Modelado y Análisis Formal del Protocolo de Comunicación de Sensores One Wire

María Emilia Cambronero Piqueras
Escuela Superior de Ingeniería Informática de Albacete
Universidad de Castilla-La Mancha

ÍNDICE

1. Motivación

Actualmente, las comunicaciones de datos entre procesos y sistemas se han convertido en uno de los pilares de nuestra sociedad.

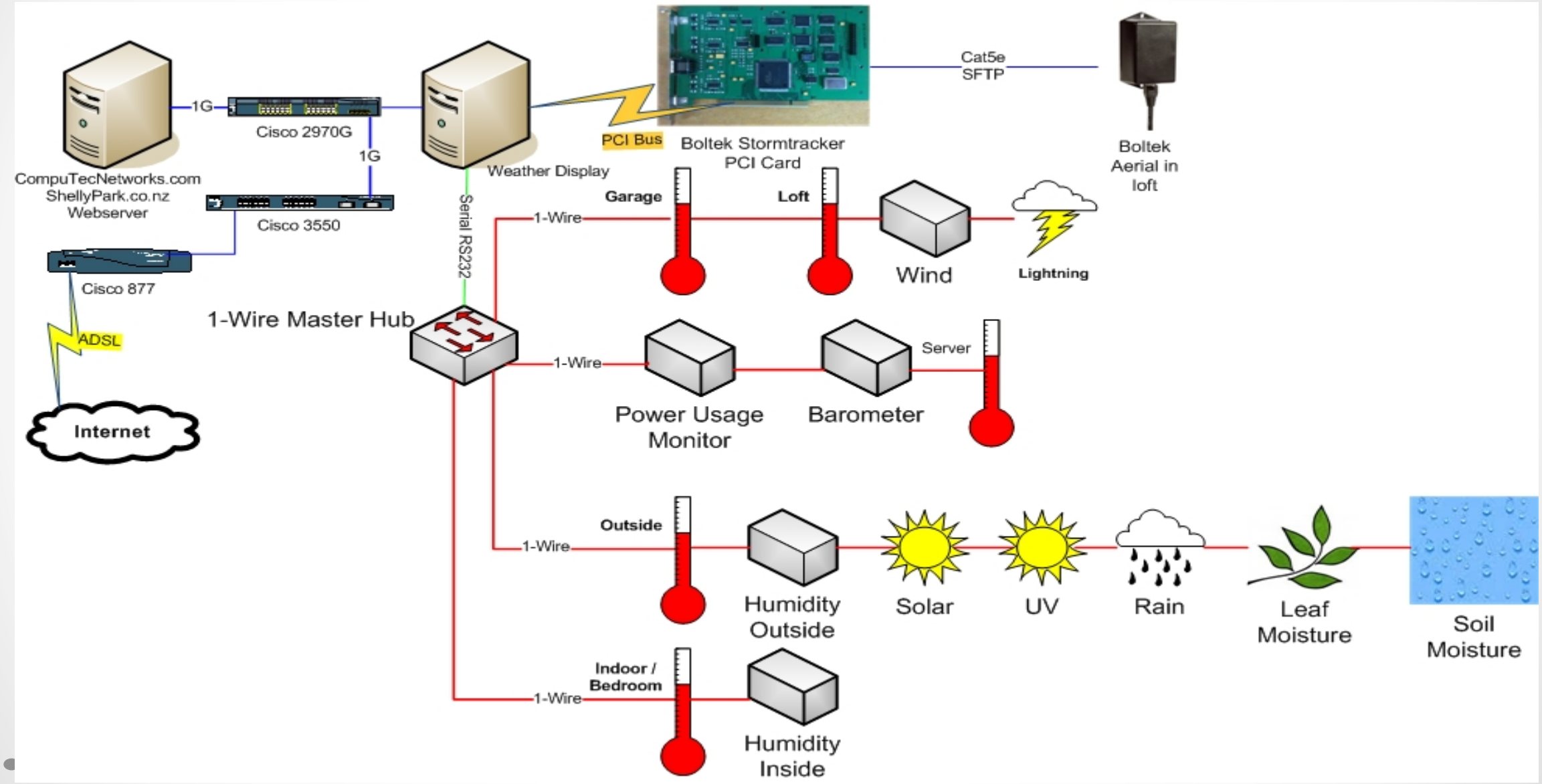
Las redes de sensores han cobrado un papel muy relevante, tanto en el mundo industrial como en el ámbito social.

La red 1-Wire, también conocida como Micro Lan, es un bus **de bajo coste** basado en un PC o un microcontrolador que se comunica digitalmente sobre un cable de 1 par con componentes 1-Wire.

