Second year internship report

Amaury Nègre

October 7, 2004

This internship was done from 05/07/2004 to 24/09/2004 at:

INRIA Rhône-Alpes
ZIRST
655 Avenue de l'Europe
Montbonnot
38334 Saint Ismier cedex
France

My objective was to evaluate the Pekee Robot and to install linux on this platform. My project leaders were Roger Pissard-Gibollet (research engineer), Gérard Baille (research engineer) and Daniel Simon (research scientists).

Contents

Word of gratitude

I would like to thank all the people who helped me during that internship, especially my project leaders Roger Pissard-Gibollet, Gérard Baille and Daniel Simon who trusted me, David Robert who helped me when I had difficulties, Soraya Arias and Jean-François Cuniberto who were always ready to help me, and Romain Lacroix, another student who helped me while I was installing linux.

Chapter 1

INRIA Rhône-Alpes Research Unit

1.1 The research unit

INRIA is a public establishment for research in scientific and technologic domain. It leads advanced research in computer science, telecommunication, multimedia, control, robotics, and signal processing.

Figure 1.1: INRIA Rhône-Alpes building

INRIA is under the control of both the Ministry of Research and the Ministry of Industry. The Rhône-Alpes research unit created in 1992 employs more than 400 people. INRIA Rhône-Alpes works with other research centers (CNRS, IMAG, UJF,...) in four main domains:

- Computer system and network
- Computer aided conception and creation
- Perception, simulation and action
- Modeling for complex phenomena

INRIA creates also companies (start-ups) as direct application of research subject. There are several companies which were created by the INRIA, for example Robosoft and Blue Eye Video.

1.2 The SED Team

In order to make experiments and to test their work, researchers need platforms (car-like robots, bi-steerable robots, biped robots,...) especially in robotics. All platforms that are needed by the INRIA Rhône-Alpes researchers and engineers are provided by the "SED" (Support Expérimentations & Développement logiciel). This is a team of about 10 engineers and technicians, which conceive, adapt and maintain all the hardware part of the robots.

1.3 Work context

I worked in to that team for my internship, although my work was intended to the team called "POP-ART". This team searches to solve the problem of the safe design of real-time control systems, and would like to use the robot "Pekee" to experiment their applications. I installed linux in this robot and wrote applications to use all its functions.

Chapter 2

Internship goal

2.1 Internship context : the Pekee Platform

In order to complete experimentations, the SED team acquired the Pekee Mobile Robot Platform. This is an open robotic development toolkit produced by Wany Robotics. This platform, intended for researchers, educators, and students in technology, provides features useful in the field of obstacle avoidance, embedded video, network, signal analysis, and artificial intelligence.

The main features of this platform are :

- Embedded computer, included a color video camera, (720 x 560 pixels) (cf Figure 2.3)
- Wi-Fi 802.11b and RJ45 Ethernet connections
- USB, keyboard, and mouse ports
- 2 differential drive DC motors with incremental encoders and integrated suspension
- Mitsubishi M16C family micro-controller running Wany robot control software
- Many built-in sensors (15 infrared distance measurement sensors, 2 gyrometers, 2 temperature sensors, 1 shock sensor, 1 light sensor)
- 1 serial port
- 4 x I2C connectors

Figure 2.1: Pekee 1

- Buzzer
- 2 x 12 volt rechargeable batteries (NiMH)
- Integrated intelligent charger
- Wany Open Parallel Platform™(OPP) bus (Cf Figure 2.3).

This platform seems to be a good toolkit to complete experimentations in the robotic field, but it doesn't include a linux support; all the software is only developed for Windows® which is an issue since the POP-ART's scientists work mainly with unix systems. Moreover, Windows® offers very low real-time performances.

2.2 Internship subject

The objective of my intership was at first to understand and to evaluate the Pekee's capabilities, and then, it was to install a linux system in this platform and to develop drivers and applications to use a maximum of functionalities, like motors and sensors control and video capture. I also tried to work with linux rtai (Realtime Application Interface) in order to build real-time applications.

Figure 2.2: Pekee Cartridge

Figure 2.3: Pekee Open

Chapter 3

Project realization

3.1 Linux installation

The installation of linux into the PC cartridge required several steps. First of all, I needed to build the toolchain (compilator, and c libraries) specific to the STPC processor. The next step was to build a linux kernel containing specifics drivers, and the last step was to make a distribution containing the common UNIX utilities.

3.1.1 Compile Toolchain

The Processor present in the Pekee PC cartridge is a STPC, based on a 486 architecture. So, we can't compile program with the classic gcc installed on my computer (using a pentium 4). Thus, I needed to compile gcc in this computer to build stpc programs, what is called a cross-compilator. First, I compiled binutils-2.15 which supplies the linkers and assembly tools, gcc-2.95.3, a c and c++ compilator needed to build the kernel and other programs, and finally glibc-2.2.4 which provides many essential libraries to build programs. This stage took me a lot of time because I tried many versions of those tools which were not compatible and some components could take more than half an hour to compile.

