IIM3
Hybrid embedded operating system for ARM Cortex M3 / M4

Ioannis Deligiannis, George M. Papadourakis

Contents List

Introduction

Embedded systems are computer systems that are part of larger systems and they perform some of the requirements of these systems

Most of such embedded systems are also characterized as real time systems, which means that the real-time properties such as response time, worse case execution time, etc., are important design concerns.

REAL-TIME OSs

In general, an operating system (OS) is responsible for managing the hardware resources of a computer and hosting applications that run on the computer.

An RTOS performs these tasks, but is also specially designed to run applications with very precise timing and a high degree of reliability

To be considered "real-time", an operating system must have a known maximum time for each of the critical operations that it performs (or at least be able to guarantee that maximum most of the time)

General Purpose OSs

In contrast to real-time operating systems, the most popular operating systems for personal computer use (such as Windows) are called general-purpose operating systems

Operating systems like Windows are designed to maintain user responsiveness with many programs and services running (ensuring "fairness").

Comparison 1/3

Setting Priorities

Most operating systems (of any type) allow the programmer to specify a priority for the overall application.

General-purpose operating systems do not always follow these programmed priorities ,in contrast, real-time operating systems follow the programmer's priorities much more strictly.

Comparison 2/3

Interrupt Latency

Interrupt latency is measured as the amount of time between when a device generates an interrupt and when that device is serviced.

While general-purpose operating systems may take a variable amount of time to respond to a given interrupt, real-time operating systems must guarantee that all interrupts will be serviced within a certain maximum amount of time.

Comparison 3/3

Performance

Real-time operating systems may or may not increase the speed of execution, they can provide much more precise and predictable timing characteristics than general-purpose operating systems.

IIM3 Hybrid Embedded OS

The IIM3 Hybrid Real-Time Operating System (RTOS) provides real-time performance within a small, configurable footprint.

It manages an unlimited number of application tasks, semaphores, mutexes, event flags, message queues, timers and memory partitions

Architectural Design

The design of the IIM3 architecture follows the microkernel principles.

The OS kernel comprises only the most necessary functionalities, like the scheduler, resource management etc., whereas all additional functions run outside the kernel as separate components, using only a predefined kernel interface

Scheduler

The Scheduler is the most important part of the Thread Management.

It implements a real-time capable scheduling scheme.

Allows to run non real-time threads in “parallel” to real-time application(s).

Thread

The Thread Management function group allow defining, creating, and controlling thread functions in the system.

A Thread can be in the following states:

- Thread.State.INITIALIZING
- Thread.State.RUNNING
- Thread.State.WAITING

Services

- A separated category of tasks.
- Called as hard real-time threads.
- Collecting data from sensors
- Keep timers and other background activities up-to-date.

Synchronization

Components for thread synchronization are provided by the Mutex Management module

The Mutex Management function group is used to synchronize the execution of threads.

This is for example used to protect access to a shared resource, such as a shared memory image.

Benefits

As a result of the presented architecture, the OS kernel can provide very detailed runtime information about its state .

The dynamic capabilities of the IIM3 architecture enable the implementation of sophisticated applications and more flexible services.

Prototypical Implementation

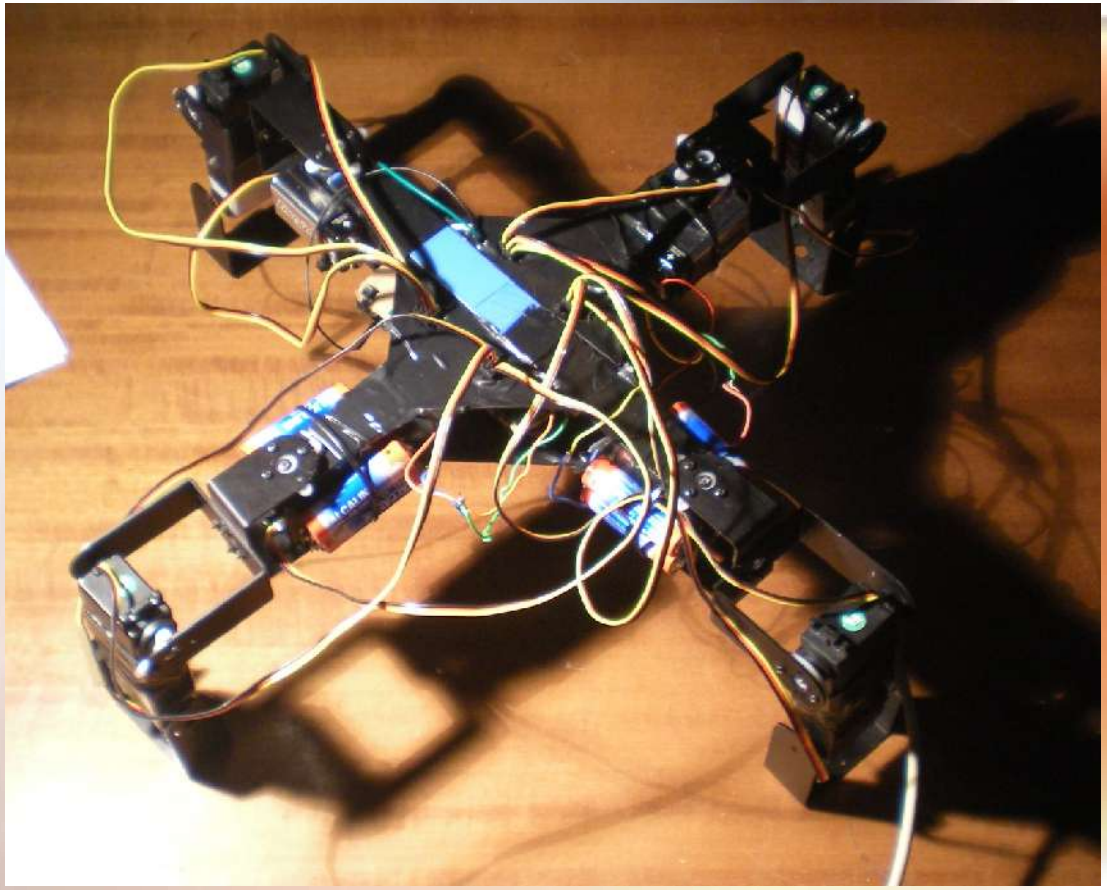
A first prototypical implementation of IIM3 was performed on the Cortex M3 microcontroller ,afterwards we tried to customize the kernel in order to use the Cortex M4 microcontroller. .

