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. Remotely controlling it with a desktop PC is well understood, but limits the utility as a research robot. By providing a capable on-board intelligence, the Roomba can be made autonomous and enable the installation of additional senses and actuators. With the addition of a sub-$100 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 other science payloads.

Motivation for Autonomous Roombas

The iRobot Roomba Open Interface (ROI) [1] is a simple serial protocol that turns the robotic vacuum cleaner into a controllable robot base with basic sensors. The release of the protocol specification in late 2005 created an explosion of robotics experimentation accessible to anyone with a Roomba [2]. The physical interface of the ROI is RS-232-like asynchronous serial, which can be converted to true RS-232 via a serial tether [3] or accessed via a Bluetooth adapter [4]. The logical interface is based on a data format of 57.6 kbps, 8N1, with a command structure consisting of one to five byte commands used to interrogate and actuate the Roomba's capabilities.

Others [5] have used the ROI to create robots as peripherals to stationary computers 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. To allow the treatment of complex problems, the added system should implement a true multi-tasking OS and be able to accept a wide range of peripherals. Its only Roomba-related requirements are single serial port and a small form-factor.

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 common wireless routers.

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

Most low-power embedded computer systems with the desired attributes are too expensive to deploy in large numbers. In the same spirit of repurposing the Roomba as a robotics platform, certain consumer wireless routers, available for under $100, are capable of running embedded Linux. These routers have integrated Wi-Fi, Ethernet, and serial ports. Some have USB and IDE. Figure 1 shows the

capabilities of three different widely available routers used as on-board Roomba controllers. With only two serial ports, the WRT54GL has limited peripheral capabilities. The WL-HDD is the lowest-cost alternative to offer USB. Its need for a regulated power supply and relatively mediocre CPU and memory make it only suitable if an IDE interface is needed for a large datastore. Of the three, the WRTSL54GS is the best performing option, with a faster CPU than the WL-HDD and USB 2.0.

Embedded Linux with OpenWrt

The original WRT54G shipped with firmware based on embedded Linux and the terms of the GPL allowed hackers to easily inspect, modify, and improve its functionality [6]. Many consumer routers use the same chipset and are amenable to similar hacking [7]. A variety of special-purpose firmware projects have been created to run on these devices. One of the most advanced alternatives to the stock firmware is the OpenWrt project [8].

OpenWrt provides features similar to a modern package-based Linux distribution: a built-in web server with CGI support, an SSH server, and most importantly, a package management tool, "ipkg". Using ipkg, new applications, tools and kernel drivers can be added and removed at runtime, without requiring a reboot. OpenWrt is loaded onto the prospective router by using the standard firmware upgrade mechanism provided by the router. If desired, OpenWrt can be removed the same way.

With OpenWrt installed, the router can be configured as a wireless access point, a wireless client, or a member of an ad-hoc network. The latter option allows mesh networks to be created, requiring no networking infrastructure. This is useful for experimenting with emergent or flocking behaviors.

Hardware Peripheral Support in OpenWrt

Beyond the standard support for Wi-Fi and Ethernet network devices built into the router, OpenWrt also has drivers for various USB devices such as mass storage, digital still cameras, webcams, and serial ports. USB hub support means several of these devices can be connected simultaneously. Multiples of the same device is also possible, allowing stereo webcam vision or the connection of several USB-to-serial adapters to control out-board microcontroller-based systems.

For example, Figure 2 shows a Roomba Red ($70 for factory refurbished) 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 non-GUI Linux/Unix pogram can be recompiled for use in OpenWrt, assuming it can fit in the smaller memory footprint of the target router. The cross-compilation and packaging techniques are simple and well documented [9]. Installation of software uses the same ipkg mechanism as for system packages. Both the system [10] and the third-party [11] package libraries contain a growing list of standard Linux tools and device drivers.

Controlling the Roomba from OpenWrt. As in the desktop case, the Roomba is controlled via its ROI port, connected to the router with a USB-to-serial adapter and Roomba serial tether. Functionally this is the same as a PC-controlled Roomba. In fact, the Roomba appears as a serial port as in the desktop case and Unix/Linux code to control the Roomba on a PC [12] can be recompiled with no source changes to work in OpenWrt.

Controlling the System Remotely. Utilizing the fact that OpenWrt contains a web server with basic CGI execution capability, a web control panel can be created to directly control and inspect the Roomba or on-board peripherals, and to initiate higher-level behaviors. For the robot of Figure 2, a small CGI webpage was created that displayed a real-time JPEG webcam stream, 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). In lieu of user-based control, web pages could be replaced with XML services to allow machine parsing and control.

References

[1] iRobot Corp. 2006. "Roomba Open Interface", http://irobot.com/developers.
[2] Kurt, T. 2006. *Hacking Roomba*, Hoboken, NJ.: Wiley.
[3] Kurt, T. 2006. "Build a Roomba Serial Tether", http://hackingroomba.com/projects/build-a-roomba-serial-tether/.
[4] Kurt, T. 2006. "Build a Roomba Bluetooth Adapter", http://hackingroomba.com/projects/build-a-roomba-bluetooth-adapter/.
[5] Dodds, Z. and Tribelhorn, B. 2006. *Erdos: Cost-effective Peripheral Robotics for AI Education*, AAAI.
[6] Weiss, A. 2005. "WRT54G Story", *Wi-Fi Planet*, http://www.wi-fiplanet.com/tutorials/article.php/3562391.
[7] OpenWrt. 2006. "Supported Devices", http://wiki.openwrt.org/TableOfHardware.
[8] OpenWrt. 2006. "OpenWrt", http://openwrt.org/.
[9] OpenWrt. 2006. "Building Packages HowTo" http://wiki.openwrt.org/BuildingPackagesHowTo.
[10] OpenWrt. 2006. "OpenWrt Packages", http://downloads.openwrt.org/whiterussian/packages/.
[11] ipkg.be team. 2006. "OpenWrt Package Repository Tracker", http://www.ipkg.be/.
[12] Kurt, T. 2006. "Embedded Linux and roombacmd", http://hackingroomba.com/code/embedded-linux/.