Multitâche, système embarqué

Microclub Lausanne, 2 mai 2014

Gestion d'un système multitâche 32 bits. Théorie et pratique

1. Définitions

Multitâche temps réel

Exécution instantanée ?

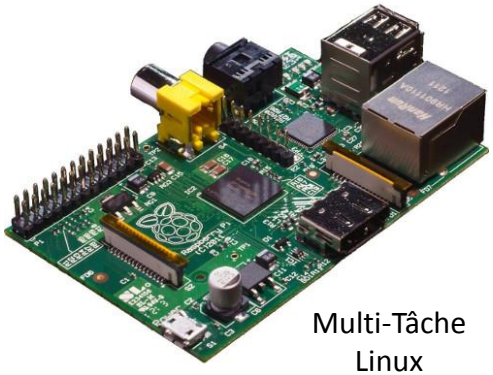
- Système organisé
- Temps de réponse prévisible (temps de réaction à un événement)
- Remplit les tâches qui lui sont imposées
  - Soft real-time (jeux vidéo)
  - Hard real-time (missiles)
  - Firm real-time (navigation)

Système embarqué

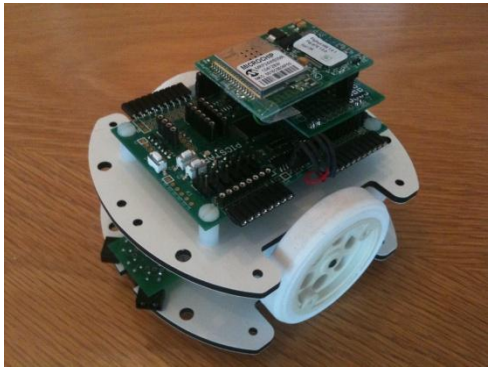
- Composé de un ou plusieurs ordinateurs/processeurs
- A un rôle central dans un système, qui n'est pas obligatoirement un ordinateur (PC)
- Machine à laver, Micro-onde, TV, Radio contiennent de nous jours des systèmes embarqués.

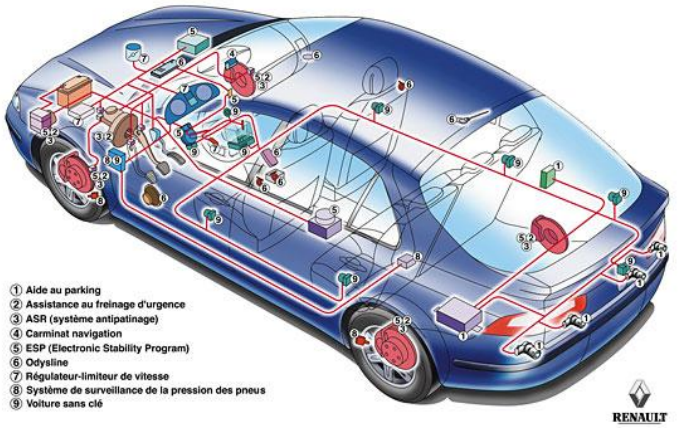
Systèmes embarqués
Exemples

Multi-processeur



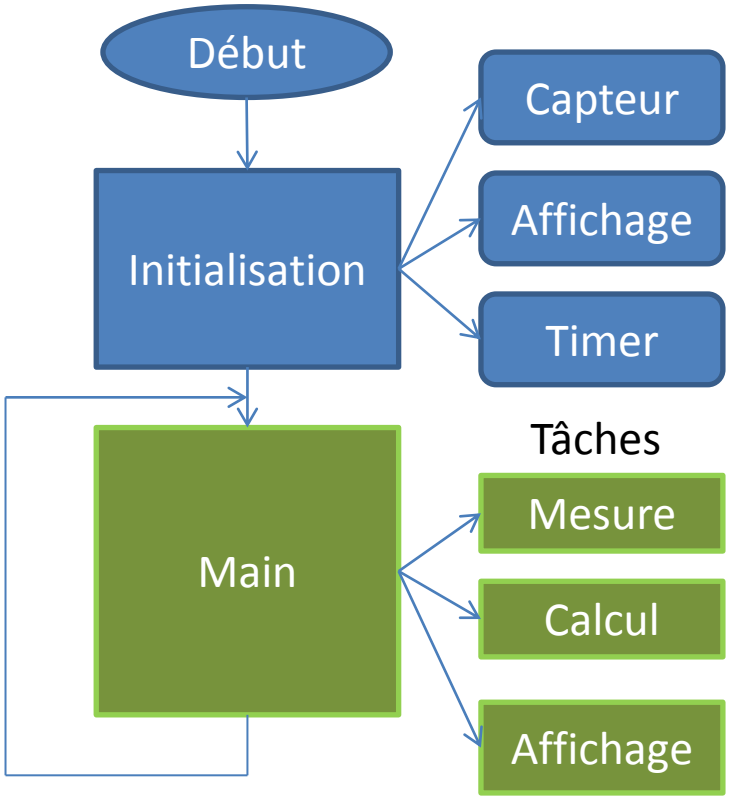
Multi-Tâche
Linux
Androis



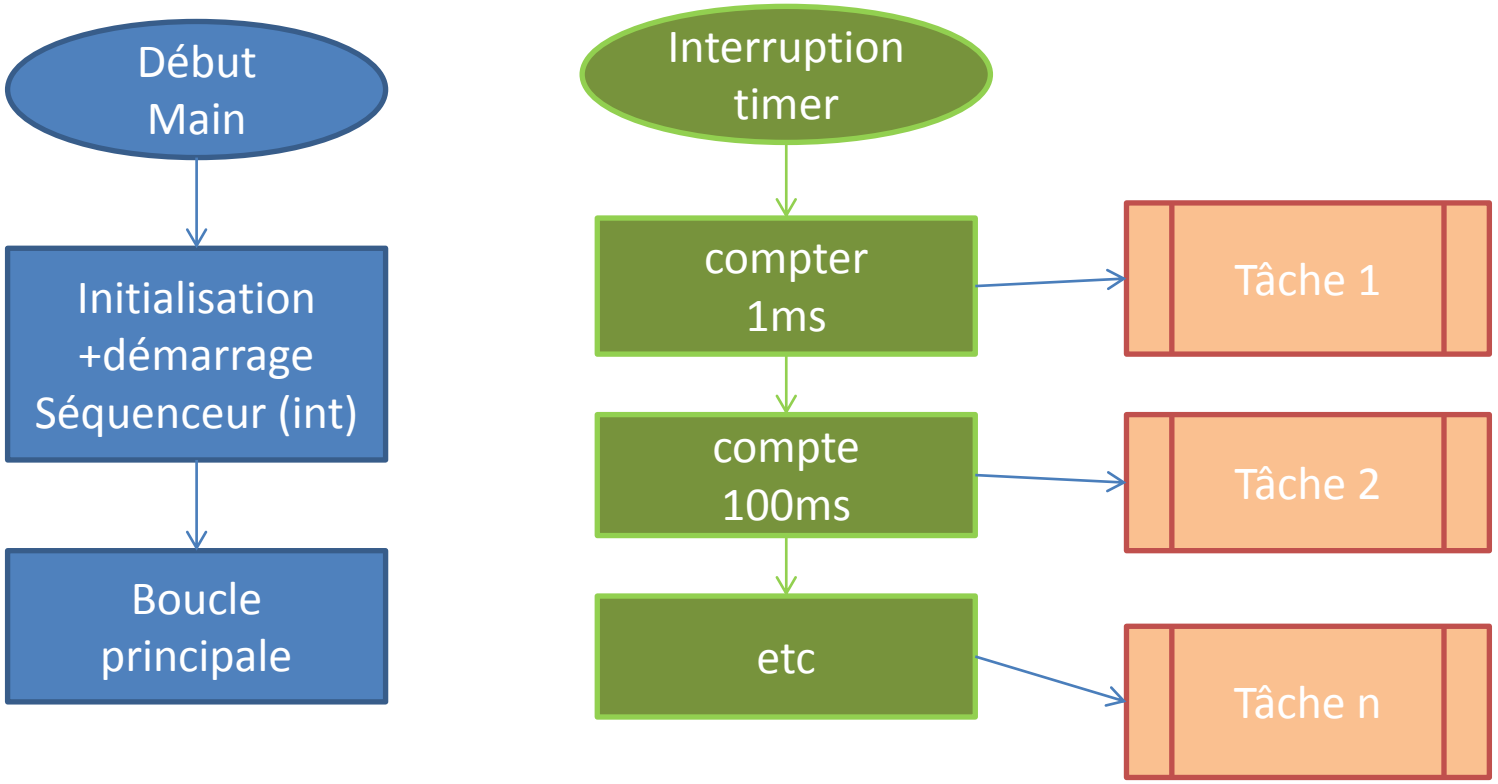


2. Exemple système simple exécution séquentielle

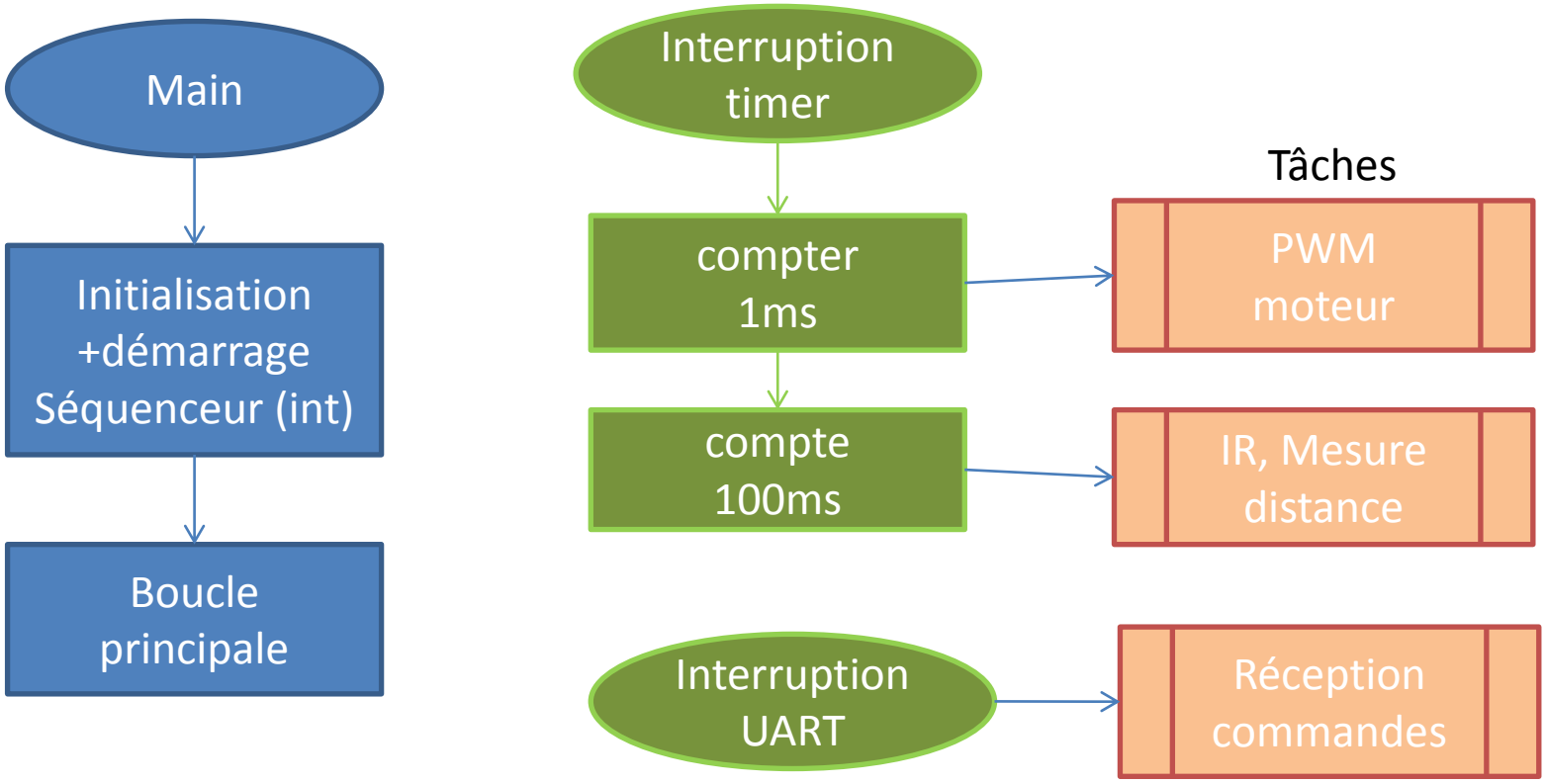
- Compte tours
  - Capteur divers
  - Mesure d'impulsions
  - Calcul de n/temps
  - Affichage du résultat



