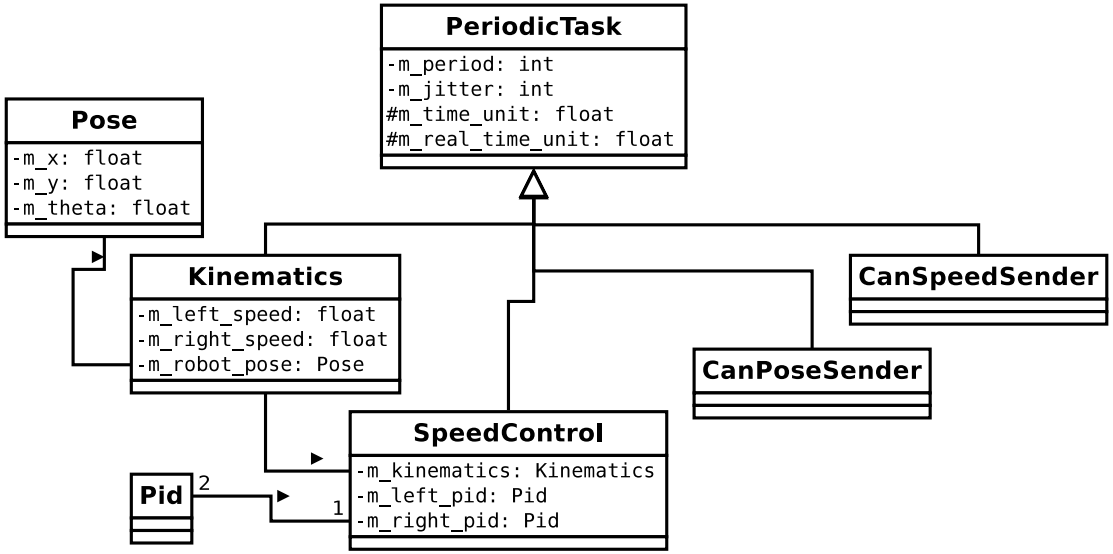
Architettura Software del Motion Control
Gestione dei Task Periodici

Corrado Santoro

ARSLAB - Autonomous and Robotic Systems Laboratory
Dipartimento di Matematica e Informatica - Università di Catania, Italy
santoro@dmi.unict.it

Programmazione Sistemi Robotici

Class Diagram



- Il software del motion control è basato su un insieme di **task periodici** ognuno dei quali effettua le attività di gestione e controllo della movimentazione.
- In più sono presenti alcune classi che rappresentano informazioni aggiuntive sullo stato del robot

Task Periodici

- Il Motion Control possiede un sistema di scheduling per i task periodici
- Ogni task periodico è rappresentato da una sottoclasse di **`PeriodicTask`** ed è caratterizzato dal **periodo di esecuzione** dove è stato ridefinito il metodo **`run()`**
- Il sistema di scheduling si occupa di eseguire il metodo **`run()`** di ogni PeriodicTask allo scadere del periodo relativo
- Il sistema si basa su un **quanto di tempo** prefissato T_{sched}: **il periodo di ogni task deve essere un multiplo intero di** T_{sched}
- Nel motion control T_{sched} è fissato a $5ms$
- Lo scheduling è **non-preemptive**, il metodo **`run()`** viene eseguito fino alla fine e non può essere interrotto
- Il metodo **`run()`** dovrà pertanto eseguire un singolo "step" del task periodico specifico

Classe PeriodicTask

```
class PeriodicTask {
 public:
    PeriodicTask(const char * name,
                 int period, float time_unit, int jitter,
                 bool insert_as_last = false);
    virtual void run() = 0;
    virtual void on() { m_on = true; };
    virtual void off() { m_on = false; };
    virtual bool on_status() { return m_on; };
    virtual void set_period(int v) { m_period = v; };
 private:
    void execute();
    const char * m_name;
    int m_ticks;
    int m_period;
    int m_jitter;
    bool m_on;
    PeriodicTask * m_task_next;
 protected:
    float m_real_time_period;
    float m_time_unit;
};
```

