Low-cost On-board Linux, Vision, Wi-Fi, and more for the Roomba Robotics Base

Tod E. Kurt

ThingM Design
1126 Palm Terrace
Pasadena, CA 91104
tod@thingm.com

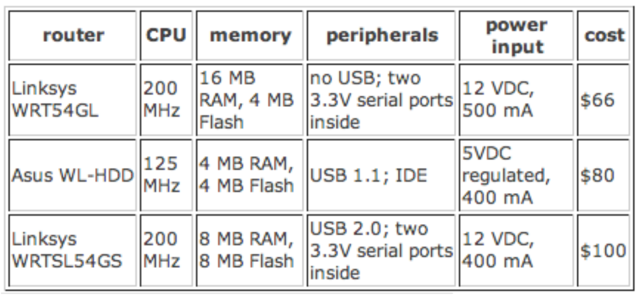
Abstract

The Roomba has become a rugged yet inexpensive peripheral robotics platform. With the addition of a sub-$100 consumer electronics wireless router, the Roomba can cut the tether and have an on-board embedded Linux system with built-in Wi-Fi and USB. The addition of USB allows the use of a wide-range of additional peripherals supported by Linux such as cameras, flash memory, and even spectrometers.

Motivation for Autonomous Roombas

The iRobot [1] Roomba Open Interface (ROI) [2] is a simple serial protocol that turns the robotic vacuum cleaner into a controllable robot base with basic sensors. Others [3] have used this interface to create robots as peripherals to larger, stationary computer systems running decision code. Much can be learned with this peripheral robotics approach. A next step would be the addition of on-board intelligence to allow local processing of high-bandwidth sensors, reduce processing loop lag, and experiment with the design challenges of a fully autonomous system. Adding an embedded computer system to a Roomba at first glance isn't cost-effective compared to other robotic systems, but the addition of few other consumer electronics devices, as hackable as the Roomba, can create an autonomous robot based on Linux with vision, audio input, and other sensors.

The release of the ROI (originally SCI) specification not only allowed this robust consumer robot to be hacked by hobbyists [4] and researchers but it's also the bellwether of a new trend of companies enabling alternative uses of their products. In the consumer electronics realm there are several others following this trend (or tacitly enabling it by not disallowing it). The most interesting devices for low-cost robotics are certain wireless routers. These devices, available for under $100, are capable of running an embedded version of Linux, have integrated Wi-Fi, Ethernet, USB and serial ports. Figure 1 shows the capabilities of three different widely available routers. These three devices have all been used as the brain for a Roomba-based robot. Of the three tested, the best performing is the WRTSL54GS, with the faster CPU and USB 2.0 interface.



router	CPU	memory	peripherals	power input	cost
Linksys WRT54GL	200 MHz	16 MB RAM, 4 MB Flash	no USB; two 3.3V serial ports inside	12 VDC, 500 mA	$66
Asus WL-HDD	125 MHz	4 MB RAM, 4 MB Flash	USB 1.1; IDE	5VDC regulated, 400 mA	$80
Linksys WRTSL54GS	200 MHz	8 MB RAM, 8 MB Flash	USB 2.0; two 3.3V serial ports inside	12 VDC, 400 mA	$100

Figure 1 Capabilities of three different wireless routers.

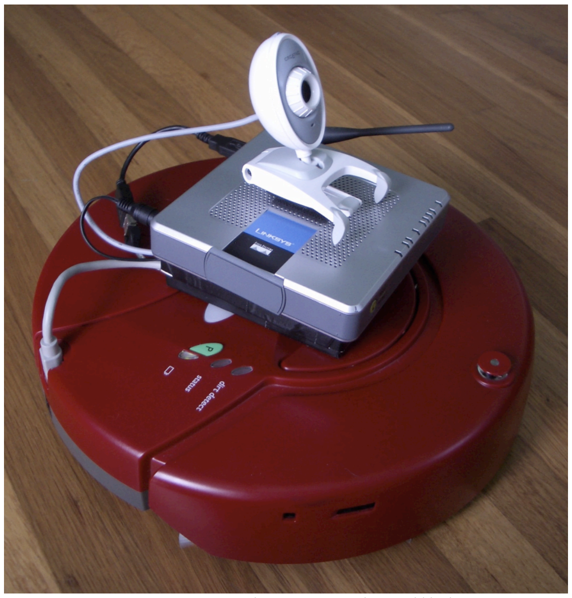


Figure 2 Autonomous Linux Roomba utilizing a WRTSL54GS, USB webcam, flash drive, and serial port.

