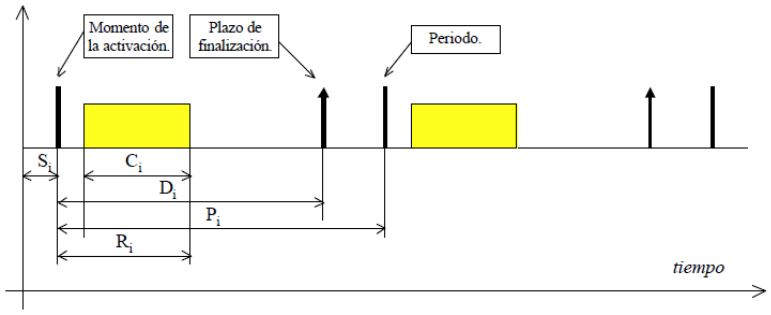
Sistemas de Tiempo Real – Planificación – Test de Garantía

1. de Determinar qué **planificación**/es **cumple**/n con los **requisitos de un sistema de tiempo real** que corre en la computadora de un automóvil, la misma debe atender las siguientes **tareas críticas** (cuyas características figuran en el siguiente cuadro siguiente).

Número de Tareas 3	Período de activación P_i	Tiempo máximo de ejecución C_i	Plazo máximo de terminación D_i
Medición de **velocidad** del vehículo ***T1***	**20**	**4**	**20**
Control de **frenado** (ABS) ***T2***	**40**	**10**	**40**
Control de **inyección** de combustible ***T3***	**80**	**40**	**80**

- **Tareas.** Sucesión de trabajos que se repiten. Cuando la tarea comienza a ejecutarse comenzará a ejecutarse su primer trabajo y cuando el último trabajo finalice terminará la ejecución de la tarea.
- **Tareas críticas**. El fallo de una de estas tareas (por no ejecutarse a tiempo) puede ser catastrófico para el sistema ($R_i < D_i$).
- **Tareas periódicas**. Su ejecución se realiza periódicamente. Sus trabajos entran en ejecución en un periodo constante.
- Los parámetros de la tarea ***Ti*** con desfase S_i, periodo P_i, tiempo de ejecución C_i y plazo de finalización D_i, los definiremos con la 4-tupla (S_i, P_i, C_i, D_i).
  - ***T1***(20, 4, 20), ***T2***(40, 10, 40) y ***T3***(80, 40, 80).



2. **Restricciones del modelo simple**

- Supondremos un **sistema** formado por **único procesador.**
- El **conjunto** de tareas es **estático**. No se destruyen ni se crean nuevas tareas, por tanto, el número de tareas permanecerá constante durante toda la vida del sistema.
- El **tiempo de ejecución** máximo de los trabajos de cada tarea es **conocido.**
- Todos los trabajos son **interrumpibles.**
- Las **operaciones** del núcleo de multiprogramación son **instantáneas**. En concreto, se considera nulo el tiempo de cambio de contexto utilizado para dejar de ejecutar una tarea y retomar otra.
- Todas las tareas son **periódicas**, o esporádicas transformadas en periódicas utilizando el tiempo mínimo entre dos activaciones consecutivas como su período.
- Los plazos de finalización de todas las tareas son iguales a sus períodos respectivos ($D_i = P_i$).
- Las tareas son independientes unas de otras. No se considera la existencia de secciones críticas ni de recursos compartidos que obligue a la sincronización entre tareas. De existir una relación de precedencia las supondremos periódicas independientes con el mismo período de la precedente.

Factor de Utilización - Es la fracción de CPU que se utiliza en el sistema y se define con la expresión:

$$U = \sum_{i=1}^{N} \frac{C_i}{P_i} \leq 1$$

Para que el conjunto de tareas periódicas se puedan ejecutar en un procesador es **condición necesaria** que el **factor de utilización sea menor o igual que uno**.

