Programación de sistemas de tiempo real

Práctica 1

Programación de dispositivos y planificador cíclico simple

Autores:

Juan José Garzón Luján (100363861) -> *Encargado de software*

Carlos García Cañibano (100363813) -> *Encargado de hardware*

Índice

PARTE I – Análisis del apartado software

Apartado A

Tenemos las siguientes tareas:

Tareas	T/D	C	Abreviatura
Leer Pendiente	10 s	0.9 s	LP
Leer Velocidad	10 s	0.9 s	LV
Encender/Apagar Acelerador	10 s	0.9 s	A
Encender/Apagar Freno	10 s	0.9 s	F
Encender/Apagar Mixer	15 s	0.9 s	M

El **uso CPU** = 4 x 0.9 / 10 + 0.9 / 15 = **0.42** <= 1

Podemos ver que no son tareas armónicas, por tanto, utilizamos el método general para sacar el ciclo principal (CP) y ciclo secundario (CS).

Ciclo principal = m.c.m(10,15) = **30 s** = Tm

Para el ciclo secundario, tenemos que cumplir varias condiciones:

a) Máx(Ci) < Ts $\leq$ Di, i = 1…n

   0.9 < Ts <= 10 => Obtenemos el intervalo [0.9 - 10] para Ts

b) Divisores de Tm

   Los posibles valores par Ts se nos queda en [1,2,3,5,6,10]

c) Algún Ti tiene que ser múltiplo de Ts

   Los posibles valores par Ts se nos queda en [1,2,3,5,10]

d) Ts + (Ts – m.c.d(Ts,Ti)) $\leq$ Di, $\forall$i = 1….n

   Ts = 1
   1 + (1 - m.c.d(1,10)) $\leq$ 10 -> 1 $\leq$ 10
   1 + (1 - m.c.d(1,15)) $\leq$ 15 -> 1 $\leq$ 15

   Ts = 2
   2 + (2 - m.c.d(2,10)) $\leq$ 10 -> 2 $\leq$ 10

$2 + (2 - m.c.d(2,15)) \leq 15 \rightarrow 3 \leq 15$

Ts = 3

$3 + (3 - m.c.d(3,10)) \leq 10 \rightarrow 5 \leq 10$

$3 + (3 - m.c.d(3,15)) \leq 15 \rightarrow 3 \leq 15$

Ts = 5

$5 + (5 - m.c.d(5,10)) \leq 10 \rightarrow 5 \leq 10$

$5 + (5 - m.c.d(5,15)) \leq 15 \rightarrow 5 \leq 15$

Ts = 10

$10+ (10 - m.c.d(10,10)) \leq 10 \rightarrow 10 \leq 10$

$10+ (10 - m.c.d(10,15)) \leq 15 \rightarrow 15 \leq 15$

Los posibles valores par Ts se nos queda en [1,2,3,5,10]

Eligiendo para el **ciclo secundario 5 s** (Ts), nos queda así la planificación:

Ciclo	Tareas
1	LP , LV , M
2	A , F
3	LP , LV
4	F , A , M
5	LP , LV
6	A , F

Respecto al código tenemos las siguientes funciones para las distintas tareas:

Tareas	Funciones
Leer Pendiente	task_slope()
Leer Velocidad	task_speed()
Encender/Apagar Acelerador	task_gas()
Encender/Apagar Freno	task_break()
Encender/Apagar Mixer	task_mix()

Para las tareas del acelerador y freno, tenemos variables globales para controlar las peticiones que hacemos, ya que por ejemplo no tiene lógica mandar la petición de encender el acelerador si ya esta activado.

Para el mixer tenemos una variable global que nos indica cuantos ciclos lleva el mixer(se considera un ciclo del mixer 15 s ya que es su periodo), cuando esa variable sea 2, entonces apagaremos o encenderemos el mixer depende lo que toque.

Respecto al planificador, tenemos una función llamada finisch_cycle(), está función nos sirve para cambiar el número del ciclo, obtener el tiempo que hay que esperar para terminar el ciclo secundario y finalmente se añade al tiempo inicial el tiempo del ciclo secundario para así no acarrear el error de solicitar el tiempo con la función clock_gettime().