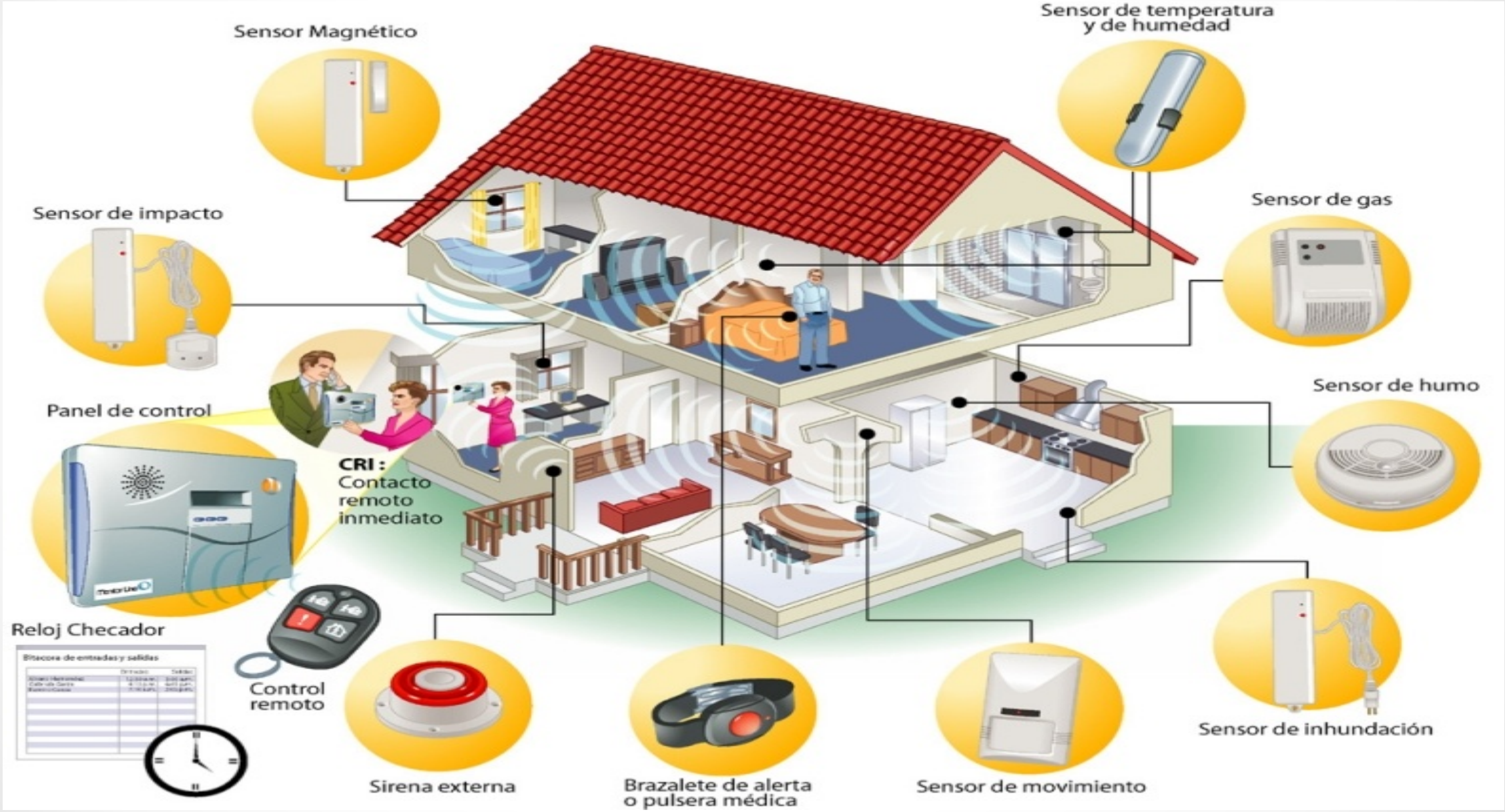
Su principal característica radica en que físicamente se compone de un único conductor, más su retorno o masa, al que se encuentran conectados todos los dispositivos 1-Wire.

Permite distancias elevadas. El límite de dispositivos depende su tipo, longitud del cableado, tipo de master, etc..., y se suele garantizar la funcionalidad de **hasta 2000 dispositivos**.

1. Motivación



1. Motivación



1. Motivación

En muchas casos, los protocolos de comunicación de sensores no han sido analizados formalmente con el fin de hacer un estudio exhaustivo y riguroso de los mismos para garantizar su coherencia.

