

# LINUX: AN EMBEDDED OPERATING SYSTEM FOR MOBILE ROBOTS

Alan FT Winfield<sup>1</sup>

## Introduction

Developed at UWE Bristol's Intelligent Autonomous Systems (IAS) Laboratory for conducting experiments in collective mobile robotics, the LinuxBot is a proven and reliable wireless-networked wheeled mobile robot capable of supporting a wide range of sensors and actuators. The robot has a Linux-based software and communications architecture, which embeds TCP/IP networking tools including FTP, Telnet and Web servers. This approach, combined with a Robot Application Programmers Interface, provides an extremely powerful and flexible platform for researching and developing a wide range of mobile robot sub-systems: for instance navigation; sensing and sensor fusion; autonomous operation; remote tele-operation or multi-robot co-operation.

When the author embarked upon the first installation of Linux<sup>2</sup> onto a mobile robot in 1998, its success as an embedded operating system for experimental mobile robotics was by no means certain. Now, five years on, that decision has been strongly vindicated. A number of research projects have made use of the LinuxBot, and their success owes much to the choice of operating system. This paper will outline and discuss the rationale for using Linux as an embedded operating system for mobile robots.

## The LinuxBot

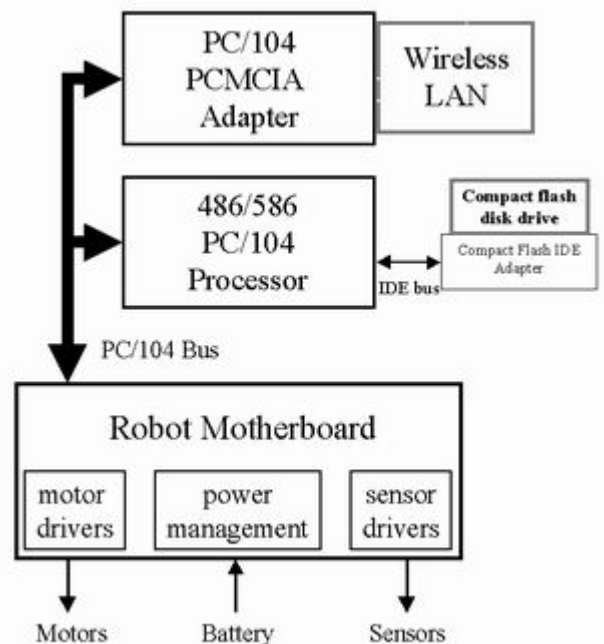
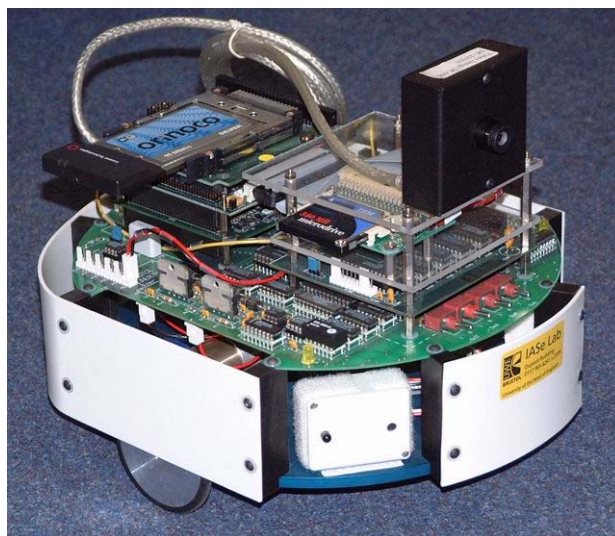


Figure 1: The LinuxBot and its Hardware Block Diagram

The LinuxBot is a differential-drive mobile robot purpose designed for conducting laboratory experiments in collective mobile robotics (multi-agent intelligent systems). Physically the robot is based upon a motherboard that integrates intelligent motion control and motor drivers

<sup>1</sup> Intelligent Autonomous Systems Laboratory, Faculty of Engineering, University of the West of England, Bristol, Frenchay, Bristol BS16 1QY. Email: [Alan.Winfield@uwe.ac.uk](mailto:Alan.Winfield@uwe.ac.uk). IAS Lab web pages: <http://www.ias.uwe.ac.uk/>.

<sup>2</sup> Or **GNU/Linux**, to give it its proper title.

with power management and basic sensor I/O. As shown in figure 1 a PC/104 bus interface allows an onboard robot controller to be ‘stacked’ onto the motherboard using commercial-off-the-shelf PC/104 cards [1]. The block diagram shows two PC/104 cards: a processor and a PCMCIA adapter for the wireless LAN card, but other cards may be added. The robot shown in the photograph in figure 1 additionally incorporates a PC/104 IEEE 1394 (Firewire) interface for the digital camera mounted on the robot. For a detailed description of the robot see Winfield and Holland [2].