Apartado B

Tenemos las siguientes tareas:

Tareas	T/D	C	Abreviatura
Leer Pendiente	10 s	0.9 s	LP
Leer Velocidad	10 s	0.9 s	LV
Encender/Apagar Acelerador	10 s	0.9 s	A
Encender/Apagar Freno	10 s	0.9 s	F
Encender/Apagar Mixer	15 s	0.9 s	M
Leer Luminosidad	5 s	0.9 s	LL
Encender/Apagar Focos	5 s	0.9 s	FO

El **uso CPU** = 4 x 0.9 / 10 + 0.9 / 15 + 2 x 0.9 / 5 = **0.78** <= 1

Podemos ver que no son tareas armónicas, por tanto, utilizamos el método general para sacar el ciclo principal (CP) y ciclo secundario (CS).

Ciclo principal = m.c.m(5,10,15) = **30 s** = Tm

Para el ciclo secundario, tenemos que cumplir varias condiciones:

a) $Máx(Ci) < Ts \leq Di$, $i = 1…n$

   $0.9 < Ts <= 5$ => Obtenemos el intervalo [0.9 - 5] para Ts

b) Divisores de Tm

   Los posibles valores par Ts se nos queda en [1,2,3,5]

c) Algún Ti tiene que ser múltiplo de Ts

   Los posibles valores par Ts se nos queda en [1,2,3,5]

d) $Ts + (Ts – m.c.d(Ts,Ti)) \leq Di, \forall i = 1….n$

   Ts = 1
   $1 + (1 - m.c.d(1,5)) \leq 5 \rightarrow 1 \leq 5$

$1 + (1 - m.c.d(1,10)) \leq 10 \rightarrow 1 \leq 10$
$1 + (1 - m.c.d(1,15)) \leq 15 \rightarrow 1 \leq 15$

Ts = 2
$2 + (2 - m.c.d(2,5)) \leq 5 \rightarrow 3 \leq 5$
$2 + (2 - m.c.d(2,10)) \leq 10 \rightarrow 2 \leq 10$
$2 + (2 - m.c.d(2,15)) \leq 15 \rightarrow 3 \leq 15$

Ts = 3
$3 + (3 - m.c.d(3,5)) \leq 5 \rightarrow 5 \leq 5$
$3 + (3 - m.c.d(3,10)) \leq 10 \rightarrow 5 \leq 10$
$3 + (3 - m.c.d(3,15)) \leq 15 \rightarrow 3 \leq 15$

Ts = 5
$5 + (5 - m.c.d(5,5)) \leq 5 \rightarrow 5 \leq 5$
$5 + (5 - m.c.d(5,10)) \leq 10 \rightarrow 5 \leq 10$
$5 + (5 - m.c.d(5,15)) \leq 15 \rightarrow 5 \leq 15$

Los posibles valores par Ts se nos queda en [1,2,3,5]

Eligiendo para el **ciclo secundario 5 s** (Ts), nos queda así la planificación:

Ciclo	Tareas
1	LP , LA , LL , FO , M
2	A , F , LL , FO
3	LP , LA , LL , FO
4	A , F , LL , FO , M
5	LP , LA , LL , FO
6	A , F , LL , FO

Respecto al código añadimos las siguientes funciones para las distintas tareas:

Tareas	Funciones
Leer Luminosidad	task_lightSensor()
Encender/Apagar Focos	task_lamps()

En el apartado A y B, las funciones lo único que cambia es el planificador ya que hay más tareas, lo demás es igual.

Apartado C

Modo Normal

Tenemos las siguientes tareas:

Tareas	T/D	C	Abreviatura
Leer Pendiente	10 s	0.9 s	LP
Leer Velocidad	10 s	0.9 s	LV
Encender/Apagar Acelerador	10 s	0.9 s	A
Encender/Apagar Freno	10 s	0.9 s	F
Encender/Apagar Mixer	15 s	0.9 s	M
Leer Distancia	15 s	0.9 s	LD
Leer Luminosidad	5 s	0.9 s	LL
Encender/Apagar Focos	5 s	0.9 s	FO

El **uso CPU** = 4 x 0.9 / 10 + 2 x 0.9 / 15 + 2 x 0.9 / 5 = **0.84** <= 1

Podemos ver que no son tareas armónicas, por tanto, utilizamos el método general para sacar el ciclo principal (CP) y ciclo secundario (CS).

