Towards a real-time software architecture for autonomous mobile robots

Overview with a case study

Erxa S.r.l.
ERXA S.r.l. is a private company that born and grew developing real-time software application for motion & processo control.

Main competencies of ERXA are

- Design and development of robotics cells control systems (from axis control to part-program interpretation)

- CAM applications for robotics cell off-line programming

Erxa S.r.l., Towards a real-time software architecture for autonomous mobile robots
OUTLINE OF PRESENTATION

- RTOS, RT Systems and Mobile Robots
- RT Software Architecture
- Macπ4Log Architecture
- Case Study
RTOS and RT Systems

What is the goal of a real-time application?

Every real-time application has to do, at least, the following:

• Read input devices
• Process the data
• Write output devices

Most applications will have more than two devices, like operator keyboards and graphics displays. Moreover some devices could require to be serviced at the same time!
RTOS and RT Systems

RT system
There exist different possible definitions to describe a RT System. The most useful for our needs are

A RT System is a system that reacts in a time-predictable way to time-unpredictable external stimuli.

A RT System is a system where the correctness of the computed data and the time they are produced have the same, fundamental, importance.
RTOS and RT Systems

There RT System definitions origin from the following key requirements

- Once an event has occurred an action has to be taken within a predetermined time limit; missing a deadline is considered a (severe) software fault (i.e. a RT system needs predictable/deterministic RTOS)

- Even if more than one event happens simultaneously, all deadlines for all of them should be met (i.e. a RT system needs pre-emption and multi-task capable RTOS)

*Deadlines arise from the underlying physical phenomena of the system under control*
RTOS and RT Systems

RTOS fundamental requirements

Other basic RTOS requirements are

- Thread priority with priority inheritance
- Predictable thread synchronization mechanism
  - Inter-task communication
  - Tasks synchronization
  - Resources lock/unlock
- Predefined latencies
  - Task switch
  - Interrupt latency (task to interrupt handler)
  - Interrupt latency (interrupt handler to task)
RTOS and RT Systems

RT systems can be further classified using its capability of respecting the required deadlines.

HARD RT System
Missing a deadline may reflect in catastrophic failures (ex: ABS – antilock braking system).

FIRM RT System
Tolerant to rare deadline missing. The delayed results of processing are unusable (ex: Food Production system).

SOFT RT System
Tolerant to deadline missing. The delayed results of processing degrade the performances of the system (ex: video conference).
Mobile Robots and RT Systems

An Autonomous (Mobile) Robot is a machine that operates in a partially unknown and unpredictable environment.

Then, leaving a robot free to move in some environment requires to give it the ability to appropriately react to its surrounding.

The surrounding coincides with the internal representation of the world a robot built using 1) perception, 2) a-priori information.
Mobile Robots and RT Systems

The concept of ‘appropriate reaction’ means in most conditions the robot must act with respect of happening external events exploiting some real-time reaction.

This is why, finally, we claim a Mobile Autonomous Robot is a/should be modelled as a RT System.

The external stimuli (i.e. typologies of) have to be modelled as the perception inputs or, more in general, as measures of the surrounding inducing change of behaviours.
RT-PLATFORM
Long term objectives

RT-Platform (aka RTP)
is a real-time software platform developed with the goal of to offer a long-living foundation made of modelling patterns, architectural patterns and software libraries for service robotics applications.

Nowadays a lot of specific robots together with their ‘specific’ software are made to solve/explore robotics problems.

Every specific robot has its sensors and actuators, its OS and sw libraries, its interfaces.

This is in contrast with parts changes/extension and software reuse/extension i.e. they miss both hardware and software modularity.
Architectural Design

Hardware Abstraction (OS Independence)
RT-Platform can execute on Windows CE/XP, Linux RTAI, VxWorks
A hardware abstraction layer (HAL) has been built on the top of OS calls.

Resources Abstraction
A layer offering templates and patterns that defines the basic bricks of
the RT architecture (tasks, inter-task communication resources,
diagnostic, timings, shared database).

