To prevent future problems, the following is the list of the major components to purchase to get started with gumstix (for Verdex Pro, and see Fig. 2):

Mother Board The verdex Pro XL6P motherboard[1]

Networking Board The netpro-vx network board[2] to allow Ethernet and/or Wifi networking:

I/O Board The console-vx[3] for serial interface with the host computer. This is required, at the very least, to communicate with the host computer.

Serial Cable The serial null-modem cable with DB-9 connector[4]. Note that it must be *null-modem*. Also note that the serial cable comes with iRobot Create does *NOT* work! It has different "architecture" with the one from gumstix.

Power Supply The 5 V wall power supply[5]. This is very tricky. Some users complained that the wall power supply connector is not a standard one. So, it would be better to purchase it from the website.

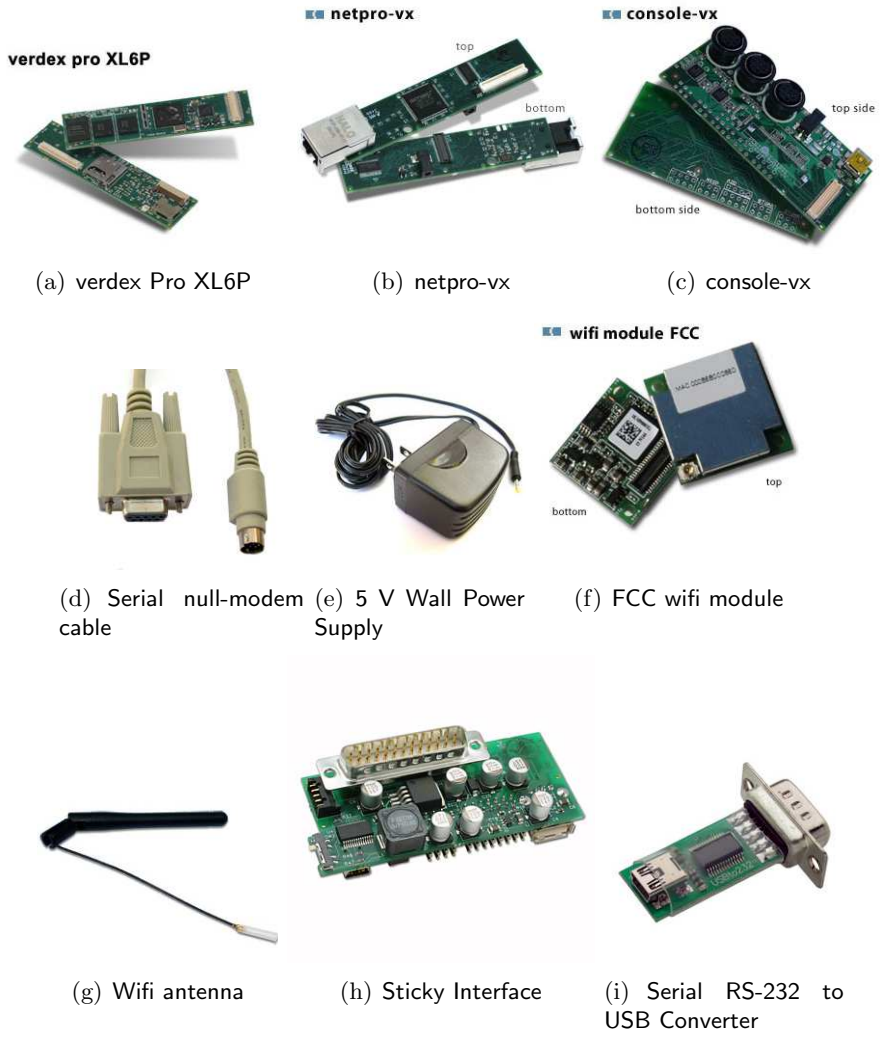
Wifi card The FCC wifi module[6].

[1] `http://gumstix.com/store/catalog/product_info.php?products_id=210`
[2] `http://gumstix.com/store/catalog/product_info.php?products_id=207`
[3] `http://gumstix.com/store/catalog/product_info.php?products_id=185`
[4] `http://gumstix.com/store/catalog/product_info.php?products_id=85`
[5] `http://gumstix.com/store/catalog/product_info.php?products_id=70`
[6] `http://gumstix.com/store/catalog/product_info.php?products_id=191`

These are the optional yet quite useful hardware:

Sticky Interface An customized board[7] that interface with iRobot Create directly with the gumstix.

USB to serial converter with USB cable If the host computer does not have an RS-232 serial port, this can be very useful to convert the port to USB so that it can interface with the host computer. It can be purchased from the Parallax[8], and the device is called Parallax USB to Serial (RS-232) Adapter.



(a) verdex Pro XL6P (b) netpro-vx (c) console-vx

(d) Serial null-modem cable (e) 5 V Wall Power Supply (f) FCC wifi module

(g) Wifi antenna (h) Sticky Interface (i) Serial RS-232 to USB Converter

Figure 2: Various gumstix modules, boards and accessories required

The general issue in working with embedded processor is that they do not have dedicated I/O devices and processing unit, even though they are installed with an operating system. Generally, we need a host computer that is able to develop codes so that it can be cross-compiled to produce executable that can be deployed in the Embedded Linux environment on gumstix. However, the

[7] http://www.elementdirect.com/wiki/tiki-index.php?page=Sticky+Interface+Documentation
[8] http://www.parallax.com

architecture between the i386 Intel and Marvell XScale ARM architecture are quite different in terms of the machine codes. Hence, it is easier to have both the host and embedded computer to run the operating system with same Linux kernel. Once the codes developed in the host computer, it can then be cross-compiled to gumstix using FTP-like software. However, as will be shown later, QuaRC will be able to eliminate Linux completely once the gumstix is configured properly with the QuaRC settings. Effectively, the advantage of using the software is to reduce the need of intermediate operating system, and the entire process can be done within Windows machine.

