快快樂樂設計
嵌入式即時作業系統

Nov 4, 2007

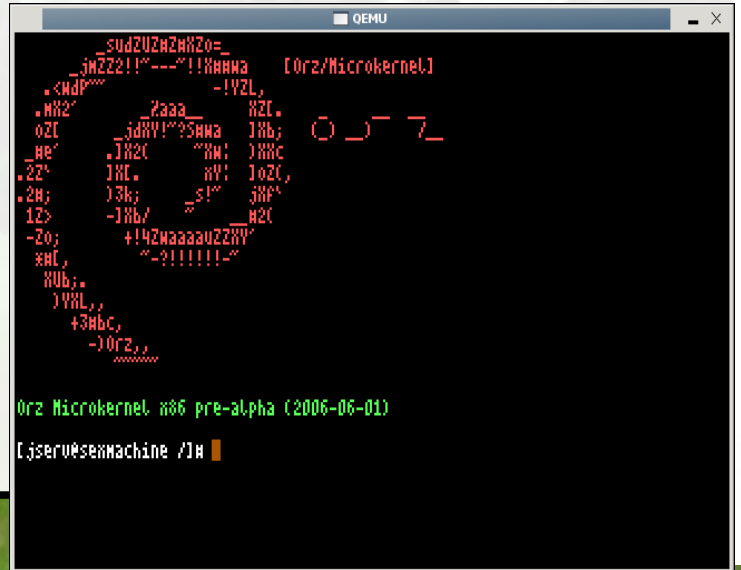
Jim Huang(黃敬群 /**jserv**)
Email: <**jserv.tw@gmail.com**>
Blog: **http://blog.linux.org.tw/jserv/**

提綱

- 動機與原則
- 機器人概況
- 嵌入式 (Embedded) 與即時作業系統 (RTOS)
- 設計自己的 RTOS
- 吃自己的狗食！

動機 (1)

- Everything can be Orz.
  - Orz Microkernel('07)



動機 (2)

- Orz Microkernel 的啟發
  - 學習作業系統與相關的系統程式該如何設計
  - 建立自信：原來一個作業系統只需幾 kb 的空間就實做出來

動機 (3)

- 設計作業系統也可很有趣
  - 以實體的機器人設計作為主軸
  - 體驗如何親手打造嵌入式系統並著手設計相關軟硬體建設

原則 (1)

- 從零到有，設計即時作業系統
  - 杜威博士：「作中學」
  - RT nanokernel (OSDC.tw 2007)
  - 模仿 Linux 經典設計並建構具體而微的 RTOS

原則 (2)

- 滿足自動控制系統需求
  - 即時處理
  - 建構嵌入式環境

原則 (3)

- 善用自由軟體
  - 從開發環境到最終結果，都採用自由軟體
  - 以 GNU Toolchain 加速開發與系統偵錯

What is Robot?(1)

可被遠端遙控的機械？

What is Robot?(2)

可程式化、自動運作的機械？

What is Robot?(3)

模仿人類行為的機械？

What is Robot?(4)

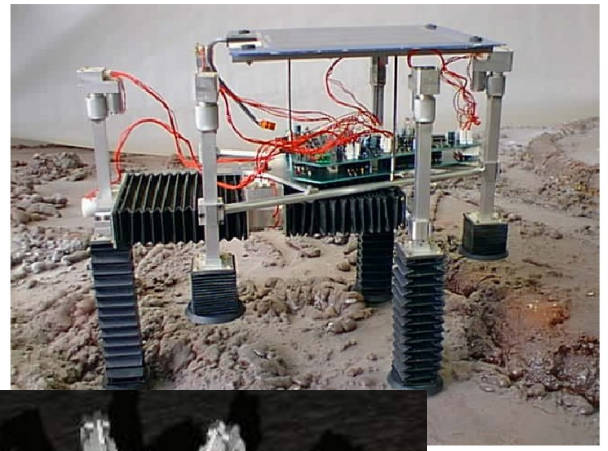
- 沒有確切的定義
  - ”Robot” 中文譯詞相當不精確
  - Robot 變成「聰明機器」的代名詞

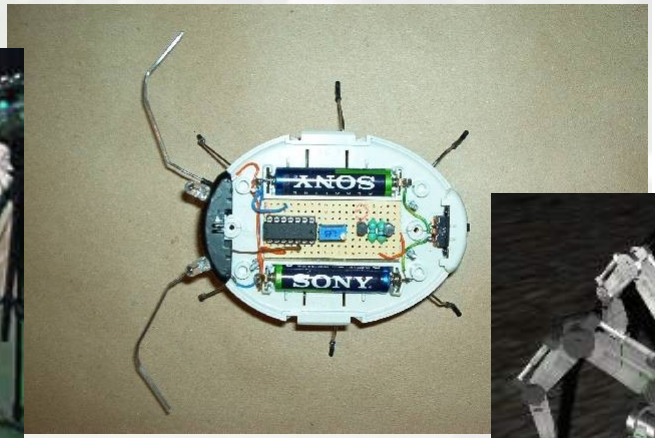
規範

- Robot
  - A mobile computer situated in the real world interacting with the environment through sensors and actuators in order to perform various intelligent tasks ***without constant attention***
- Robotics
  - The science of building and programming robots