3.1.2 Linux kernel

The linux kernel is the heart of all linux systems, it manages the processes and all the devices. I installed the kernel linux-2.4.25 with some patches to improve real-time performances :

- low-latency : it decreases the latency time when a task is loaded

Figure 3.1: BusyBox logo

- preemptible : another patch which reduces latency
- hight-resolution-timer : it increases the timer resolution, this patch doesn't work with an stpc architecture because the timer seems to be incompatible

I also installed the kernel linux-2.4.25-adeos with rtai modules needed to program real-time applications. Those patches replace the linux scheduler with a real-time scheduler and runs linux as a low-priority process.

3.1.3 Linux utilities and Busybox

To work with a linux system, it is essential to have a minimum number of utilities, like a shell, functions to handle files, users managers, networking utilities, etc. As I had a few disc space and limited resources, I didn't choose the common GNU utilities but the BusyBox package. This tool is indeed the "Swiss Army Knife of Embedded Linux", it combines tiny versions of many common UNIX utilities into a single small executable and it was written with size-optimization and limited resources in mind. This package is used in many embedded systems and in the installation disk of many linux distributions like red-hat, debian or mandrake.

After the Busybox installation, I needed to write some configuration files and init scripts which initialize the system. I also had to install some other utilities like "ssh" to have all network support. To boot the system, I used the GRUB loader, which is a great tool but which requires a minimum amount of time to understand.

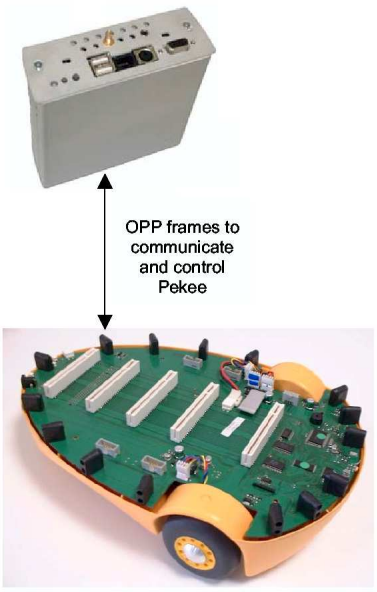

Figure 3.2: OPP bus

At this stage, I didn't have the Pekee PC-cartridge yet, so I tested all in an old notebook computer with a pentium 133 So I could test and develop my distribution, but I wasn't sure it would work on the Pekee PC. Finally, when I received the card I had no problem installing my distrib, and in spite of its delay, I didn't really waste time.

3.2 Drivers programming

3.2.1 OPP driver

To control the robot and to analyze sensor values, we need to communicate with the Pekee micro-controller, this communication uses the OPP (Open Parallel Platform™) bus (cf Figure 2.3). This bus has been created by Wany Robotics society and no drivers were available for linux, so I needed to make it myself. To get started, I read a program from Stephane Mocanu, a research scientist from the LAG; this program used the OPP bus in user space, I adapted and modified it to build a module which would work in the kernel space and be more efficient. The driver works as a character driver and allows to send and receive requests to the OPP bus using the standard read/write functions on a special device file.

When I finished this driver, I wrote a library to use this bus easily. The library provides functions to send and receive informations to the Pekee Microcontroller, and other functions to use the bus in multi-threads applications.

3.2.2 Camera driver

The Pekee PC-cartridge integrates a color video camera (720 x 560 pixels). To capture frames, a video input port is included in the STPC Microprocessor. Searching documentation of this device, I found the STPC Development Kit, a package made by STMicroelectronics, which contains a library to use all the STPC graphic functions. I began to write a program which could capture frames, but later, I realized that it would be great to have a "Video for Linux" compatible driver, so, after reading more documentation, I created such a driver. This driver was a minimal driver which only contains enough functions to capture frames and to be used by common video capture programs.

3.3 Application programs

3.3.1 Tracking program

In order to test the OPP communication, I wrote a first program which tracks a ball or other object with the infrared distance measurement sensors and follows it. This simple program used the OPP bus in the two sides, and so can check the driver.

When I had finished the camera drivers I could take care of another tracking program using the camera. This program made a color tracking to follow a red ball and used the infrared sensors to avoid obstacles.

For the moment, I can validate my drivers but all those programs worked on the Pekee cartridge and thus, I was quickly blocked by the cpu speed, I decided then to write a tcp server to transmit OPP commands and video frames by the network.

3.3.2 OPP and Video servers

To make it possible to control the robot by the network, I created a server which links the opp bus to the network, It receives commands through the network and transmits these to the OPP bus, and on the other side, each

OPP received frame are sent to the network. With this server, it is possible to control Pekee and to analyze sensors measurements with a distant computer. I also wrote a second server witch sends camera frames on the network. This server workes fine but, as images weren't compressed the data rate was very low (at best two frames/second), that's why I tried to use existing programs which encode and stream video data, but the STPC microprocessors were not enough powerful to real-time encode video, so I did not manage to obtain better results; moreover, I didn't have enough time to follow up my work.

3.3.3 Remote control program

To use those two servers, I realized a little remote control application. This program can be launch in a computer linked to the Pekee network. It draws the camera view in an windows; it sends a motion order to the robot when we press a directional key, and it also draws the infrared telemeter status in the windows. This application is not only an amusing program to play with the robot, but a great test of all the functions I wrote; furthermore, the POP-ART team wanted to experiment such programs.