Erxa S.r.l., Towards a real-time software architecture for autonomous mobile robots
Architectural Design

Hierarchical layered organization split in four levels

FIELD
Wraps and hides the access to low-level physical devices (actuators, sensors)

EXECUTOR (u-mission)
Manages and orchestrates the physical devices (through the FIELD interface)

MANAGER (mission)
Models and manages a real world entity or high level mission.

SUPERVISOR (task)
Manages and coordinates the managers at task/goal level
Architectural Design

Elements of the architecture can be mainly grouped as

ACTIVE OBJECTS
• event-driven fashion behaviour
• one task with an execution context
• provide commanded services in an asynchronous way.

PASSIVE OBJECTS
They can be think of as libraries providing synchronous services by function calls. The calls are executed in the caller context.
Architectural Design

Finite State Machine Modelling

The formal methods related to Finite State Machine models (FSM or Automaton) is a helpful concept that can help designers and programmers to develop their software models by reducing the complexity both in specification and implementation tasks.

Intuitively a robotics application can always be decomposed (i.e. represented) by a set of finite states and a state-transition function.

The same observation applies to every active component of the architecture.

Thus we assumed the modelling using FSM’s would be the most correct way to approach the design and implementation of architecture components.

Erxa S.r.l., Towards a real-time software architecture for autonomous mobile robots
RTP-Kernel

Prototype Task Library (PTL)
It’s, basically, a FSM defining the systemic behaviour of a RT Task that
- Represents an independent execution flow
- Definitely separates the systemic behaviour from applicative behaviour
- Defines and implements the event-driven infrastructure
- Defines and implements the hooks to trace and log the system activities

Systemic FSM
The ‘RUNNING’ state of a PTL task is the systemic state where the applicative control flow takes place.

This control flow can be modelled, it-self, by an applicative FSM reacting to incoming messages/events (event-driven FSM)
RTP-Kernel

Communication (COM)

Both the systemic (PTL) and applicative FSM use COM as support to message passing.
RTP-Kernel

Virtual Clock Library (VCL)

Lets access to timing functionalities such as

Interrupt context
• Periodic function call (ex: watchdogs)
• One-shot delayed function call

PTL context
• Periodic event generation (ex: periodic tasks)
• One-shot delayed event generation

Erxa S.r.l., Towards a real-time software architecture for autonomous mobile robots
RTP-Kernel

Real-time Database (RDB)

The ‘net’ composed by single automata can exchange information through a real-time database.

This allows to increase the decoupling between components (even on same layer) and to collect and store information about the state of the whole system.
RTP-Kernel

Booter (BTR)
To correctly start-up the system its components should be initialized and started in the right order eventually breaking the procedure because of some unexpected condition (configuration file).

Diagnostic (DIA)
It offers functionalities to signal, monitor and store anomalies of the system both for systemic and applicative layers.

Trace Library (TRL)
Trace are used both at debug level and production level to have a feedback about the works running on.
THE MACn4LOG ARCHITECTURE
The main goals of MACπ4Log have been to develop a mobile robotic platform with on-board vision systems, sensors and wireless communication in order to (mainly)

- build and update maps of both indoor and outdoor logistic spaces
- perform programmed and pro-active surveillance
- locate and, eventually follow, moving objects
- locate on the map specific items
- be able to achieve coordination with other fix/mobile platforms
Information Model

The information model was created starting from the functions required to the robotic system. It consists in the identification and description of entities, relationships and functionalities of the system.

The analysis of requirements followed a top-down approach in order to simplify the highest level functions in simpler ones and then identifying tasks, decomposing them in missions and the latter in u-missions.

This process let to 1) highlight the intrinsic hierarchical structure of the application and 2) isolate functional sub-components to be mapped onto the software objects.
The capabilities (tasks) required to the robot can be split with respect of a set of lower level, sometimes overlapping, missions.

