

Linux Robotics Framework

Operational Concept Description

May 31, 2008

Contents

1	Scope	1
1.1	Identification	1
1.2	System Overview	1
1.3	Document Overview	1
2	Referenced Documents	1
3	Current System or Situation	2
3.1	Background, Objectives, and Scope	2
3.2	Operational Policies and Constraints	2
3.3	Description of Current System or Situation	2
3.4	Users or Involved Personnel	2
3.5	Support Concept	2
4	Justification for and Nature of Changes	3
4.1	Justification for Change	3
4.2	Description of Needed Changes	3
4.3	Priorities Among the Changes	3
4.4	Changes Considered but not Included	3
4.5	Assumptions and Constraints	3
5	Concept for a New or Modified System	4
5.1	Background, Objectives and Scope	4
5.2	Operation Policies and Constraints	4
5.3	Description of the New or Modified System	4
5.4	Users/Affected Personnel	4
5.5	Support Concept	4
6	Operational Scenarios	5
6.1	Scenario 1: Model car performing basic navigation	5
6.2	Scenario 2: Extending LRF with a robotic arm component	5

7	Summary of Impacts	6
7.1	Operational Impacts	6
7.2	Organizational Impacts	6
7.3	Impacts During Development	6
8	Analysis of the Proposed System	6
8.1	Summary of Advantages	6
8.2	Summary of Disadvantages/Limitations	6
8.3	Alternatives and Trade-offs Considered	7
9	Notes	7
9.1	Glossary	7

1 Scope

1.1 Identification

This document applies to the Linux Robotic Framework as described in the Software Project Proposal from Nias Digital (2008_NiasDigitalProposal.pdf).

1.2 System Overview

The Linux Robotics Framework (LRF) will provide a code-base and well defined API for robotics software developers from hobbyists to professionals. The framework will be efficient enough to run on the small, affordable AVR32 processor (which has significant limitations compared to a standard desktop PC).

The project is being sponsored by Nias Digital as both an entry into the robotics market, and a personal interest of the CEO. It is hoped that this framework will be publicised and that future maintenance will be done by a community of open source developers.

The framework will be released under an LGPL making it further development and usage available to all interested parties.

1.3 Document Overview

This document shall describe the clients requested functionality and our intended scope. It shall cover the current systems we are replacing/modifying. This document is for the use of project stakeholders including Nias digital staff, ANU project team members, and ANU staff involved with the project.

2 Referenced Documents

Software Project Proposal - Nias Digital 2008 (2008_NiasDigitalProposal.pdf).

3 Current System or Situation

3.1 Background, Objectives, and Scope

In recent years Linux has been used increasingly in the robotics field. The increase of computational power and ability has reduced the need for real-time systems, opening the door for general purpose operating systems such as Linux.

There are currently a number of highly competitive open source (GPL and LGPL) projects which are working toward providing a framework for developing both industrial and consumer level robotics systems. They each aim to provide a common base and software components to streamline the development of robots. These projects are highly competitive with their commercial counterparts and their accessibility makes them well placed to define and become industry standards. Businesses who adopt and/or sponsor such development will be well placed to benefit from their future development and adoption by the industry.

3.2 Operational Policies and Constraints

The current projects do not have strict targets for hardware requirements. Most projects are developed to run on similar hardware power to that found on desktop systems. The LRF project however is focusing on low-cost, low-power general-purpose processors which are rapidly shaping up as direct competitors to dedicated micro-processors for a wide range of applications.

3.3 Description of Current System or Situation

Systems developed on this kind of architecture need to implement everything from drivers for common components, to library functions for common tasks. There is a large overhead ensuring these are implemented correctly and completely.

Robotics is a combination of many engineering disciplines. Those working in the field have other concerns than the elegance of their source code (reusability, quality). If a framework was provided, their coding efforts could be focussed on their domain specific areas.

3.4 Users or Involved Personnel

The intended users will range from amateur individual hobbyists and open source teams through to professional commercial developers.

3.5 Support Concept

The LRF project intends to adopt the support concepts widely used through the open-source Linux robotics projects. Namely, the development of a dedicated website, the provision of extensive and useful API documentation for the software, including tutorials and examples of use and the provision of subscription mailing lists to allow interaction and mutual support between enthusiasts and developers.