3 VMware and Fedora

First, Fedora[9] should be installed on a host computer. This is useful in the initial setup of the verdex. Fedora can be installed in a number of ways, but we are not going to get into too much details on this aspect. To minimize Fedora's invasion to the already installed Windows XP computer, we chose to install VMWare so that Fedora can be run as a *virtual machine* behind the Windows system. VMWare is a commercial software, and the particular software that we will be using is a free software called VMWare Player, which the latest version can be downloaded from `http://www.vmware.com/download/player`. The software should be able to access most of the resources on the PC. However, depending on the intermediate interface that we are going to use with the verdex, other VMWare software might be needed - we will discuss this aspect later.

After installing VMWare Player, we then download Fedora 9 (Cambridge) virtual machine image from `http://www.thoughtpolice.co.uk/vmware`. Place the image file in the VMWare Player installation folder (for good housekeeping). The image file can be opened using VMWare, which turn on and boot Fedora virtually on the computer. Start VMware, the window shown in Fig. 3 is then displayed. Click Open, search for the corresponding image file with extension `.vmx` to start the Fedora virtual machine. The next time when VMWare is started again, the link under the list of `Recent Virtual Machines` can be clicked to start the corresponding Fedora system. Note that all the settings that were done on OS within the virtual machine will be remained and saved on the same `.vmx` file. Hence, you can potentially working with different settings with different `.vmx` file, but this is not important.

After booting Fedora, obtain all the appropriate updates and install them. Change the desktop settings appropriately to prepare for further Linux package installations. While it is easy to work in the GUI mode like Windows, a lot of the Linux/Fedora operations rely on command lines. Go to `Applications - System Tools - Terminal`, the console terminal window can be opened, which require you to interact using the command lines. Most of the operations would require the "super-user (`su`)" to make[10]. You can access this mode by:

```
[<username>@locahost ~]$ su -
```

Enter the password as `thoughtpolice`, you will be working under the root:

```
[root@localhost ~]$
```

In the remainder of this guide, we will use `$` to indicate the command that you need to enter to execute specific operation. Whenever needed, we will also indicate which "user" are you by padding the information in front of the `$` shell.

The following steps will be summarized from:

`http://www.gumstix.net/Software/cat/Software-verdex-pro/111.html`

[9] One of the distribution of Linux OS.

[10] This is like the admin rights in Windows setting.

Figure 3: VMWare Player Startup Window

for our purposes. While the guide has been made as complete as possible, a lot of the steps were either unappropriate or outdated. Also, while this document is also making every effort to make the guide complete, it should be first completely read to get a general idea before proceeding. Most of the operations will require the knowledge of fundamental Linux commands. A general information about basic Linux commands can be found at: `http://www.ss64.com/bash`. You can also obtain instant help by doing:

```
$ man <command name>
```

in terminal to get the "manual" of the command.

We first need to install a number of packages required to interface with verdex, and they are:

- `gcc`
- `patch` (these first 2 are often bundled with other developer tools in the build-essential package)
- `help2man` (Centos 5 package available from `atrpms` repository)
- `diffstat`
- `texi2html` (texinfo on SUSE)
- `makeinfo` (texinfo on Ubuntu and Fedora)
- `ncurses-devel` (`libncurses5-dev` on Ubuntu)
- `cvs`
- `gawk`
- `python-dev` (`python-devel` on Fedora)
- `python-pysqlite2` (`python-sqlite2` on SUSE and python-sqlite on Fedora)

There maybe some packages for which name differs with different Linux distribution. So try to use appropriate name and make sure you install all the above listed packages. The way to install these packages is using the `yum` updater service:

```
$ yum install <package name>
```

For instance, to install `gcc`, simply do:

```
$ yum install gcc
```

This will install the `gcc` package. In this way you can install all the required packages. Also to download the gumstix source code, you will need also need `svn`, which is a version control package. Just do

```
$ yum install svn
```

During the installation, you may encounter error messages indicating that you might need more packages. Simply do `yum install` the missing packages will do the trick.

We will continue to prepare the host computer for verdex interface. We will rely on the serial communication mode to interact with verdex at the initial stage. We need to send files serially from the host to verdex. Two software can be used to do this, namely Kermit and Minicom. Minicom is a built-in software, so we will illustrate the Kermit installation process next. In fact, we found that Kermit is much easier to use than Minicom.

The Kermit complete installation procedure can be found here: `http://www.columbia.edu/kermit/ck80.html`. Download the `cku211.tar` or `cku211.zip` Unix complete package from the website to `kermit` directory (for good housekeeping again). To `untar` or `unzip` to unzip the zipped packages, we can use the commands, respectively:

```
$ tar xvf cku211.tar
```

or

```
$ unzip -a cku211.zip
```

in the specific directory. Then, do:

```
$ make linux
```

It will complete the installation. There will be one `wermit` binary file in the same directory where you made. Copy that the directory to `/usr/local/bin/kermit` by:

```
$ mv wermit /usr/local/bin/kermit
```

We are now ready to establish the communication between the host computer and verdex.