Hiperperíodo - Es el mínimo común múltiplo de los períodos de todas las tareas

$$\boldsymbol{H = mcm(P_1,\ P_2, P_3)}$$

3. Planificador cíclico con tramas

En lugar de utilizar planificaciones cíclicas en las que los trabajos se activan en instantes arbitrarios, resulta más interesante dotar a las planificaciones de una cierta estructura. Una estructura adecuada es que los instantes de decisión no sean arbitrarios si no que se realicen periódicamente. De esta forma, los instantes de decisión dividirán el tiempo de ejecución en segmentos que se denominan **tramas**. Cada trama tendrá una duración ***f***, que denominaremos **tamaño de la trama**.

Idealmente, es deseable elegir las tramas para que cada trabajo pueda comenzar en una trama y finalizar su ejecución en la misma trama. Este objetivo se puede alcanzar si hacemos el tamaño de la trama lo bastante grande para que sea mayor que el tiempo de ejecución de todas las tareas,

$$\boldsymbol{f \geq máx(C_i)\,, 1\ \leq i\ \leq N\ (1)}$$

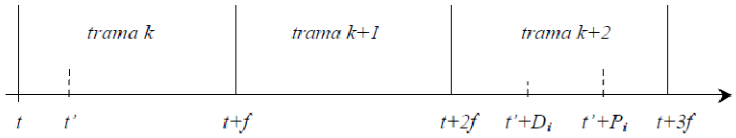
Para mantener la longitud del ciclo de ejecución lo más corta posible, interesa elegir ***f*** de forma que sea un divisor de ***H***. Esto se cumple si ***f*** divide el periodo ***Pi*** de al menos una tarea ***Ti***, es decir,

$$\boldsymbol{\exists i:\ \frac{P_i}{f} - \left\lfloor \frac{P_i}{f} \right\rfloor = 0\ (2)}$$

Cuando esto ocurre, existe un número entero de **tramas** (***F***) en un hiperperíodo ***H***. (***F*** = ***H*** / ***f***).

Por otro lado, para que el planificador pueda determinar cuando un trabajo finaliza antes de su plazo de ejecución, el tamaño de la trama debe ser lo suficientemente pequeño para que, entre el instante de activación de cualquier trabajo y su plazo de finalización, exista al menos una trama completa.

Supongamos una tarea definida por $\boldsymbol{T_i}$, = ($\boldsymbol{P_i}$, $\boldsymbol{C_i}$, $\boldsymbol{D_i}$) en la que queremos que se cumpla la condición anterior.



Si ***t'*** es el instante de activación de la tarea, supongamos que se activa en la trama ***k***, que comienza en el instante $\boldsymbol{t \leq t'}$.

En el caso que $\boldsymbol{t = t'}$ bastaría con que ***f*** fuera menor que $\boldsymbol{D_i}$. Consideremos el caso más desfavorable en que $\boldsymbol{t' > t}$. Nos interesa que, al menos, la trama (***k***+1) esté completa en el intervalo comprendido entre $\boldsymbol{t'}$ y $\boldsymbol{t'+D_i}$. Si esto ocurre, entonces $\boldsymbol{t+2f \leq t'+D_i}$, o lo que es lo mismo $\boldsymbol{2f - (t' - t) \leq D_i}$. Como $\boldsymbol{(t' - t)}$ es mayor o igual que el $\boldsymbol{mcd(P_i, f)}$, entonces la condición se cumple si es cierta la siguiente desigualdad:

$$\boldsymbol{\forall i: 2f - mcd(P_i, f)\ \leq Di\ (3)}$$

El sistema cumple con la **condición necesaria** (factor de utilización menor que 1). Por la restricción **(1)** $\boldsymbol{f \geq 40}$. Como el hiperperíodo es **80.** Los valores posibles para el tamaño de trama según **(2)** son **40**, **80**. Ninguno de estos valores de satisface la restricción **(3)** por lo tanto **no hay solución aceptable** que garantice $\boldsymbol{\forall i: (R_i < D_i)}$.