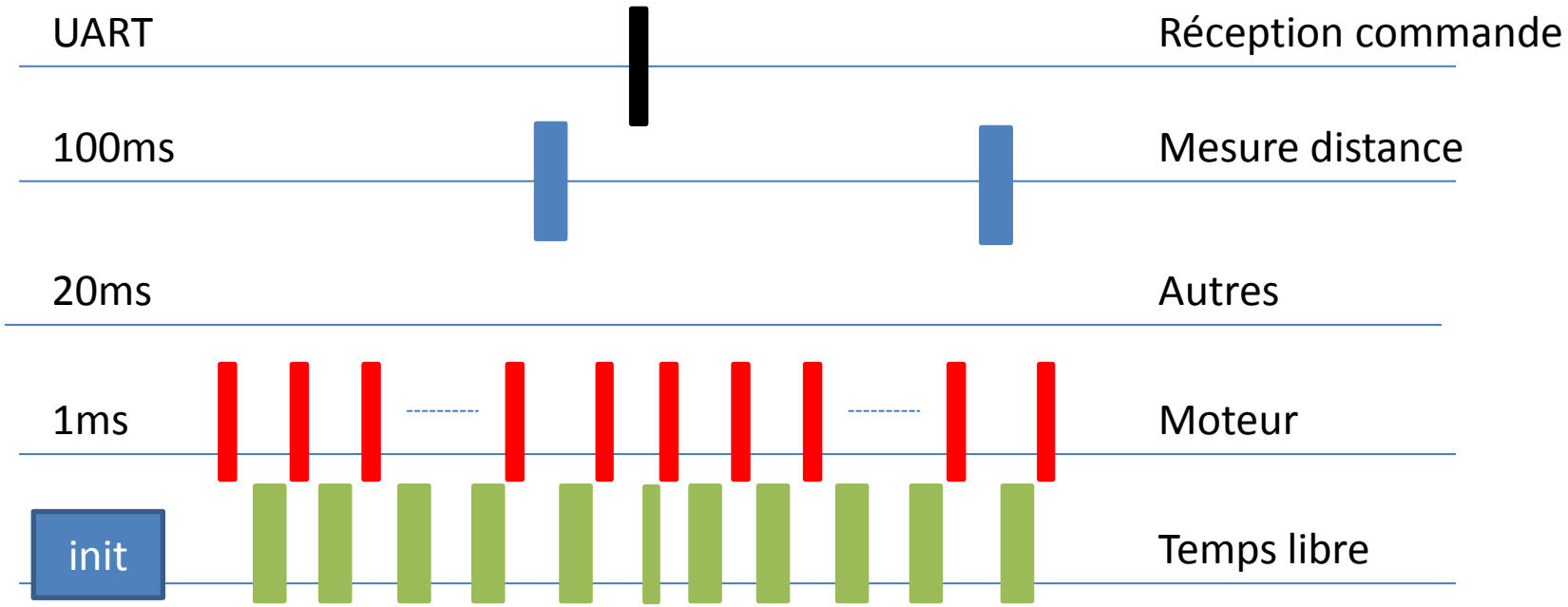
Multitâches
avec séquenceur



Multitâches
Exemple Robot



Multitâches TR simple



4. Utilisation d'un RTOS
Exemple: Station météo

433.92mHz

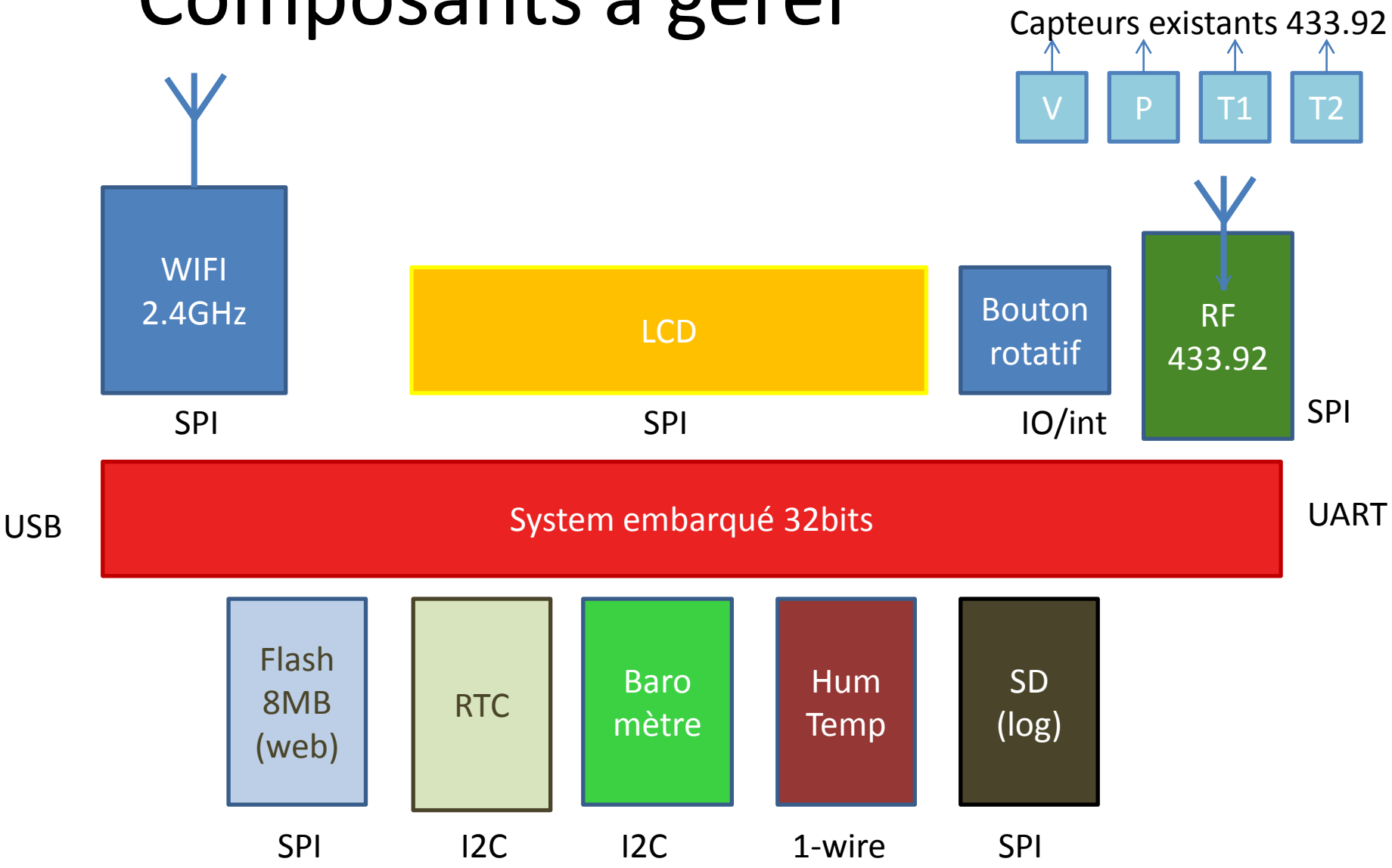
433.92mHz

wifi

Low-cost

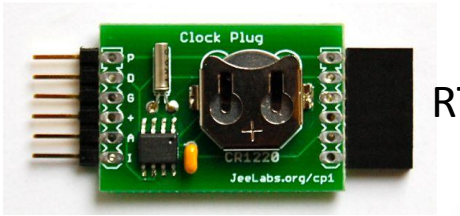
Wind 12
Temp 24

Composants à gérer

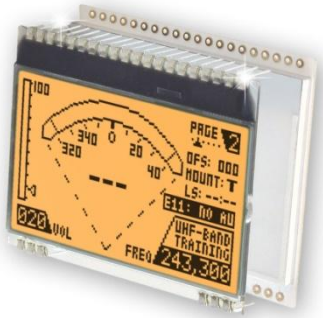


Projet Station météo

WIFI



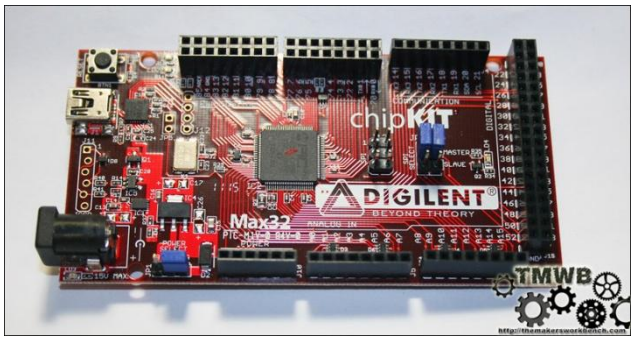
RTC



LCD 128x64

Encodeur rotatif

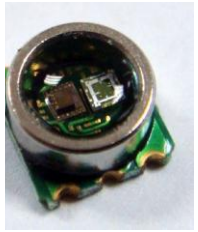
USB



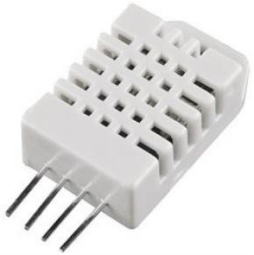
Carte SD

RF433.92

FLASH 8M

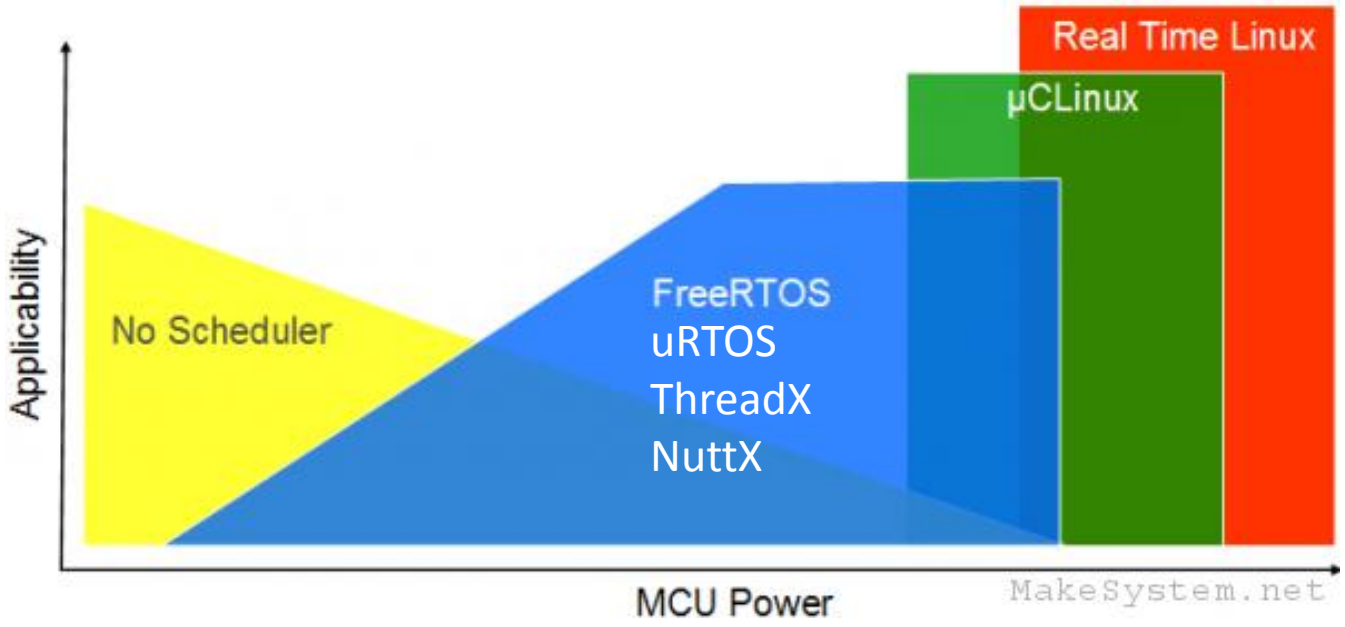


BARO



Humidité DHT22

5. Positionnement des Solutions Multitâches



Système 8 bits ⟶ Système 32 bits

6. Choix d'un système multitâches

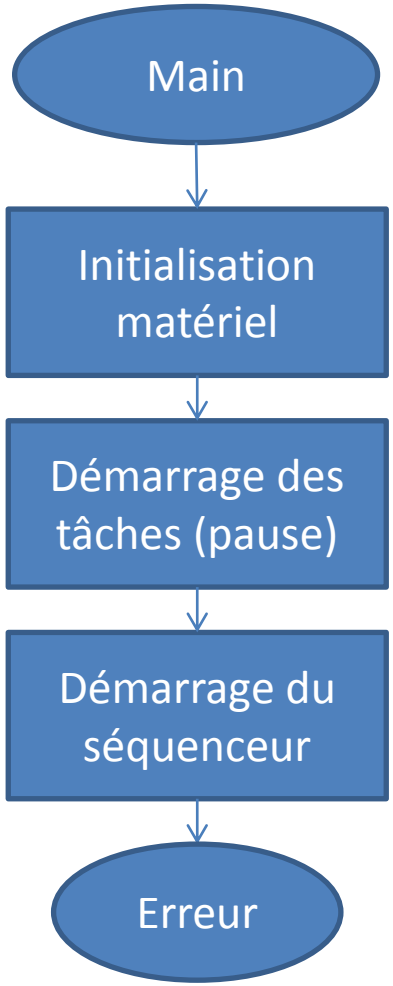
- Quel RTOS choisir
  - Compatible TCP-Stack (wifi)
  - Bonne documentation
  - Exemple disponibles
    - Wifi + Int ext,…
  - Supporte de multiple plateformes
  - Si possible gratuit
- FreeRTOS
  - Simple
  - Paramétrisable