4 Fedora Configuration for verdex

We now download the `verdex` source code. Go to home directory in terminal by:

```
$ cd /home/User/
```

Create a directory called `gumstix`:

```
$ mkdir gumstix
```

Be ready to get the updated Open Embedded (OE) in the `gumstix` folder:

```
$ cd gumstix
```

Use `svn` to get the updated OE:

```
$ svn co http://gumstix.svn.sourceforge.net/svnroot/gumstix/trunk gumstix-oe
```

This process may take up 10 - 15 minutes depending upon the download speed. After this command completes you will have a copy of `gumstix` build system source code placed in your home directory in `/gumstix/gumstix-oe`.

Next, `gumstix OE` requires some environment setup in order to function properly. The following method is the most convenient (and highly recommended) to set up via your `bash` profile:

```
$ cat gumstix-oe/extras/profile >> ~/.bashrc
```

Now once the build environment is ready we can start using the verdex. However, let's prepare QuaRC before moving on while we are still working on the host computer.

5 QuaRC Installation

QuaRC comes with a comprehensive installation guide for installation to both Windows host computer and verdex target processor. While it is straightforward to install a software on a Windows machine, care should be exercise since the system needs to ultimately run in real-time.

First, make sure MATLAB is properly installed on the host computer. Type `ver` in the MATLAB command line to check if Real-Time Workshop is also available. Close MATLAB before installing. Then, we will need a compiler. While MATLAB itself comes with its own compiler, we suggest to use Microsoft Visual Studio. The free Visual Studio Express can be downloaded at: `http://www.microsoft.com/Express`. Make sure this is installed *before* installing QuaRC. If your machine has the full version of Visual Studio, it can then serve as the compiler for QuaRC. Then, do the following setup:

1. Right click on `My Computer` and select `Properties`. In the `System Properties` window, select `Advanced` tab.

2. Click on `Environment Variables`. In the `Environment Variables`, we need to create a new user variable.

3. Click on `New` in the `User variable for <username>` section, enter `MSSdk` in `Variable name:` entry. For `Variable value:`, it should be the location where you install Microsoft Platform SDK. Commonly, it should be `C:\Program Files\Microsoft SDKs`. Click on `OK` when you are done.

4. Open MATLAB and type `mex -setup` in the MATLAB command line and follow the instruction to select `Microsoft Visual C++ 2008 Express Edition` as your default MATLAB compiler.

The above steps will prepare the system to configure QuaRC to choose Microsoft Visual Studio C++ as the default compiler.

Once the above preparation is ready, close all the application and run QuaRC autorun installation file `autorun.exe` located in the `win32` folder. Follow the procedure to install the necessary

QuaRC components. During the installation, you should choose to configure both license manager and QuaRC. Restart the computer after the installation. QuaRC will start up by itself when Windows is started. You can open MATLAB to check if QuaRC is installed by typing `ver` in MATLAB command line. `Quanser Real-Time Control (QuaRC)` should appear in the list. You can also open Simulink and the `QuaRC Targets` hierarchy should also appear. This completes QuaRC installation on the host machine.

6 verdex Hardware

It is extremely important to discharge yourself before working with the hardware with bare electronic boards. We will first need the following boards to get started:

- verdex pro XL6P motherboard
- netpro-vx
- wifi module FCC
- console-vx

We can first connect the wifi module (and the antenna) to netpro-vx. Then, the rest of the boards should be connected in stack. From top to bottom, the order of the stack should be (see also Fig. 4):

1. netpro-vx (with wifi and antenna)
2. verdex motherboard
3. console-vx

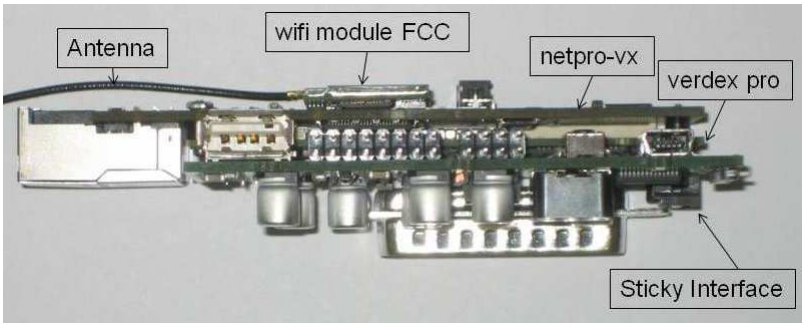


Figure 4: Connection of verdex boards with correct sequence.

6.1 Checking the life sign of gumstix in Windows

We can very quickly check the life sign of the verdex in the Windows environment by using the HyperTerminal program. Connect the null modem serial cable, one end to the RS-232 port of the host computer, another end to the *middle connector* of the console-vx. The following is adopted from `http://docwiki.gumstix.org/index.php/Connecting_via_Serial_-_Windows`. Go to Window's `Start - All Programs - Accesories - Communications - HyperTerminal`.

Name the new connection as `verdex`. Then select to connect to `COM1` (or appropriate serial connection). Do the following setup:

Bits per second: 115200

Data bits: 8

Parity: None

Flow control: None

Plug the power to the verdex, you should see the system boot up in the HyperTerminal. The username is `root`, password is `gumstix`. If the above is successful, it means the verdex is working fine. If not, restart everything, check the hardware connection and try again.