N = 3	P_i	C_i	D_i
T1	20	4	20
T2	40	10	40
T3	**80**	**40**	80

$$\boldsymbol{U = \sum_{i=1}^{N} \frac{C_i}{P_i} = \frac{4}{20} + \frac{10}{40} + \frac{40}{80} = 0,95 \leq 1}$$

$$\boldsymbol{H = mcm(P_1, P_2, P_3) = 80}$$

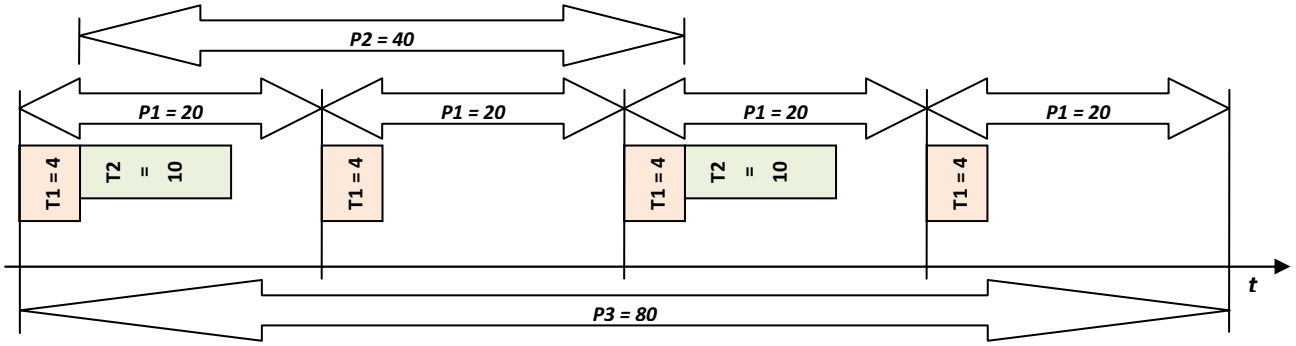
$f \geq máx(C_i), 1 \leq i \leq n$ (1)	$f \geq 40$

$\exists i : \frac{P_i}{f} - \left\lfloor \frac{P_i}{f} \right\rfloor = 0$ (2)	$f \in \{40, 80\}$

$\forall i : 2f - mcd(P_i, f) \leq D_i$ (3)	$f = 40$	$f = 80$
$2f - mcd(20, f) \leq 20$	$2 * 40 - mcd(20, 40) \leq 20$ $80 - 20 > 20$	$2 * 80 - mcd(20, 80) \leq 20$ $160 - 20 > 20$
$2f - mcd(40, f) \leq 40$	$2 * 40 - mcd(40, 40) \leq 40$ $80 - 40 \leq 40$	$2 * 80 - mcd(40, 80) \leq 40$ $160 - 40 > 40$
$2f - mcd(80, f) \leq 80$	$2 * 40 - mcd(80, 40) \leq 80$ $80 - 40 \leq 40$	$2 * 80 - mcd(80, 80) \leq 80$ $160 - 80 \leq 80$

Como en el sistema el factor de ***U ≤ 1*** la solución es dividir cada trabajo de las tarea con tiempo de ejecución mayor en porciones o rodajas (slices) con tiempos de ejecución menores en las que su ejecución sea indivisible. De esta forma, podemos reducir el tamaño mínimo de ***f*** que impone la condición **(1)** tanto como sea necesario.

Dividamos entonces ***T3***(80, **40**, 80) a fin de reducir el tiempo máximo de ejecución para llevar la restricción **(1)** a ***f*** ≥ **15** (máximo C_i), pero antes de hacerlo dispongamos en un gráfico temporal las tareas ***T1***(20, 4, 20), ***T2***(40, 10, 40). Del gráfico surgen dos cosas los posible tamaños de la trama (***f = 20, 40, 80***) así como los huecos (6 + 16 + 6 + 16 = **44**) para alojarlas porciones o rodajas (slices) de ***T3***(80, **40**, 80).



