5.1

ARDUINO

Real time OS, FreeRTOS

Farouk MEDDAH

PLAN

- **REAL TIME OPERATING SYSTEM**
  - Présentation
  - Architecture vue générale
- **FREE RTOS**
  - Présentation
  - Contrôle des taches
  - Exemples de programmes

REAL TIME OPERATING SYSTEM

REAL TIME OPERATING SYSTEM

Système d'exploitation temps réel

- Un système multitâches destiné aux applications temps réel [Wikipédia].
- Il existe un variété de RTOS:
  - LynxOS, RT-Linux, Nucleus, uC/OS-II.
- ISP Flash memory, read-while-write.

REAL TIME OPERATING SYSTEM

Système d'exploitation temps réel

- Un système multitâches destiné aux applications temps réel [Wikipédia].
- Il existe un variété de RTOS:
  - LynxOS, RT-Linux, Nucleus, uC/OS-II.
- ISP Flash memory, read-while-write.

FREERTOS

FREERTOS (1)

- Ecrit par Richard Barry & FreeRTOS Team
- http://www.freertos.org/. (un téléchargement chaque 260s).
- Dernière version 9.0.0 16/08/2016
- RL78 peut créer 13 tasks, 2 queues et 4 software timers dans moins de 4K octets de RAM.
- Micro-kernel
  - 3 fichiers sources communs.
  - Un fichier spécifique pour chaque microcontrôleur.
- f

FREERTOS (2)

FreeRTOS v1.0.1

FreeRTOS v2.0.0

+Scalability
+New Task API

FreeRTOS v3.0.0

+API Changes
+Directory Names Changed
+Changes in Kernel

FreeRTOS v4.0.0

+Co-routines

FreeRTOS v5.0.0

+API Changes

FreeRTOS v7.6.0

+ Tick Suppression
+ Queue sets
+ Port optimized task selection

FreeRTOS v9.0.0

+ Completely Statically Allocated OS
+ Static allocated RAM (Tasks & objects)
+ Force task to leave blocked state
+ Deleting Tasks

FREERTOS (3)

- **Task (50%)**
  - task.c et task.h tout les routines de création, gestion des taches.
- **Communication (40%)**
  - queue.c et queue.h pour la gestion du communication.
  - Les taches et les interruptions échangent les données, et signalent l'utilisation des ressources critique par les sémaphores et les mutexes.
- **Hardware (10%).**

FREERTOS (4)

- **Main fonction**
  - La fonction main de FreeRTOS ne fait que la création des taches.
- **Contrôle des taches**
  - La tache ayant la priorité la plus haute s'exécute en premier et ne libère la CPU que si elle dort.
  - Si deux taches ou plus ont la même priorité, le temps CPU sera partagé entre eux.

FREERTOS (4)

- **Contrôle des taches (suite)**
  - **vTaskDelay()** Met la tache en état bloqué (sleep); [n'utiliser pas la fonction delay].
  - **xQueueSend()** Si la (Queue) où vous envoyez est pleine la tache sera bloquer (sleep).
  - **xQueueReceive()** Si la (Queue) où vous recevez est vide la tache sera bloquer (sleep).
  - **xSemaphoreTake()** La tache sera bloque si la sémaphore est prise par une autre tache.