6.2 RS-232 vs. USB

Working with RS-232 can be an issue with VMware Player/Fedora since VMware Player does not support RS-232 communication. There are two workarounds:

Method 1: Using RS-232 to USB converter Since VMware Player allows accessing the USB ports, the simplest solution is to use an RS-232 to USB converter. After connecting the RS-232 to USB converter, you can see an additional USB icon at the bottom right side of VMware Player. Click on it and select `connect`. This then connects the converter to the Linux system. This can also be first tested by the procedures described in the previous section with appropriate USB `COM` port number.

Method 2: Using VMware Workstation You can download a 30-day trial VMware Workstation version which allows you to access all the resources on the PC. In this case, you can also access the RS-232 interface.

7 verdex Configurations

In this section, we will install QuaRC on verdex. To get started, we will need to replace the file system image with the one QuaRC provided. We will mainly follow the *Replacing the file system image* document from the gumstix support site.

We first boot Fedora and setup the serial connection between the host computer and the verdex. In this case, the verdex motherboard should be connected with the console board, and the serial cable should be connected to the host computer and the middle port of the console board. Unplug the power supply from verdex for the moment and do NOT boot it up yet! Open a terminal in Fedora in the host computer. Sign in as an `su`:

```
$ su -
```

Launch Kermit by:

```
$ kermit -l /dev/ttyS0
```

or

```
$ kermit -l /dev/ttyUSB0
```

(if you are using a RS232 to USB connector as a serial port). Now, you should see the `C-Kermit>` prompt. Then, set up the serial connection parameters to connect to verdex:

```
C-Kermit> take /gumstix/gumstix-oe/extras/kermit-setup
```

Depending on the functionality of Kermit, you may need to change to the directory by manually doing `cd` to `/gumstix/gumstix-oe/extras`, then:

```
C-Kermit /gumstix/gumstix-oe/extras> take kermit-setup
```

Finally, connect to verdex (do NOT power it yet!):

```
C-Kermit /gumstix/gumstix-oe/extras> connect
```

Now, connect the power supply to verdex. When connected and powered, you should see a message from U-Boot and counting down to boot the system, hit any key to stop the autoboot process and drop into U-Boot. At this point, you should verify that your U-Boot version is 1.2.0. You should also see the `GUM>` prompt.

We then first prepare U-Boot to receive a file download at RAM location at `a2000000`:

```
GUM> loadb a2000000
```

To return to the Kermit prompt in the host computer, simply do `Ctrl-\` and then `c`. Then, locate the `JFFS2` file:

```
Angstrom-gumstix-quarc-verdex-image-glibc-ipk-2007.9-test-20090427-gumstix-custom
-verdex.rootfs.jffs2
```

(the file should be in the `verdex` folder)[11] and send it to the UBoot of verdex:

```
C-Kermit> send Angstrom-gumstix-quarc-verdex-image-glibc-ipk-2007.9-test-20090427
-gumstix-custom-verdex.rootfs.jffs2
```

A screen displaying the file upload progress should display. Since it is about 23 MB, it might take about 45 - 50 minutes to transfer over the serial rate. Make sure you do not unplug the power during this process since the file is only stored in the RAM. Once the file is finished transferred, we connect back to verdex serially by:

```
C-Kermit> connect
```

If you cannot connect using `connect` command, then do `take kermit-setup` again.

Getting the new filesystem copied into the flash memory is a two-step process. First, the old file system must be erased. Then, the new filesystem must be written from RAM to flash. To erase the old filesystem, we do:

> This is very important, do NOT make any mistake, or else the entire flash will be erased and the hardware may become useless.

```
GUM> protect on 1:0-1
GUM> erase all
```

The first line protects flash sectors 0 and 1, which contain the U-Boot bootloader. This line is very important! If you omit it, your verdex will no longer be usable! The second line erases all of the flash except the 2 sectors we just protected. Now, we commit the new filesystem to the flash:

```
GUM> cp.b a2000000 40000 ${filesize}
```

[11] Hitting the Tab key may be extremely helpful.

This copies the new filesystem into flash (at address 40000). ${filesize} is a reference to the variable filesize, which was set by the file transfer.

After this, we use the above similar process to load the kernel:

```
GUM> loadb a2000000
```

Return to host computer by Ctrl-\ and c, transfer the file:

```
uImage-2.6.21-r1-gumstix-custom-verdex.bin
```

available in the verdex directory:

```
C-Kermit> send uImage-2.6.21-r1-gumstix-custom-verdex.bin
C-Kermit> connect
GUM> katinstall 100000
GUM> katload 100000
GUM> bootm
```

verdex is then rebooted. After all the booting process, QuaRC installation process would display. Since we have not sent the license file into verdex, we will first abort the process. To login to verdex, the username is root, and the password is quanser. We now send the license file to verdex. Note that we might need to rename the license file such that it does not have special characters or space in order to transfer through Kermit. The easiest way to rename a file in Fedora is locate the file using the GUI, right click on the file and click on Rename.