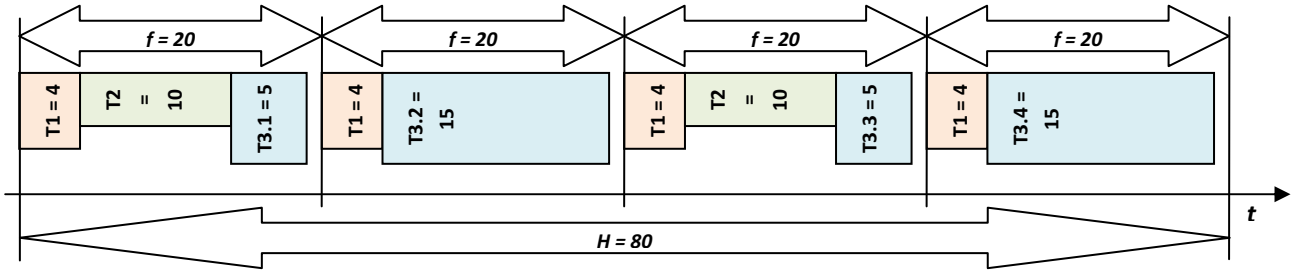
Dividamos entonces ***T3***(80, **5 + 15 + 5 + 15**, 80) llevando así la restricción **(1)** a ***f*** ≥ **15** (máximo C_i) y repitiendo las prueba podemos ver que se **cumplen** (**1**), (**2**) y (**3**) para ***f*** = **20**, con hiperperíodo ***H*** = **80** y número de tramas **F** = ***H*** / ***f*** = **4**, **hay solución aceptable** que garantice $\forall i : (R_i < D_i)$.

N = 6	P_i	C_i	D_i
T1	20	4	20
T2	40	10	40
T3.1	**80**	**5**	**80**
T3.2	**80**	**15**	**80**
T3.3	**80**	**5**	**80**
T3.4	**80**	**15**	**80**

$$U = \sum_{i=1}^{N} \frac{C_i}{P_i} = \frac{4}{20} + \frac{10}{40} + \frac{40}{80} = 0,95 \leq 1$$

$$H = mcm(P_1, P_2, P_3) = 80$$

$$F = H/f = 4$$



$f \geq máx(C_i), 1 \leq i \leq n$ (1)	$f \geq 15$

$\exists i : \frac{P_i}{f} - \left\lfloor \frac{P_i}{f} \right\rfloor = 0$ (2)	$f \in \{20, 40, 80\}$

$\forall i : 2f - mcd(P_i, f) \leq Di\ (3)$	$f = 20$
$2f - mcd(20, f) \leq 20$	$2 * 20 - mcd(20, 20) \leq 20$ $40 - 20 \leq 20$
$2f - mcd(40, f) \leq 40$	$2 * 20 - mcd(40, 20) \leq 40$ $40 - 20 \leq 40$
$2f - mcd(80, f) \leq 80$	$2 * 20 - mcd(80, 20) \leq 80$ $40 - 20 \leq 40$

Existen tres decisiones de diseño en el planificador:

1) Elegir el tamaño de la trama
2) Dividir los trabajos en rodajas
3) Situar las rodajas en las tramas

Por lo general, estas decisiones no se pueden tomar de forma independiente.

Para no penalizar el rendimiento, interesa subdividir los trabajos en el menor número de rodajas posibles, sin embargo, esto no siempre es factible. Si las rodajas son grandes, es posible que no exista un planificador válido al no poder encajar las rodajas en las tramas. Y al contrario, si las rodajas son pequeñas es más fácil encontrar un planificador válido.

Además, si las tareas no son divisibles, cuanto menor tengan que ser las rodajas más fácil será encontrar problemas debido a las zonas no divisibles de los trabajos.

¿Cómo configuraría FreeRTOS para cumplir con ésta planificación?

4. Planificador con prioridades estáticas

Planificador Rate Monotonic (RM)

El planificador de prioridades monótonas en frecuencia (Rate Monotonic) será un planificador con prioridades con las siguientes características:

- Todas las **tareas** son **periódicas** e **independientes** unas de otras.
- El plazo de finalización de cada tarea se tomará igual a su periodo ($D_i = P_i$).
- La asignación de **prioridades** se hará de forma **inversa al periodo**, es decir, una tarea de menor periodo que otra tendrá mayor prioridad (solo tiene importancia el valor relativo de la prioridad, el absoluto es indiferente).
- El planificador será **expulsivo**, es decir, si al activarse una tarea la CPU la está utilizando otra de menor prioridad (mayor valor), la de menor prioridad pasará al estado de 'lista' y tomará la CPU la tarea que se acaba de activar.
- El tiempo de cambio de contexto se considera despreciable.
- El conjunto de tareas es síncrono: ($Si = 0,\ \forall i$).

El planificador **RM no es** un planificador **óptimo** en el caso general, pero sí que lo es cuando las tareas son periódicas simples.