Ciclo principal = m.c.m(5,10,15) = **30 s** = Tm

Para el ciclo secundario, tenemos que cumplir varias condiciones:

a) Máx(Ci) < Ts $\leq$ Di, i = 1…n

   0.9 < Ts <= 5 => Obtenemos el intervalo [0.9 - 5] para Ts

b) Divisores de Tm

   Los posibles valores par Ts se nos queda en [1,2,3,5]

c) Algún Ti tiene que ser múltiplo de Ts

   Los posibles valores par Ts se nos queda en [1,2,3,5]

d) Ts + (Ts – m.c.d(Ts,Ti)) $\leq$ Di, $\forall$i = 1….n

   Ts = 1
   1 + (1 - m.c.d(1,5)) $\leq$ 5 -> 1 $\leq$ 5
   1 + (1 - m.c.d(1,10)) $\leq$ 10 -> 1 $\leq$ 10
   1 + (1 - m.c.d(1,15)) $\leq$ 15 -> 1 $\leq$ 15

Ts = 2

$2 + (2 - m.c.d(2,5)) \leq 5 \rightarrow 3 \leq 5$

$2 + (2 - m.c.d(2,10)) \leq 10 \rightarrow 2 \leq 10$

$2 + (2 - m.c.d(2,15)) \leq 15 \rightarrow 3 \leq 15$

Ts = 3

$3 + (3 - m.c.d(3,5)) \leq 5 \rightarrow 5 \leq 5$

$3 + (3 - m.c.d(3,10)) \leq 10 \rightarrow 5 \leq 10$

$3 + (3 - m.c.d(3,15)) \leq 15 \rightarrow 3 \leq 15$

Ts = 5

$5 + (5 - m.c.d(5,5)) \leq 5 \rightarrow 5 \leq 5$

$5 + (5 - m.c.d(5,10)) \leq 10 \rightarrow 5 \leq 10$

$5 + (5 - m.c.d(5,15)) \leq 15 \rightarrow 5 \leq 15$

Los posibles valores par Ts se nos queda en [1,2,3,5]

Eligiendo para el **ciclo secundario 5 s** (Ts), nos queda así la planificación:

Ciclo	Tareas
1	LP , LV , LL , FO , M
2	LD , A , F , LL , FO
3	LP , LV , LL , FO
4	A , F , LL , FO , M
5	LD , LP , LV , LL , FO
6	A , F , LL , FO

Modo Frenada

Tenemos las siguientes tareas:

Tareas	T/D	C	Abreviatura
Leer Pendiente	10 s	0.9 s	LP
Leer Velocidad	5 s	0.9 s	LV
Encender/Apagar Acelerador	5 s	0.9 s	A
Encender/Apagar Freno	5 s	0.9 s	F
Encender/Apagar Mixer	15 s	0.9 s	M
Leer Distancia	10 s	0.9 s	LD
Encender Focos	30 s	0.9 s	FO

El **uso CPU** = 2 x 0.9 / 10 + 0.9 / 15 + 3 x 0.9 / 5 + 0.9 / 30 = **0.81** <= 1

Podemos ver que no son tareas armónicas, por tanto, utilizamos el método general para sacar el ciclo principal (CP) y ciclo secundario (CS).

Ciclo principal = m.c.m(5,10,15,30) = **30 s** = Tm

Para el ciclo secundario, tenemos que cumplir varias condiciones:

a) Máx(Ci) < Ts $\leq$ Di, i = 1…n

   0.9 < Ts <= 10 => Obtenemos el intervalo [0.9 - 5] para Ts

b) Divisores de Tm

   Los posibles valores par Ts se nos queda en [1,2,3,5]

c) Algún Ti tiene que ser múltiplo de Ts

   Los posibles valores par Ts se nos queda en [1,2,3,5]

d) Ts + (Ts – m.c.d(Ts,Ti)) $\leq$ Di, $\forall$i = 1….n

   Ts = 1
   1 + (1 - m.c.d(1,5)) $\leq$ 5 -> 1 $\leq$ 5
   1 + (1 - m.c.d(1,10)) $\leq$ 10 -> 1 $\leq$ 10
   1 + (1 - m.c.d(1,15)) $\leq$ 15 -> 1 $\leq$ 15
   1 + (1 - m.c.d(1,30)) $\leq$ 30 -> 1 $\leq$ 30