應用型態

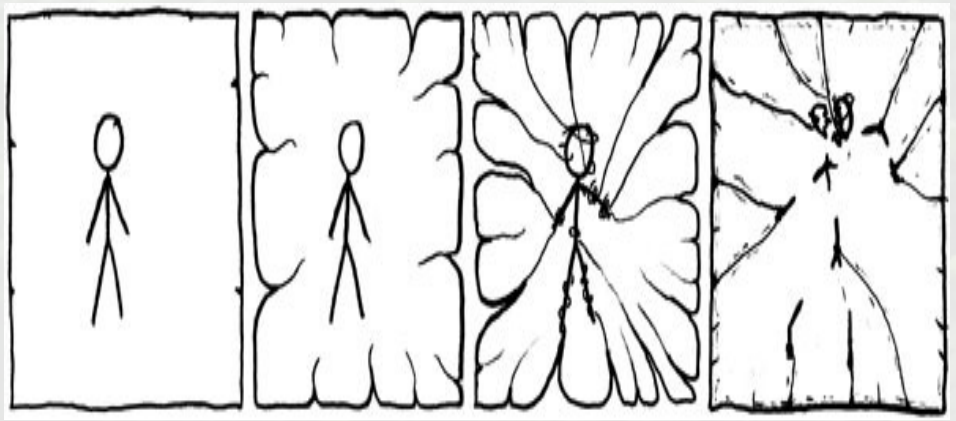
- 國防軍事
- 科學搜救探索
- 彷生物型態
- 機器人足球賽





親手打造機器人軟硬體

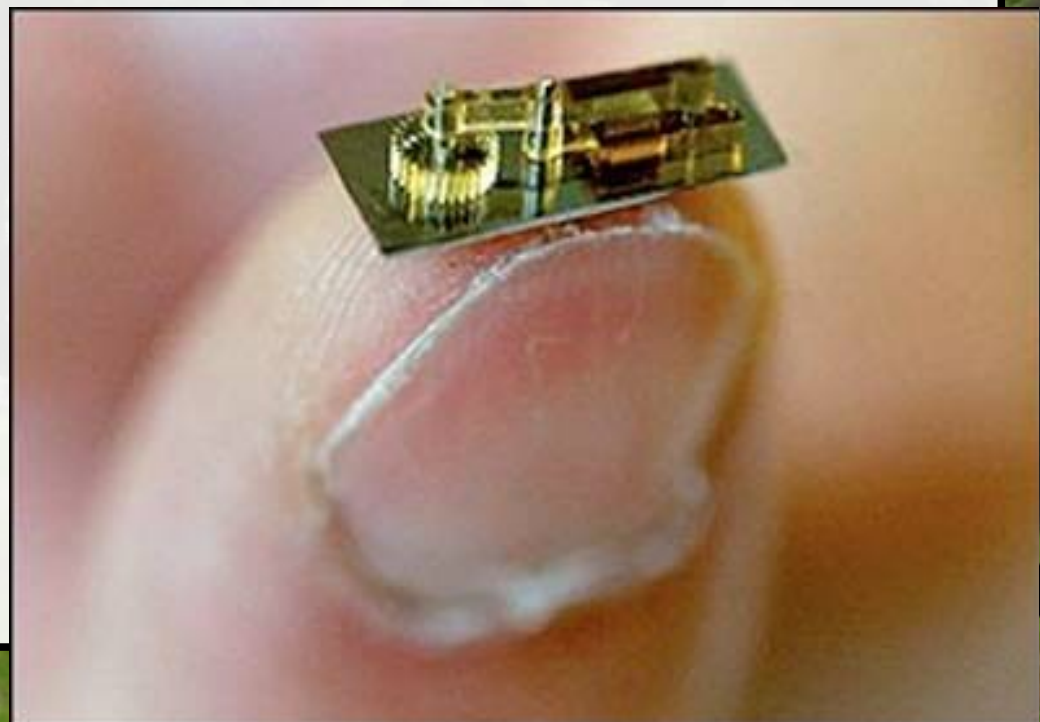
- 透視



  - 硬體設計
  - 軟體需求
    - 嵌入式系統 + 即時作業系統
  - 機器人控制

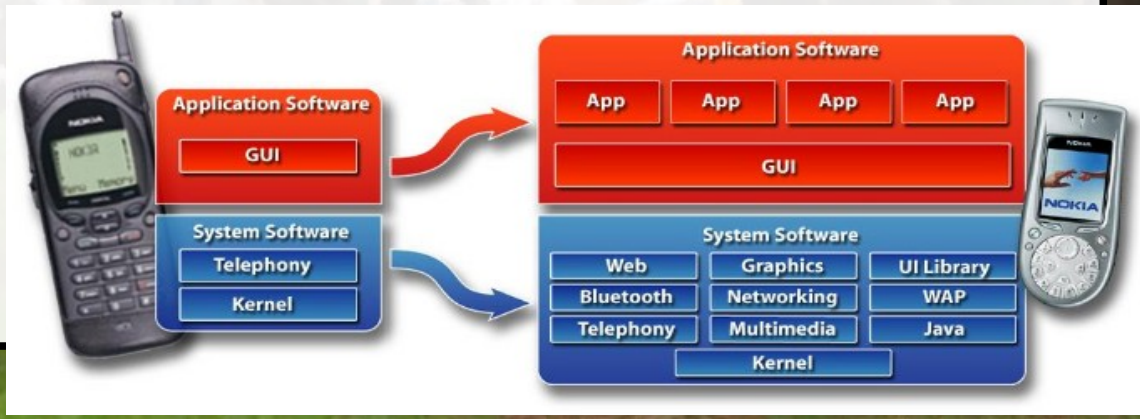
猜猜看

- Information at Your Fingers Tips
  - 語出 Bill Gates
  - 汽車引擎控制器 〉

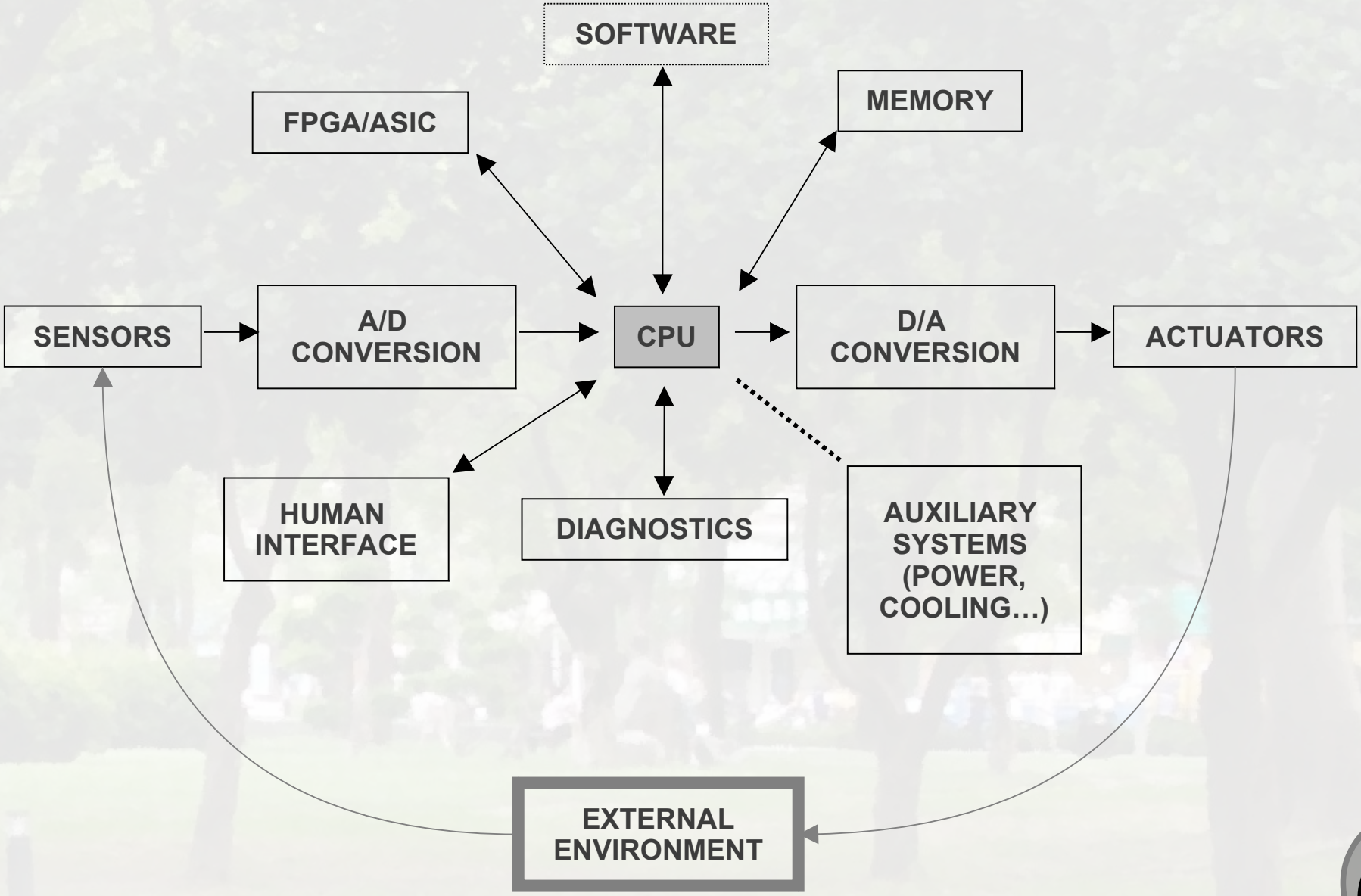


嵌入式系統

- Embedded System = System software + User Application + Hardware
- 根據 IEEE 的定義
  - Embedded 為控制、監視 / 輔助設備、機器或工廠運作的裝置
  - 電腦軟體與硬體的綜合體
  - 基於某種特殊用途



嵌入式系統組織


SOFTWARE
FPGA/ASIC
MEMORY
SENSORS
A/D CONVERSION
CPU
D/A CONVERSION
ACTUATORS
HUMAN INTERFACE
DIAGNOSTICS
AUXILIARY SYSTEMS (POWER, COOLING…)
EXTERNAL ENVIRONMENT

RT = Realtime(1)

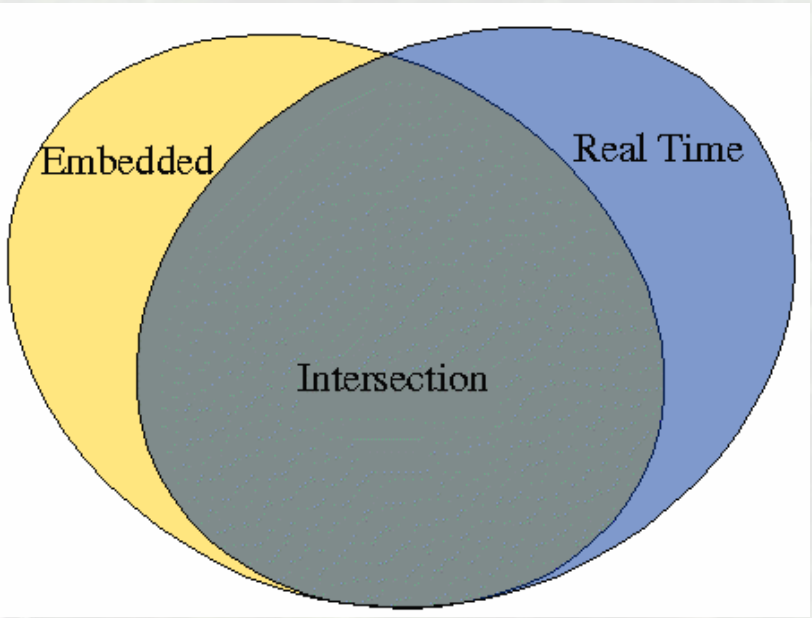
- 強調對於時間需求的處理
  - 若系統不理會時間需求，將可能有生命危險
    - 飛機失事
- POSIX Standard 1003.1

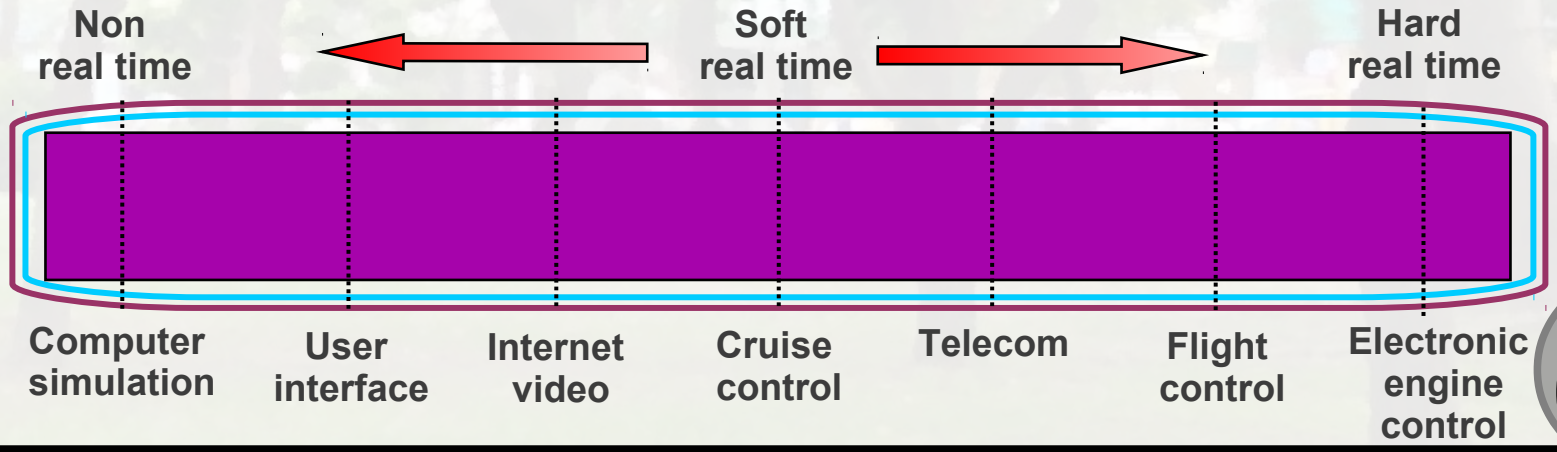
"Real time in operating systems:
The ability of the operating system to **provide a required level of service in a bounded response time**."

	POSIX.1	ISO 9945-1
Shell and Utility	POSIX.2	ISO 9945-2
Realtime API	**POSIX.1b**	**IEEE 1003.1b**
Operating Security API	POSIX.1e	IEEE 1003.1e
Operating Security Commands	POSIX.2c	IEEE 1003.2c

RT = Realtime(2)

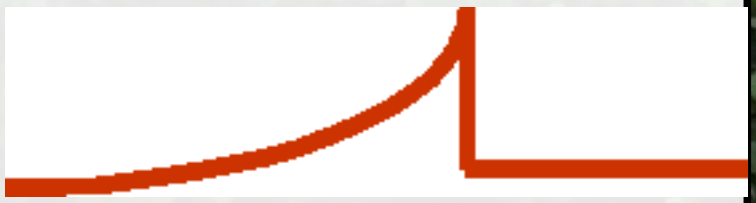
- Realtime 與 Embedded 無關
  - 但產品有交集



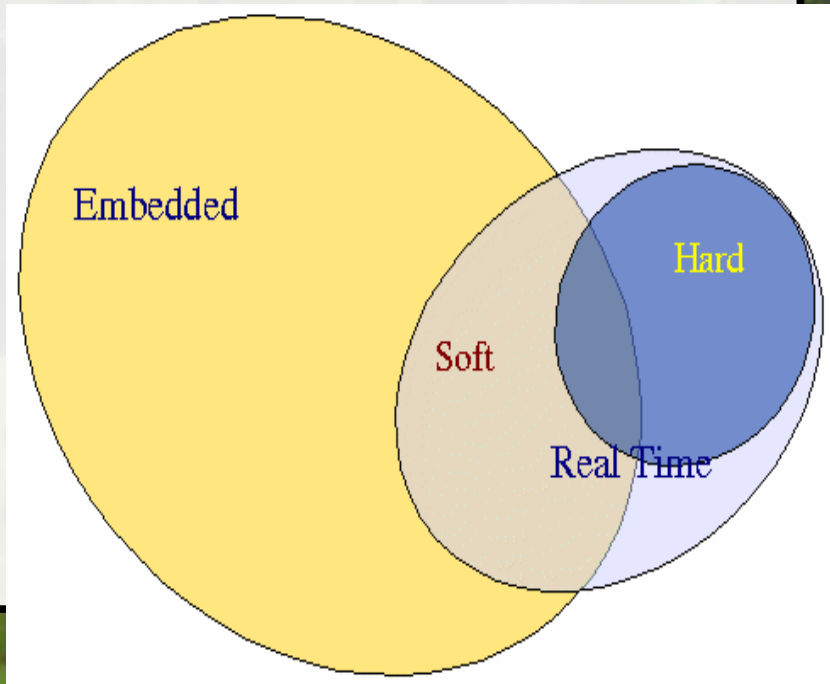


RT = Realtime(3)

- Hard Realtime
  - 當系統無法達到功能上或時間上的需求時，將會造成相當高的損害
  - 有 deadline 的設計考量
  - 應用：工業醫療控制



- Soft Realtime
  - 允許少量的工作延遲
  - 應用：多媒體



RTOS 應用探討

- 可決定性 (deterministic)
  - 以飛行模擬為例，飛行模型的模擬過程必須有可決定性
- 可預測的 (predictable)
  - 用於 I/O 的時間必須可預測

[Warning]

- 電腦工程師殺人
  - 程式沒寫好，不僅是過失，也可能是無法挽救的悲劇
  - 由 Real Time 看 Real Life

RTOS 需求 (1)

典型 RTOS 需求

- 快速的 context switch
- Interrupt Management
  - 有能力對於外界所引發的中斷做迅速的反應
- 多工 (Multi-Tasking) 核心
  - Fixed priority, preemptive scheduling
- 提供行程與行程之間溝通的機制
  - Semaphores、 Message Queues 等等
  - Priority Inheritance
- Timer
  - 特殊的 Alarm 與 Time out 機制
- Device Drivers

RTOS 需求 (2)

典型 RTOS 評估項目

- 可決定性 (Determinism)
  - 可自行估算工作在期限內完成
- 反應速度 (Responsiveness)
  - 確認中斷要求後，需多少時間來處理該中斷要求
- 使用者控制 (User control)
  - 對於各個行程的優先權精確地加以控制
- 可靠性 (Reliability)
- 錯誤容忍性 (Fail-soft operation)
  - 當系統發生錯誤時，是否有足夠的能力去維持資料與系統效能