Test de garantía mediante el Factor de Utilización

Un conjunto de **N** tareas será planificable bajo la política de planificación **Rate Monotonic** si se cumple la siguiente desigualdad:

$$U = \sum_{i=1}^{N} \frac{C_i}{P_i} < N\left(2^{1/N} - 1\right)$$

N = 3	P_i	C_i	Pr	U_i
T1	20	4	3	0,20
T2	40	10	2	0,25
T3	80	40	1	0,50

$$U = \sum_{i=1}^{N} \frac{C_i}{P_i} = 0,95 < 1$$

$$N\left(2^{\frac{1}{N}} - 1\right) = 3\left(2^{\frac{1}{3}} - 1\right) = 0,799$$

$0,95 > 0.779$ => No cumple

No sirve de nada dividir la tarea con tiempo de ejecución mayor en porciones o rodajas (slices) pues aumenta N.

Test de garantía mediante el Análisis de Tiempo de Respuesta

Recurrir al test de garantía basado en el factor de utilización tiene dos inconvenientes importantes:

- Es inexacto. Da una **condición suficiente**, pero **no necesaria**.
- No se puede aplicar a un modelo de procesos más general, en el que las prioridades no se asignen según indica el **RM**.

Vamos a ver ahora un test de garantía diferente.

Este test consta de dos fases. En la primera se utiliza una aproximación analítica para predecir el caso más desfavorable del tiempo de respuesta en cada tarea. Luego se comparan estos valores con los plazos de ejecución de cada tarea. Este proceso requiere que se analice cada tarea de forma independiente.

Para el proceso de más prioridad, el caso más desfavorable en el tiempo de respuesta corresponde a su tiempo de ejecución (esto es, $R_i = C_i$), pues ningún otro proceso lo puede retardar. El resto de procesos sufrirán una interferencia producida únicamente por los procesos de mayor prioridad. De forma general, para una tarea T_i su tiempo de respuesta será:

$$R_i = C_i + I_i$$

I_i es el valor máximo de la interferencia a la puede verse afectada la tarea T_i.

Para el proceso de mayor prioridad ya hemos visto que este valor es cero. Para el resto de procesos el valor máximo se dará cuando todas las tareas de mayor prioridad se activen al mismo tiempo que la tarea ***Ti*** (este es el instante crítico).

Con esto, podríamos formular el siguiente teorema:

Teorema:

Un conjunto de **N** tareas será planificable bajo cualquier asignación de prioridades, si y solo si, cada una de las tareas cumple su plazo de finalización en el peor de los casos.

El tiempo de finalización de cada tarea en el peor de los casos ocurre cuando todas las tareas de prioridad superior se inician a la vez que esta.

Un instante crítico (el peor caso) válido para todas las tareas será el momento en que todas las tares piden ejecutarse a la vez. En un sistema síncrono esto ocurre, al menos, en el instante inicial.

Consideremos una tarea T_j de mayor prioridad que la tarea T_i. En el intervalo [0, R_i] se activará un número determinado de veces (una por lo menos). El número de veces que se activará será:

$$numero\ de\ activaciones = \left\lceil \frac{R_i}{P_j} \right\rceil$$

La función techo (⌈ ⌉) da el número entero más pequeño superior al resultado de la fracción sobre la que actúa. Por ejemplo, de 1/3 será 1 y de 5/6 será 2. La definición para los valores negativos no afecta en nuestro caso.

Cada activación de la tarea *Tj* producirá una interferencia sobre la tarea T_i de valor:

$$interferencia\ máxima = \left\lceil \frac{R_i}{P_j} \right\rceil C_j$$

El total de la interferencia a la que se ve sometida la tarea *Ti* por todas las de mayor prioridad será:

$$I_i = \sum_{j \in hp(i)} \left\lceil \frac{R_i}{P_j} \right\rceil C_j$$

donde ***hp(i)*** es el conjunto de tareas de mayor prioridad que la tarea T_i.

Sustituyendo este valor en la expresión del tiempo de respuesta tendremos:

$$R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{P_j} \right\rceil C_j$$

Aunque la formulación de la ecuación de la interferencia es exacta, el valor de la interferencia sigue siendo desconocido pues está en función de R_i que es justo lo que queremos calcular.