Classe PeriodicTask

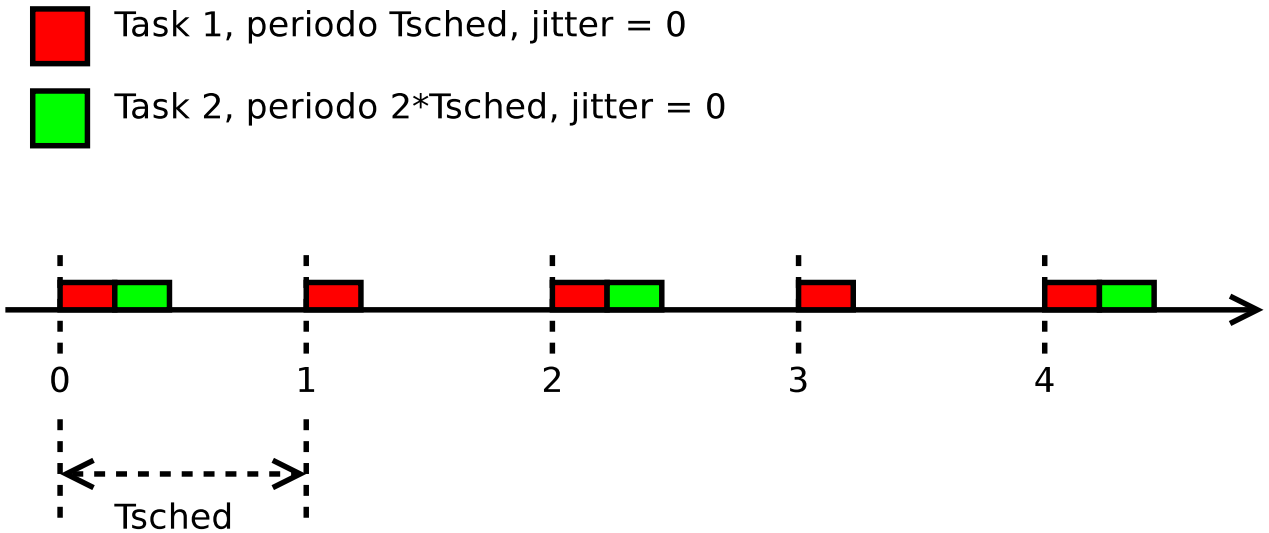
```
class PeriodicTask {
 public:
    PeriodicTask(const char * name,
                 int period, float time_unit, int jitter,
                 bool insert_as_last = false);
...
};
```

Parametri del costruttore:

- **name**, nome del task
- **period**, periodo, in multipli di T_{sched}
- **time_unit**, valore di T_{sched} in millisecondi
- **jitter**, latenza di esecuzione rispetto al periodo, in multipli di T_{sched}