  - Light (5-10kb rom, 256bytes ram for scheduler, 76 bytes/ queue, 64 bytes /task + stack)
- Processeurs
  - ARM
  - AVR
  - MPS430
  - PIC18, 24, 32
  - etc
- Choix
  - uRTOS
  - ThreadX
  - NuttX
  - FreeRTOS

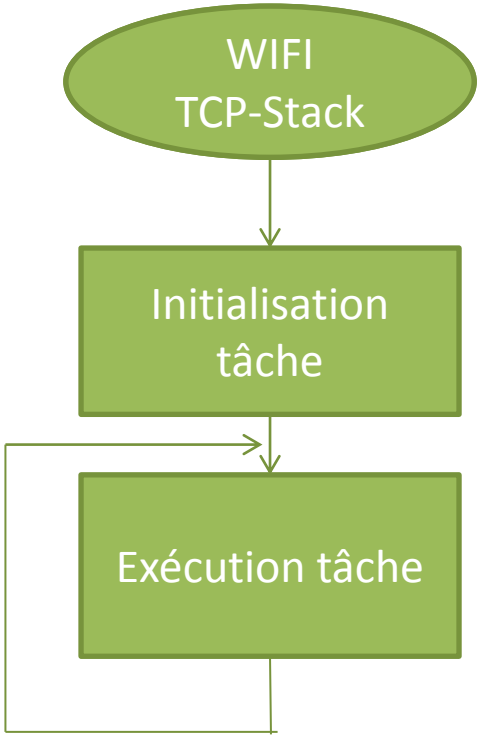
7. Mise en route d'un RTOS Commet s'y prendre?

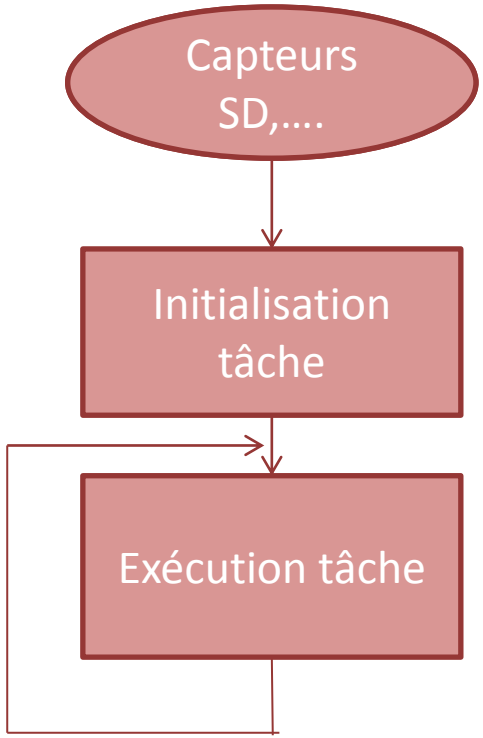
Read the Fxxx manual

- Définir les tâches
- Définir des dépendances
- Synchroniser les actions
- Définir les priorités des tâches
- Gérer les périphériques (vitesse, fréquence d'accès)
- Coordonner l'accès aux périphériques

8. Concept Multitâche

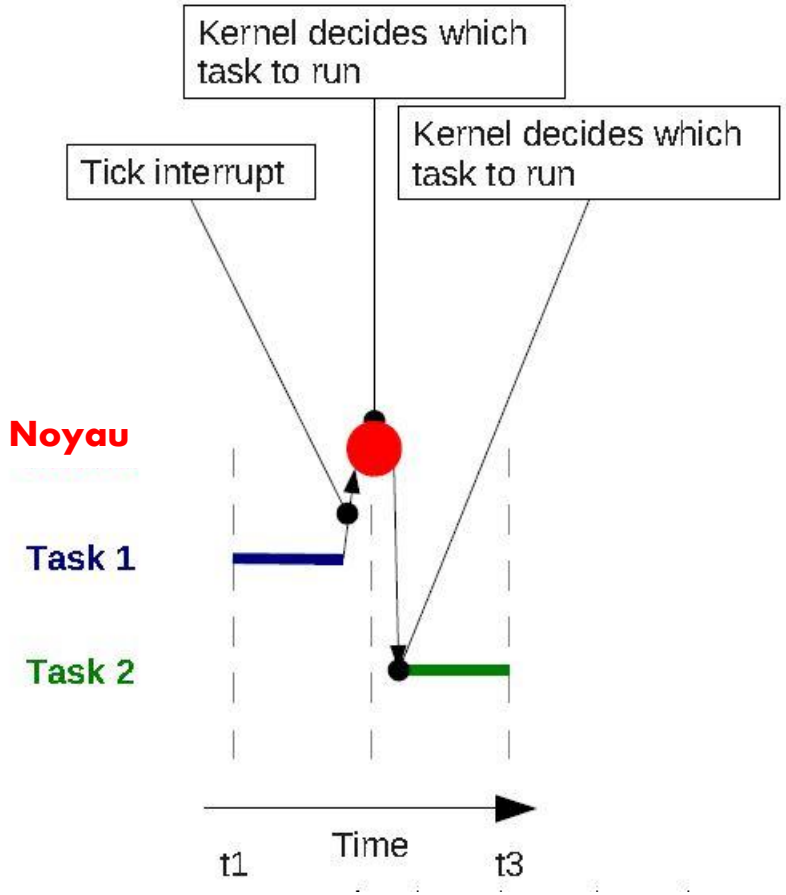






Une fois les tâches démarrées, le séquenceur va commuter les tâche en fonction d'un paramètre temps défini et d'une priorités allouées.

9. Gestion des tâches, Commutation

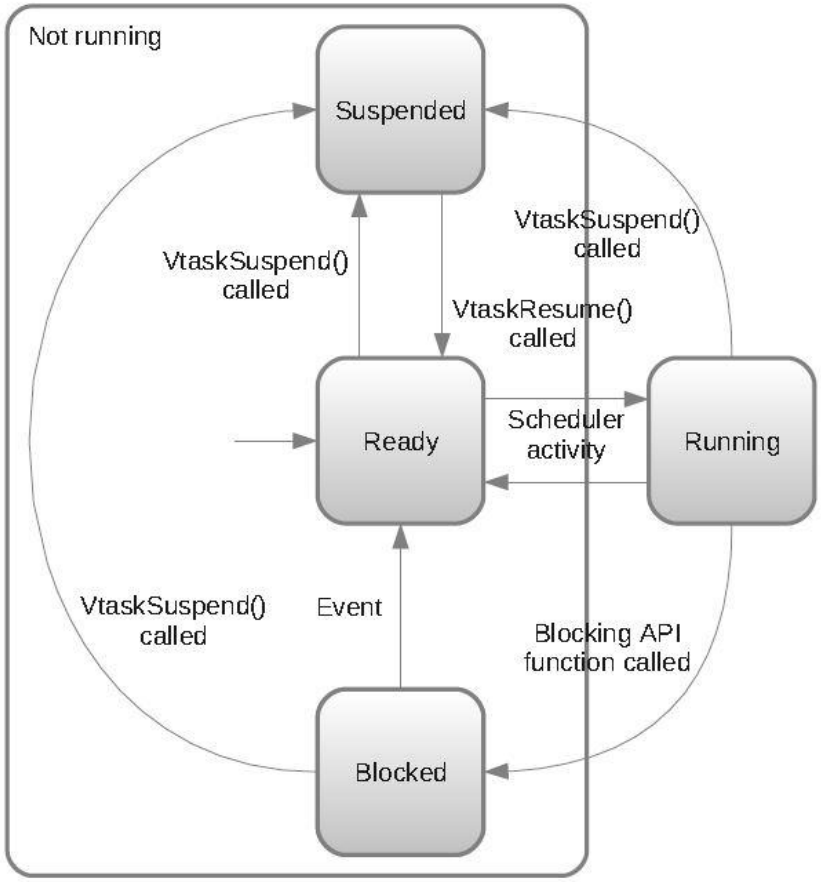


Une horloge interne (tick) permet à un séquenceur (scheduler) de commuter les tâches

La tâche suivante sera choisie en fonction de sa priorité, priorité définie au démarrage ou en fonction du déroulement du programme.