   Ts = 2
   2 + (2 - m.c.d(2,5)) $\leq$ 5 -> 3 $\leq$ 5
   2 + (2 - m.c.d(2,10)) $\leq$ 10 -> 2 $\leq$ 10
   2 + (2 - m.c.d(2,15)) $\leq$ 15 -> 3 $\leq$ 15
   2 + (2 - m.c.d(2,30)) $\leq$ 30 -> 2 $\leq$ 30

   Ts = 3
   3 + (3 - m.c.d(3,5)) $\leq$ 5 -> 5 $\leq$ 5
   3 + (3 - m.c.d(3,10)) $\leq$ 10 -> 5 $\leq$ 10
   3 + (3 - m.c.d(3,15)) $\leq$ 15 -> 3 $\leq$ 15
   3 + (3 - m.c.d(3,30)) $\leq$ 30 -> 3 $\leq$ 30

   Ts = 5
   5 + (5 - m.c.d(5,5)) $\leq$ 5 -> 5 $\leq$ 5
   5 + (5 - m.c.d(5,10)) $\leq$ 10 -> 5 $\leq$ 10
   5 + (5 - m.c.d(5,15)) $\leq$ 15 -> 5 $\leq$ 15
   5 + (5 - m.c.d(5,30)) $\leq$ 30 -> 5 $\leq$ 30

   Los posibles valores par Ts se nos queda en [1,2,3,5]

Eligiendo para el **ciclo secundario 5 s** (Ts), nos queda así la planificación:

Ciclo	Tareas
1	LV , A , F , LD , M
2	LV , A , F , LP , FO
3	LV , A , F , LD
4	LV , A , F , LP , M
5	LV , A , F , LD
6	LV , A , F ,LP

Modo Parada

Tenemos las siguientes tareas:

Tareas	T/D	C	Abreviatura
Leer fin de descarga	5 s	0.9 s	LF
Encender/Apagar Mixer	15 s	0.9 s	M
Encender Focos	5 s	0.9 s	FO

El **uso CPU** = 0.9 / 15 + 2 x 0.9 / 5 = **0.42** <= 1

Podemos ver que son tareas armónicas, por tanto, el ciclo principal (CP) es el Máx (Ti) i=1..n y ciclo secundario (CS) Mín (Ti) i=1..n.

Ciclo principal = Máx (5,15) = **15 s** = Tm

Ciclo secundario = Mín (5,15) = **5 s** = Ts

Ciclo	Tareas
1	LF , FO , M
2	LF , FO
3	LF , FO

Respecto al código añadimos las siguientes funciones para las distintas tareas:

Tareas	Funciones
Leer Distancia	task_distance()
Leer fin de descarga	task_unload()

Al tener diferentes modos en este apartado, tenemos una variable global para saber en qué modo estamos, dentro de las funciones contemplamos en qué modo estamos ya que no siempre hace lo mismo la función. Un ejemplo sería la de acelerador ya que cuando estamos en modo normal nos tenemos que mantener sobre 55 m/s y cuando estamos en modo frenada sobre 2.5 m/s.

Tenemos una función para cada modo y dentro de cada función su planificador, a parte, tenemos una función select_mode(), que irá cambiando según corresponda. Como en el modo parada solo tenemos 3 ciclos, tenemos la función change_cycle() que irá cambiando de ciclo como toque en cada modo y resetea el ciclo a 0 cuando se cambia de modo. La función finish_cycle() ya solo calcula cuanto tiempo hay que esperar para terminar el tiempo del ciclo secundario y añadir la duración del ciclo secundario al tiempo inicial.