Return the prompt to the host computer again by Ctrl-\ and c, and send the license file to the root:

```
C-Kermit> send license.lic
```

Once this is done, we can reconfigure QuaRC on verdex. Connect to verdex:

```
C-Kermit> connect
```

Then:

```
root@quarc-gumstix:~$ quanser_license_registration -i QUARC
```

Then follow the instructions of configure both license manager and QuaRC. Keep the file in /home/root preferably and then locate the license file in the same folder. The configuration is successful if it shows:

```
root@quarc-gumstix:~$ The license has been configured successfully.
```

Reboot the verdex. QuaRC installation on verdex should be complete.

8 Networking Configurations

In order to establish communication between the host computer and the target verdex processors, they must be within a local area network. The simplest way to establish this network is the use of a (wireless) router. We use a Linksys router model WRT54GL. The remainder of the setup assumes the reader understands router settings well.

In this particular system, we setup such that the router is in DHCP mode. To ensure security, the SSID is set to be robotnet and not broadcasted. We also filter the MAC addresses of the wireless accesses - we will see how to obtain the MAC address of the verdex wifi module later. Most of the other settings remained default, but they are listed here for completeness sake:

Subnet Mask: 255.255.255.0

Gateway: 192.168.1.1

The IP address remains starting at `192.168.1.100`.

Going back to Fedora, we again access verdex using Kermit. In verdex, change directory to `/etc/network`:

```
root@quarc-gumstix:~$ cd /etc/network
```

Then use `vi` to edit the `interface` file:

```
root@quarc-gumstix:~$ vi interfaces
```

We want to be able to communicate with `verdex` wirelessly with fixed IP address so that we can identify the particular verdex. Hence, we should make the IP address static. Search for the line with `# Wireless interfaces`. Comment by `#` the lines with (first hit `Insert` key to activate editing in `vi`):

```
# auto wlan0
# iface wlan0 inet dhcp
# wireless_mode managed
# wireless_essid any
```

Go to the line with:

```
# Quanser ad-hoc wifi domain, as wlan0 cannot be ....
```

Uncomment and edit the following lines:

```
auto wlan0
iface wlan0 inet static
address 192.168.1.111
netmask 255.255.255.0
gateway 192.168.1.1
    pre-up /sbin/iwconfig $IFACE mode managed
    pre-up /sbin/iwconfig $IFACE key off
    pre-up /sbin/iwconfig $IFACE enc off
    pre-up /sbin/iwconfig $IFACE essid robotnet
    pre-up /sbin/iwconfig $IFACE txpower 100mW
```

The major change should only be the `address`, `netmask`, `gateway`, `mode` and `essid`. It is suggested to change the IP address to be "far enough" from `192.168.1.100` since the other computers might pre-occupy the nearer addresses. Also note that do not change the `eth0` (wired ethenet) to `192.168.1.100` setting, since the network somehow prioritizes wired mode, and correspondingly the wireless mode will not work together. If you need to work with wired mode, configure the router instead.

Once the `interfaces` file is edited, hit `Esc` to exit from edition mode, type `:w` to save changes, and type `:q` to exit from `vi`. Finally, do:

```
root@quarc-gumstix:~$ ifconfig
```

You can then see the MAC address appears. Take the one under `wlan0` category. Enter the MAC address to the MAC address filter list under `Wireless - Wireless MAC Filter`. You should do `Permit only PCs listed to access the wireless network`. Reboot the verdex by:

```
root@quarc-gumstix:~$ reboot
```

Once rebooted, you will see a blue LED on the netpro-vx card near the wifi module blinking fast and finally settled indicating that it is connected to the wireless network. Then go to Window's `Run`, enter `cmd` to open a DOS command prompt, enter `ping 192.168.1.111` to check if you can establish the connection to verdex. You can do `ping 192.168.1.111 -t` to ping continuously till you see the communication is established. Stop it by doing `Ctrl-c`.

Once the network is established, a lot of the communication can be much simpler. We can use PuTTY, which is available at: `http://www.chiark.greenend.org.uk/~sgtatham/putty`, to access to command prompt through wireless ethernet. Simply invoke PuTTY, enter the IP address `192.168.1.111`, you should be able to see the login prompt. After all these convenient settings, you can now send files over TCP/IP to the verdex in the PuTTY console using the `scp`:

```
$ scp <file to send> 192.168.1.111:/path/to/place/file in the target.
```

8.1 Optional Configurations

We can see that the bootup process might take a long time to search for the available external Bluetooth devices over different baud rates. To reduce this search, one way is to terminate the process completely from the boot-up process. To do this, go to the `/etc/init.d` directory:

```
$ cd /etc/init.d
```

Move the bluetooth configuration directory `bluetooth` to somewhere else (say the `home` directory) for potential future use:

```
$ mv bluetooth /root/home
```

Once this is done, reboot it:

```
$ reboot
```

You should experience much lesser booting time.

Another house-keeping that we can potentially do is to change the startup greeting of the verdex. This can be important since we might be working with a number of verdex, and we can use this to identify the window that we are working on. To do this, go to `/etc` directory:

```
$ cd /etc
```

The message can be edited in the file `motd` using `vi`:

```
$ vi motd
```

You should put as much information as possible so that you can identify the verdex that you are working on. Say:

```
My name is Snoopy.
My IP address is 192.168.1.111.
```

Again, rebooting it:

```
$ reboot
```

You should see the message you left in the file displays right after the booting process.