4 Justification for and Nature of Changes

4.1 Justification for Change

Past and current projects have been very isolated in their design and implementation. Given progress made in the software industry, it is time that robotic applications were standardised and a distinct framework was created. This will speed up the development of the industry and open the market to many under-funded developers.

4.2 Description of Needed Changes

Necessary changes that need to be incorporated into current robotics applications include a standardised design, a standardised implementation procedure and suitable documentation. A distinct framework or structure needs to be maintained throughout current projects. The need for a universal, extendable robotics API stands out in terms of importance when we speak of needed changes. The API must encompass methods and functionalities which deal with general robotics. It must enable the user, regardless of their skill/knowledge level to implement his general robotics idea.

4.3 Priorities Among the Changes

A well thought out design with common interfaces is the top priority of this framework. As it is intended to be extensible, the organisation of components needs be considered before anything else. This design needs to be well documented so that it can be understood by users hoping to add their own implementations. Last would be a small collection of common component implementations and some example systems to demonstrate the use of the framework.

4.4 Changes Considered but not Included

Standardising the robotic software effort was considered as part of this project, but has not been included as it is beyond the capabilities of the team, and the constraints in which the project is being run. We will instead try to adopt what standards already exist in other projects.

4.5 Assumptions and Constraints

The primary constraint of the framework will be its focus on low-end, but consequently cheap hardware. This constraint may result in a increase in the development time for features and modules but will enable the framework to be accessible to the hobbyist using affordable retail grade hardware, while being of sufficient quality for it to potentially be used in industrial applications. The second major constraint, which will also increase the development time, is the requirement that the library be compatible across a range of processor architectures. It is assumed that there will be a community to further develop the framework upon the completion of the involvement of the ANU student group.

5 Concept for a New or Modified System

5.1 Background, Objectives and Scope

The intended objectives are to streamline and standardise the development of robotic applications for both research and product development. Current implementations take on a more ad-hoc approach with little re-usable code between projects.

5.2 Operation Policies and Constraints

The framework must run on affordable, low-level hardware such as the Atmel NGW100. It must be portable to several architectures, including at least the Atmel AVR32 and Intel x86. It should be built on the Linux kernel and must be extensible enough for developers to create drivers for their own components.

5.3 Description of the New or Modified System

The LRF will provide an API for users to quickly and easily develop and test robotic systems. The API will be strongly documented and designed with both the professional and the hobbyist in mind. The final product will be a package containing both source code and accompanying documentation.

The framework will utilise component-based architecture. The framework will provide the basis with which components are built, as well as provide some basic pre-built components for common robotic parts.

Where needed, the framework will define generic interfaces for classes of hardware. Possible classes that this might apply to could include, but are not restricted to, GPS devices, accelerometers and motor control units. However, use of existing interfaces, such as the Video4Linux API for video devices, will be strongly preferred over re-creating existing material which is open-source, easily accessible, well-documented and supported.

5.4 Users/Affected Personnel

The LRF is designed to be used by both robotics experts and hobbyists. Obviously, the current community of experts and hobbyists in the field of robotics generally, and open-source robotics software development in particular, will be affected by the emergence of a usable, accessible framework with the support of a its own community built around the project. But it is also expected that as the field continues to expand in size and interest to the wider computing community and society in general that this project will be a major and early port of call for newcomers with no previous participation in the robotics area but have some interest may also be affected with the simpler and more straight forward method of robotics programming offered by the framework.

5.5 Support Concept

As one of the goals of the LRF project is to make the LRF libraries an industry-wide resource and standard, the project will need a strong support strategy, one which will endure after the initial project is itself completed. In order to make the future development of the framework possible and self-sustaining the following strategies will be

employed:

Because the framework is open source, the development of an active self-sustaining LRF community participation will be a key part of the support system. Therefore the development of a functional website will accompany the framework and will be marketed/communicated to the wider open-source robotics community. This site will include a description of what the framework is, what it does, report on the current state of the software development and provide access to archives and copies of the framework. A mailing list will also run in tandem with the website to facilitate communication between members of the community.