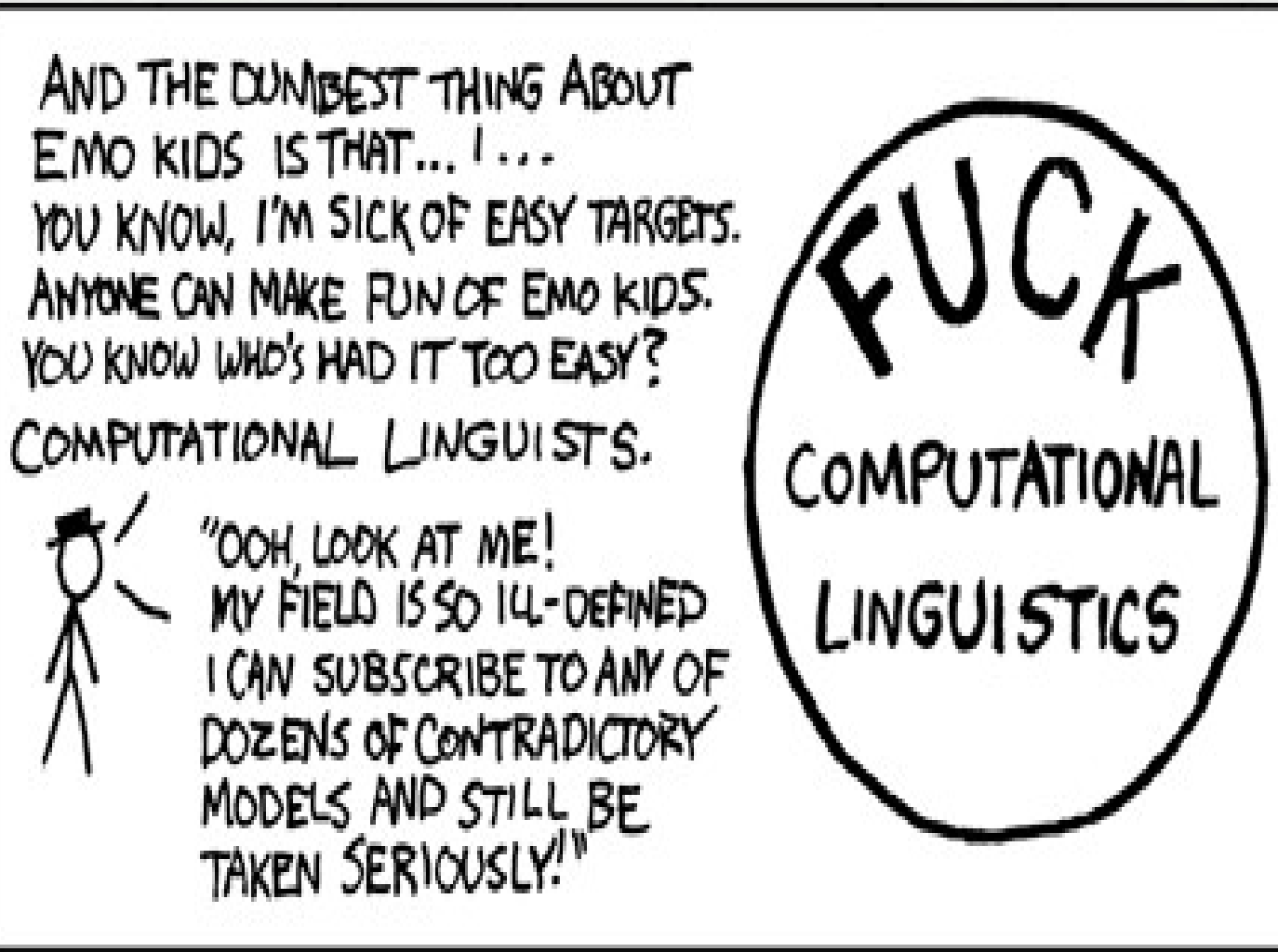
RTOS 需求 (3)

Response Time 考量

- Execution Time
  - 用於非資源競搶的時間總和
- Higher Priority Non Schedulable Entities
  - Interrupt Handlers
- Scheduling
  - Time taken by higher priority work
- Shared Resources
  - Time taken by lower priority work
    - 考慮共享資源競搶

不要再拿教科書壓我啦！

- 作中學



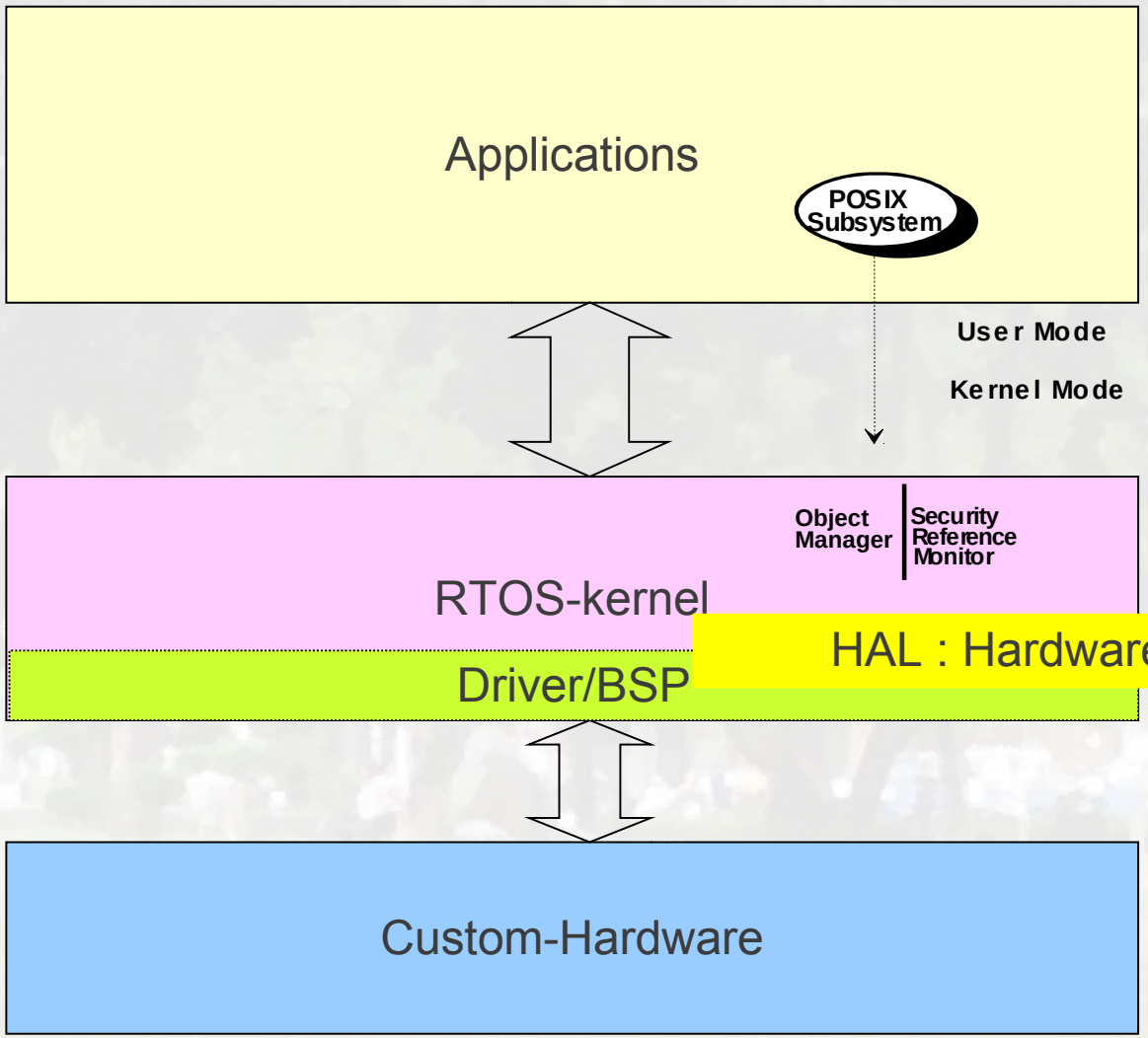
設計自己的 RTOS

Jamei = **J**ust **A**nother **M**icroprocessor **E**mbedded **I**nfrastructure

- 輕巧並可調整組態
- 仿造 Linux kernel 部份設計
  - arch(平台相依實做)
  - device driver model
  - vfs
- 部份 POSIX Thread
- 部份 Realtime API (IEEE 1003.1b)
- New BSD License 與 GNU GPLv2(部份)
- 支援 i386 與 ARM9(S3C24xx)

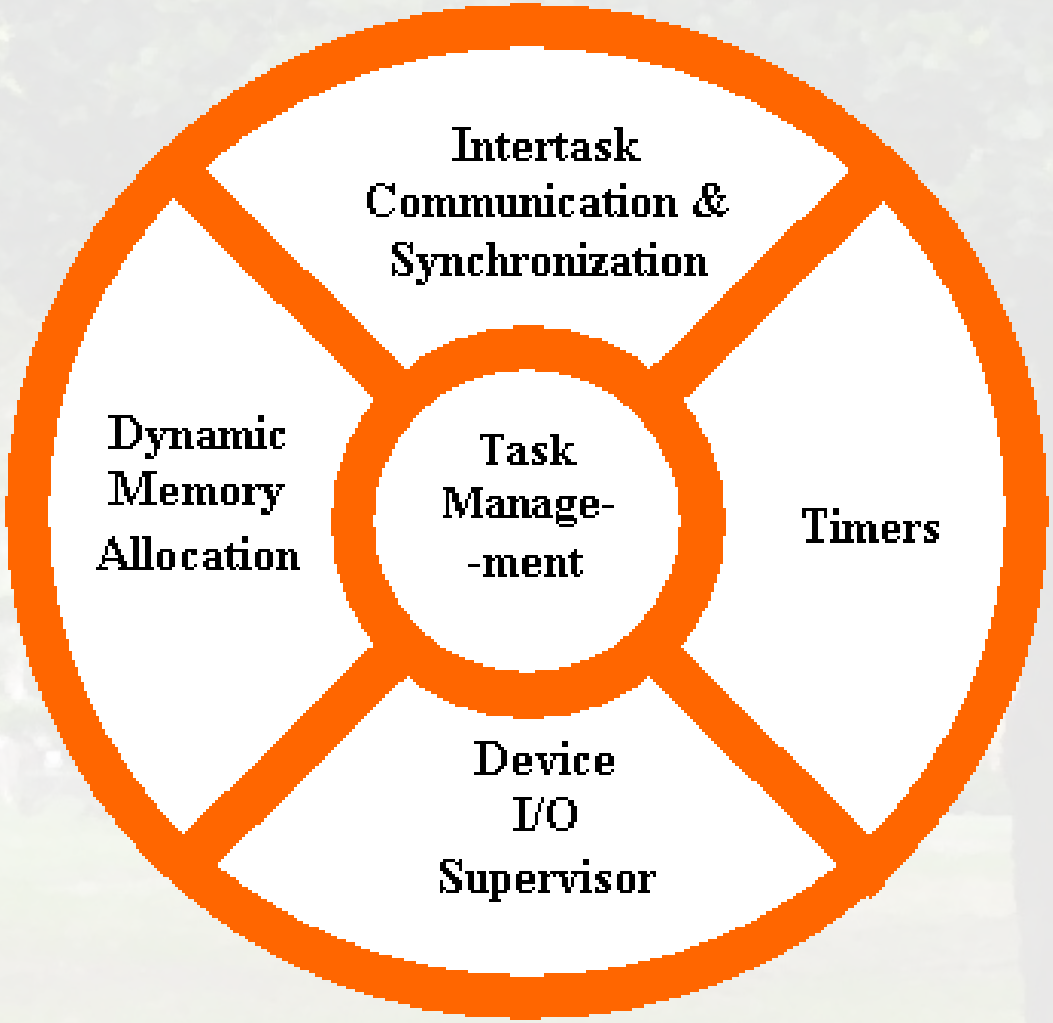
RTOS 結構

- Introduction
- Structure
- RTOS Kernel
- Tasks
- Memory
- Timers
- I/O
- IPCs
- Device Driver
- In an Action

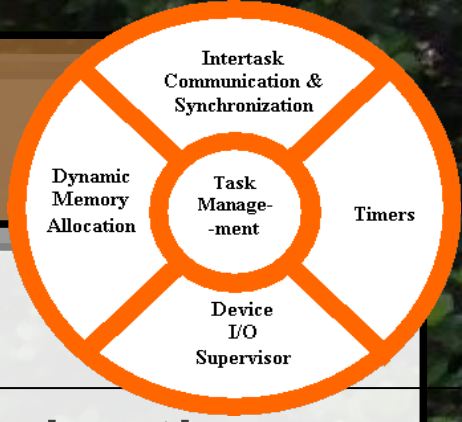


RTOS Kernel

- Introduction
- Structure
- RTOS Kernel
- Tasks
- Memory
- Timers
- I/O
- IPCs
- Device Driver
- In an Action