La ecuación final tiene en ambos lados el valor R_i, pero es difícil de despejar debido al operador techo. En general existirán más de un valor que den solución a esta ecuación, pero a nosotros sólo nos interesará el valor más pequeño. Todos serán válidos para el tiempo de respuesta en el peor de los casos de la tarea T_i, pero para que se cumplan los plazos bastará con que uno sea menor o igual que D_i, y si hay alguno siempre será el menor.

Una forma sencilla de resolver la ecuación es por medio de un mecanismo iterativo. Supongamos la sucesión:

$$\omega_i^{n+1} = C_i + \sum_{j \in hp(i)} \left\lceil \frac{\omega_i^n}{P_j} \right\rceil C_j$$

La sucesión $\{\omega_i^0 = C_i,\ \omega_i^1, \omega_i^2, \omega_i^n\}$ es monótonamente no decreciente.

Esto es evidente ya que la fracción del sumatorio primero será 0 y después del numerador tomará valores que nunca serán más pequeños que el anterior, debido al signo '+' de la expresión.

Cuando se obtenga que $\omega_i^n = \omega_i^{n+1}$ se habrá encontrado una solución para la ecuación. Si $\omega_i^n < D_i$ entonces ω_i^n será la solución más pequeña, y habremos encontrado la solución que buscábamos.

Si la ecuación no tiene solución, entonces la sucesión aumentará indefinidamente (esto ocurrirá para las tareas de baja prioridad si el total del factor de utilización del procesador supera el 100%). Si se alcanza un valor superior al periodo de la tarea, se puede asumir que no cumplirá su límite temporal.

Si obtenemos todos los tiempos de respuesta de cada tarea menores que sus correspondientes periodos, entonces el sistema será planificable.

N = 3	P_i	C_i	Pr	U_i	R_i
T1	20	4	3	0,20	4
T2	40	10	2	0,25	14
T3	80	40	1	0,50	76

$$U = \sum_{i=1}^{N} \frac{C_i}{P_i} < N\left(2^{\frac{1}{N}} - 1\right)$$

$0,95 > 0,779 \rightarrow$ *No cumple*

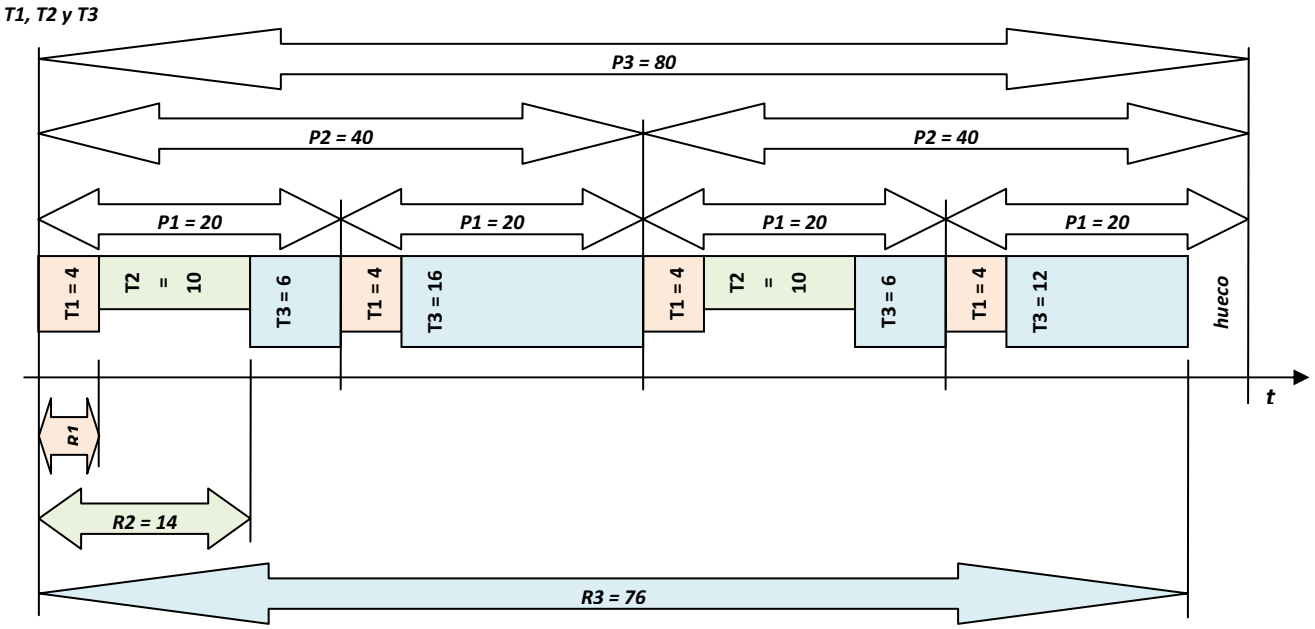
$$\omega_i^{n+1} = C_i + \sum_{j \in hp(i)} \left\lceil \frac{\omega_i^n}{P_j} \right\rceil C_j$$