9 QuaRC on iRobot Create

We now in the position to have QuaRC to control the iRobot Create over the wireless network. Since we can now interact with verdex in the networking mode, we do not need the console-vx and the serial connection anymore. Unplug the power supply and disconnect console-vx board from the verdex motherboard. Substituting it with the Sticky Interface. Make sure they stay intact using the nuts, washers, supporters and screws provided to prevent disconnection during operation. Then, we are ready to install the entire verdex unit on the DB-25 connector on iRobot Create.

> WARNING - SUPPLY NO POWER and TURN EVERYTHING OFF BEFORE CONNECTING THE HARDWARE!

- Make sure that the Sticky Interface is in the OFF mode.
- Make sure the power supply is disconnected from verdex.
- Make sure the battery of iRobot Create is disconnected, and the power supply to iRobot Create is also disconnected.

Connect verdex stack on the DB-25 connector. Make sure it is fully connected. Connect the battery ONLY. Turn ON the Sticky Interface. You should see the LEDs on the unit are on. After a while, again, `ping` it.

Open MATLAB, open Simulink, create an `.mdl` model as shown in the Fig. 5. Most of the blocks can be found under: `QuaRC Targets - Devices - Third Party - iRobot - Roomba - Interfacing`. The `HIL Initialize` block can be found under: `QuaRC Targets - Data Acquisition - Generic - Configuration`. The model is a simple program to run the Demos on iRobot Create.

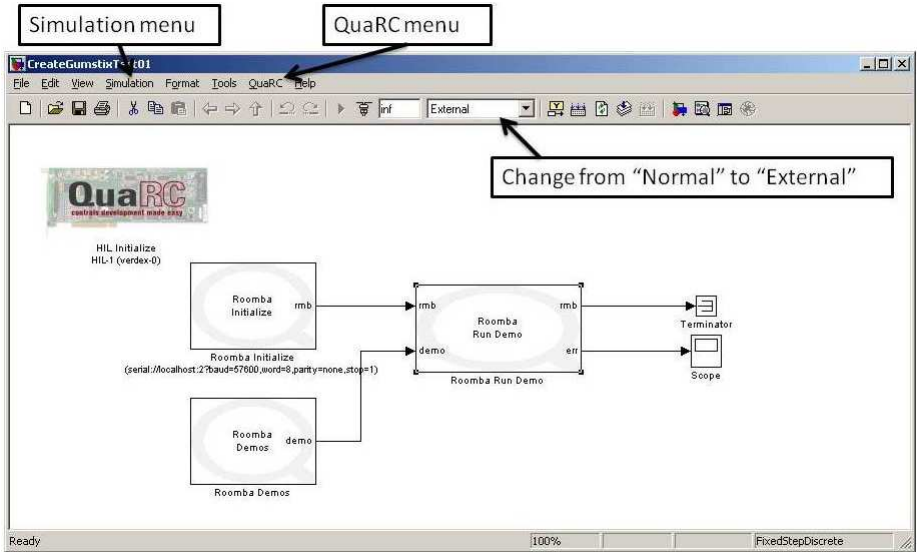


Figure 5: Simple Simulink model - Demo mode of iRobot Create

First, change `Normal` to `External`. In the `Roomba Initialize` block (see Fig. 6), change the parameter of `URI` of `Roomba to which to connect:` to:

```
serial://localhost:2?baud=57600,word=8,parity=none,stop=1"
```

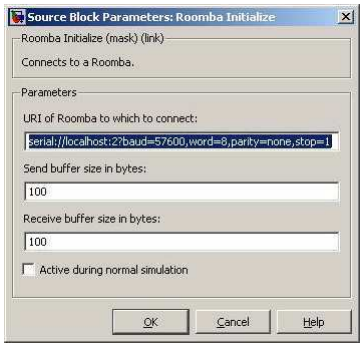


Figure 6: Roomba Initialize block settings.

This is to ensure that Port `#2` of `verdex` should be accessed. Change the sampling time in blocks `Roomba Demos` and `Roomba Run Demo` to 0.02.

In the `HIL Initialize` block (see Fig. 7), in the `Main` tab, under `Board type`, select `verdex`. Then, we go and set the QuaRC Preferences: In the Simulink model Window, under QuaRC

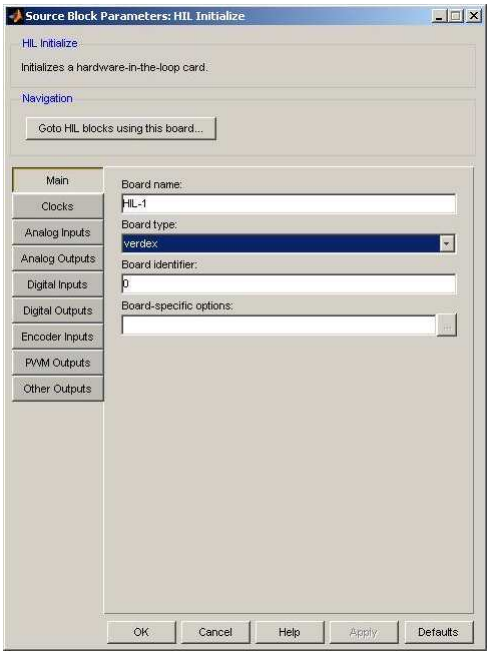
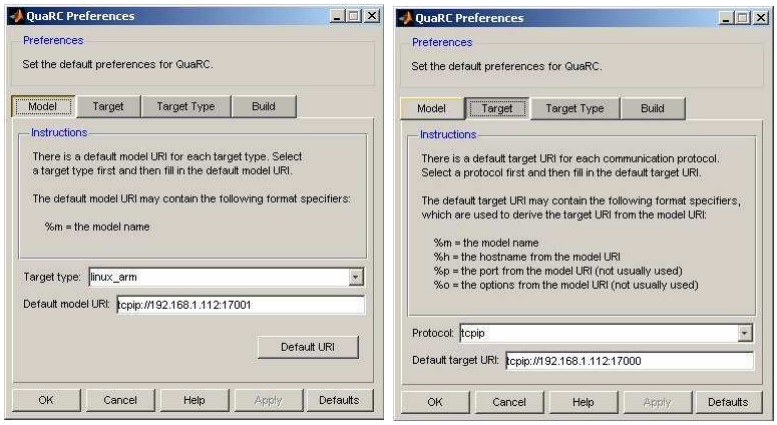