El uso de **métodos formales** que nos permitan hacer este análisis riguroso es un paso importante en el uso de estos protocolos, ya que permiten realizar simulaciones previas a la instalación de este tipo de redes.

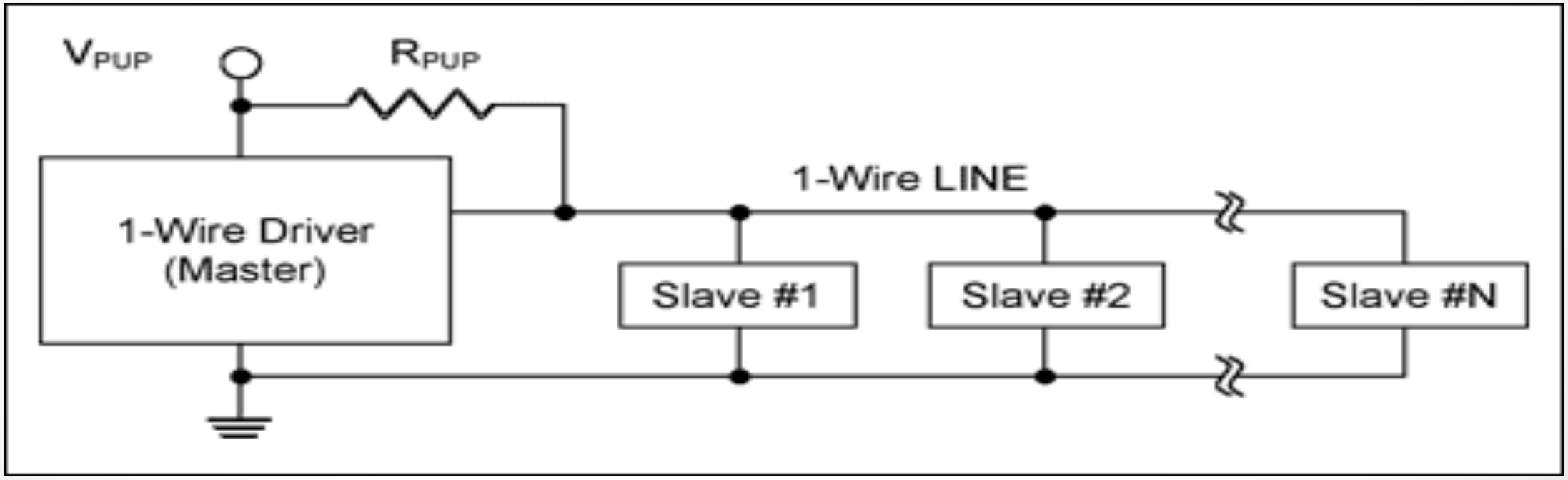
Usamos las Redes de Petri (PN) como método formal para simular y analizar el comportamiento del protocolo 1-Wire. En concreto, las **Redes de Petri Coloreadas Temporizadas (Timed Colored Petri Net (TCPNs)**.

ÍNDICE

2. Protocolo One Wire (1-Wire)

1-Wire es un protocolo de comunicaciones en serie diseñado por Dallas Semiconductor.

Está basado en **un bus, un maestro y varios esclavos de una sola línea** de datos en la que se alimentan y que se utiliza para el intercambio de datos. Por supuesto, necesita una referencia a tierra común a todos los dispositivos.



- Comunicación **bidireccional y half-duplex**.

2. Protocolo One Wire

Cada dispositivo tiene un **número de identificación, único e Inalterable (64 bits)**.

El maestro inicia y controla la comunicación.

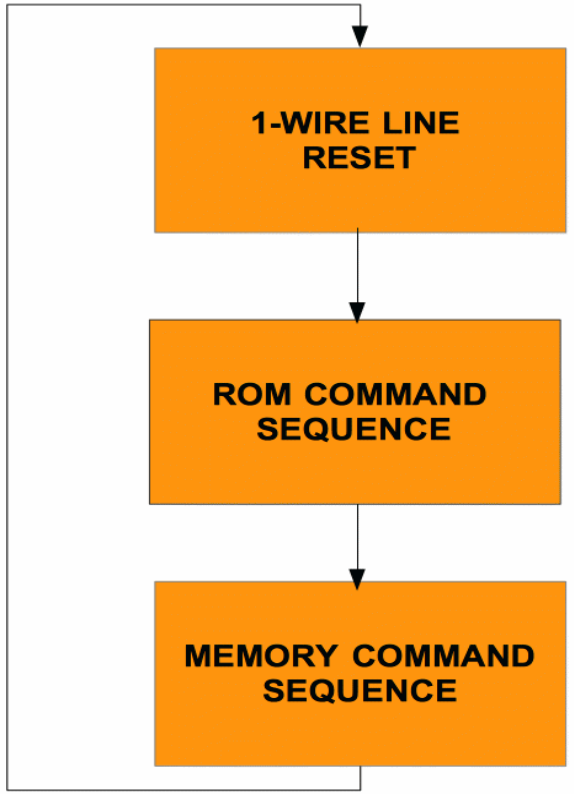
Esta comunicación se produce mediante Time Slots de 60 microsegundos.