Apartado D

Modo Emergencia

Tenemos las siguientes tareas:

Tareas	T/D	C	Abreviatura
Leer Pendiente	10 s	0.9 s	LP
Leer Velocidad	10 s	0.9 s	LV
Apagar Acelerador	10 s	0.9 s	A
Encender Freno	10 s	0.9 s	F
Encender/Apagar Mixer	15 s	0.9 s	M
Activar Modo Emergencia (1 vez Arduino)	10 s	0.9 s	AE
Encender Focos	10 s	0.9 s	FO

El **uso CPU** = 6 x 0.9 / 10 + 0.9 / 15 = **0.6** <= 1

Podemos ver que no son tareas armónicas, por tanto, utilizamos el método general para sacar el ciclo principal (CP) y ciclo secundario (CS).

Ciclo principal = m.c.m(5,10,15) = **30 s** = Tm

Para el ciclo secundario, tenemos que cumplir varias condiciones:

a) Máx(Ci) < Ts $\leq$ Di, i = 1…n

   0.9 < Ts <= 5 => Obtenemos el intervalo [0.9 - 5] para Ts

b) Divisores de Tm

   Los posibles valores par Ts se nos queda en [1,2,3,5]

c) Algún Ti tiene que ser múltiplo de Ts

Los posibles valores par Ts se nos queda en [1,2,3,5]

d) $Ts + (Ts - m.c.d(Ts,Ti)) \leq Di, \forall i = 1....n$

Ts = 1

$1 + (1 - m.c.d(1,5)) \leq 5 \rightarrow 1 \leq 5$
$1 + (1 - m.c.d(1,10)) \leq 10 \rightarrow 1 \leq 10$
$1 + (1 - m.c.d(1,15)) \leq 15 \rightarrow 1 \leq 15$

Ts = 2

$2 + (2 - m.c.d(2,5)) \leq 5 \rightarrow 3 \leq 5$
$2 + (2 - m.c.d(2,10)) \leq 10 \rightarrow 2 \leq 10$
$2 + (2 - m.c.d(2,15)) \leq 15 \rightarrow 3 \leq 15$

Ts = 3

$3 + (3 - m.c.d(3,5)) \leq 5 \rightarrow 5 \leq 5$
$3 + (3 - m.c.d(3,10)) \leq 10 \rightarrow 5 \leq 10$
$3 + (3 - m.c.d(3,15)) \leq 15 \rightarrow 3 \leq 15$

Ts = 5

$5 + (5 - m.c.d(5,5)) \leq 5 \rightarrow 5 \leq 5$
$5 + (5 - m.c.d(5,10)) \leq 10 \rightarrow 5 \leq 10$
$5 + (5 - m.c.d(5,15)) \leq 15 \rightarrow 5 \leq 15$

Los posibles valores par Ts se nos queda en [1,2,3,5]

Eligiendo para el **ciclo secundario 5 s** (Ts), nos queda así la planificación:

Ciclo	Tareas
1	LV , LP , AE
2	A , F , FO , M
3	LV , LP , AE
4	A , F , FO
5	LV , LP , AE , M
6	A , F , FO

Respecto al código añadimos las siguientes funciones para las distintas tareas:

Tareas	Funciones
Activar Modo Emergencia (1 vez Arduino)	task_emergency()

Se ha añadido una función para el modo emergencia con su planificador dentro y al tener que comprobar que no hemos sobrepasado el pasado el tiempo del ciclo secundario hemos creado la función check_cycle() para ello. En las tareas se ha contemplado el nuevo modo ya que por ejemplo hay que desactivar el acelerador y encender el freno constantemente en el modo emergencia. Para evitar que se envían más de una vez tarea de modo emergencia a arduino,

hemos creado una variable global que se pondrá a uno una vez se haya solicitado y la respuesta sea la correcta, y así no se volverá a solicitar el modo emergencia.