Avec un processeur tournant a 80MHz, on effectue 80’000 instructions en 1ms

Etats et Fonctions des tâches



Fonctions de gestion des tâches

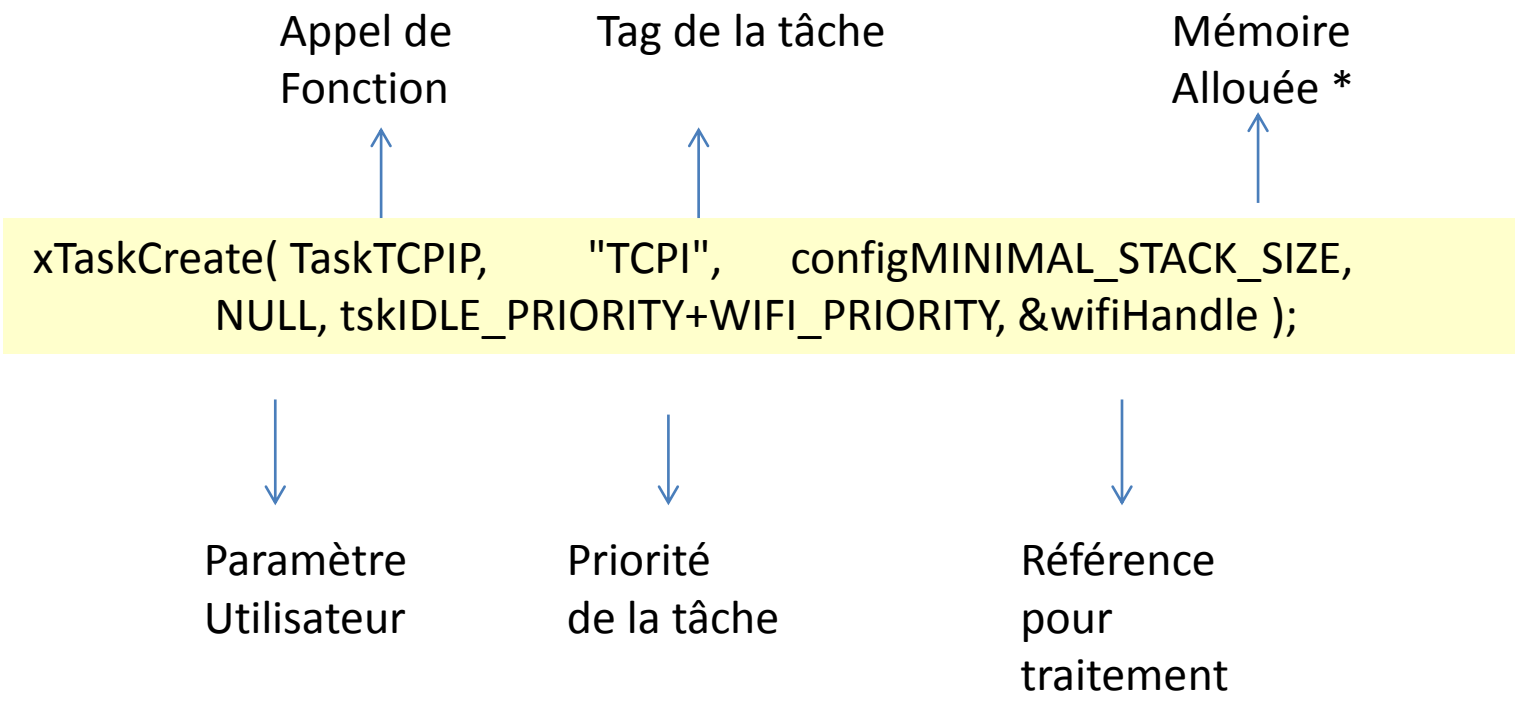
xTaskCreate(…)

vTaskPrioritySet(taskhandle, newpriority)

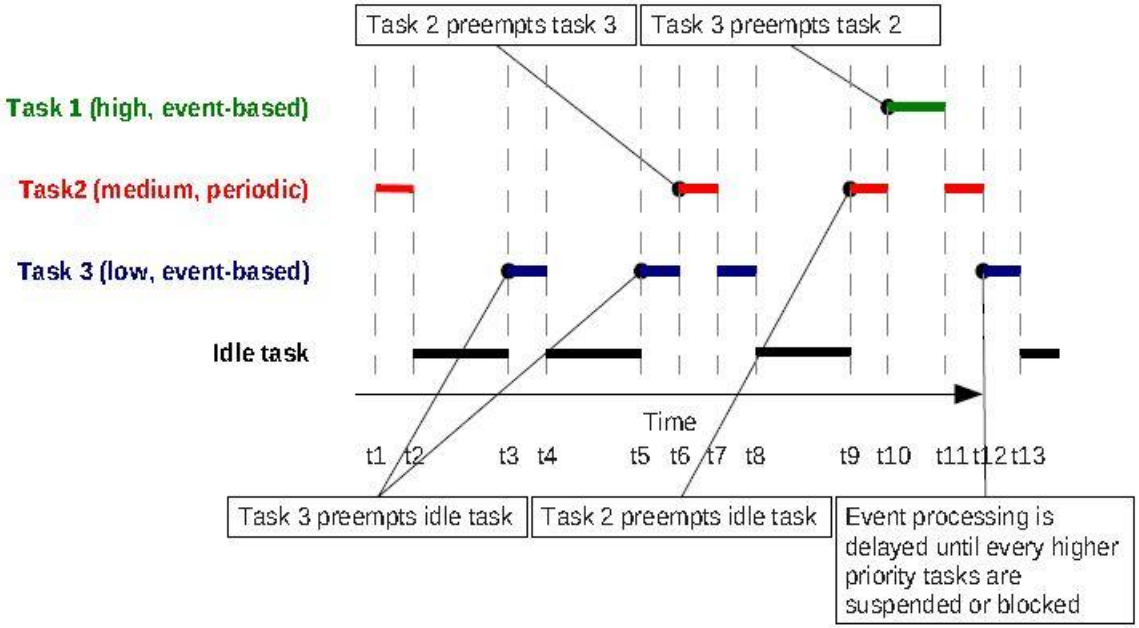
vTaskSuspend ()
vTaskSuspendAll()
vTaskDelete()
vTaskDelay()
vTaskDelayUntil()
….

taskhandle = nom de la tâche

Démarrage de tâches

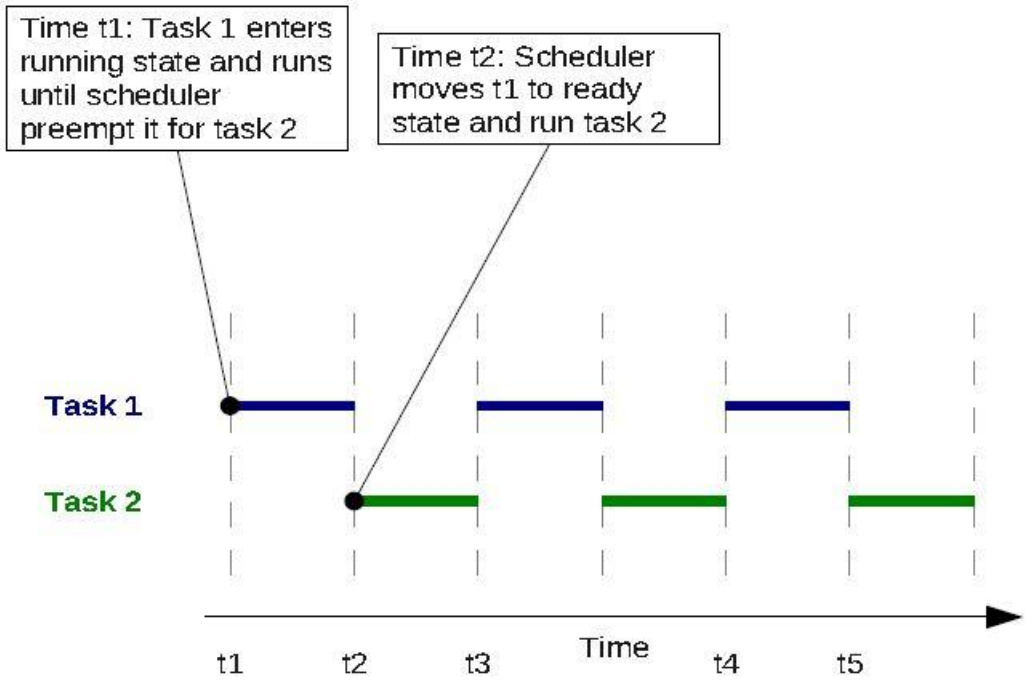


Priorité des tâches

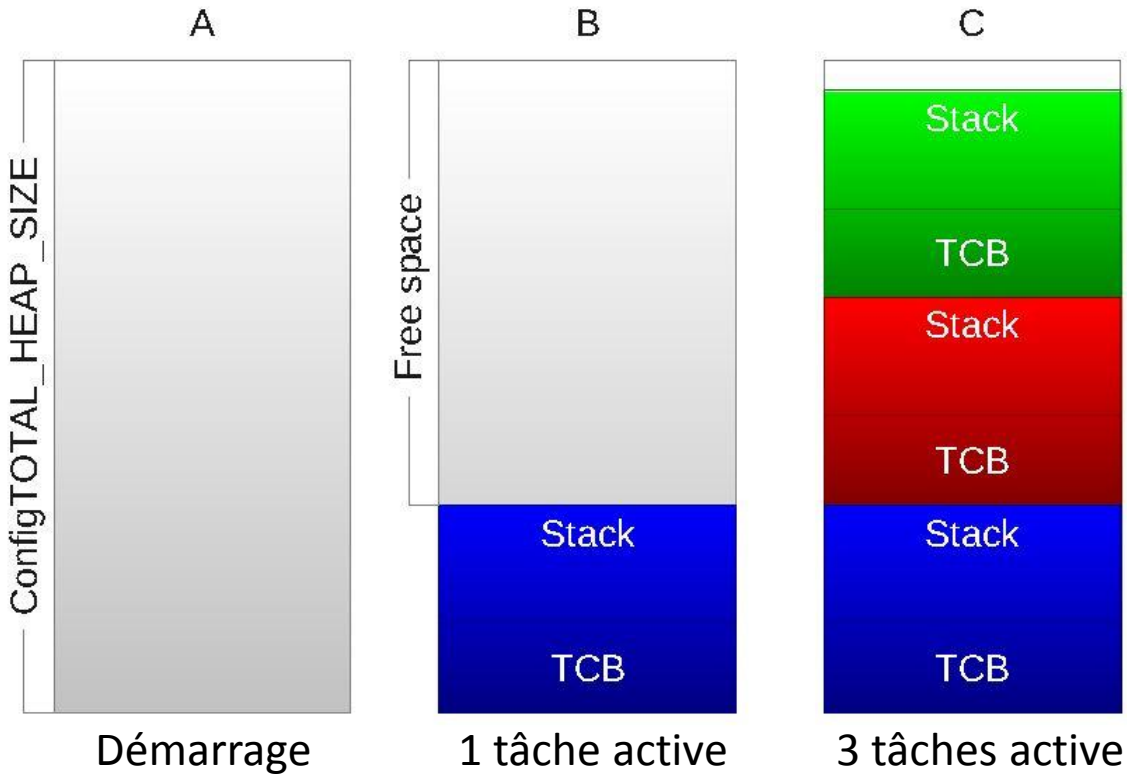


Une tâche de haute priorité ne doit monopoliser le CPU pour éviter un blocage du système. FreeRTOS n'est pas protégé contre un blocage.

Priorité égale des tâches



Gestion de mémoire des tâches

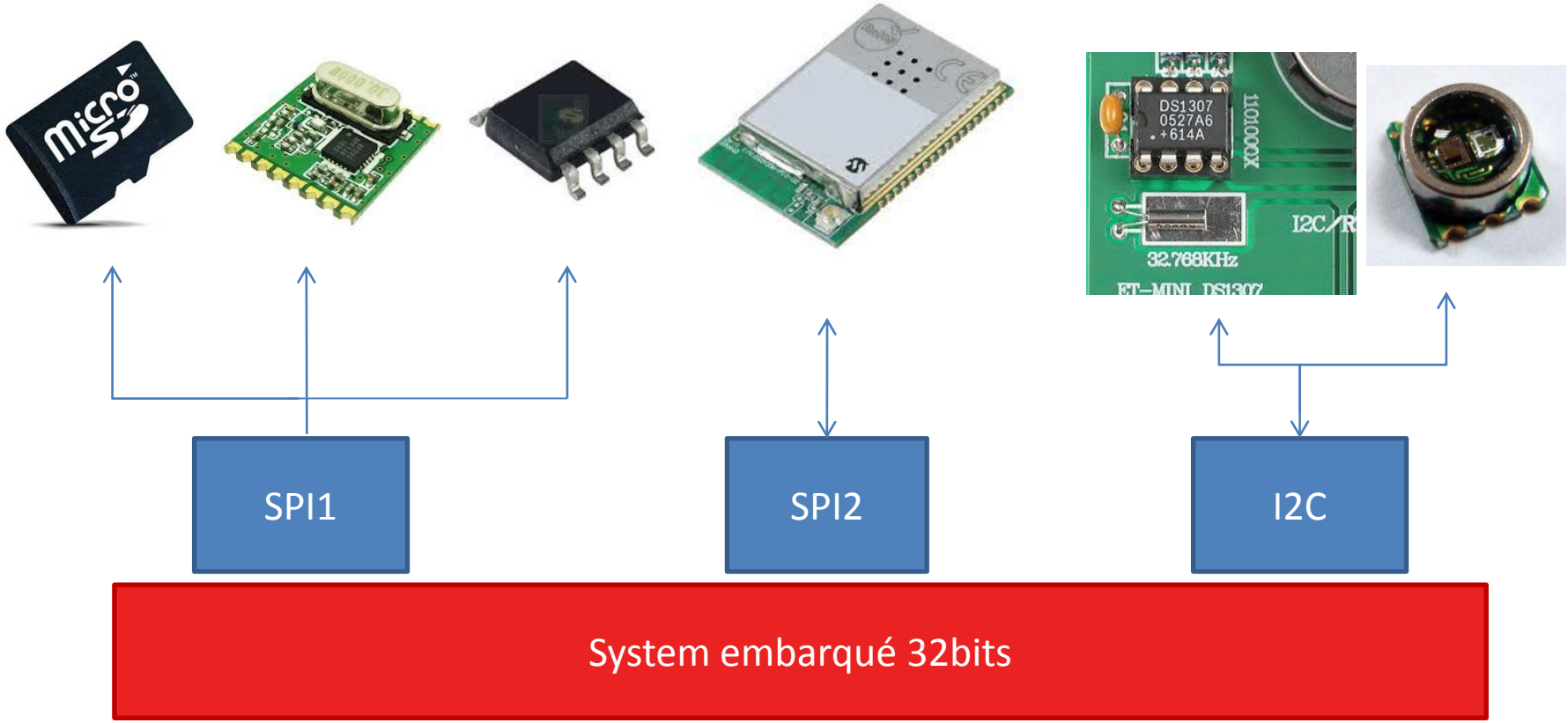


On réservera assez de mémoire « heap_size » pour faire tourner le max. de tâches prévues dans les paramètres de FreeRTOS