Los esclavos se sincronizan con el reloj del maestro a través de la línea.

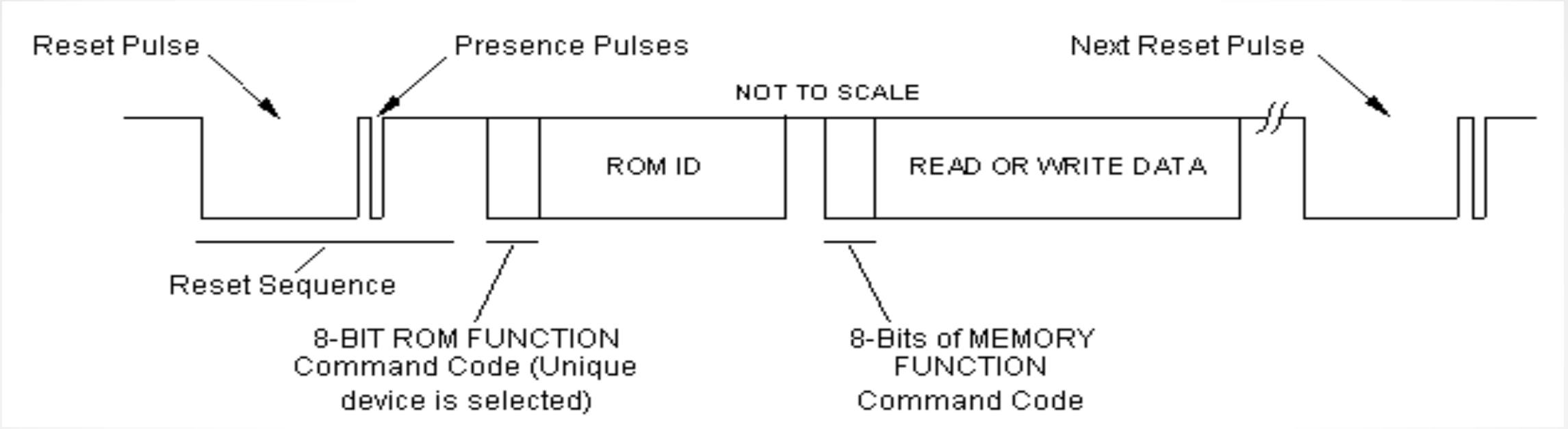
A grandes rasgos, la comunicación se realiza **en tres fases**:

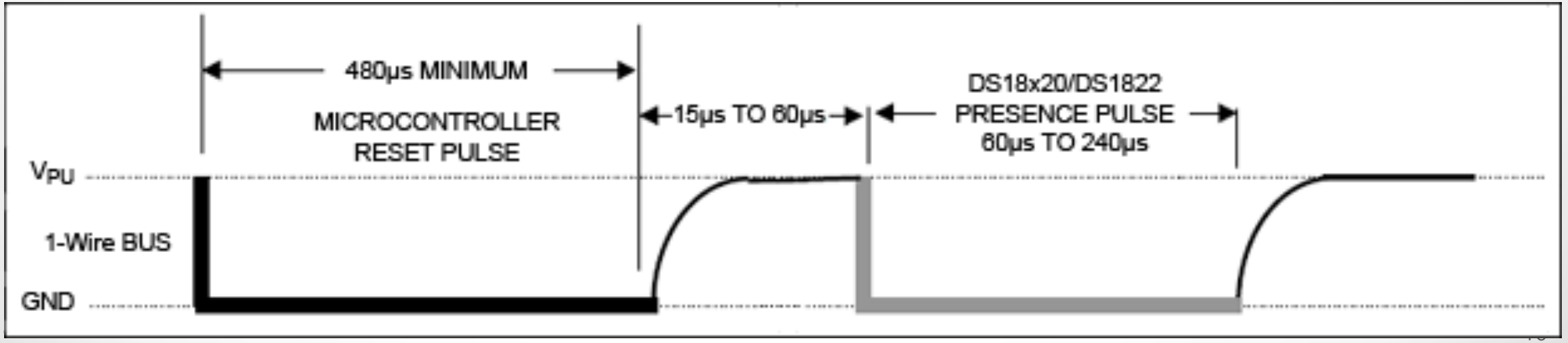
1. **Reset/Sincronización** de dispositivos.
2. **ROM Command**, p.e., seleccionar un dispositivo.
3. Función con uno de los dispositivos (**Memory Command**).