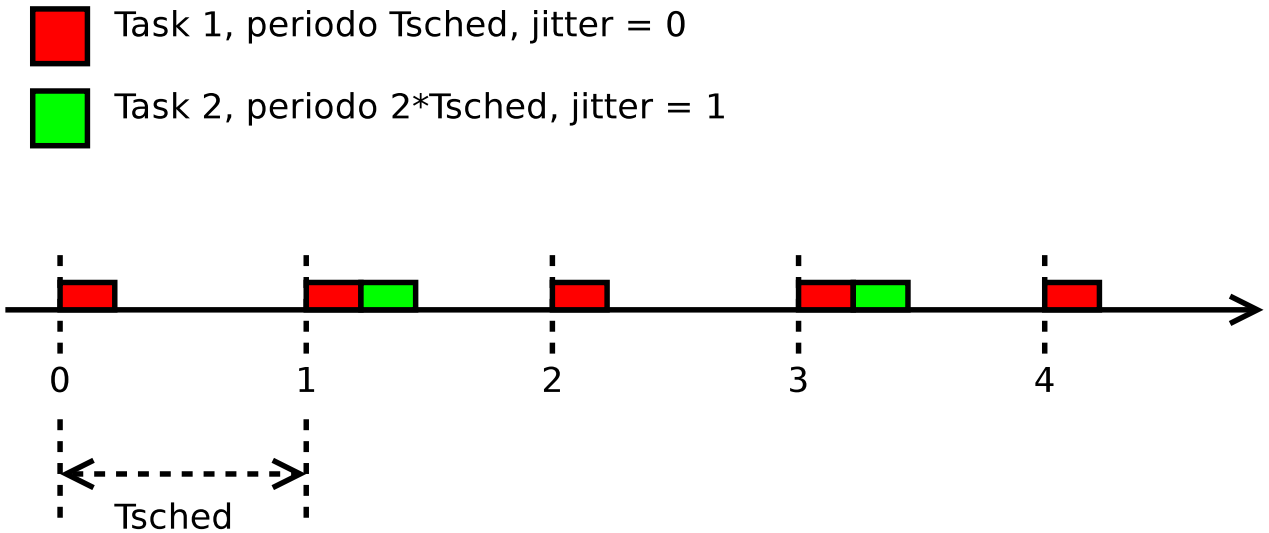
Periodo e Jitter

- Supponiamo di avere due task periodici, con periodi rispettivamente, T_{sched} e $2T_{sched}$ e $jitter = 0$
- L'esecuzione avverrà secondo quanto riportato in figura:

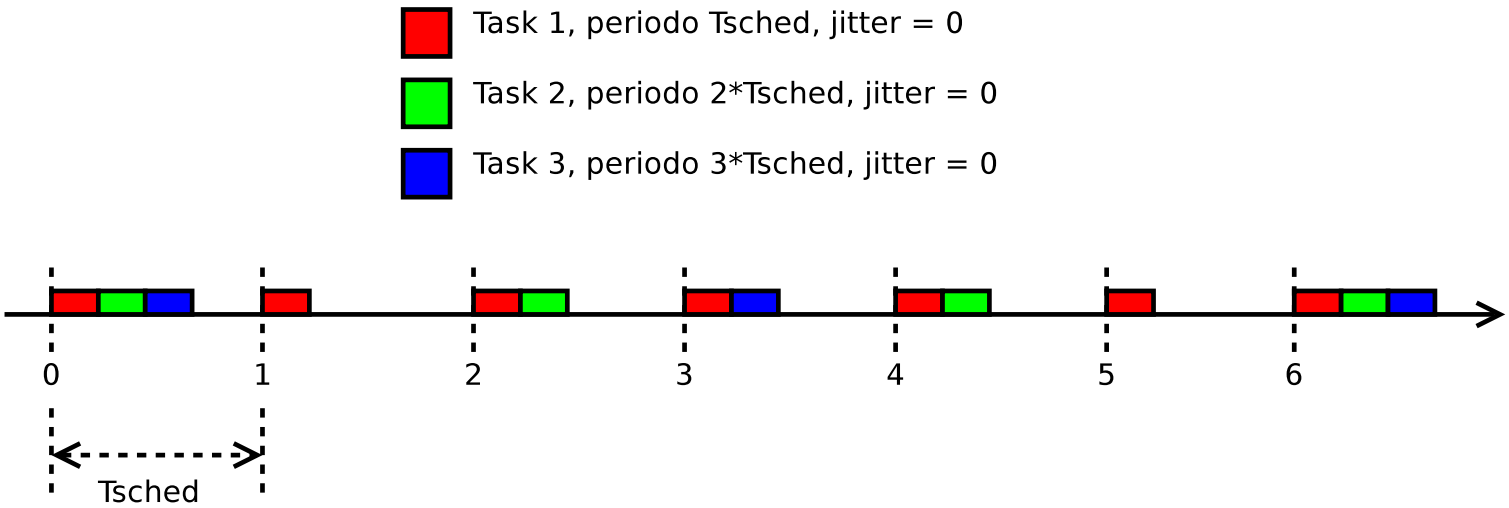


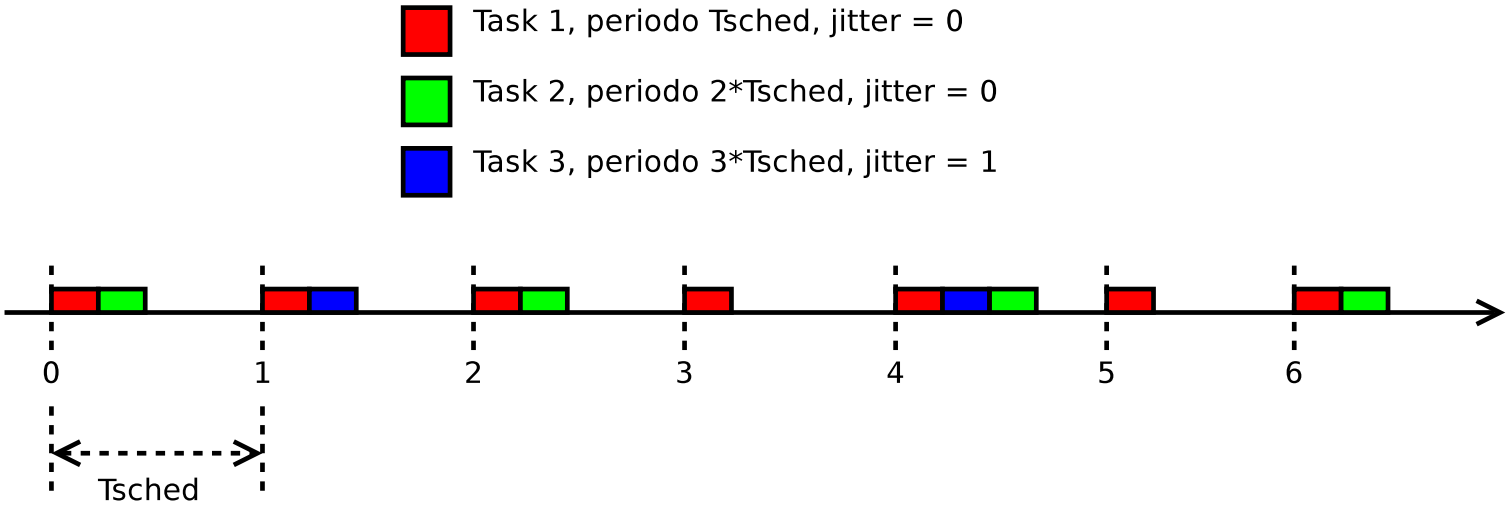
Periodo e Jitter

- Il *Jitter* specifica la **traslazione nel tempo** (in termini di numero di quanti T_{sched}) dell'esecuzione del task.
- Esempio:



- Esempio 2:


Task 1, periodo Tsched, jitter = 0
Task 2, periodo 2*Tsched, jitter = 0
Task 3, periodo 3*Tsched, jitter = 0
0
1
2
3
4
5
6
Tsched


Task 1, periodo Tsched, jitter = 0
Task 2, periodo 2*Tsched, jitter = 0
Task 3, periodo 3*Tsched, jitter = 1
0
1
2
3
4
5
6
Tsched

Classe PeriodicTask

```
class PeriodicTask {
 public:
    PeriodicTask(const char * name,
                 int period, float time_unit, int jitter,
                 bool insert_as_last = false);
    virtual void on() { m_on = true; };
    virtual void off() { m_on = false; };
    virtual bool on_status() { return m_on; };
};
```

Altri metodi:

- **`on`**, rende il task “schedulabile” (di default un task è sempre in off)
- **`off`**, rende il task “non-schedulabile”
- **`on_status`**, restituisce il flag on/off

Fattibilità dei task periodici

- Sia tr_i il **massimo tempo di esecuzione** del metodo `run()` per il task i-esimo.
- Siano $tr_1, tr_2, \ldots, tr_m$ i tempi massimi di esecuzione di **tutti i task in "ON"**
- Allora la seguente disuguaglianza è una **condizione sufficiente** per la corretta esecuzione dei task:

$$\sum_{i=1}^{m} tr_i \leq T_{sched}$$

Periodi di esecuzione dei task del MotionControl

- Tutte le definizioni dei periodi sono riportate nel file ``time_defines.h''
- Kinematics: 5 ms ($1\,T_{sched}$)
- SpeedControl: 10 ms ($2\,T_{sched}$)
- Telemetria velocità: 40 ms ($8\,T_{sched}$)
- Telemetria posizione: 250 ms ($50\,T_{sched}$)
- Controllo in posizione (da implementare): 40 ms ($8\,T_{sched}$)

Architettura Software del Motion Control
Gestione dei Task Periodici

Corrado Santoro

ARSLAB - Autonomous and Robotic Systems Laboratory
Dipartimento di Matematica e Informatica - Università di Catania, Italy
santoro@dmi.unict.it

Programmazione Sistemi Robotici