### **The Choice of Linux as Operating System**

From an Operating Systems point of view the LinuxBot is a *PC on wheels*, albeit with limited resources. There is no keyboard or display, and the operating system can only be accessed via the wireless Local Area Network. The first generation LinuxBots employed a 25MHz 386 processor with 4MB RAM and an 80Mbyte solid-state IDE disk drive. Current robots use a 266MHz 586 processor with 32MB RAM and typically a 256MB compact-flash solid-state disk drive. The operating system thus has to meet two fundamental criteria:

1. Support for PCMCIA wireless LAN devices, TCP/IP networking protocols and associated programming tools (Berkeley Sockets API), and
2. Modest resource requirements (processor performance, RAM and disk-drive).

Initially the operating system was MSDOS. With a maximum of 2Mbytes of solid-state disk drive - at the limit of this technology in 1996 - we had little option but to use MSDOS. The addition of a proprietary DOS TCP/IP stack from FTP Inc provided a Berkeley sockets compatible Application Programmers Interface (API) for developing robot applications with TCP/IP network communications, so we could - in a limited sense - achieve our goal of wireless networking. Of course the single-tasking nature of MSDOS meant that we couldn’t use network tools like telnet, or ftp. Instead we had to write our own client-server applications from scratch, using the sockets API. Sockets programming is not trivial (in MSDOS or in Linux). Thus a good deal of ingenuity was required to set up wireless networked co-ordinated command and control of these MSDOS-based machines. Despite these limitations, single-tasking MSDOS enhanced with a TCP/IP stack was sufficient for simple multi-robot experiments in flocking and swarm behaviour.

By 1998, 80Mbyte IDE solid-state disk drives became available, opening up the possibility of breaking out of MSDOS. I might at this point claim that we underwent a rigorous process of evaluating alternative operating systems before selecting Linux, but that would be misleading. The fact is that we were running MSDOS (albeit with the problems I have already alluded to) when Linux presented itself as a viable alternative. Here (with a measure of post-hoc rationalisation) are the reasons that Linux meets our particular need.

*Integrated TCP/IP networking.* Wireless networking is a fundamental requirement for these robots and the fact that Linux integrates TCP/IP networking, together with telnet and ftp is a very significant advantage. The ability to be able to telnet ‘log-in’ to any individual robot via the wireless LAN, ftp upload code into a robot, and then start, stop or monitor robot applications code remotely (and while the robot is physically on the move) is a massive benefit when setting up complex multi-robot experiments. Although web interfacing was not a requirement for these robots, the fact that Linux also integrates http and a web server is a bonus and is allowing us to experiment with web based interfaces for robot tele-operation and tele-presence.

*Support for Wireless LAN and other devices.* The availability of device drivers for wireless LAN interfaces, and for PCMCIA, was of absolutely fundamental importance. In fact the open source code policy of Linux developers has also proven a significant advantage, since it enabled us to solve a number of obscure wireless networking problems which would have

been impossible without the ability to examine source code. The Linux newsgroups are another one of its strengths, having come to our aid on more than one occasion.

*Multi-tasking.* We badly needed a multi-tasking operating system, so that robot applications code can be run as a background task, while we are remotely monitoring the application in the foreground (say, via a telnet session). Actually there is another reason that multi-tasking is particularly attractive to roboticists. There is a powerful paradigm for robot control called behaviour-based robotics, in which complex robot activity is broken down into simple discrete behaviours. A low-level behaviour might be 'avoid walls and other robots'. This behaviour would be continuously active unless overridden by a higher-level behaviour, such as 'plug into battery charger in the wall'. From a programming point of view it makes sense to code each behaviour as a separate process, or task, using the Linux (or Unix) inter-process communication mechanisms to signal between behaviours. From a debugging point of view the ability to be able, independently, to start and stop (i.e. kill) individual behaviours is a significant advantage.

*Modest hardware requirements.* Clearly our robot operating system has to be able to run and give at least an adequate level of performance on a modest processor, with limited memory and disk resources. Arguably, Linux is the only operating system currently available that can meet the three core requirements of multi-tasking, integrated TCP/IP networking and wireless LAN support - and provide a rich set of compilers, tools and utilities - yet run comfortably in the limited hardware we have available on the LinuxBot. In practice performance has not been an issue. Even the 25MHz 386 based LinuxBots have comfortably run sophisticated experimental applications code.