The deliverables will contain a wide provision and range of support. This will include complete and accurate documentation to explain the usage and theory behind the design and code developed by the project. Ongoing support such as maintenance will be the responsibility of interested parties who join the community around the project.

To improve the robustness of the code in development and to test the design, several demonstration robots will be built using the framework. The demonstration robots will be made publicly available along with the framework to act as examples for new users to study.

6 Operational Scenarios

The emphasis on the LRF project is on development of a consistent, standardised set of functions and interfaces as a basis for future development rather than the creation of operational robots as such. Nonetheless the project intends to modify or create one or more operational, LRF controlled systems to demonstrate the basic functionality of the library and approach. The robots to be developed will probably be along the lines of the following scenarios, which demonstrate the strengths of the LRF approach.

6.1 Scenario 1: Model car performing basic navigation

Hardware: A remote control car with an Atmel NGW100 replacing the receiver.

The developer of the car will install LRF onto their NGW100 board using the directions in the provided documentation. They will select several LRF components to simplify the use of their hardware, such as a component which talks to the motor control unit, thus removing their need to know the exact details of using such a device. They will write the code to control the car using these components. This code might include straightforward instructions such as forward, turn left, accelerate etc, as well as flexible control-flow instructions such as if unable to detect forward motion, reverse back a distance and attempt to go around.

6.2 Scenario 2: Extending LRF with a robotic arm component

Hardware: A robotic arm controlled an Atmel NGW100 deploying telnet/wireless communication.

A developer wishing to control a robotic arm in a specific manner may decide to extend LRF and make a new component for the task. They will build the component using the provided component base and specifications provided by LRF. They may use an MCU component already developed for LRF, allowing their component to work with arms constructed using any type of MCU supported by the component without them having to code for specific devices. The development of the LRF website and community will provide opportunity for them to benefit from the input and experience of other developers and allow the wider community to benefit from reuse of their work.

7 Summary of Impacts

7.1 Operational Impacts

In the short term few operational impacts are anticipated, the project being concerned with laying the foundations of the library and its associated community. As the project is essentially working on a green field, new and existing users should experience little impact or difficulty transitioning to the LRF.

7.2 Organizational Impacts

The successful development of this framework will give NIAS the basis for future new product lines and applications involving the Atmel range of processors and increase the range of services they can offer clients.

7.3 Impacts During Development

During the development, this framework and the development of its associated community will effectively act as an advertisement working to gain interest and support for NIAS and other involved parties.

8 Analysis of the Proposed System

8.1 Summary of Advantages

The LRF project has a number of advantages over existing open source projects:

- The framework will run on cheaper and lower power hardware than any previous project studied.
- The framework will use standardised, generic interfaces for hardware devices.
- The framework will target a currently neglected section of the industry/hobby.

8.2 Summary of Disadvantages/Limitations

It should also be noted that the project will not provide a 'complete' range of capabilities but instead is focusing on providing a well-documented range of basic standardised functions and interfaces as a basis for future development.

8.3 Alternatives and Trade-offs Considered

Many similar projects currently exist in development and were considered for use in this one. Each of the projects listed below are well developed and have useful ideas for the design and implementation of such a framework but do not insist that their code meet specifications to be run on the hardware required by the LRF project.

Orocos is a component based, EU funded project to develop a set of libraries for use in industrial robotic applications. It has a real time layer, kinematics library and bayesian filtering library.

Orca/Hydro is similarly component based, and has a focus on creating standardized interfaces. Hydro is a library to complement Orca which implements algorithms common to robotics devices.

Player acts as a hardware abstraction layer for robotic devices, with control of robots performed through a network interface. This characteristic allows the control code of such devices to be developed in any language or operating system. It defines standard interfaces for common robotic hardware.

9 Notes

9.1 Glossary

Component a component represents a unit of software controlling some part of a robot. It could be anything from a button or motor to a drive system or robotic arm. Components are designed to be strung together in hierarchies so that more complex component systems can be built up from smaller ones.

Hardware abstraction layer Software that forms an abstraction from the specific details of the hardware, allowing code written against it to work on any hardware supported by the layer.

MCU Motor control unit. Hardware used to control a motor.