RTOS Kernel::Tasks

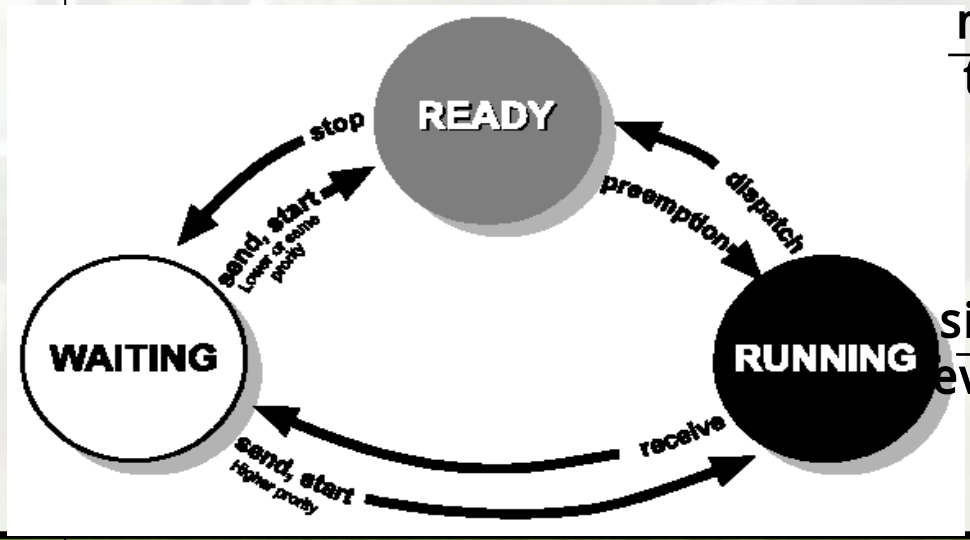


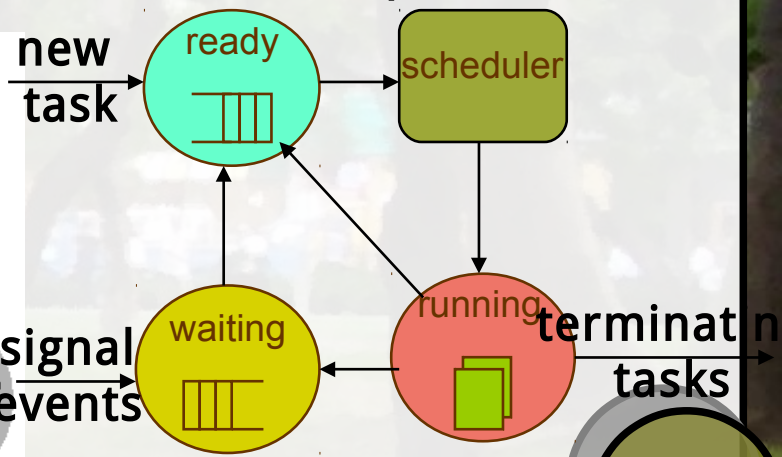
Task 是RTOS 最重要的項目

- Introduction
- Structure
- RTOS Kernel
- Tasks
- Memory
- Timers
- I/O
- IPCs
- Device Driver
- In an Action

- RTOS scheduler 必須 determinstic
  - O(1) or O(n)
- Scheduling policy
  - Clock driven
  - Priority driven (RMS & EDF)

笨蛋，問題在 scheduling ！

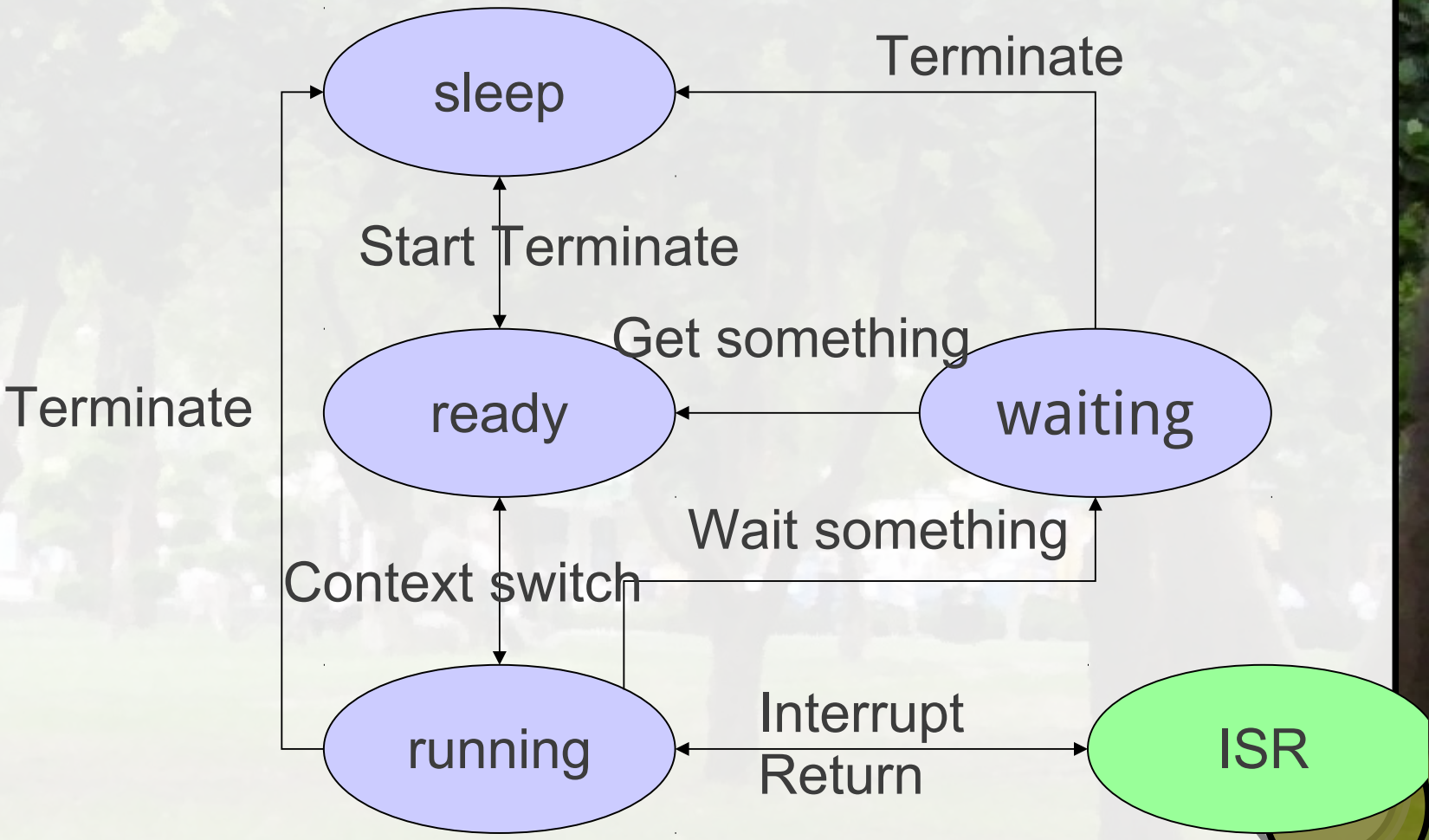




RTOS Kernel::Tasks

Task 的狀態移轉

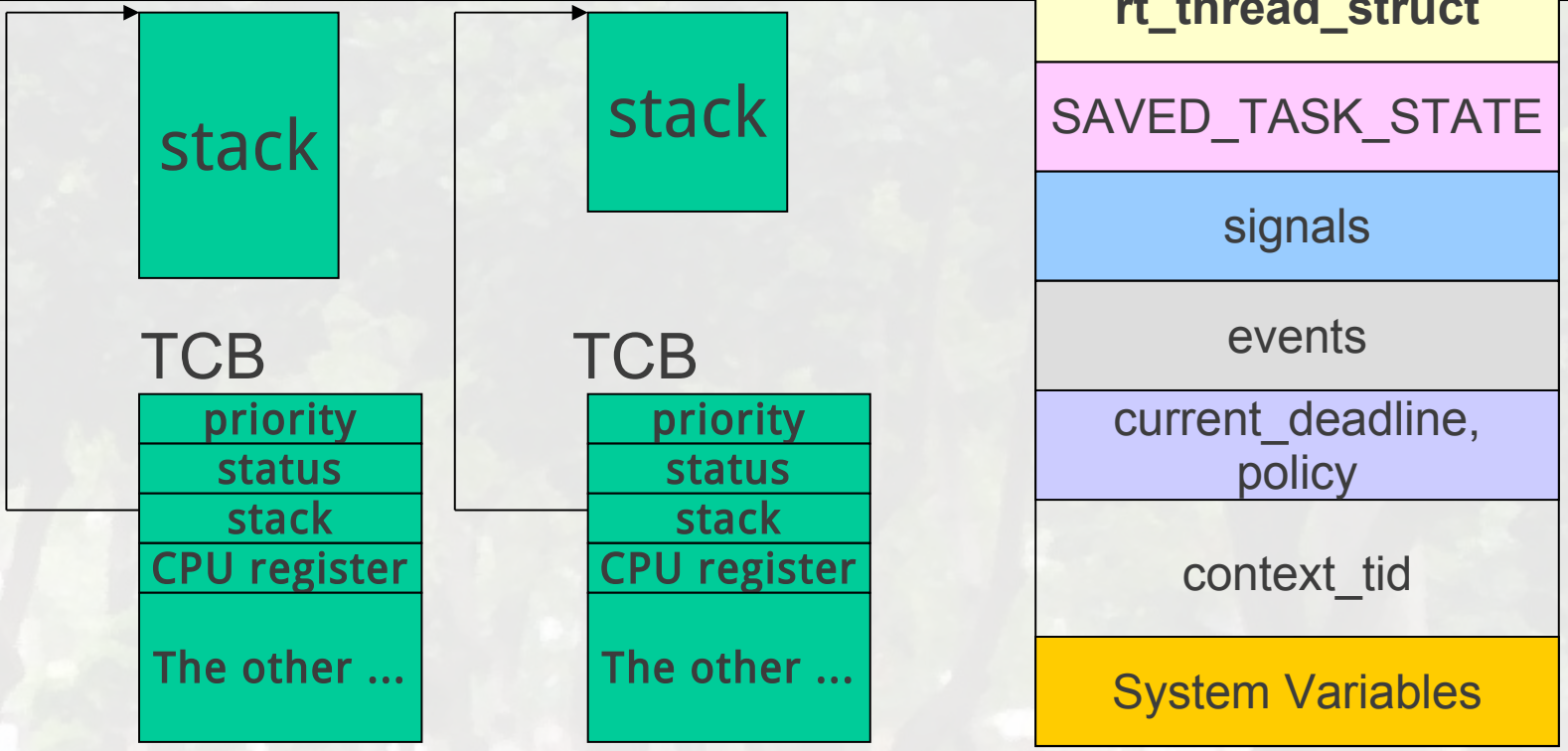
- Introduction
- Structure
- RTOS Kernel
- Tasks
- Memory
- Timers
- I/O
- IPCs
- Device Driver
- In an Action

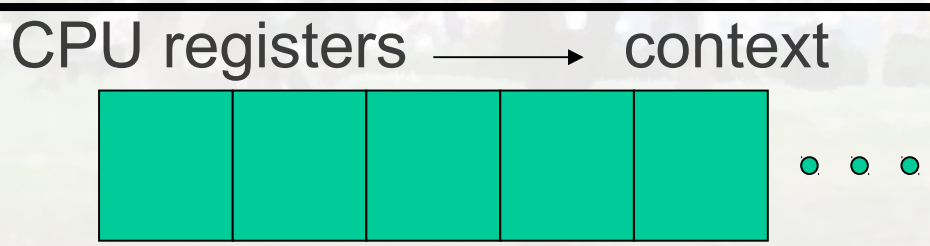


RTOS Kernel::Tasks

TCB (Task Control Block)

- Introduction
- Structure
- RTOS Kernel
- Tasks
- Memory
- Timers
- I/O
- IPCs
- Device Driver
- In an Action





RTOS Kernel::Tasks

Jamei 中 thread 是 Task 的單元

- Introduction
- Structure
- RTOS Kernel
- Tasks
- Memory
- Timers
- I/O
- IPCs
- Device Driver
- In an Action

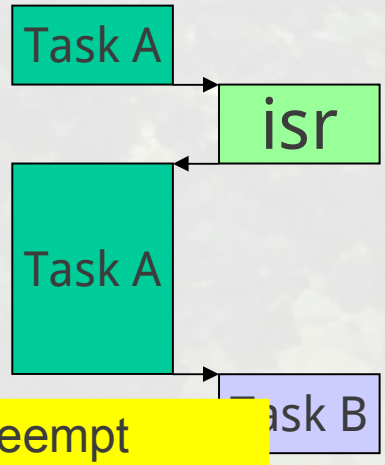
- typedef struct rt_thread_struct *pthread_t;
- pthread_create
- pthread_exit
- pthread_kill
- pthread_wakeup_np
- pthread_suspend_np
- pthread_wait_np
- ...