TBC = Task Control Block

10. Gestion des périphériques

Certains périphériques sont utilisés pas plusieurs tâches, il faut donc coordonner l'utilisation à fin d'éviter des mal fonctions du système.



Gestion de périphérique
Utilisation exclusive

Dans un avion on désire un utilisation exclusive des WC

On signale si les WC sont occupés par in signe visuel, on bloque la ressource avec une clef. Ainsi l'utilisation exclusive est assurée.

Plusieurs tâches écrivent sur le même interface. Exemple UART (Interface série)

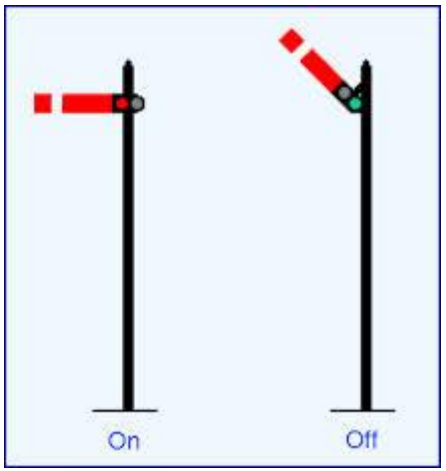
La tâche 1 envoie « 1234 »

La tâche 2 envoie « ABCD »

Sans coordination, on obtiendrait un mélange de données
Style « 1A2BC3 »
Alors que nous désirons
'1234'
'ABCD'

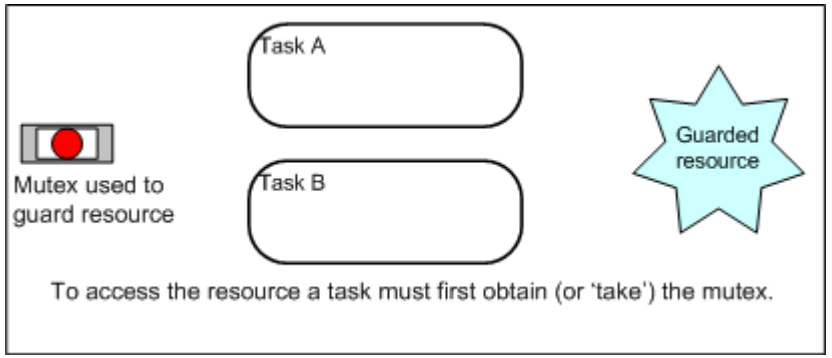
Gestion de périphériques
Sémaphores

Nous allons donc utiliser un Indicateur (sémaphore) comme Indicateur d'occupation



Et il nous faudra bloquer l'utilisation de la ressource jusqu'à achèvement du travail (de la tâche)

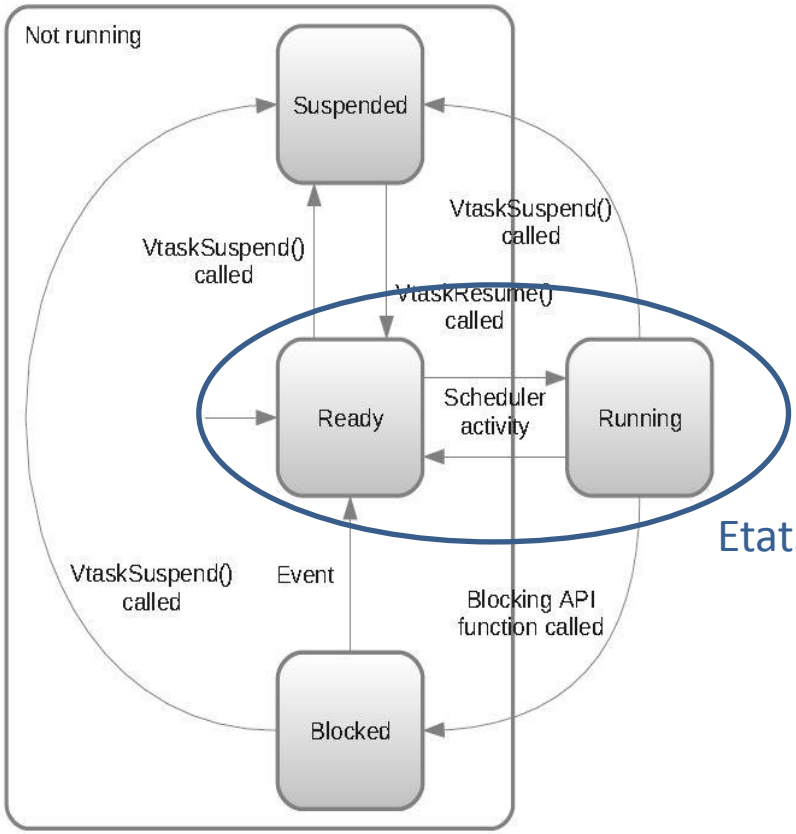
Fonction Sémaphore (exemple FreeRTOS)



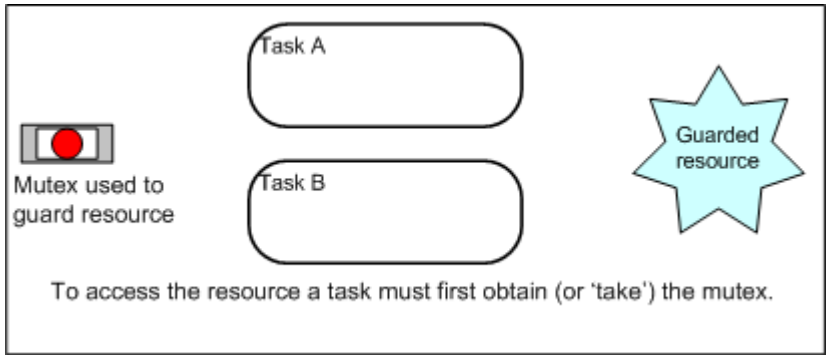
Fonctions Semaphore

- xSemaphoreCreateMutex
- xSemaphoreCreateBinary
- xSemaphoreTake
- xSemaphoreGive

Gestion de périphériques
Sémaphores



Fonction Sémaphore (exemple FreeRTOS)