R_1, R_2 *y* R_3 *cumplen OK*

T1(20, **4**, 3)	$\omega_1^0 = C_1 = 4 \rightarrow R_1 = 4$
T1(**20**, **4**, 3) *T2*(40, **10**, 2)	$\omega_2^0 = C_2 = 10$
	$\omega_2^1 = C_2 + \left\lceil \frac{\omega_2^0}{P_1} \right\rceil C_1 = 10 + \left\lceil \frac{10}{20} \right\rceil 4 = 14$
	$\omega_2^2 = C_2 + \left\lceil \frac{\omega_2^1}{P_1} \right\rceil C_1 = 10 + \left\lceil \frac{14}{20} \right\rceil 4 = 14 \rightarrow R_2 = 14$
T1(**20**, **4**, 3) *T2*(**40**, **10**, 2) *T3*(80, **40**, 2)	$\omega_3^0 = C_3 = 40$
	$\omega_3^1 = C_3 + \left\lceil \frac{\omega_3^0}{P_1} \right\rceil C_1 + \left\lceil \frac{\omega_3^0}{P_2} \right\rceil C_2 = 40 + \left\lceil \frac{40}{20} \right\rceil 4 + \left\lceil \frac{40}{40} \right\rceil 10 = 58$
	$\omega_3^1 = C_3 + \left\lceil \frac{\omega_3^0}{P_1} \right\rceil C_1 + \left\lceil \frac{\omega_3^0}{P_2} \right\rceil C_2 = 40 + \left\lceil \frac{58}{20} \right\rceil 4 + \left\lceil \frac{58}{40} \right\rceil 10 = 72$
	$\omega_3^1 = C_3 + \left\lceil \frac{\omega_3^0}{P_1} \right\rceil C_1 + \left\lceil \frac{\omega_3^0}{P_2} \right\rceil C_2 = 40 + \left\lceil \frac{72}{20} \right\rceil 4 + \left\lceil \frac{72}{40} \right\rceil 10 = 76$
	$\omega_3^1 = C_3 + \left\lceil \frac{\omega_3^0}{P_1} \right\rceil C_1 + \left\lceil \frac{\omega_3^0}{P_2} \right\rceil C_2 = 40 + \left\lceil \frac{76}{20} \right\rceil 4 + \left\lceil \frac{76}{40} \right\rceil 10 = 76 \rightarrow R_3 = 76$

El ejemplo no cumple el test de garantía mediante factor de utilización pero si **cumple** el test de garantía por análisis del **tiempo de respuesta**, por lo tanto **puede implementarse** mediante una planificación con prioridades estáticas del tipo **Rate Monotonic**.

El gráfico temporal correspondiente (en el que se observa un hueco de 4 unidades de tiempo) en el que cada tarea sería el siguiente:



¿Cómo configuraría FreeRTOS para cumplir con ésta planificación?

5. Referencias

Real-Time Systems and Programming Languages (Fourth Edition), Alan Burns and Andy Wellings (Addison-Wesley, 2009)

Real-Time Systems: Specification, Verification and Analysis, Mathai Joseph (Revised version with corrections, 2001)

Apuntes de Sistemas Informáticos de Tiempo Real, Francisco Pastor (2004/5) www.uv.es/gomis/Apuntes_SITR/