•試圖滿足 POSIX Thread 語意
•_np 表示 non-POSIX

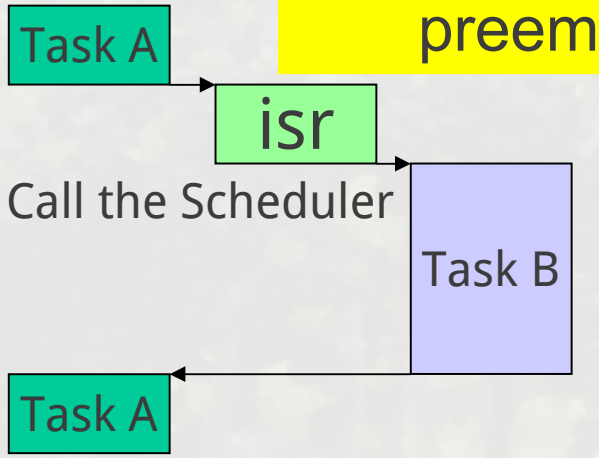
RTOS Kernel::Tasks

Jamei 中 thread 是 Task 的單元

- Introduction
- Structure
- RTOS Kernel
- Tasks
- Memory
- Timers
- I/O
- IPCs
- Device Driver
- In an Action



non-preempt



preempt

preemptive scheduling

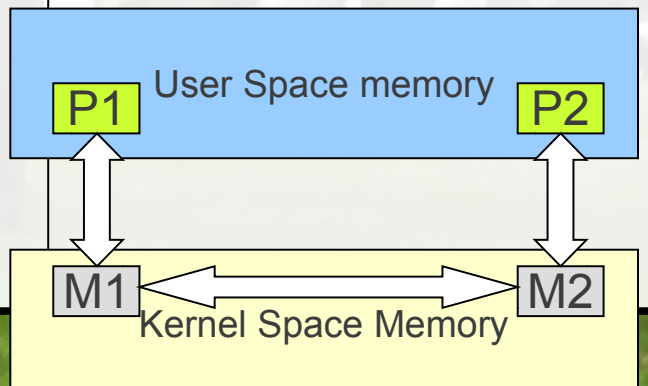
```
int rt_schedule(void)
{
  struct rt_thread_struct *preemptor = 0;
  ...

  if ((s->clock->mode == RT_CLOCK_MODE_ONESHOT)) {
    if ((preemptor = find_preemptor(s,new_task)))

    {
      (s->clock)->settimer(s->clock, preemptor->resume_time - now);
    } else {
      (s->clock)->settimer(s->clock, (HRTICKS_PER_SEC / HZ) / 2 );
    }
    set_bit (RT_SCHED_TIMER_OK, &s->sched_flags);
  }
```

優先權 Task A < Task B

RTOS Kernel::Memory

- Introduction
- Structure
- RTOS Kernel
- Tasks
- Memory
- Timers
- I/O
- IPCs
- Device Driver
- In an Action

- Jamei 可支援 MMU (virtual memory) 與 MPU (memory protection)
- 區分 user-space 與 kernel-space memory




```
void start_kernel(void)
{
  /* architecture-dependent */
   init_arch();
  /* NOTE:
       - Mask all interrupts.
       - Interrupts are mapped in 0x20-0x30 IDT entries
  */

#if CONFIG_KERNEL_MEMORYPROT
 init_page();
#endif

#if CONFIG_CONTEXT_MEMORYPROT
 init_context();
#endif
...
```

RTOS Kernel::Memory

Jamei 仿效 POSIX/Linux 的處理

- Introduction
- Structure
- RTOS Kernel
- Tasks
- Memory
- Timers
- I/O
- IPCs
- Device Driver
- In an Action

- APIs
  - kmalloc / kfree
  - mmap
  - shm (shared memory area)
- 有彈性的記憶體管理機制
  - 放任 (no protected memory)
  - 最低限度記憶體保護機制
    - 不可寫入 RT executive 記憶體
  - 保護 context 執行單元記憶體

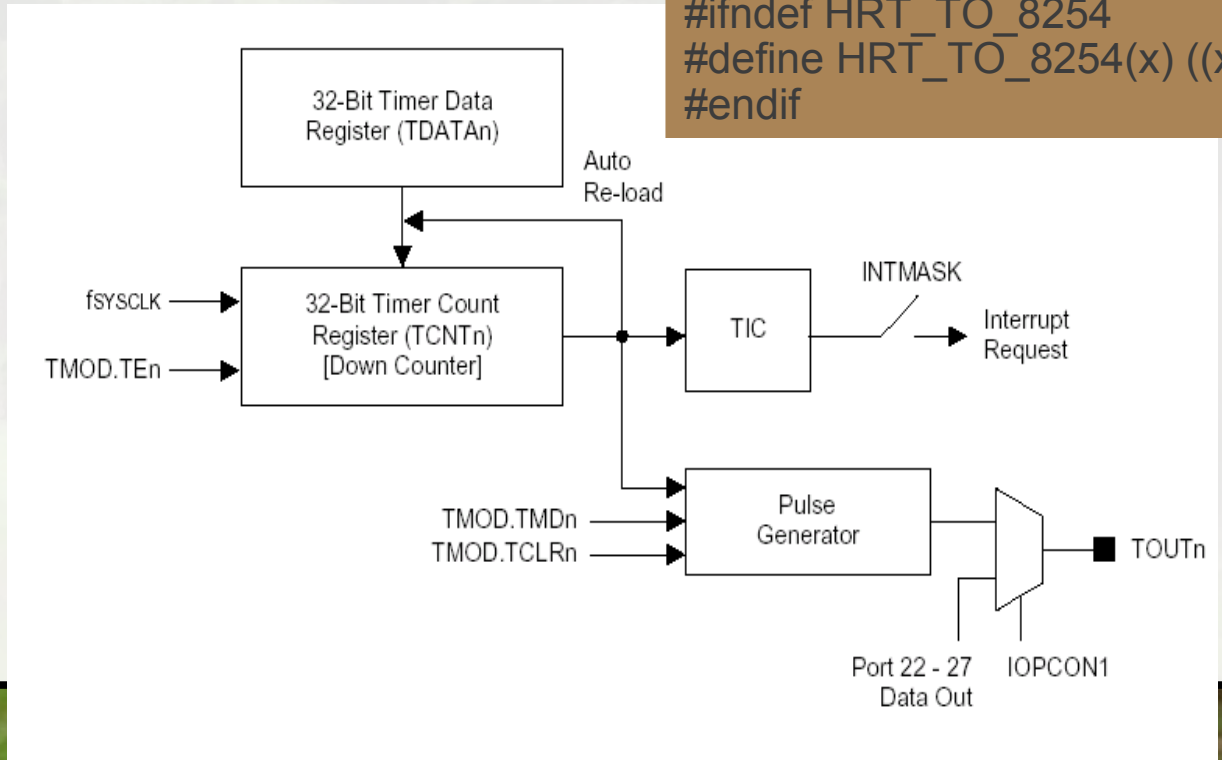
RTOS Kernel::Timer

Timer 本質上是硬體時鐘的軟體呈現

- Introduction
- Structure
- RTOS Kernel
- Tasks
- Memory
- Timers
- I/O
- IPCs
- Device Driver
- In an Action

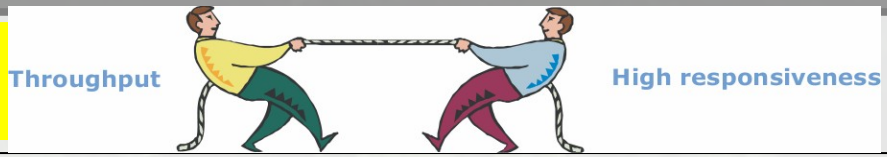
- 型態
  - watchdog timer
  - programmable timer

```
#define HRT_FROM_NS(x) (x)
#define HRTICKS_PER_SEC 1000000000

#ifndef HRT_FROM_8254
#ifndef HRT_FROM_8254
#define HRT_FROM_8254(x) ((x) * 838)
#endif

#ifndef HRT_TO_8254
#define HRT_TO_8254(x) ((x) / 838)
#endif
```



RTOS Kernel::Timer

萬惡 HZ



- Introduction
- Structure
- RTOS Kernel
- Tasks
- Memory
- Timers
- I/O
- IPCs
- Device Driver
- In an Action

- 實做 HRT (High Resolution Timer)
- 用於 sched 、 sync 等系統實做

```
struct rl_clock {
        ...
        int (*sethrtime)(
                struct rt_clock *,
                hrtime_t t);

        int (*settimer)(
                struct rt_clock *,
                hrtime_t interval);
        ...
        hrtime_t resolution;
        hrtime_t value;
        hrtime_t delta;
        pthread_spinlock_t lock;
        struct rt_clock_arch arch;
};
```

```
int rt_schedule(void)
{
...
  if ((s->clock->mode == RT_CLOCK_MODE_ONESHOT)) {
    if ((preemptor = find_preemptor(s,new_task)))

    {
      (s->clock)->settimer(s->clock, preemptor->resume_time - now);
    } else {
      (s->clock)->settimer(s->clock, (HRTICKS_PER_SEC / HZ) / 2 );
    }
...
```

RTOS Kernel::I/O

everything is file

- Introduction
- Structure
- RTOS Kernel
- Tasks
- Memory
- Timers
- I/O
- IPCs
- Device Driver
- In an Action

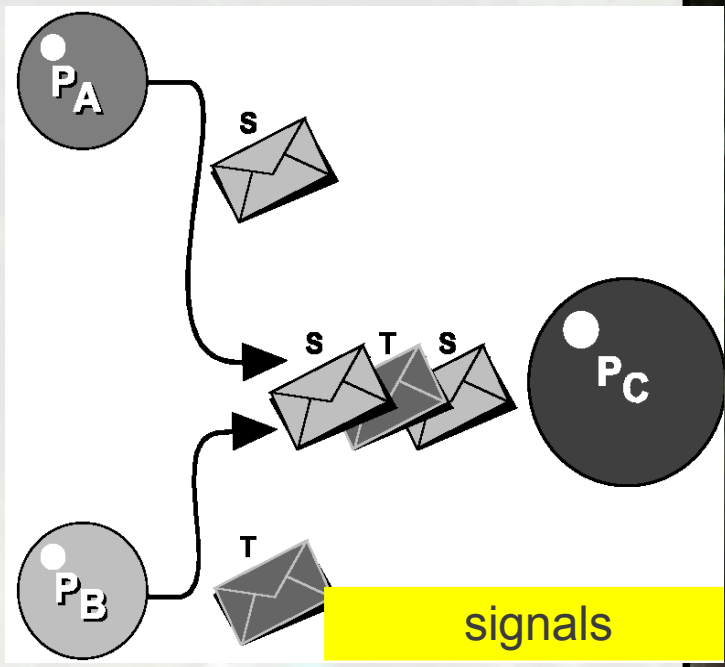
- Interrupt-driven
  - polling / DMA
- I/O mapping
  - memory space 與 I/O 操作的對應
- APIs
  - open / close
  - read / write
  - mmap
  - register_rtdev / unregister_rtdev

RTOS Kernel::IPC

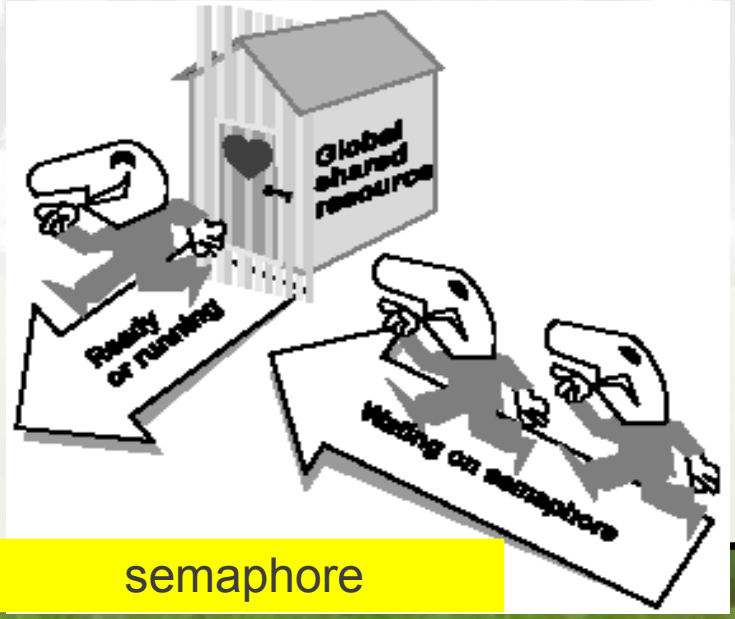
其實是 Inter-Task Communications

- Introduction
- Structure
- RTOS Kernel
- Tasks
- Memory
- Timers
- I/O
- IPCs
- Device Driver
- In an Action