Both ARM Microcontrollers used in a prototype Quadruped robotic application.

Prototypical Implementation

IIM3 Robot v01

- Dynamic Movement
- Simple obstacle avoidance



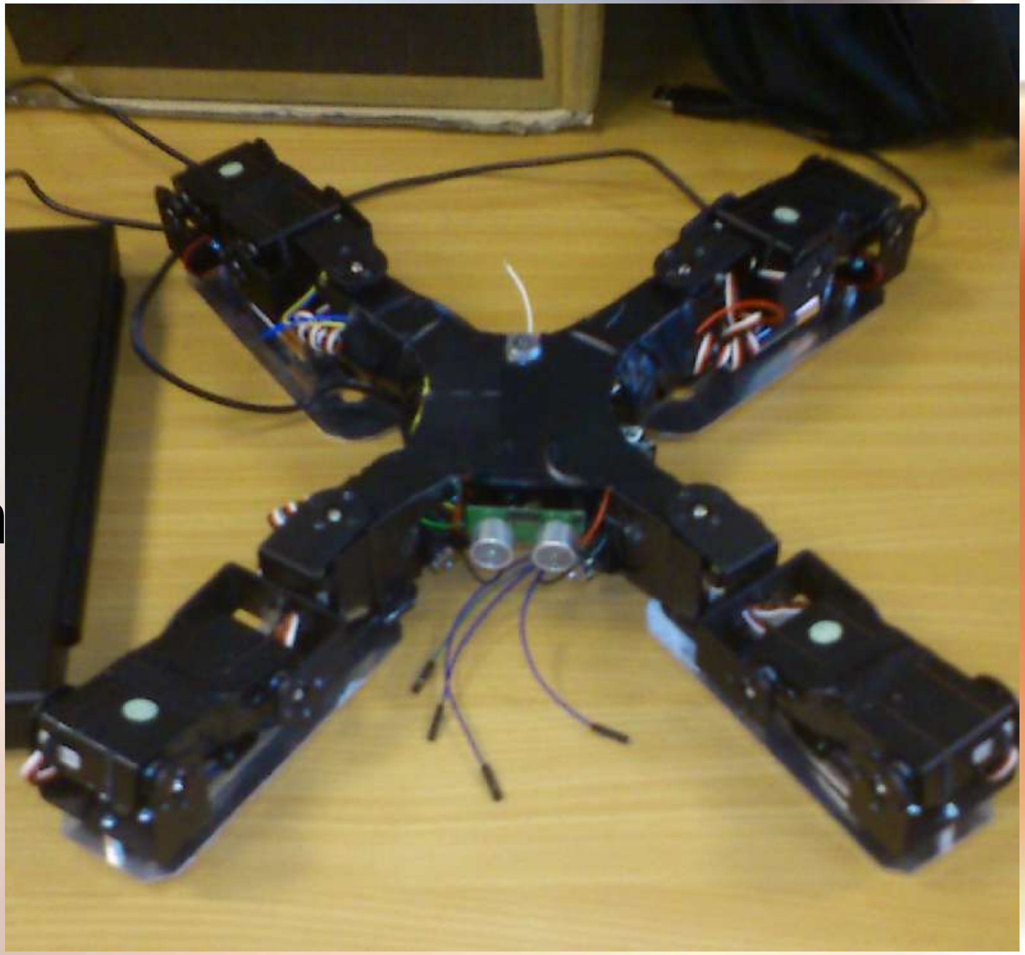
Prototypical Implementation

IIM3 Robot v02

•Dynamic Movement

•Sophisticated obstacle avoidan

•Area Mapping



Prototypical Implementation

IIM3 Robot v03 is currently under development.

- High-end sensors and mechanical parts
- Dynamic Movement
- Object Recognition and Avoidance
- Mapping of areas inaccessible by humans.

Thank you