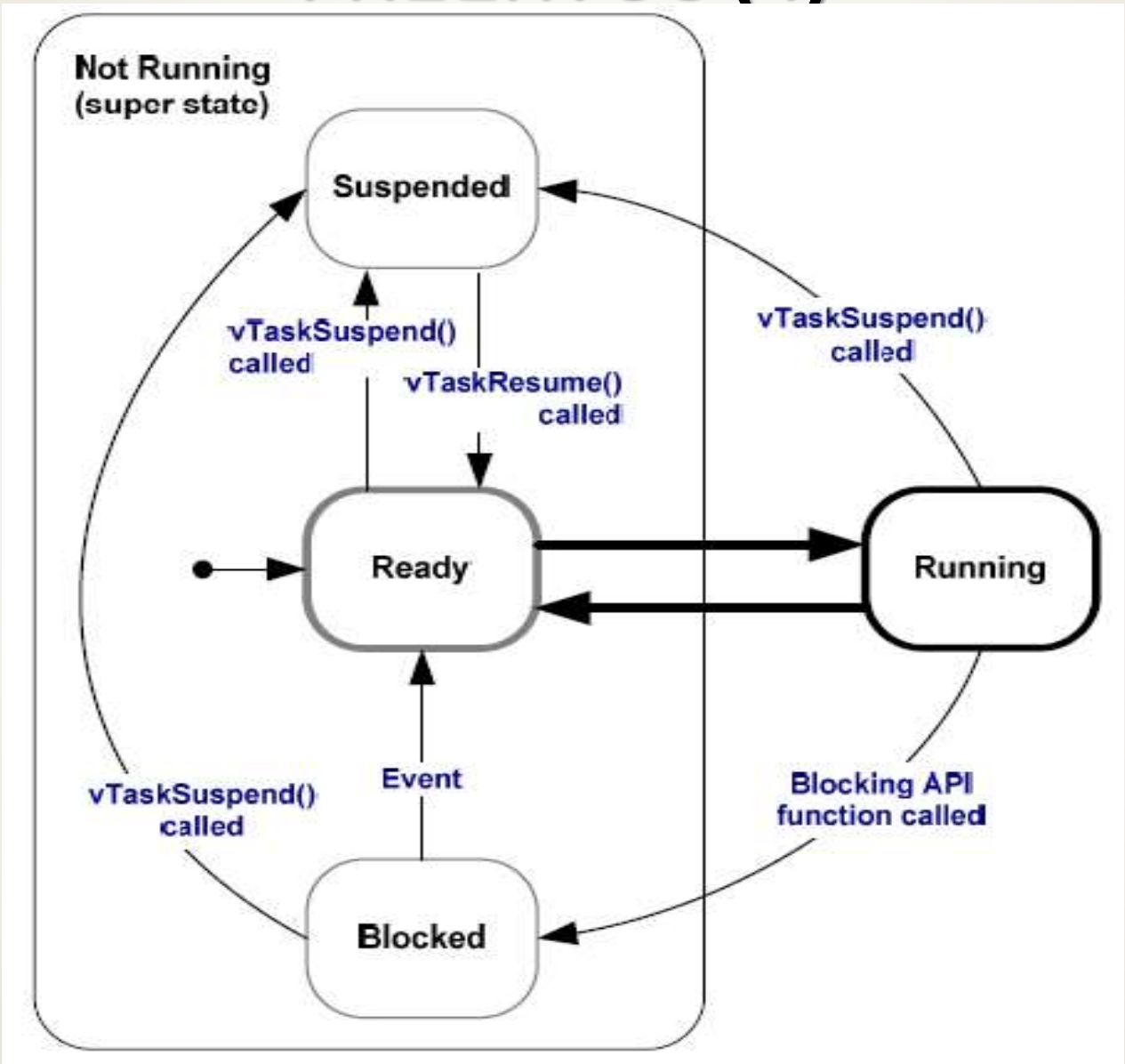
EXEMPLE

```
void hello_world_task(void* p)
{
    while(1) {
        puts("Hello World!");
        vTaskDelay(1000);
    }
}

int main()
{
    xTaskCreate( hello_world_task, (signed char*)"task_name",
         STACK_BYTES(2048), 0, 1, 0);
    vTaskStartScheduler();

    return -1;
}
```

FREERTOS (4)


Not Running
(super state)
Suspended
vTaskSuspend()
called
vTaskResume()
called
vTaskSuspend()
called
Ready
Running
vTaskSuspend()
called
Event
Blocking API
function called
Blocked

EXEMPLE (1/3)

```
#include <Arduino_FreeRTOS.h>

void TaskBlink( void *pvParameters );
void TaskAnalogRead( void *pvParameters );

void setup() {
   xTaskCreate( TaskBlink;  (const portCHAR *)"Blink",  128,  // Stack size
        NULL,  2,  /* priority*/  NULL );
   xTaskCreate( TaskAnalogRead,  (const portCHAR *) "AnalogRead“,
       128 ,  NULL,  1,  /*priority*/  NULL );
}

void loop()
{
  // Empty. Things are done in Tasks.
}
```

EXEMPLE (2/3)

```
void TaskBlink(void *pvParameters)  // This is a task.
{
  (void) pvParameters;
  pinMode(13, OUTPUT);

  for (;;) // A Task shall never return or exit.
  {
    digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
    vTaskDelay( 1000 / portTICK_PERIOD_MS ); // wait for one second
    digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
    vTaskDelay( 1000 / portTICK_PERIOD_MS ); // wait for one second
  }
}
```

EXEMPLE (3/3)

```
void TaskAnalogRead(void *pvParameters)  // This is a task.
{
  (void) pvParameters;

  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);

  for (;;)
  {
    // read the input on analog pin 0:
    int sensorValue = analogRead(A0);
    // print out the value you read:
    Serial.println(sensorValue);
    vTaskDelay(1);  // one tick delay (15ms) in between reads for stability
  }
}
```

MERCI