Embedded Linux with OpenWrt

The above devices are only a few examples of "WRT"-style routers, so named because the original hackable router, the Linksys WRT54G. The WRT54G is based on embedded Linux and the terms of the GPL allowed hackers to easily inspect and modify its functionality [5]. Many commercially available routers use the same chipset and are amenable to similar hacking [6]. Over time a variety of special-purpose firmware projects have been created that run on these devices. One of the most advanced alternatives to the stock firmware is the OpenWrt project [7].

OpenWrt attempts to provide features similar to a modern package-based Linux distribution. It provides a built-in web server with CGI support, an SSH server, and most importantly, a package management tool "ipkg". With ipkg, one can add new applications, tools and kernel drivers. It can also install them without requiring a reboot of the router.

OpenWrt is loaded onto the prospective router by using the TFTP or HTTP firmware upgrade mechanism provided by the manufacturer. If desired, OpenWrt can be removed from the device and the original firmware reinstated using the same technique.

Once OpenWrt is installed, the router can either be configured as a wireless access point, a wireless client, or a member of an ad-hoc network. The latter option allows a mesh-network to be created, requiring no networking infrastructure. This is ideal if several similarly configured Linux Roombas would like to coordinate their movements.

Hardware Support in OpenWrt

Beyond the built-in support network devices such as the Wi-Fi and Ethernet chipsets, OpenWrt also has in its ipkg package repository drivers for various USB devices such as mass storage, digital still cameras, webcams, serial ports. Of course USB hubs are also supported meaning that several of these devices can be connected simultaneously. Multiples of the same device are also supported, enabling stereo webcam vision or the connection of several USB-to-serial adapters to control microcontroller-based sub-systems.

For example, Figure 2 shows a $70 Roomba equipped with a $100 WRTSL54GS running OpenWrt, a $30 webcam providing a real-time JPEG-compressed video feed, a $20 USB-to-serial adapter to connect to the Roomba, a $20 flash drive to archive the JPEG stream, and a $10 USB hub to tie it all together. Thus a complete semi-autonomous telepresence robot was constructed for approximately $250.

Software Applications in OpenWrt

Almost any standard console Linux application can be recompiled for use in OpenWrt, assuming it can fit in the smaller memory footprint of the router. The cross-compilation and packaging techniques are simple and well documented [8]. Installation of created software uses the same ipkg mechanism as for system packages. In addition to the system package library [9] there exists a growing repository of third-party packages [10] created by hackers worldwide.

Controlling the System

Because the default installation of OpenWrt contains a web server with basic CGI execution capability, a user-friendly mechanism to control the on-board intelligence installed is to create a series of dynamic web pages that update describing internal state and offering web controls to alter behavior. In the robot of Figure 2, a page was created that showed the real-time JPEG stream, showed telemetry data from all sensors, and offered buttons to control Roomba movement at both a low-level (turn right, stop) and a high-level (go towards bright light, retrace steps).

Alternatively, lightweight custom protocol servers and clients in either TCP or UDP can be run on the router.

References

[1] http://irobot.com/
[2] http://irobot.com/developers
[3] Dodds, Z. and Tribelhorn, B. 2006, *Erdos: Cost-effective Peripheral Robotics for AI Education*, AAAI, 2006.
[4] Kurt, T., *Hacking Roomba*, Wiley Publishing, 2006.
[5] http://www.wi-fiplanet.com/tutorials/article.php/3562391
[6] http://wiki.openwrt.org/TableOfHardware
[7] http://openwrt.org/
[8] http://wiki.openwrt.org/BuildingPackagesHowTo
[9] http://downloads.openwrt.org/whiterussian/packages/
[10] http://www.ipkg.be/