Figure 7: HIL Initialize block settings.

menu, select Preferences (see Fig. 12). In the `Model` tab (Fig. 8(a), select `linux_arm` to be `Target type`, and change the `Default model URI:` to `tcpip://192.168.1.111:17001`. Then, in the `Target` tab (Fig. 8(b), in the `Protocol` field, select `tcpip`. The `Default model URI:` should be `tcpip://192.168.1.111:17000`. Click `OK` when you are done.

Then, under `Simulation` menu (see Fig. 9), go to `Configuration Parameter`, under `Solver` tab, choose the solver type to be `Fixed-step` and solver to be `Discrete (no continuous state states)`. Change also the `Fixed-step size` to be 0.02. Click `Apply`.

Then, go to `Real-Time Workshop` tab (see Fig. 10, make sure that the `System target file` is `quarc_linux_arm.tlc`. Click `Apply`.



(a) Model (b) Target

Figure 8: QuaRC Preferences settings

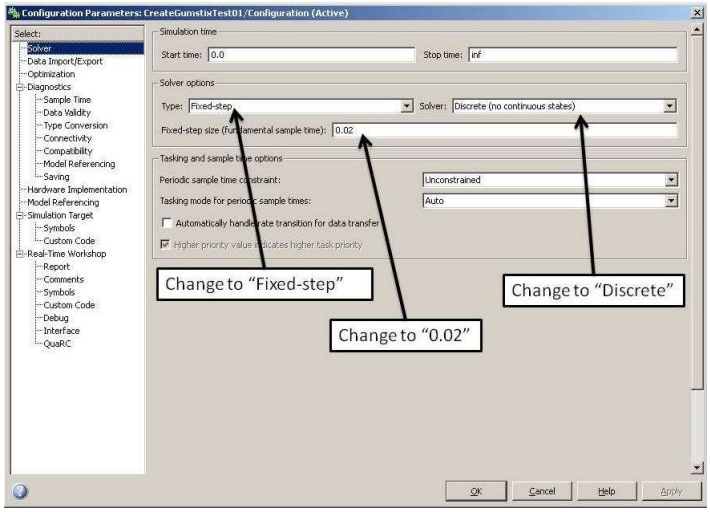


Figure 9: Configuration Parameters Solver settings.

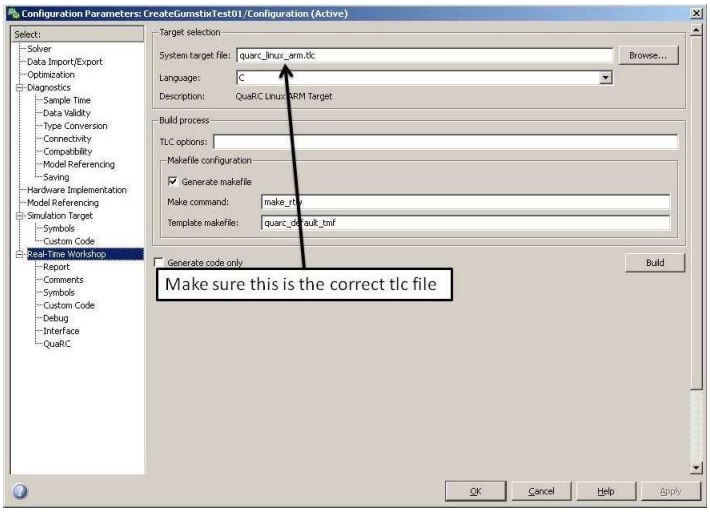


Figure 10: Configuration Parameters Real-Time Workshop settings.

Under `Real-Time Workshop - Interface` tab (see Fig. 11), in the `MEX-file arguments:`, change it to:

`'-w -d /tmp -uri %u','tcpip://192.168.1.111:17001'`

Click `OK`.