- 可配置的組態
  - Signals
  - Semaphore



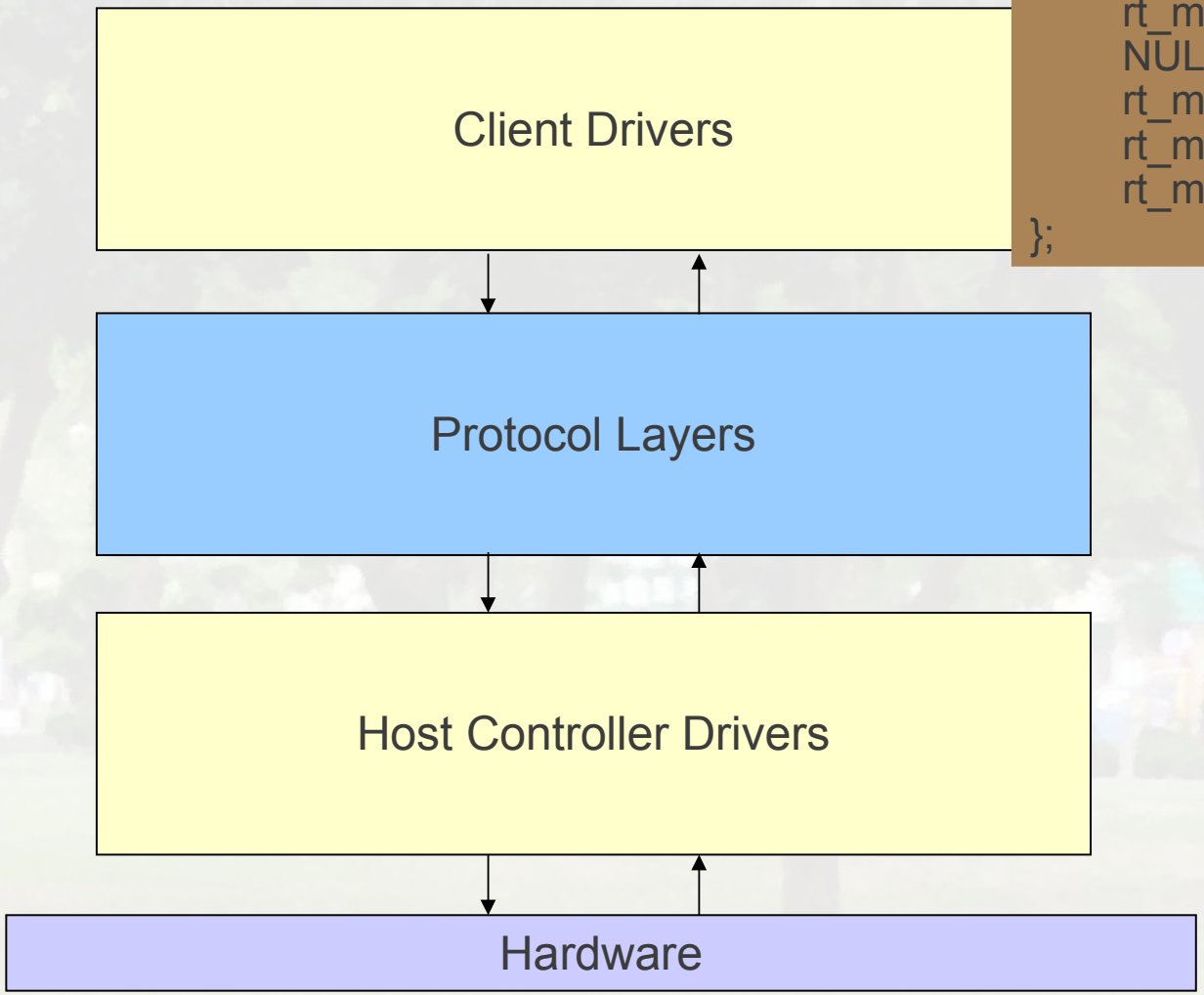
signals



semaphore

RTOS Kernel::Device Driver

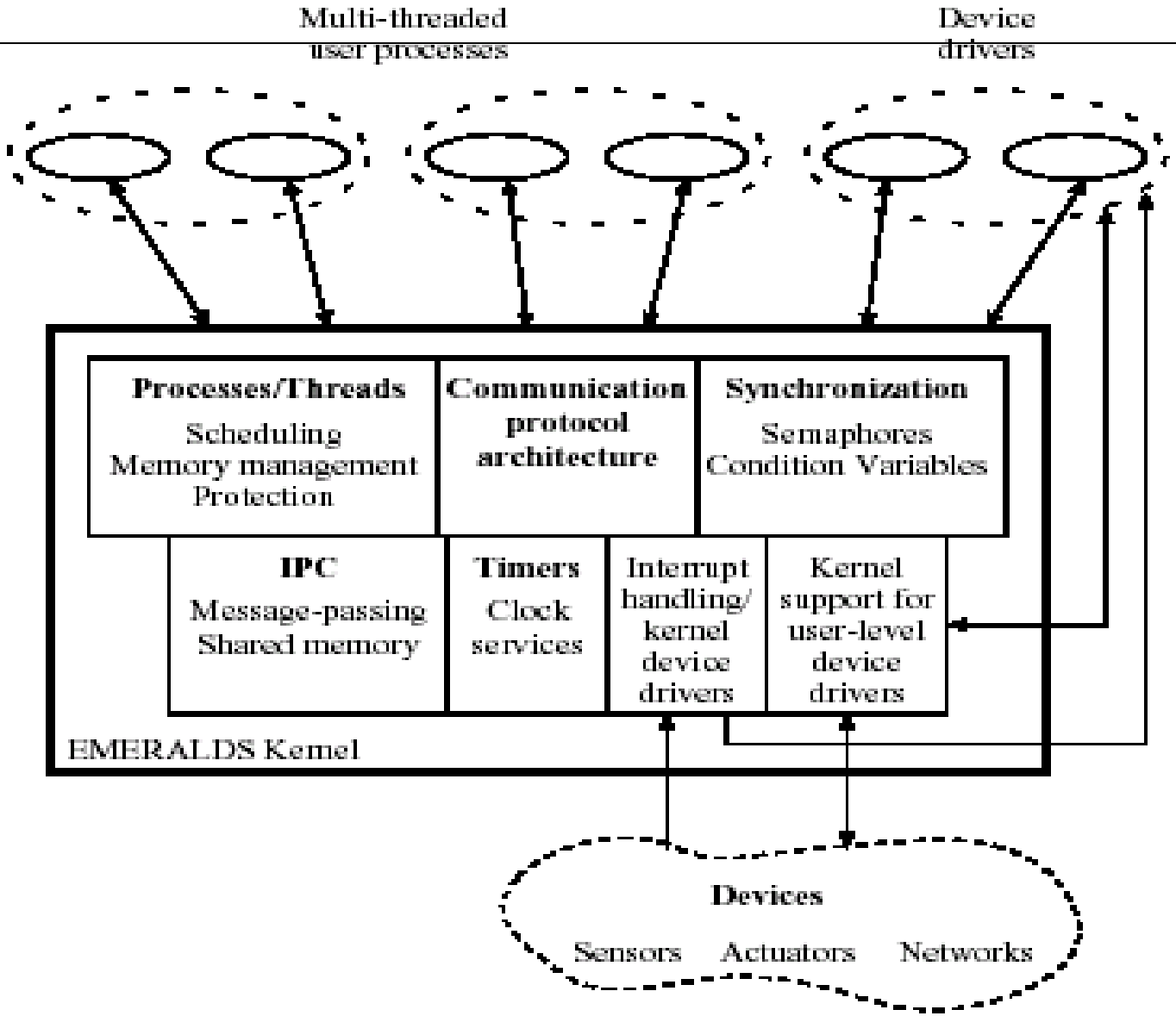
- Introduction
- Structure
- RTOS Kernel
- Tasks
- Memory
- Timers
- I/O
- IPCs
- Device Driver
- In an Action




```
static struct rt_file_operations
rt_mem_fops = {
        rt_mem_llseek,
        rt_mem_read,
        rt_mem_write,
        NULL,
        rt_mem_mmap,
        rt_mem_open,
        rt_mem_release
};
```

In an Action

- Introduction
- Structure
- RTOS Kernel
- Tasks
- Memory
- Timers
- I/O
- IPCs
- Device Driver
- In an Action

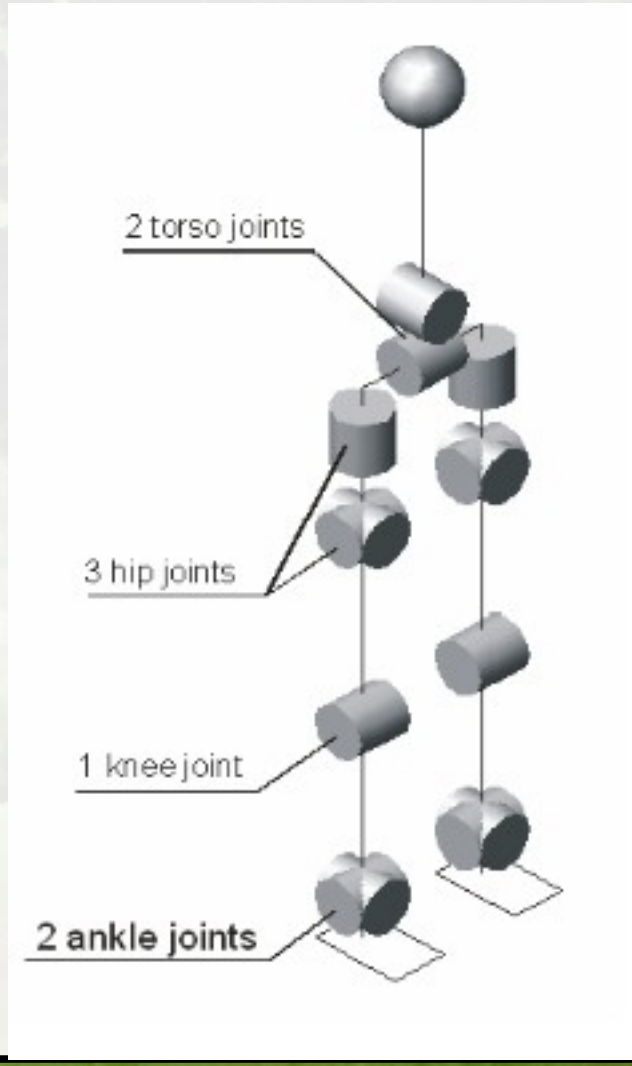


應用 RTOS 到嵌入式環境

- 吃自己的狗食
  - Eat Our Own Dog Food
  - 「在電腦業界中，吃自己的狗食是個怪名稱，表示真正使用自己產品的動作」
    - Joel on Software
  - 用 RTOS 控制機器人