<table>
<thead>
<tr>
<th>TASKS TO MISSIONS TABLE</th>
<th>Build/Update Maps</th>
<th>Surveillance</th>
<th>Pursuing</th>
<th>Locate</th>
<th>Coordination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safe path moving</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Navigation</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Mapping</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Inspection (items &amp; motion detection)</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Synchronization and communication</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
Supervisors perform the tasks of

- **Exploration for mapping**
  The goal of this task is to create the map of the whole logistic space.

- **Surveillance**
  The goal of this task is to be aware of changes in surroundings trying to identify and eventually pursuing the source of this change.

- **Identify and Locate**
  The goal of this task is to locate known items on the map.

- **Communicate**
  The goal of this task is to manages the conditions for an exchange of information with other entities of the team.
A higher level supervisor is the ‘Autonomy’. It commands the above mentioned supervisors in order to

- Establish the current task to be performed
- Bridge the commands received from an external Coordinator
MACn4LOG - Managers Layer

Managers perform the missions of

- Organize and maintains a set of local maps (Atlas)
  Which goal is to decompose the whole logistic map in a set of local maps.

- Safely move along path
  Which goal is to let the robot moves following a known sequence of targets avoiding dynamic obstacles and optimizing the trajectory.

- Inspect the surrounding
  Which goal is to identify items, members team or other moving objects in its surrounding

- Share data
  Which goal is to accomplish with a data exchange with another robot
MACπ4LOG - Executors Layer

Executors perform the actions of

- Navigate (estimate/evaluate the current location of the robot)
  The navigation makes possible to compensate for the odometry errors

- Wheels control

- Process vision data

- Special executors that act as ‘observer’ or ‘virtual sensors’
  The laser scans can be also used as proximity sensors while the scenes viewed can fused with laser/sonar scans etc.
MACπ4LOG - Field Layer

Field masks the specific characteristics of sensors and actuators by wrapping them with:

- Laser Range
- Axis (wheels) controller
- Camera
- Digital or Analog I/O signals

Now used to access information about the robot state (i.e. robot battery level)

Erxa S.r.l., Towards a real-time software architecture for autonomous mobile robots
MACπ4LOG – Complete view

The Applicative View of MACπ4Log Architecture

Erxa S.r.l., Towards a real-time software architecture for autonomous mobile robots
THE CASE STUDY
Case Study

The robotic hardware platform used by ERXA to perform the design, development and tests of software modules was composed by:

- Pioneer P3DX wheeled mobile robot
- SICK laser range
- Vision system (omni-vision camera)
- Netbook Acer Aspire One (running a Windows XP tailored OS)
- EPIA motherboard (running WindowsCE 6)
Case Study

The Functions

• Robot motion control

• SLAM

• Map Building

• Safe motion planning

• Synchronization with an external coordinator
Case Study

The Software Components Architecture

Erxa S.r.l., Towards a real-time software architecture for autonomous mobile robots
# Case Study

## The Software Components

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASC FIELD</td>
<td>interface to loose actuators (wheels)</td>
</tr>
<tr>
<td>LSR</td>
<td>interface (and driver) of SICK laser scan</td>
</tr>
<tr>
<td>CMR</td>
<td>interface (and driver) of camera</td>
</tr>
<tr>
<td>VSN EXECUTOR</td>
<td>interface and services of artificial vision</td>
</tr>
<tr>
<td>INT</td>
<td>interface and driver of WMR kinematics</td>
</tr>
<tr>
<td>NAV</td>
<td>interface and services of navigation</td>
</tr>
<tr>
<td>MAP</td>
<td>interface and services of map building</td>
</tr>
<tr>
<td>ERT</td>
<td>interface and services for synchronization</td>
</tr>
<tr>
<td>PPO MANAGER</td>
<td>manager of path planning functions</td>
</tr>
</tbody>
</table>

Erxa S.r.l., *Towards a real-time software architecture for autonomous mobile robots*
Case Study – Robot Motion Control

The control loop implemented in INT component and its FSM
Case Study – Robot Motion Control