More recently the resource issue has come to the fore again with the development of our uLinuxBot and LinuxBlimp robots. These designs make use of the remarkable DIMM-PC [3] as the core processor (see figure 2), which integrates typically 16MB RAM and 32Mbytes of solid-state disk drive. Although challenging, we have been able to build a version of Linux (based upon White Dwarf Linux [4]) which integrates all of the wireless networking components within the 32M disk-drive constraint to create perhaps the world's smallest wireless networked Linux robots. Linux is perhaps the only operating system in which it is possible to compile a customised kernel to include only those components required by the target.

### Real-time Issues

Linux is not, intrinsically, a real-time operating system and for some roboticists this would rule it out as a suitable candidate for mobile robots. In our experience of using the LinuxBots very successfully in a number of serious multi-robot experiments during the past five years the real-time issue has rarely been a problem. There are two reasons for this. Firstly, we have adopted the approach of, wherever possible, devolving critical I/O functions to either smart devices (for motion control, for instance), or PIC based sub-systems. For instance our infra-red localisation system uses a PIC micro-controller that interfaces with the main LinuxBot processor via one of the serial COM ports.

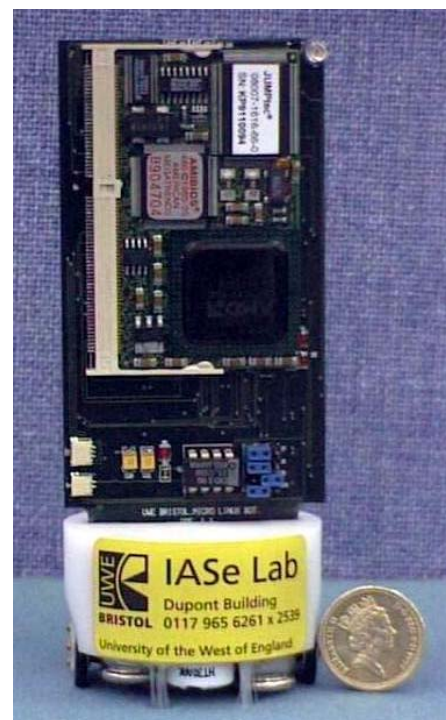


Figure 2: The uLinuxBot

Secondly, when we have needed to program time-critical functions on the main processor there has been a simple workaround. We have, for instance, made use of the POSIX real-time scheduling functions (i.e. *nanosleep( )*) to create short but precise timing waveforms on digital output pins [5]. For seriously real-time applications there are the Linux real-time extensions but, to date, we have not needed to use these. Thus, real-time Linux remains in the tool kit for such time as it is needed [6].

## Discussion

We have developed a Robot Application Programmer's Interface (RAPI); a functions library that gives the programmer access to the robot's sensors and actuators. The aim of the RAPI is to provide hardware independent functions and we are currently extending the library to incorporate uLinuxBot and LinuxBlimp robots so that (within the constraints of very different robot dynamics) the same applications code can be run across different robots. Thus, using this RAPI together with the Berkeley sockets API within Linux, researchers are able to develop, very rapidly, complex multi-robot experiments using C or C++ within a familiar program development environment. Indeed, perhaps the single most significant benefit of Linux has been that LinuxBot users have been able to focus maximum energy on research goals. Slackware [7], the chosen Linux distribution, has proven itself extremely reliable and robust, yet unobtrusive.

Linux was not designed as an embedded operating system and indeed, at the time the author first started to use Linux in this way, to do so was regarded by some as eccentric. The value and potential of Linux as an embedded operating system is, however, now much more widely recognised. The Embedded Linux portal [8] provides links to a growing body of relevant resources, including a number of Linux distributions for embedded applications.

Linux has, in all respects, met or exceeded our expectations as an operating system for experimental mobile robotics. Colleagues in the laboratory agree that Linux has been critical to the success of a number of multi-robot projects. Needless to say, Linux now features heavily in our forward plans for new mobile robot designs.

## Acknowledgements

Development of the LinuxBot has been a team effort, and I must especially acknowledge my colleagues Owen Holland and Ian Horsfield. I am also grateful to the BAe Systems Sowerby Research Centre for supporting the initial project, which lead up to the LinuxBot.

## References

1. PC/104 resources: <http://www.ampro.com/>
2. Winfield AFT and Holland OE, 'The Application of Wireless Local Area Network technology to the control of Mobile Robots', Microprocessors and Microsystems Journal (Elsevier), Vol 23, pp597-607, 2000.
3. DIMM PC: <http://www.jumpotec.de/>
4. White Dwarf Linux: <http://www.whitedwarflinux.org/>
5. Gallmeister BO, POSIX. 4: Programming for the Real World, O'Reilly & Associates, 1995.
6. Real-time Linux: <http://www.fsmlabs.com/>
7. The Slackware Linux home page: <http://www.slackware.com/>
8. The Embedded Linux portal: <http://www.linuxdevices.com/>