機器人硬體機構概況

硬體：就是電腦系統中，可以讓你踢一腳的地方

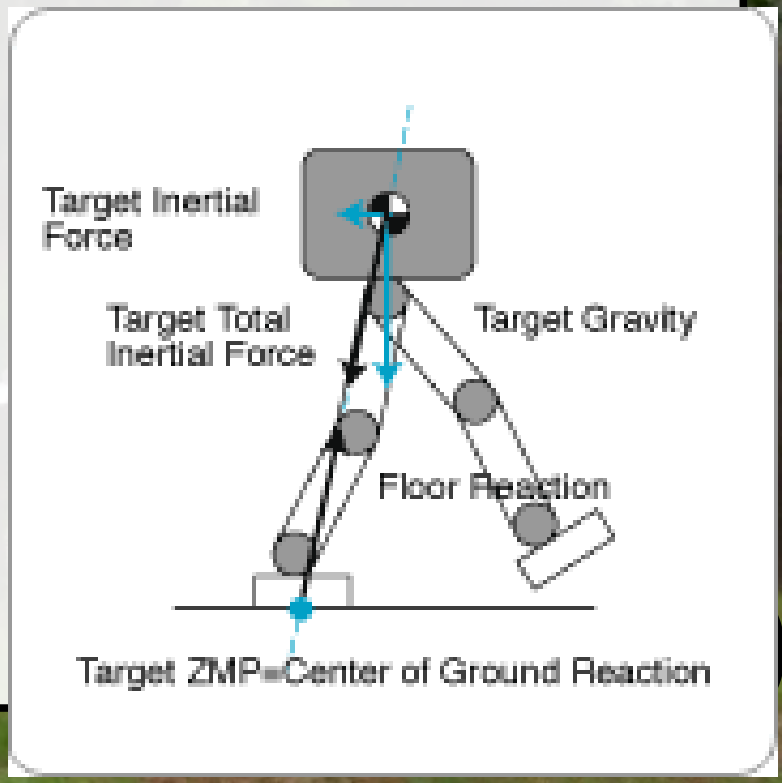


二足運動原理

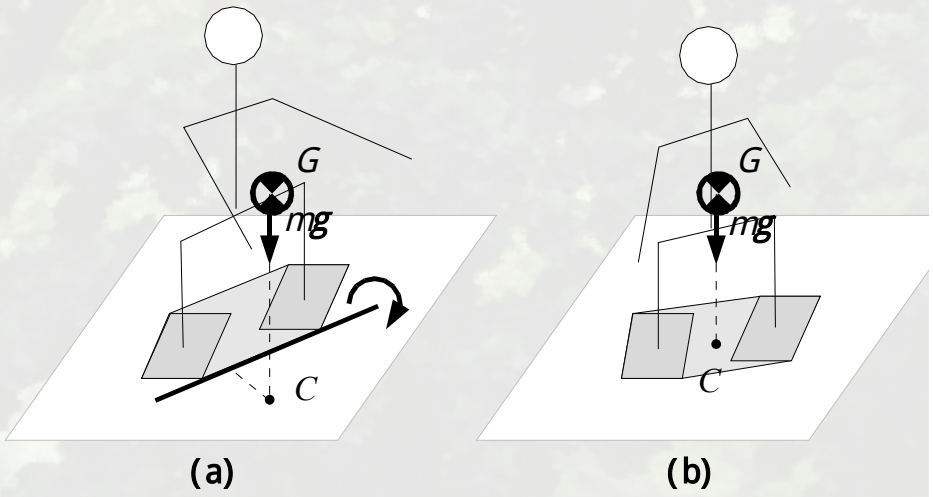
ZMP

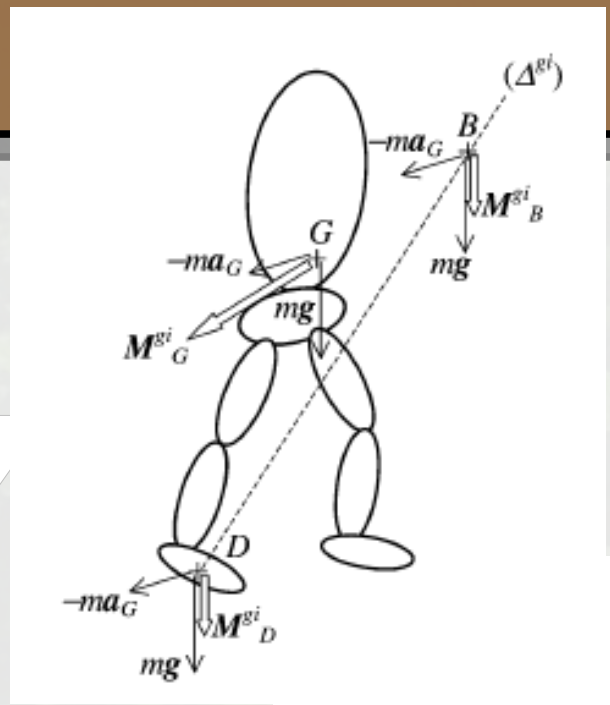
Zero Moment Point

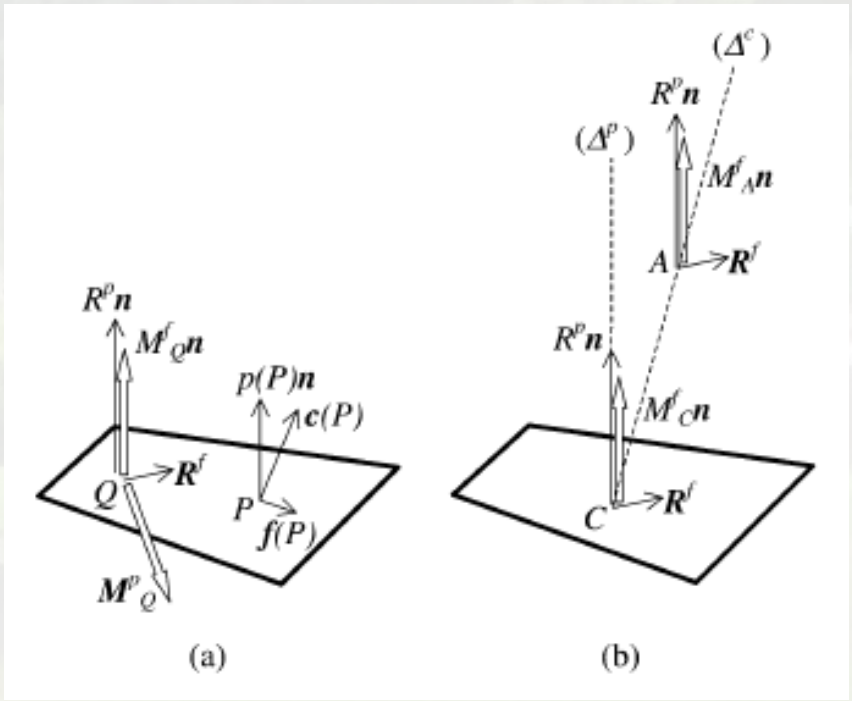
- 二足運動簡化模型
- 機器人「重心位置的重力」與「由加速度產生的作用力」兩者合力，落在支撐面內部即可

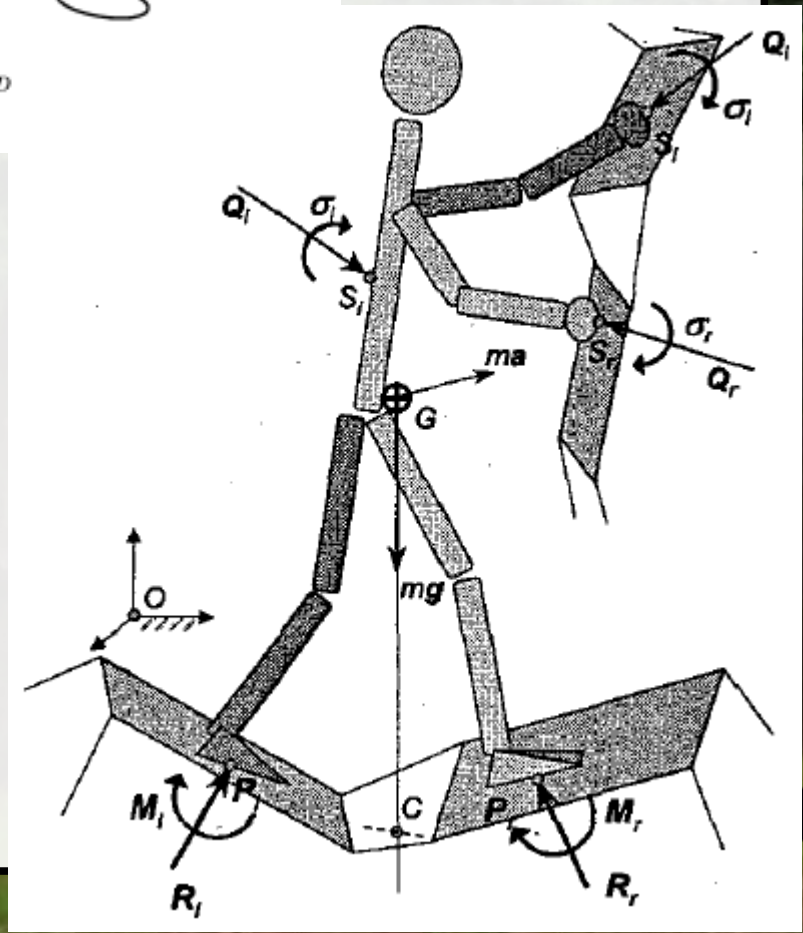


二足運動原理

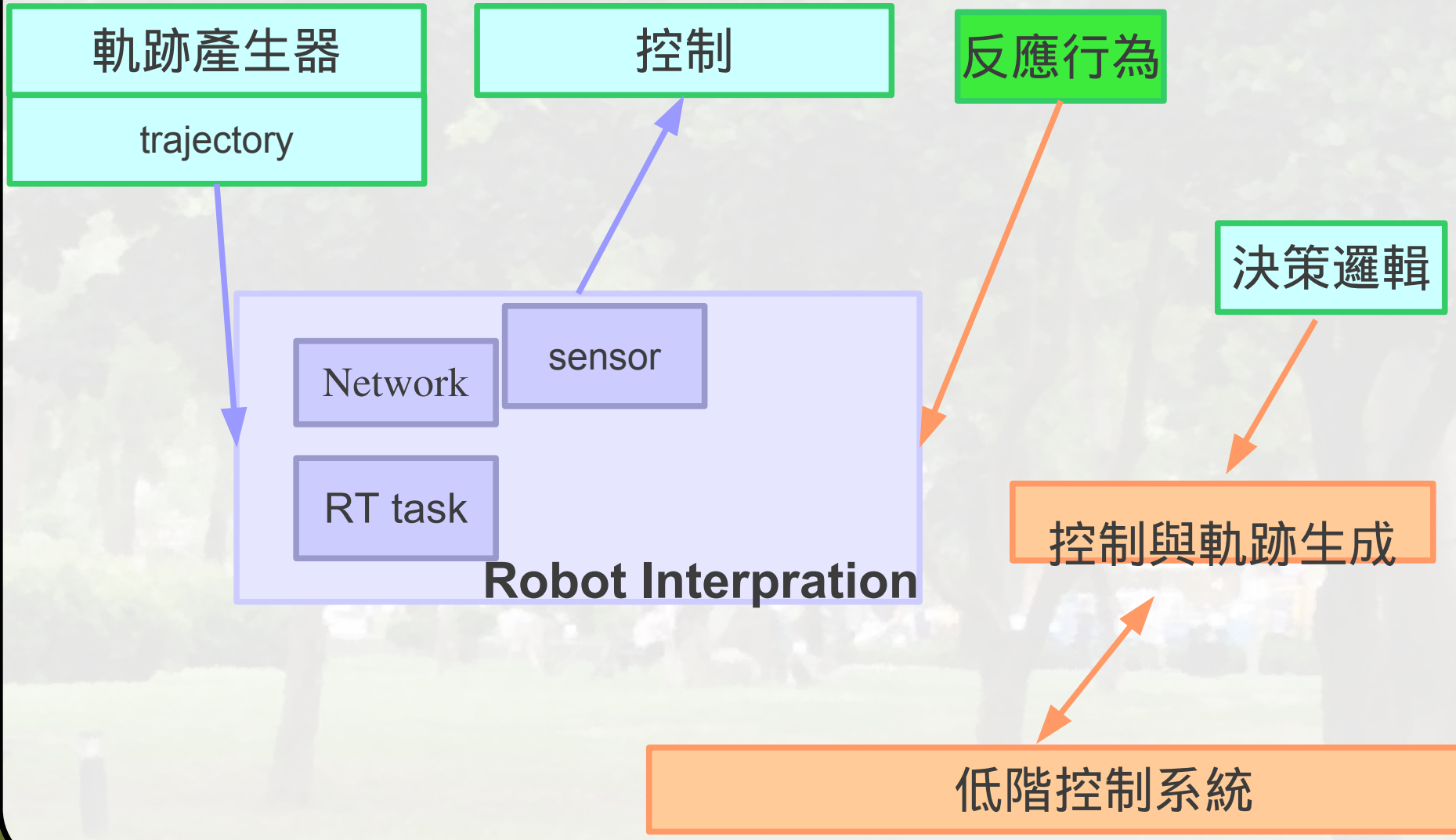

G
mg
C
(a)
G
mg
C
(b)