The differential kinematics of a unicycle be given in cartesian coordinates

\[
\begin{bmatrix}
\dot{x}(t) \\
\dot{y}(t) \\
\dot{\theta}(t)
\end{bmatrix}
= 
\begin{bmatrix}
v(t)\cos(\theta(t)) \\
v(t)\sin(\theta(t)) \\
o(t)
\end{bmatrix}
\]

\[
\begin{align*}
v(t) &= \frac{\omega_R(t) + \omega_L(t)}{2}R \\
o(t) &= \frac{\omega_R(t) - \omega_L(t)}{D}R
\end{align*}
\]

it can be transformed in polar coordinates becoming of the form

\[
\begin{bmatrix}
\dot{\rho} \\
\dot{\gamma} \\
\dot{\delta}
\end{bmatrix}
= 
\begin{bmatrix}
-v \cos \gamma \\
\frac{v}{\rho} \sin \gamma - \omega \\
\frac{v}{\rho} \sin \gamma
\end{bmatrix}
\]

(ref. Aicardi, Casalino – Closed loop smooth steering of unicycle-like vehicle)

Erxa S.r.l., Towards a real-time software architecture for autonomous mobile robots
Case Study – Robot Motion Control

The Liapunov function

\[ V = \frac{1}{2} ( \rho^2 + \gamma^2 + k_3 \delta^2 ) \]

together with the control laws

\[ \nu = k_1 \rho \cos \gamma \]
\[ \omega = k_2 \gamma + k_1 \rho \frac{\cos \gamma \sin \gamma}{\gamma} (\gamma + k_3 \delta) \]

let the controlled system be G.A. stable i.e. \( \dot{V} \leq 0 \)

The choice of parameters \( K_j \) allows some further control tuning

Erxa S.r.l., *Towards a real-time software architecture for autonomous mobile robots*
Case Study – Vision

Artificial Vision (passive component / library)

It concerned with the identification in the (omni-)viewed scene of

• Vertical lines extraction
  combined use of segmentation and
  SIFT-like features extraction lets the base
  point of vertical lines becomes a robust
  visual cue

• Color blobs extraction

Erxa S.r.l., Towards a real-time software architecture for autonomous mobile robots
Case Study – SLAM

Navigation

The SLAM function is founded on EKF with corner/lines laser features

Erxa S.r.l., Towards a real-time software architecture for autonomous mobile robots
Data association relies on the construction of a correspondence graph and its exploration with a MAXIMAL CLIQUE algorithm.

A graph of all possible pairings landmark/feature is built by associating to each arc a weight depending on node types i.e.

- corner to corner ➔ corners distance
- corner to line ➔ point-line distance
- line to line ➔ intersection angle
Case Study – Features extraction

Corner & Lines Features Extraction (passive component / library)

Based on following main steps

1. Laser scan clusterization
2. Cluster scaling and smoothing
3. Dominant lines identification
4. Lines and corner interpolation
Case Study – Features extraction

Laser scan clusterization

\[
\Delta = |\rho_i - \rho_{i+1}|
\]

\[
\Delta_{MAX} = C_m + C_r \min \{\rho_i, \rho_{i+1}\}
\]

\[
\begin{align*}
\text{if } \Delta < \Delta_{MAX} & \quad \rho_{i+1} \in \text{current cluster} \\
\text{else} & \quad \rho_{i+1} \in \text{new cluster}
\end{align*}
\]

\(C_m\): compensation for sensor noise
\(C_r\): range variation of cluster

Cluster scaling and smoothing

Smooth and preserve second derivative (curvature) independently of the scale.

(ref “Multi-scale interest regions from unorganized point clouds” - R. Unnikrishnan, M. Hebert)

Erxa S.r.l., Towards a real-time software architecture for autonomous mobile robots
Case Study – Features extraction

Dominant lines identification

Gaussian weighted kernel in a 2-D Hough table. Selecting highest ‘peaks’ and their associated support points in order to identify dominant lines in a cluster. The quality of a line depends on size (cardinality) of support points set.

Corners and Lines Detection

Support points of selected dominant lines, in every cluster, are used to identify lines equations; the lines intersections are the corners.