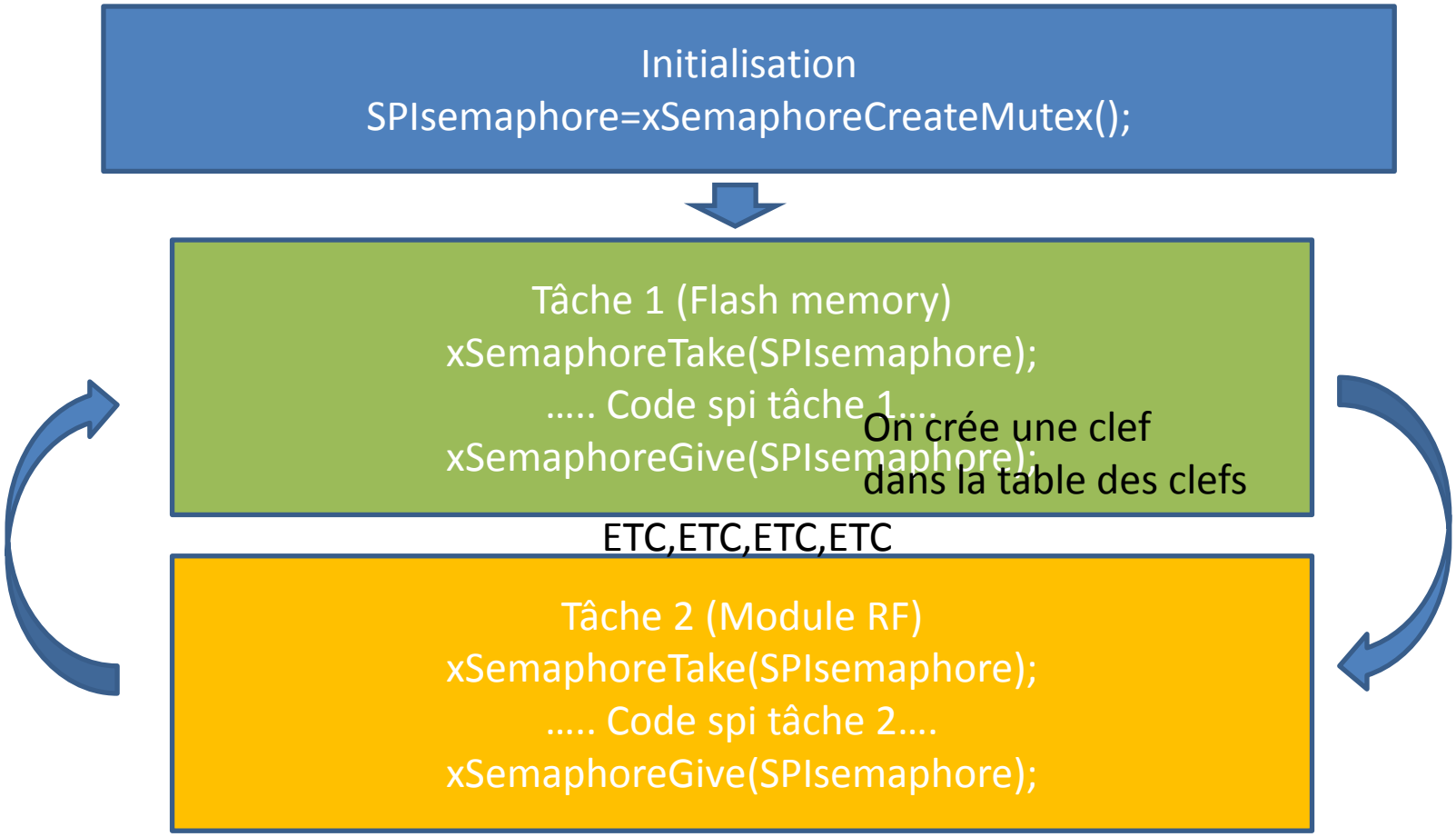


Etats de base

Fonctions Semaphore

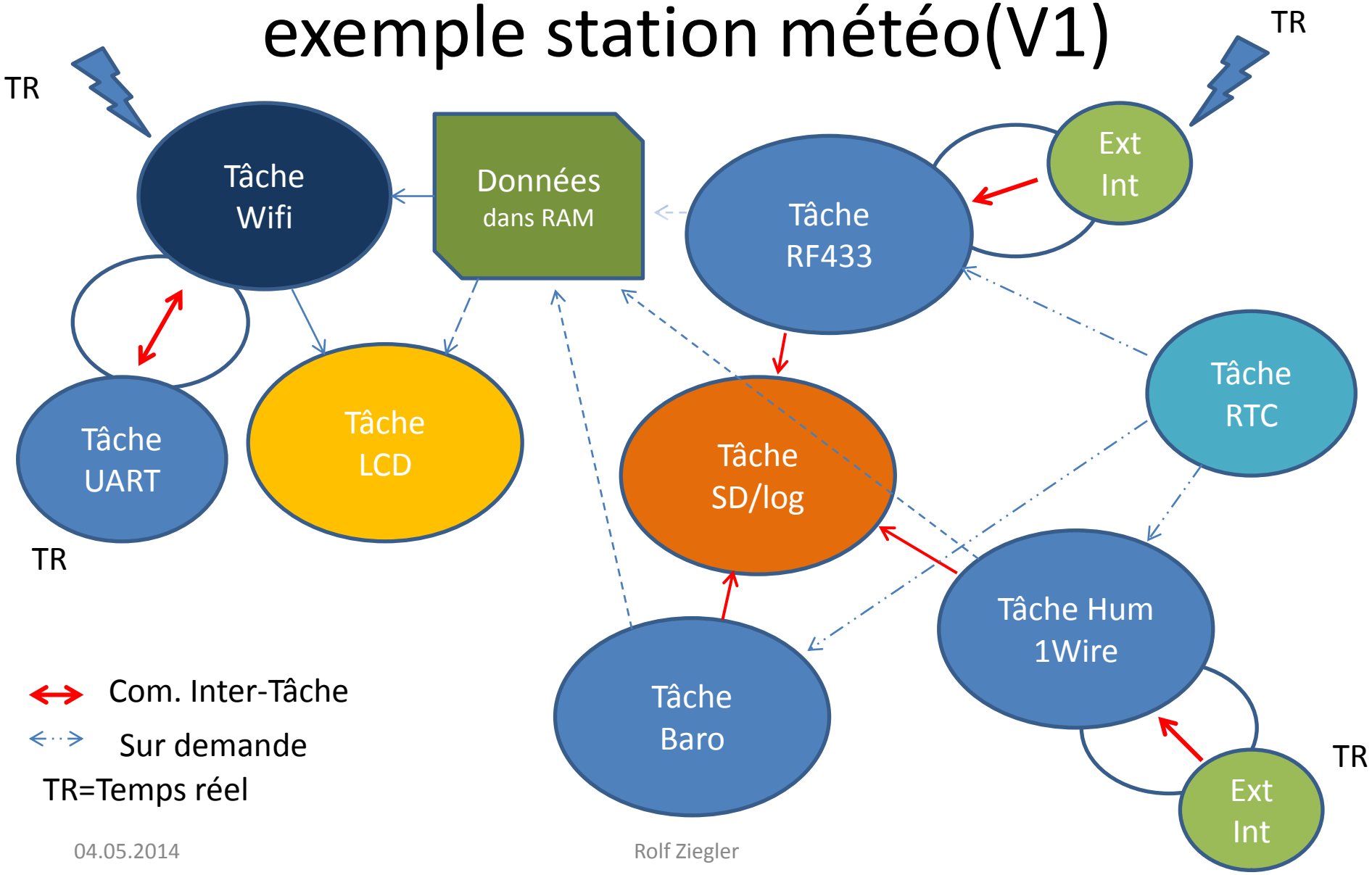
- xSemaphoreCreateMutex
- xSemaphoreCreateBinary
- xSemaphoreTake
- xSemaphoreGive

Fonctions Sémaphore
Utilisation, exemple interface SPI

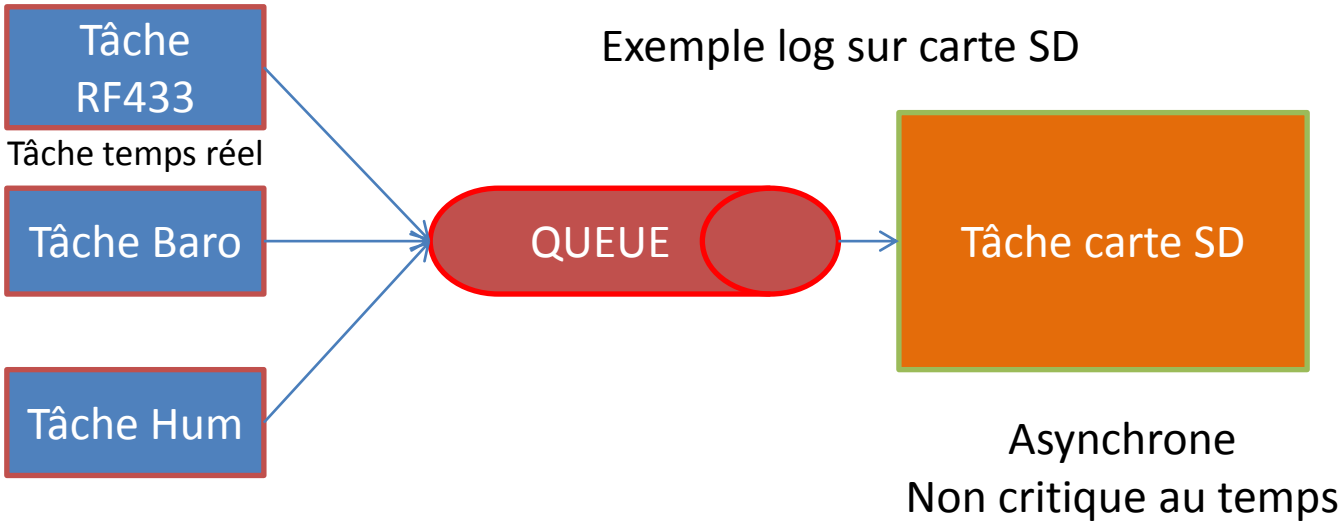


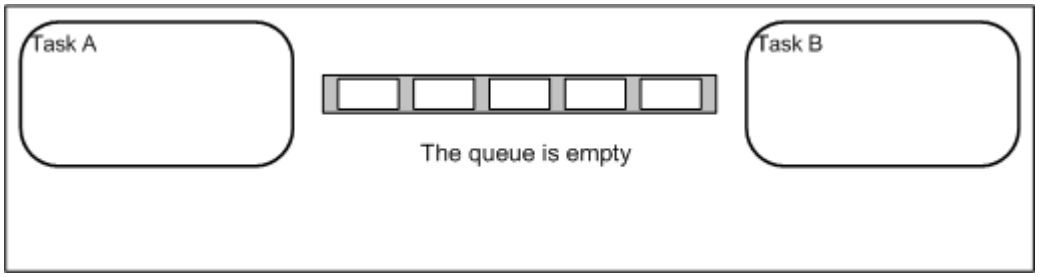
Il n'y a pas de limites du nombre de tâches qui peuvent utiliser une ressource.

11. Communication entre tâches exemple station météo(V1)

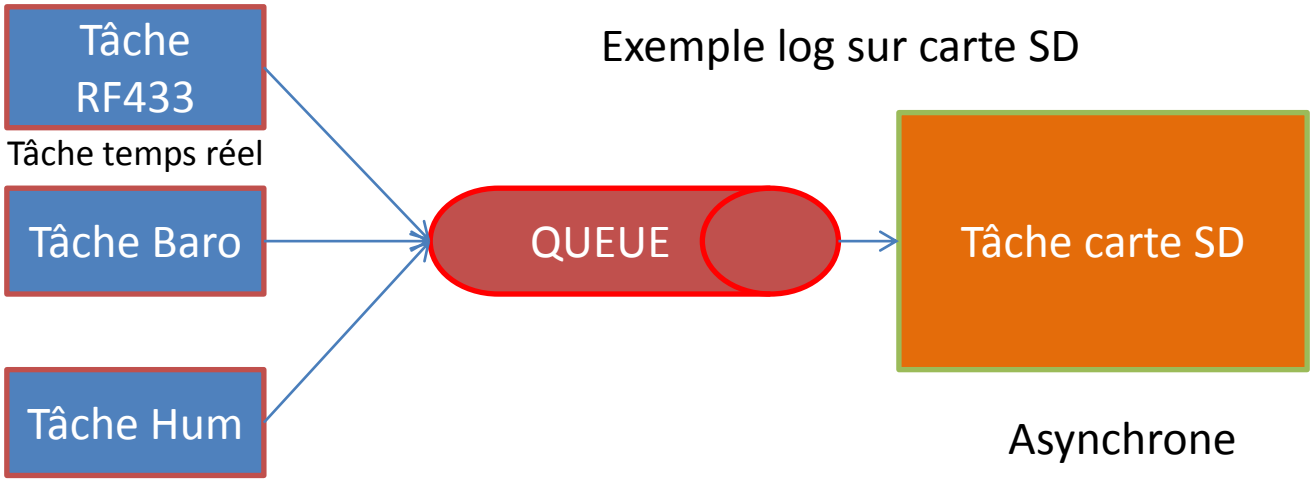


Communication entre tâches
Notion de « Queue »





Communication entre tâches
Notion de « Queue »

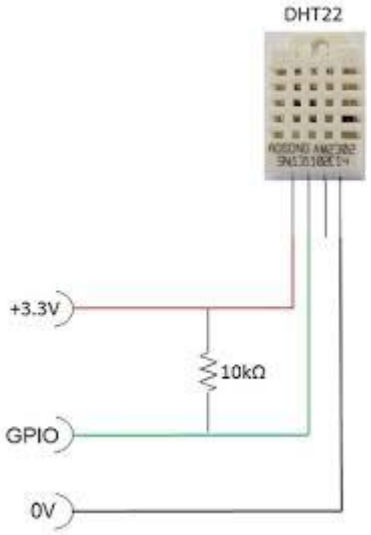


Une structure de données permet de copier un paquet de données simultanément, par exemple: Capteur, humidité, température, heure/date

Exemple chaine de caractères provenant de plusieurs tâches

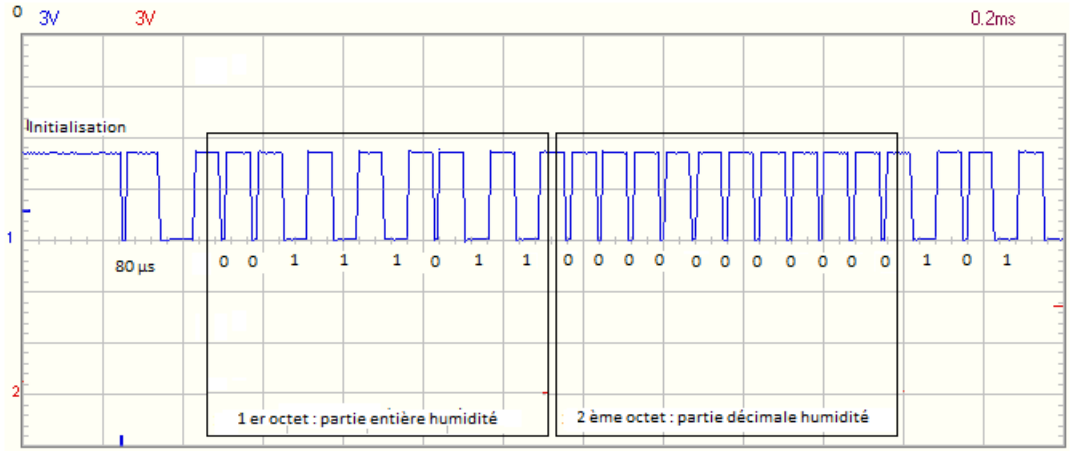
2014-05-01 00:47:33,DHT2,24.2,32.8
2014-05-01 00:47:33,HP03,1011.1,08.9
2014-05-01 00:47:35,TEMP,df,1,35,26.9