(Δ^{gi})
B
$-ma_G$
$M^{gi}{}_B$
mg
G
$-ma_G$
mg
$M^{gi}{}_G$
D
$-ma_G$
$M^{gi}{}_D$
mg


$R^p n$
$M^f{}_Q n$
$p(P)n$
$c(P)$
Q
R^f
P
$f(P)$
$M^p{}_Q$
(a)
(Δ^c)
$R^p n$
(Δ^p)
$M^f{}_A n$
A
R^f
$R^p n$
$M^f{}_C n$
C
R^f
(b)

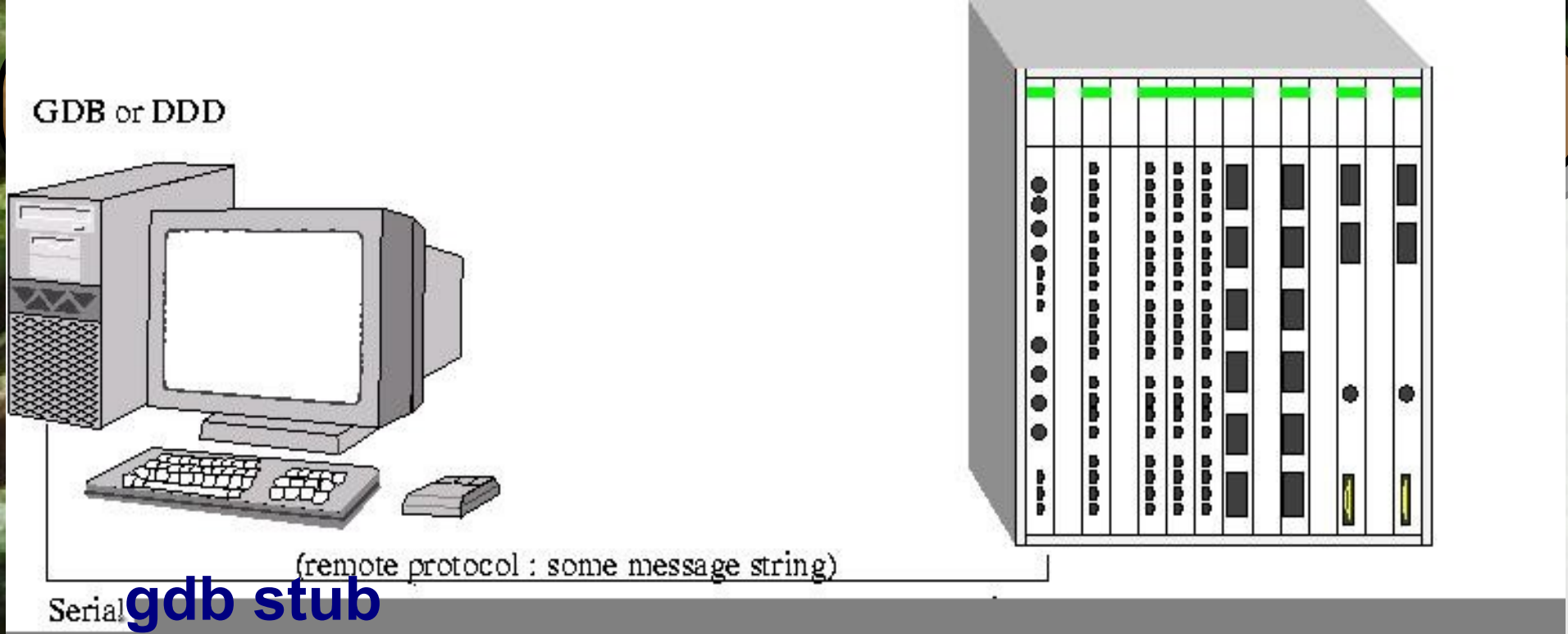

Q_l
σ_l
S_l
Q_l
σ_l
S_l
σ_r
S_r
Q_r
ma
G
mg
O
M_l
R
C
P
M_r
R_l
R_r

控制系統


軌跡產生器
trajectory
控制
反應行為
決策邏輯
Network
sensor
RT task
Robot Interpration
控制與軌跡生成
低階控制系統

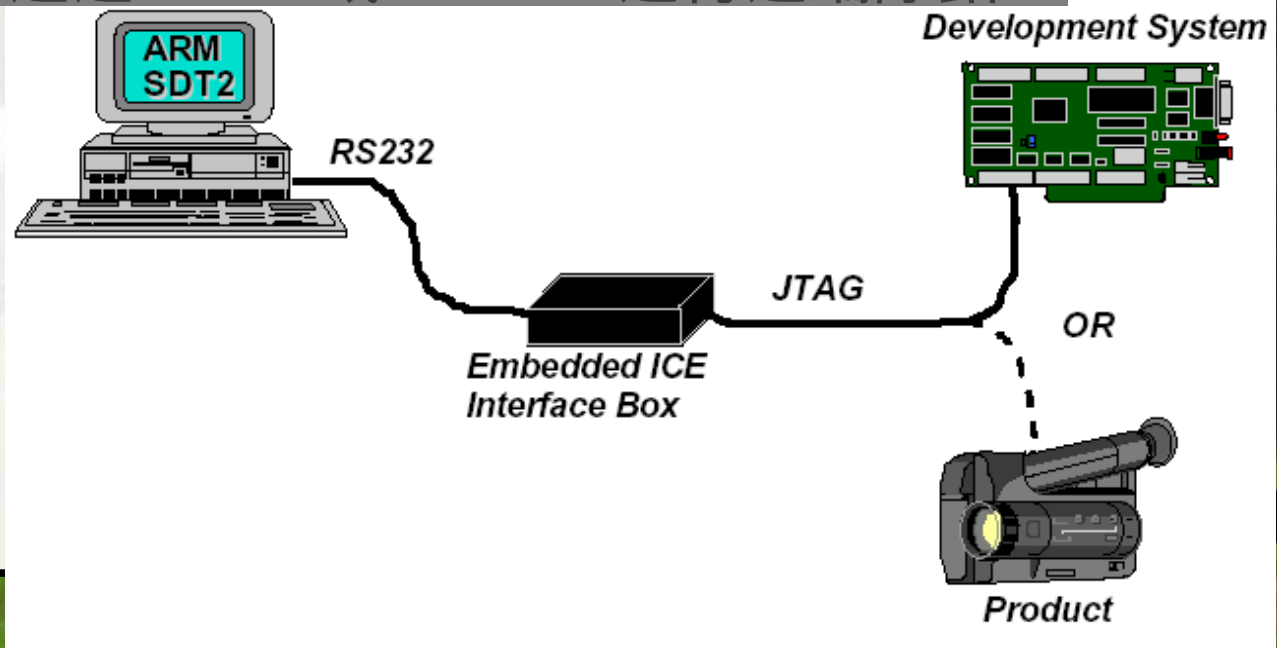
工欲善其事

- 自由軟體工具
  - GNU Compiler Collection(gcc)
  - GNU Debugger(gdb)
  - emulator (qemu / skyeye)
  - profiling (gprof)
  - Memory Debugging Tools
    - valgrind / electric-fence
  - Hardware Debugging Support and Tools
    - In-Circuit Emulator (ICE)
    - JTAG

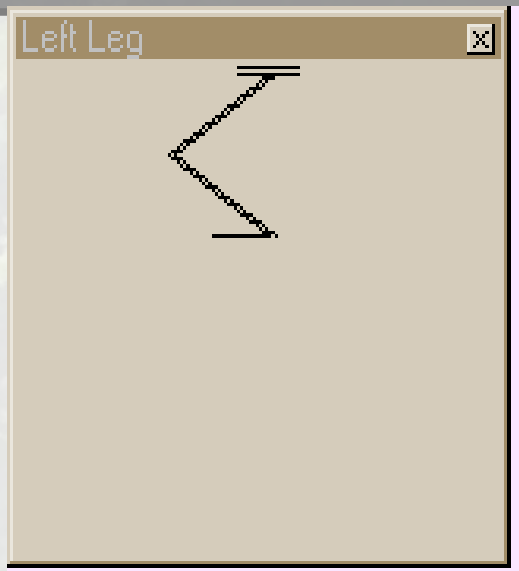


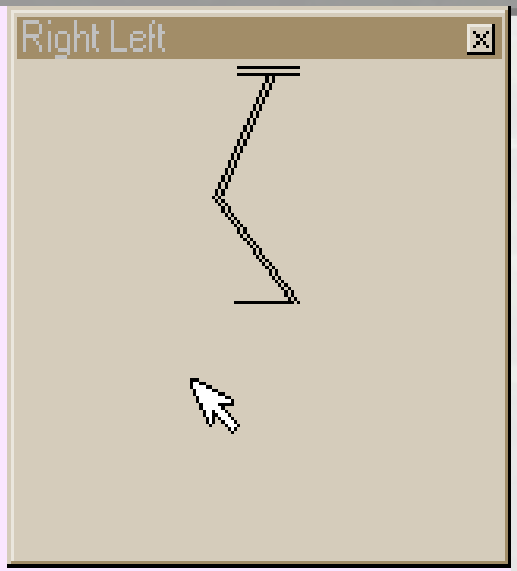
gdb stub

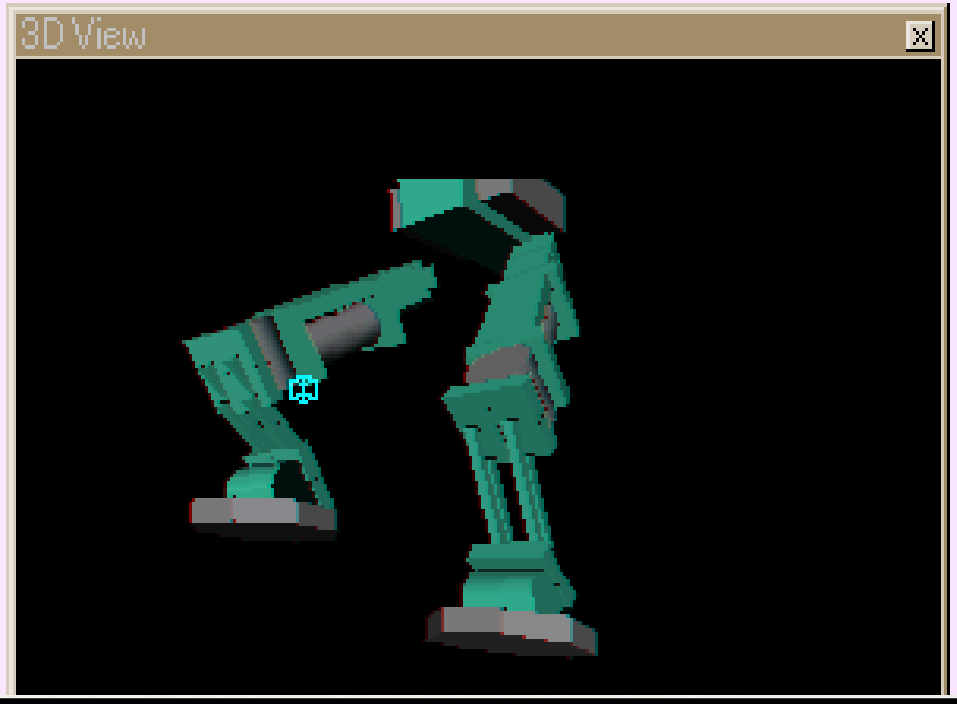
- 考慮在 emulation/target 模式下，該如何喚起 gdb ？
- Remote Debugging： 透過 serial 或 TCP/IP 進行遠端除錯



Show me the Robot







結論

- 無所不在的嵌入式系統
  - Invisible Computer
- 機電整合與自由軟體的機會
- 作中學
  - POSIX-like RTOS
  - 截長補短

參考資料

- 《The Invisible Computer》
  - MIT Press, Donald A. Norman
- 《Real-time Systems》, Jane W. S. Liu
- 部份圖片出自 xkcd
  - http://xkcd.com/
- 〈Real Time in Embedded Linux Systems〉, Michael Opdenacker - Free Electrons
  - http://free-electrons.com

注意：授權條款

- 簡報採用創意公用授權條款 (Creative Commons License: **Attribution-ShareAlike**) 發行
- 議程所用之軟體，依據個別授權方式發行