Erxa S.r.l., Towards a real-time software architecture for autonomous mobile robots
Case Study – Features extraction
Case Study – Map Building

Map Building

- Polygonal description to simplify the map representation.

- Boolean operation let to take into account for slow changes of surroundings.

- Triangular mesh generation let to quickly generate a free space graph for the path planning.

Erxa S.r.l., Towards a real-time software architecture for autonomous mobile robots.
Case Study – Path Planning

Roadmap Generation

**RG:** Generate a roadmap representing the undirected graph $G = (V; E)$
- $V$ = set of all centroids of the triangles forming the mesh in the free-space
- $E$ = set of all edges that connect each node $V$ with all adjacent nodes

**MST:** using as root the centroid of triangle closer to the robot position, a MST covering the roadmap, is built with Dijkstra algorithm

**SP:** Using MST we may found the path in the roadmap graph joining $C_S$ (centroid closer to the start) and $C_G$ (centroid closer to the goal) plus two edges.

Erxa S.r.l., *Towards a real-time software architecture for autonomous mobile robots*
Case Study – Path Planning

Roadmap smoothing

- PP online uses as main attraction point an appropriate point onto the road map

- Reduce redundant paths and robot acceleration/deceleration phases.

LookAheadON ← near($P_r, T_c$)
if LookAheadON = true then
    $T_c \leftarrow \alpha \frac{d_1}{d_1 + d_2} (T_{k+2} - T_{k+1})$
else
    $T_c \leftarrow T_k$
end if

Erxa S.r.l., Towards a real-time software architecture for autonomous mobile robots
Case Study – Path Planning

Dynamic Obstacles Avoidance

- The path-planning is managed evaluating an artificial potential field. The function that describes this field is generally the sum of distinct components depending on attractive and repulsive forces (Rimon and Koditschek, 1992).

1. The current attraction point attracts the robot with force $F_a$

2. Repulsive force assigned to all the surround obstacles $F_r$

$$F = F_a + F_r.$$ 

$F_a = \frac{(x - p(x))}{d} \cdot \min(d, b).$

$F_r = - \sum_{i=1}^{m} \frac{y_i - x}{d_i} \cdot F(d_i).$

$$F(d_i) = \begin{cases} 
F_{\text{rep}} & \text{if } d_i < s \\
F_{\text{sat}} \cdot \left| \frac{d_i - DZ_i}{s - DZ_i} \right|^n & \text{if } s \leq d_i \leq DZ_i \\
0 & \text{if } d_i > DZ_i 
\end{cases}$$
Case Study – Synchronization

Synchronization Component

A team of exploring and surveilling robots that share information about the surrounding state and receive coordination commands from a control station has the main requirement of to share the same wall-clock time; i.e. exchanged data need to be labelled with a common reference time.

The Synchronization provides every robot with a high level protocol based on UDP/IP protocol able to distribute a common clock over the team.
Case Study – Synchronization

T²DMA protocol

The proposed protocol, referred as Tokenized Time Division Multiple Access, needs the presence of a single master and a set of slaves.

The exchanged messages (UDP datagram) contain information that let every manager reconstructs the master ‘absolute’ time and even foreseen it in case of lacks of transmission.

A prediction-correction filter can be provided in the manager to estimate the linear characteristic of t_{slave} versus t_{master}

\[ t_{slave} = \alpha t_{master} + \Delta_{eth} + \Delta_{offset} \]
DA TEAM ...

Erxa Administratives
Monica Bresciani, Stefania Anné

Erxa Designers and Managers
Marco Ricci, Alessandro Dosio, Nicola Smaldone, Dario Getto, Lorenzo Bonardi

Researchers
Giuseppe Zuffanti, Andrea Guadagnuolo, Vito Macchia

Consultants
Marco Pagliano

A special thanks to Marina Indri for her support in administrative issues (aka troubles).
THANK YOU FOR YOUR ATTENTION!

[:|]:

costantino.scozzafava@erxa.it

costantino.scozzafava@erxa.it

Erxa S.r.l., Towards a real-time software architecture for autonomous mobile robots