Communication entre tâches
Exemple DHT22, Capteur 1Wire



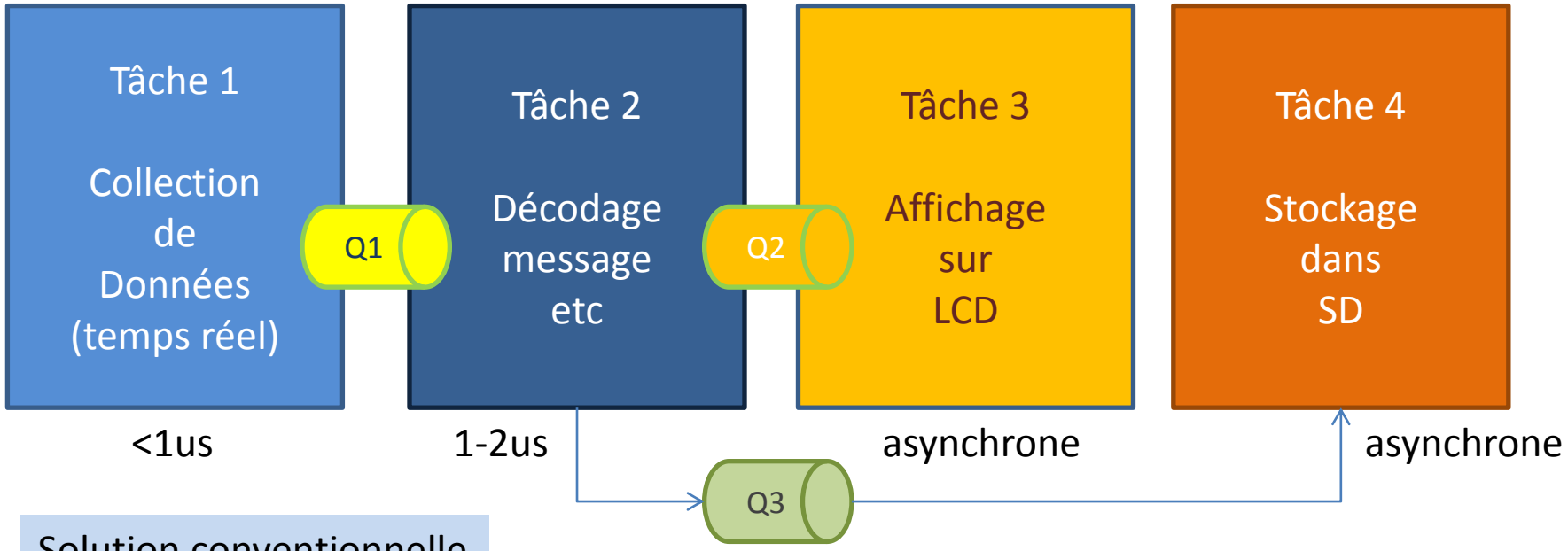
Durée de la trame ~ 700+ 40*50=2700us (2.7ms)
Avec notre processeur ceci équivaut a 108000 instructions!

1. Mettre la ligne GPIO en sortie
2. Envoyer une impulsion (neg) 500us
3. Mettre la ligne en entrée
4. Lire les 40 bits envoyés par le capteur



Communication
Asynchrone avec des queues

Exemple DHT22, 1-Wire (temps total de la transmission 2ms)

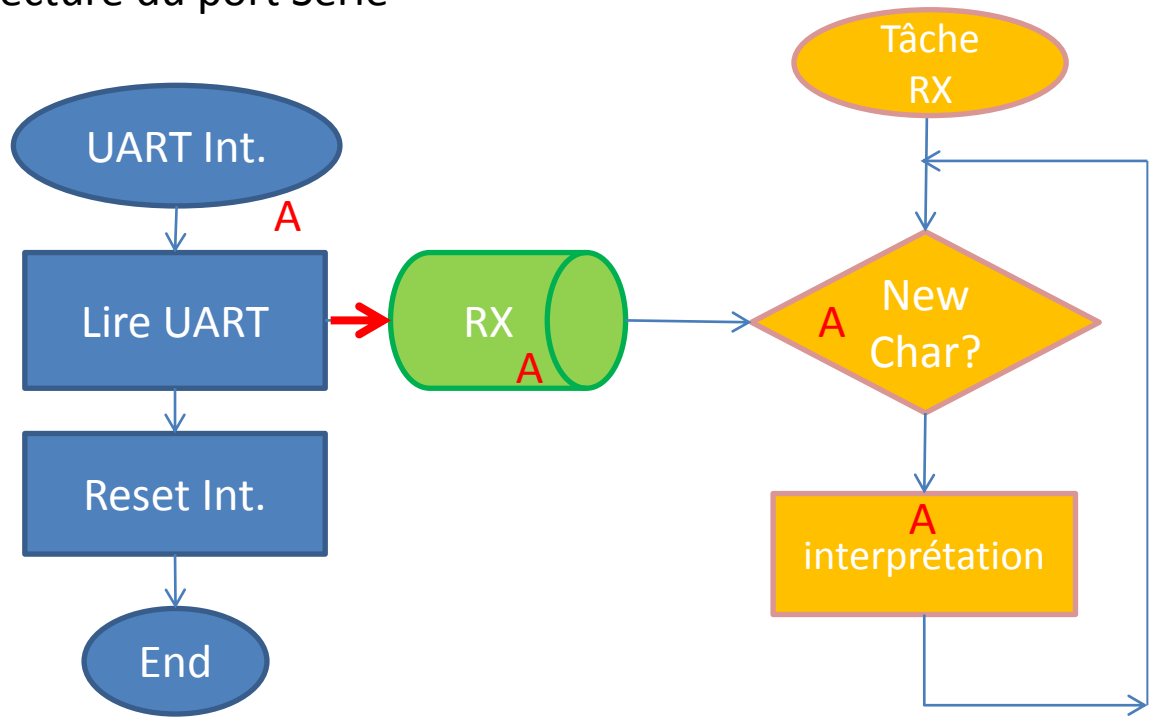


Solution conventionnelle
Avec des variables
et un flag indiquant de
nouvelles données

Temps

Tache simple UART

Lecture du port Série



Partie RT

Partie asynchrone

Fonction QUEUE

- xQueueCreate(..)
- xQueueDelete(..)
- xQueueSendfrom ISR(..)
- xQueueReceive(..)
- xQueueReceivefromISR(..)
- xQueueMessagesWaiting(..)

A réception d'un caractère
-on appelle xQueueSendfromISR(…
-on transmet le caractère
-et on termine l'interruption

Sans urgence,
La tâche qui attend un message dans la queue, lit la queue RX avec xQueueReceivefromISR et l'interprète.

12. Réalisation du projet Structure du Code

- Main = démarrage du système
- Drivers(code matériel)
- Tâches
  - WIFI-TCP-Stack
  - Module RF (433.92) *
  - Carte SD (log)
  - Module Hum. DHT22*
  - Tâche RTCC

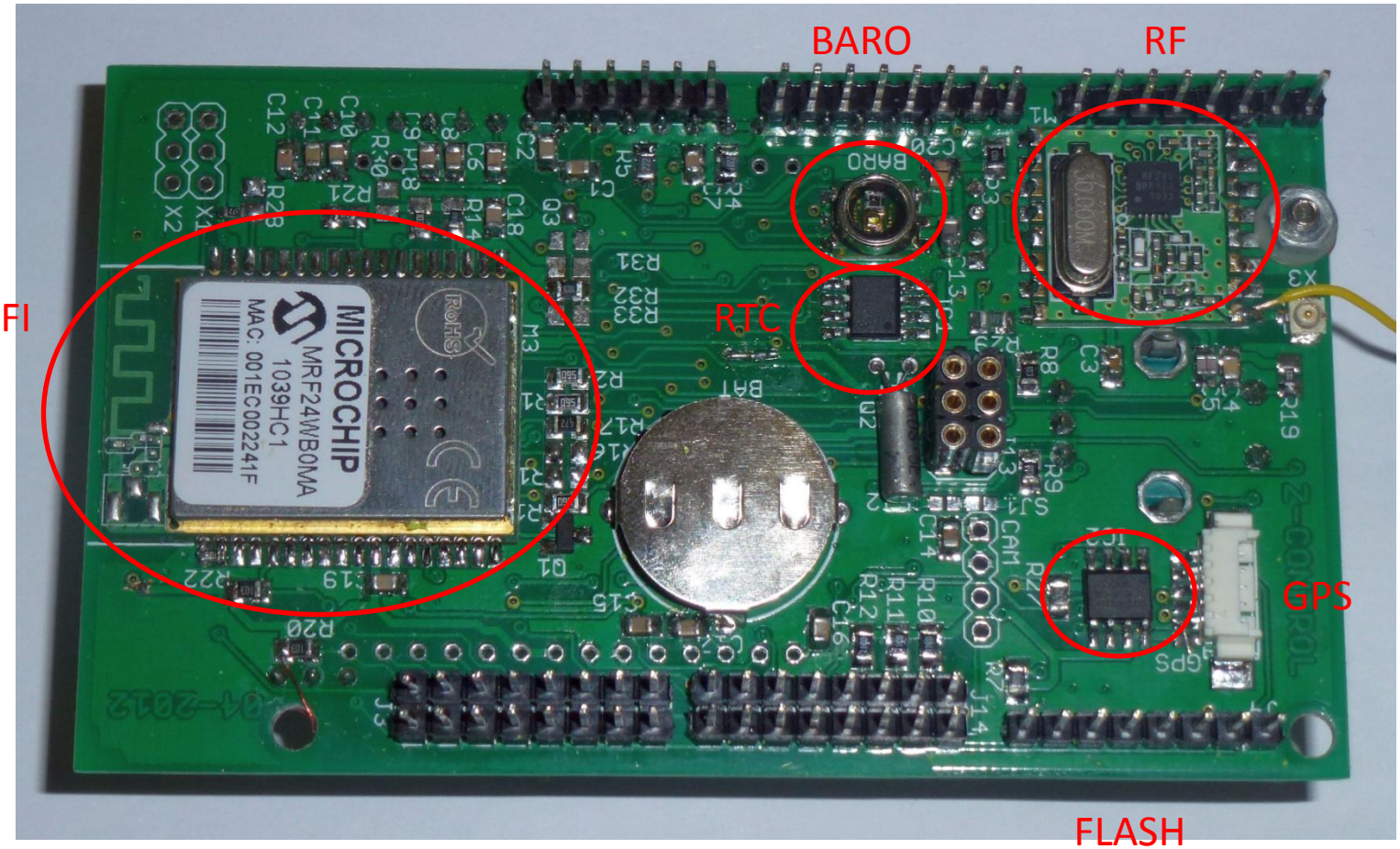
* avec interruption

- Travail à effectuer
  - Main (codé neuf)
  - TCP-IP existant (modif.)
  - WIFI existant (adapté)
  - RTOS existant (config)
  - Tâches à coder (neuf)
  - Driver périphériques
    - LCD (existant)
    - FAT (existant)
  - Page WEB (html)
- 100-200 fichiers