TIEMPO DE LAS TAREAS

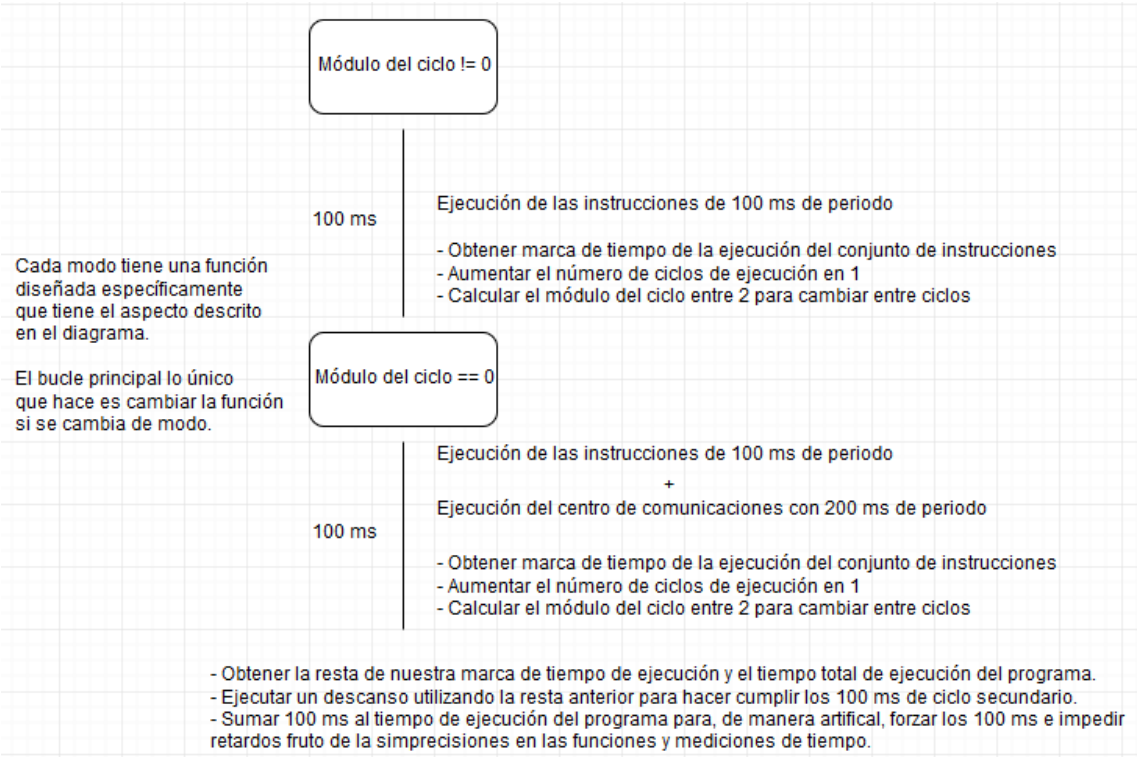
Todas las tareas duran alrededor de 0.633333348 segundos. Esto se debe a que la petición al simulador dura unos 0.5 segundos y la función display unos 0.13, entonces como cada tarea cada vez que se ejecuta va a tener que realizar esas dos funciones su tiempo de tarea es la misma.

PARTE II – Análisis del apartado hardware

I. Planificación

Se ha establecido el ciclo primario en 200 ms y el ciclo secundario en 100 ms. De esta manera todas las funciones necesarias para el control de la carretilla caben en el ciclo secundario y se permite la ejecución cada 200 ms del centro de control ya que es la tarea que se ejecuta cada más tiempo. Siendo el periodo de las demás de 100 ms.

El mecanismo de control del tiempo, el ciclo y la ejecución de las instrucciones según el modo de ejecución, está implementado con la función **micros()**. De esta manera, podemos medir el tiempo de ejecución de un ciclo y esperar si nuestras tareas tardan menos de 100 ms en ejecutarse. El siguiente esquema reflejaría el funcionamiento del planificador:



Cabe destacar el último apartado descrito en el diagrama. Con ayuda de un profesor de prácticas en sesiones de tutoría, se llegó a la conclusión de que la mejor manera para que funcionase de manera robusta nuestro planificador basado en milisegundos, era no sumar el tiempo de ejecución/espera al tiempo total de ejecución con marcas de tiempo de la función **micros()**, si no añadir de forma artificial 100 ms a la variable de control del tiempo. El sistema de esta manera no se encuentra comprometido en términos de imprecisiones en el cálculo y la medición de los tiempos.