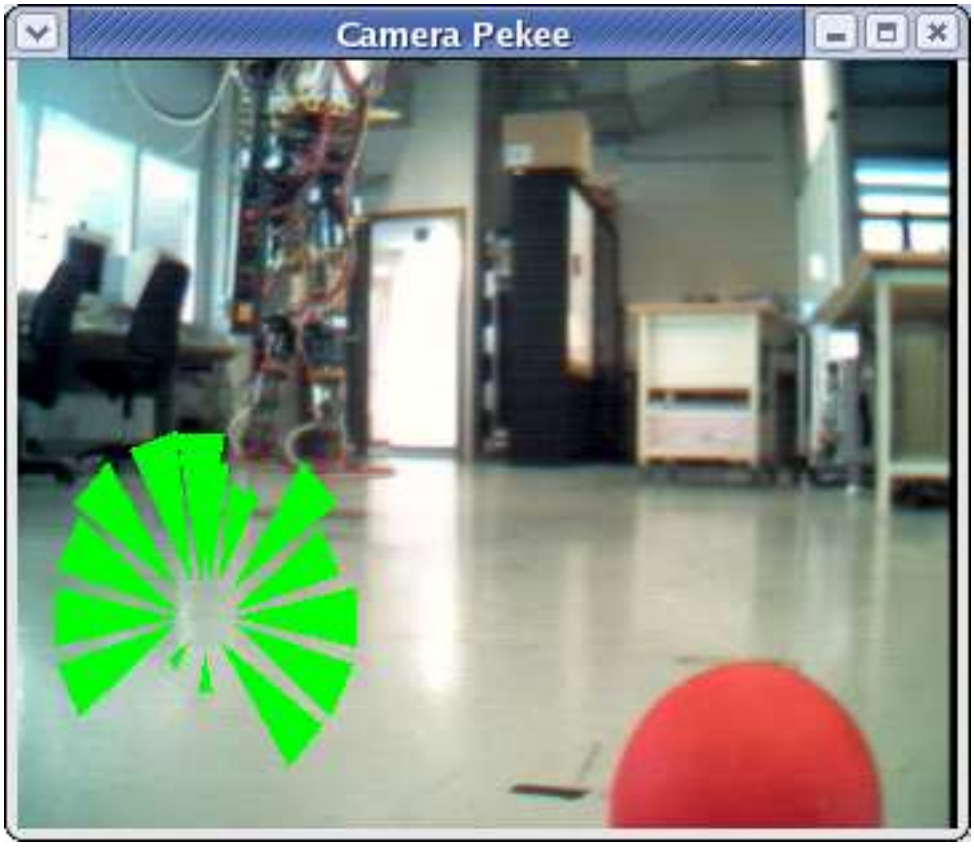

Figure 3.3: Remote control program

Chapter 4

Conclusion

This internship was very fruitfull to me because I had to cover many different fields. I also learnt new concepts and new ways of working.

First, as I installed an embedded Linux system, I could learn all the Linux structure, from the kernel to basic utilities. I also learnt to write device drivers and to work in kernel space, which change the way to program.

Next, I worked with real-time system, and even if I saw it a bit during System 2 classes, I learnt how such a system is built and how it is used.

In the image field, I understood how the low level is implemented and how to use or program a camera driver.

Finally, in the robotic field, I could test some obstacle avoidance, color tracking and remote control methods.

I realized that before programming, a long time was necessary to find and understand documentations; indeed, I spent between fifty and seventy percent of time reading manuals, web pages or forums. I also had to use other people's code, which I needed to understand and sometime correct, so I saw it is necessary to comment all code and to write manuals.

On the human side, I learnt how to work in a small team, and I often needed to meet expert person to resolve some problems. For example, I had to take contact with Wany Robotics society to have some information.

To conclude, I think that internship was very benefic to me as I learnt a lot, and it made me discover the research and work's world.

Bibliography

URLs

[1] INRIALPES
`http://www.inrialpes.fr`

[2] SED
`http://www.inrialpes.fr/iramr/`

[3] POP-ART
`http://www.inrialpes.fr/pop-art.html`

[4] Wany Robotics
`http://www.wanyrobotics.com`

[5] Busybox
`http://www.busybox.net`

Articles and books

[6] Karim Yaghmour, *Building Embedded LINUX SYSTEM*, O'Reilly & Associates, Inc, USA, First Edition, April 2003.

[7] Alessandro Rubini, Jonathan Corbet, *LINUX pilotes de périphériques*, O'Reilly, Paris (FR), Second Edition, 2002.

[8] Pierre Ficheux, *Linux embarqué*, Eyrolles, Paris (FR), Second Edition, 2003

[9] Herman Bruyninckx, *Real-Time and Embedded Guide*, K.U.Leuven, Mechanical Engineering, Belgium

[10] DIAPM-RTAI, *A Hard Real Time support for LINUX*
`http://www.rtai.org/documentation/reference/rtai_man.pdf`