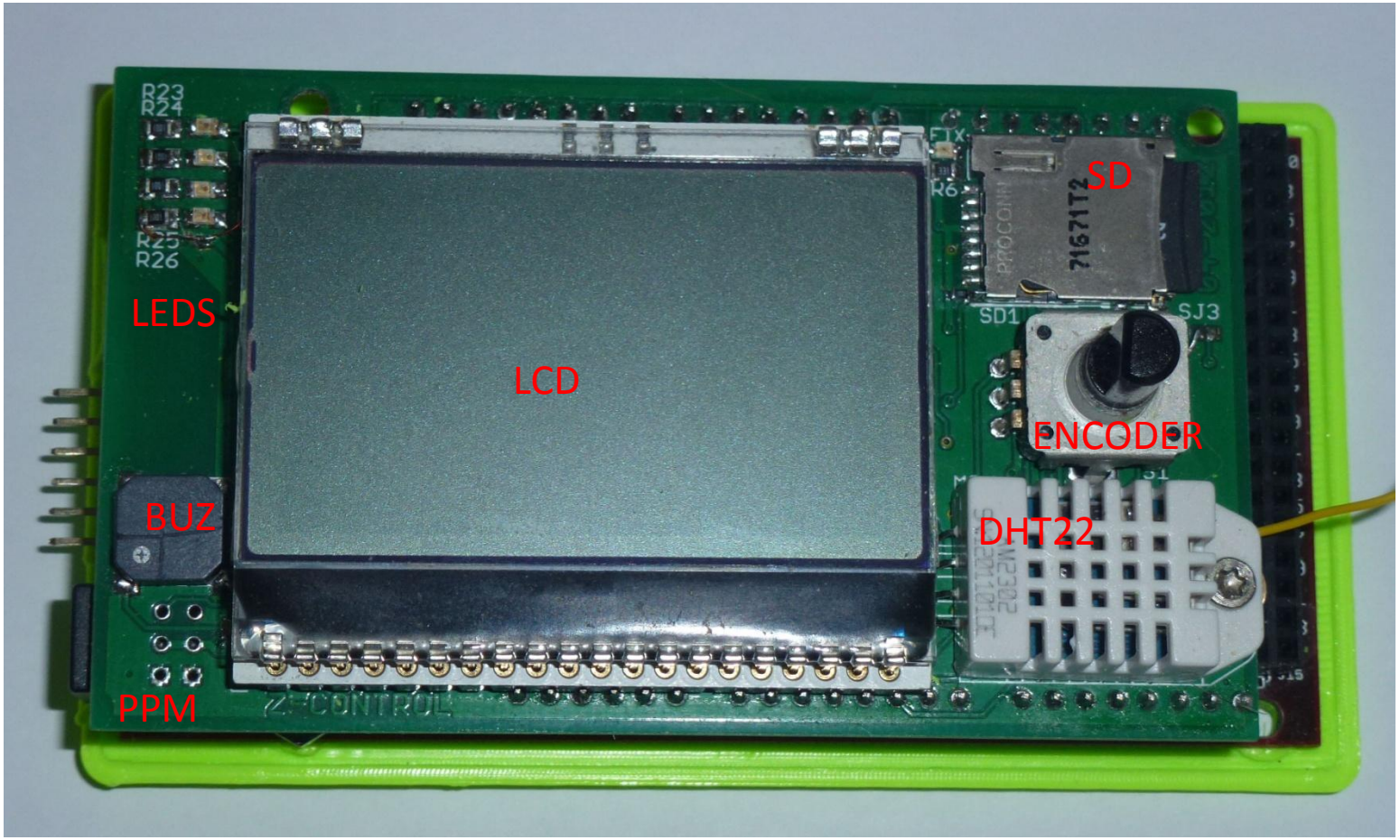
Paramétriser FreeRTOS

```
#define configUSE_PREEMPTION                    1
#define configUSE_PORT_OPTIMISED_TASK_SELECTION 1
#define configUSE_IDLE_HOOK                     0
#define configUSE_TICK_HOOK                     0
#define configTICK_RATE_HZ                      ( ( portTickType ) 1000 )
#define configCPU_CLOCK_HZ                      ( 80000000UL )
#define configPERIPHERAL_CLOCK_HZ               ( 40000000UL )
#define configMAX_PRIORITIES                    ( 6UL )
#define configMINIMAL_STACK_SIZE                ( 500 )
#define configISR_STACK_SIZE                    ( 800 )
#define configTOTAL_HEAP_SIZE                   ( ( size_t ) 36000 )
#define configMAX_TASK_NAME_LEN                 ( 8 )
#define configUSE_TRACE_FACILITY                0
#define configUSE_16_BIT_TICKS                  0
#define configIDLE_SHOULD_YIELD                 1
#define configUSE_MUTEXES                       1
#define configCHECK_FOR_STACK_OVERFLOW          3
#define configQUEUE_REGISTRY_SIZE               0
#define configUSE_RECURSIVE_MUTEXES             0
#define configUSE_MALLOC_FAILED_HOOK            1
#define configUSE_APPLICATION_TASK_TAG          0
#define configUSE_COUNTING_SEMAPHORES           1
#define configGENERATE_RUN_TIME_STATS           0
/* Co-routine definitions. */
#define configUSE_CO_ROUTINES                   0
#define configMAX_CO_ROUTINE_PRIORITIES         ( 2 )
```

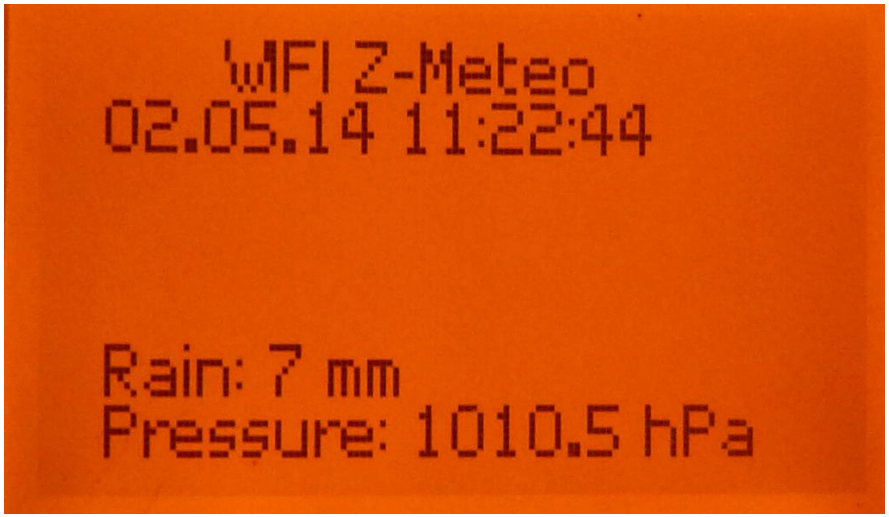
13. Réalisation du projet Hardware, mega-shield



Et dans un monde plus complexe ?

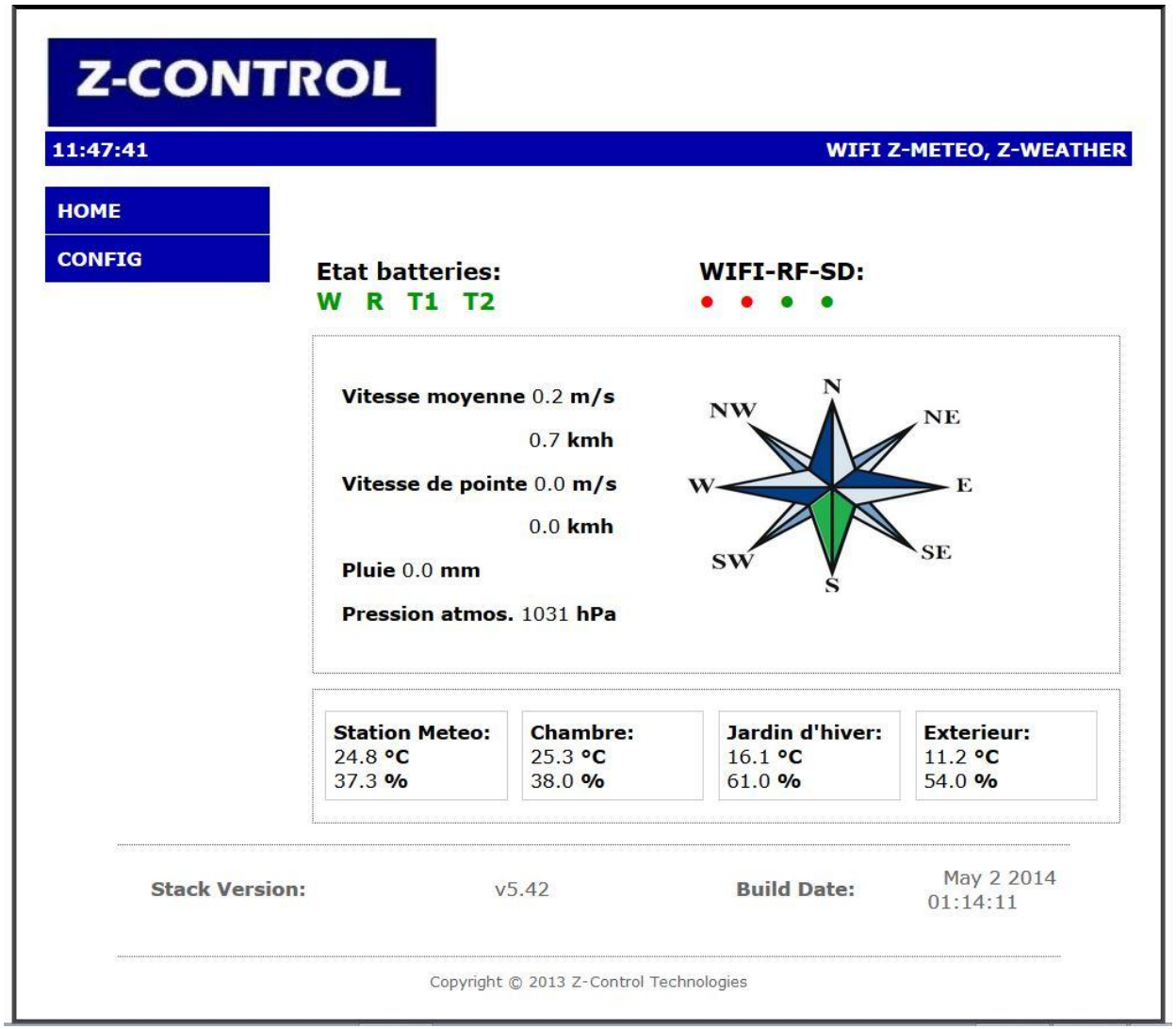


Contenu
Ecran LCD



Station météo basée RTOS

Interface WEB



Résumé
Multitâche Système embarqué

Modularité est un avantage certain (définition de tâches)

Implémentation simple des sémaphores, fonctionne mieux que prévu, même avec WIFI etc.

Surprenante utilisation des Queues, simple et efficace, attention à vérifier que les queues soient bien vidées!

Reste un mystère

- Gestion mémoire
- Gestion des priorités probablement plus basic que prévu

A résoudre

- Plantées dans wifi, disparu après changement de l'allocation mémoire

A ajouter

- Graphes avec historique température, humidité, vent, pa.

Démonstration
Questions ?

Fonctions FreeRTOS Résumé

- Tâches
  - xTaskCreate(…)
  - vTaskPrioritySet(taskhandle, newpriority)
  - vTaskSuspend ()
  - vTaskSuspendAll()
  - vTaskDelete()
  - vTaskDelay()
  - vTaskDelayUntil()
- Sémaphores
  - xSemaphoreCreateMutex
  - xSemaphoreCreateBinary
  - xSemaphoreTake
  - xSemaphoreGive
- Queues
  - xQueueCreate(..)
  - xQueueDelete(..)
  - xQueueSendfrom ISR(..)
  - xQueueReceive(..)
  - xQueueReceivefromISR(..)
  - xQueueMessagesWaiting(..)

Travail d'intégration

- MPLABX microchip
- Compilateur XC32

Côté matériel

- Debugger ICD3 (microchip)
- Carte ChipKit
- Shield fait maison

Côté Logiciel

- TCP-Stack Microchip
  Exemple EzConfig
- FreeRTOS V7.06
- FS-FAT existant*
- Code Baromètre*
- Module RF*
- LCD*

Réalisé

- Bouton Encodeur, 1Wire, RTC

*Code existant de vieux project 8 bits

Station Météo
Choix du processeur

- PIC32MX
  - Code/CPU 80MHz
  - Périf. 40MHz
  - 512k Flash (interne)
  - 128k RAM
  - 4 SPI
  - 5 I2C
  - RTC intégré
  - 5 Timer (16/32 bits)
  - USB2.0
  - 8 DMA *
  - 16 ADC *
  - CAN-Bus *
  - Input Capture 5
  - Output Compare 5 *
  - 100 pins (85 IO)

* Pas utilisé dans ma solution

Liste des queues

DHT22 queue : stockage des données réceptionnées par 1 wire
SD queue : réception des messages à mettre sur carte SD
RF queue : stockage des données série reçues par le module RF
UART queue : réception de la ligne série
Timer queue : utilisation internet de FreeRTOS