Para la justificación del funcionamiento entre modos, cabe destacar su modularidad al estar implementados en funciones separadas y su intercambio seguro. Es decir, se termina el ciclo anterior y se completan todas sus operaciones antes de cambiar de modo. De esta forma la ejecución es regular.

El cambio de ciclo se ha decidido meter dentro de cada ciclo secundario, de esta forma, no hace falta hacer manipulaciones de ciclo de manera externa a costa de sacrificar algo de tiempo dentro de cada ciclo secundario. Aún así nos sobra mucho tiempo como veremos más adelante por lo que no representa un problema.

Por poner un ejemplo, si el primer ciclo del modo 0 se ejecuta pero se cambia al modo 1 sin que dé tiempo a ejecutar el segundo ciclo del modo 0, directamente se ejecutaría el segundo ciclo del modo 1 para no dejar descolgada la ejecución de la función **comm_server**.

II. Análisis de tiempos

A continuación, se encuentra una relación de los tiempos medidos en la última versión del controlador antes de la entrega. La medición se ha realizado por funciones separadas y finalmente por ciclo (los valores son aproximados):

Función	Tiempo
task_gas()	8-12 microsegundos
task_brake()	8-16 microsegundos
task_mixer()	8-12 microsegundos
task_slope()	8-16 microsegundos
task_speed()	72-80 microsegundos
task_lamps()	8-12 microsegundos
task_distanceSelector()	152-168 microsegundos
task_lightSensor()	112-116 microsegundos
task_displayDistance()	96-108 microsegundos
task_distanceSelect()	8-12 microsegundos
comm_server()	4-152 microsegundos
Ciclo secundario	500 microsegundos

Como se puede observar, todos los tiempos son muy reducidos y concuerdan con la planificación expuesta en el apartado anterior.

III. Comentarios adicionales

La función **comm_server** recibe los comandos del servidor central y envía las contestaciones pertinentes mediante condicionales. Por decisiones de diseño, no realiza ninguna operación en sí misma nada más que enviar las respuestas y formatear la salida de los datos. Par activar las funciones, se utilizan variables globales para cada funcionalidad. (Acelerador, freno, mezclador, lámparas, pendiente, etc.) Cuando llega el turno de ejecución dentro del planificador de la tarea concreta, se comprueba el valor de dicha variable y entonces se lleva a cabo la acción pertinente. (Encender un led, realizar un cálculo, etc.)

Las funciones que utilizan rangos de datos como **task_lightSensor** y **task_distanceSelector** pueden tener un funcionamiento errático dependiendo del entorno en donde se encuentren. Aunque en el caso de la fotorresistencia se sabe que los valores arrojados se encuentran entre 0-1023, para que sea apreciable la diferencia entre luz y oscuridad correctamente según el entorno puede llegar a ser necesario un ajuste manual. En el caso del potenciómetro hemos observado que los valores se mueven entre 500-1000, pero pueden fluctuar. Probablemente se deba a la calidad de los componentes electrónicos. Quitando esto, el funcionamiento siempre es correcto.

Para las funciones que utilizan cálculos físicos como **task_speed** y **task_displayDistance** se ha establecido el intervalo de cálculo de tiempo a siempre 0.1 segundos. Esto se debe a que es el tamaño del ciclo secundario y siempre se ejecutan todas las funciones tras este intervalo de tiempo. Con marcas de tiempo externas se introducirían imprecisiones en la medición.

La función **task_displayDistance** sirve para controlar el manejo de la pantalla entre los diferentes modos. En el modo de selección de distancia 0 controla el valor directamente recibido del potenciómetro para imprimirlo en pantalla. Para el modo de acercamiento a la estación de carga 1 la pantalla muestra la distancia restante a la estación de carga. Es necesario recordar que el modo de ejecución se controla mediante una variable global.

La función **task_distanceSelect** controla para el modo de selección de distancia 0 cuando se pulsa el botón y se suelta para cambiar de modo. (Pulsación completa, no solo pulsar) El control se hace con una bandera. Esta función también sirve para controlar cuando se termina el modo de parada pudiendo pulsar el mismo botón.

Para obtener información adicional acerca del funcionamiento, consultar el código directamente. Está comentado y estructurado de tal manera que cada funcionalidad y componente puede entenderse por separado.