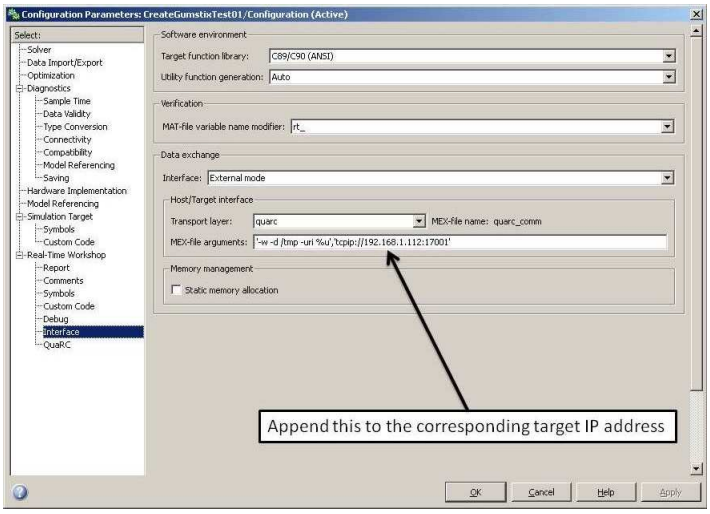
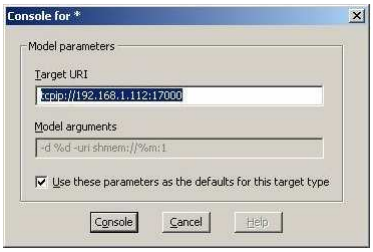


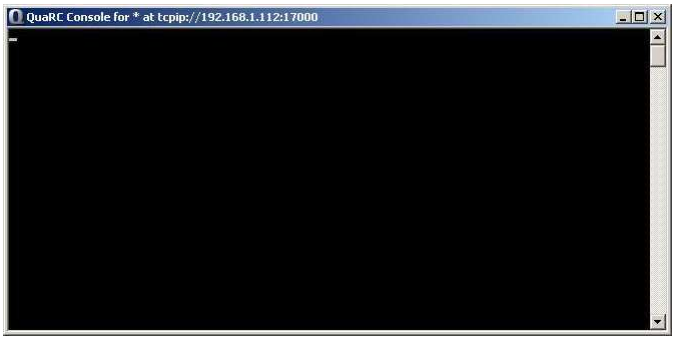
Figure 11: Configuration Parameters Real-Time Workshop Interface settings.

We should now be ready to run the system in real-time. Before we do this, it makes more sense to also understand what will be going on in the target side. To do this, we open a `Quanser Console`. Go to Window's `Start - All Programs - Quanser - QuaRC - Console for Target`, you will get the window shown in Fig. 12(a). Enter `tcpip://192.168.1.112:17000` in the `Target URI`, then hit `Console`. This will open a target console to monitor the compiling process remotely, shown in Fig. 12(b). Make sure the target is turned on, otherwise you will get the following error message shown in Fig. 12(c).

Return to the Simulink model, hit `Ctrl-b`, it will compile the Simulink codes and download to the target processor. You can also monitor the process on the console window. Finally, you should



(a) Model



(b) Target

(c) Target

Figure 12: QuaRC Preferences settings

see the download is complete. Hit the `Connect to Target` button (mouse over to see descriptions), see Fig. 13, you should see the model is running.

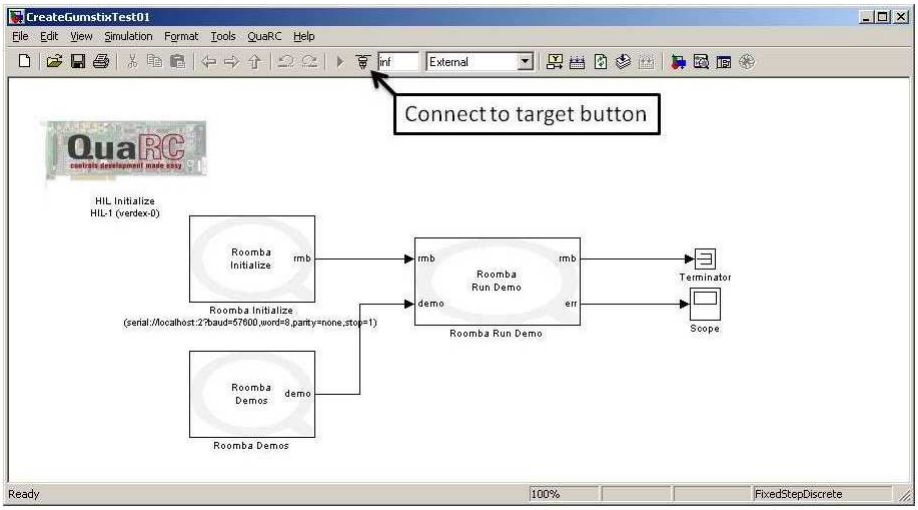


Figure 13: Simulink `Connect to Target` button.

Double click on the `Roomba Demos`, change `Demo` to other than `Stop` (see Fig. 14). You should

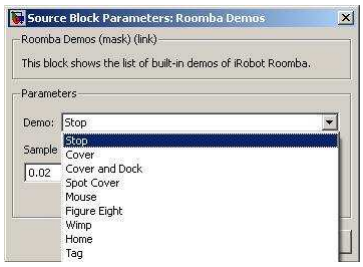


Figure 14: Roomba Demo block - change to anything other than Stop. You should see that the iRobot Create should start moving around.

see that it should operates. This shows that the entire process is complete.

10 Summary

Now that the hardware can be programmed using Simulink, the applications and integrations are seamless. QuaRC also provided many pre-programmed blocks that allow the user to interface with the iRobot Create in many different ways since it can access all the resources, including sensors and actuators, on board. It also allows integration of other third party accessories, such as a force-feedback joystick and driving wheel, and Wii, etc. This can then allow user to interact with the hardware with various different user interface. Future work include integrating force-feedback devices to allow user to control multiple robots.