Algunas veces la comunicación puede reiniciarse después de hacer un Comando de ROM, si así lo considera el master.

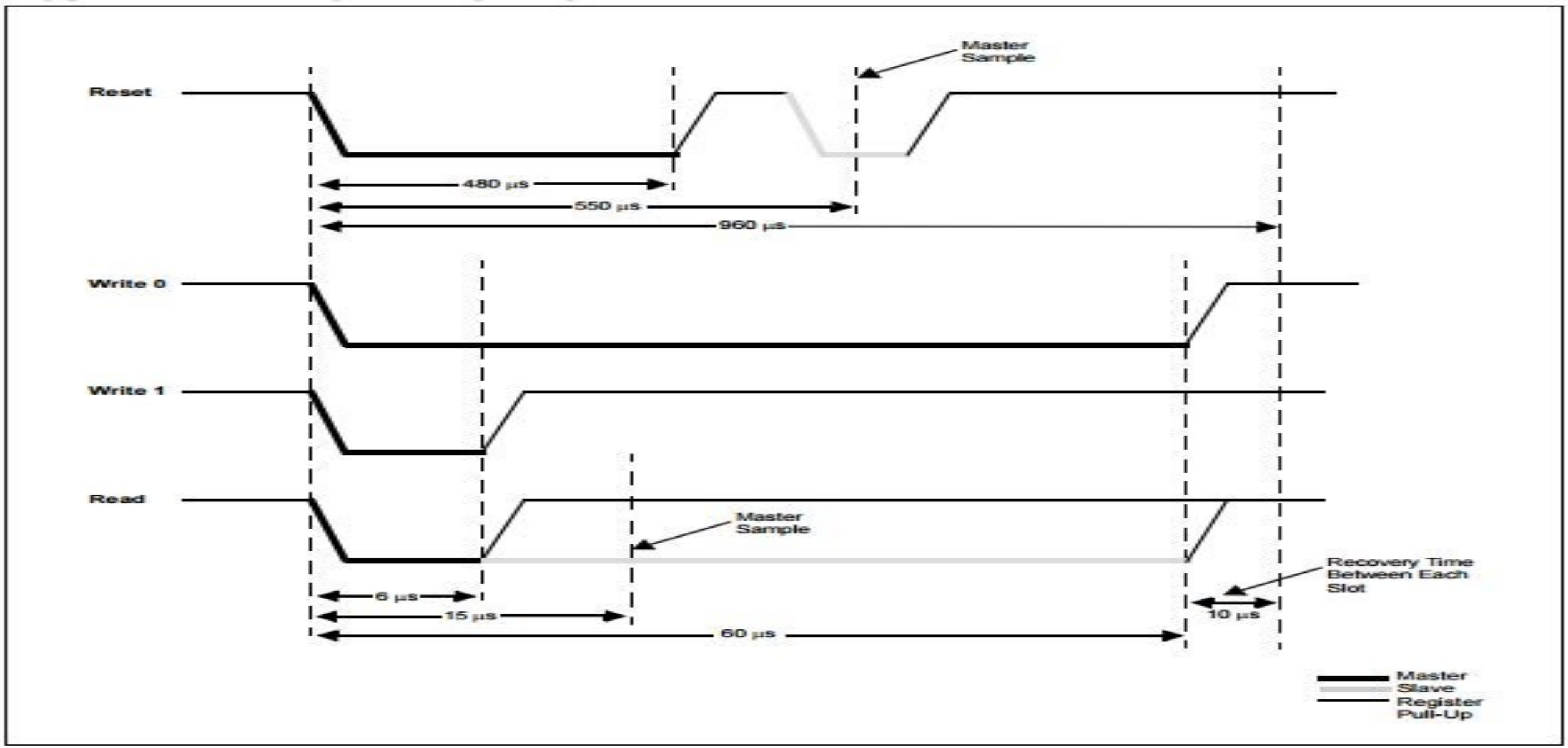


2. Protocolo One Wire





2. Protocolo One Wire (1-Wire)



2. Protocolo One Wire (1-Wire)

Comandos de ROM:

- **Read ROM**: Se usa para leer el identificador de 64 bits de **un esclavo**.

- **Skip ROM:** Se usa cuando sólo hay **un esclavo** conectado al bus, este comando es suficiente para direccionarlo. Si hay varios esclavos conectados este comando se usa para direccionar todos los esclavos de una vez.

- **Match ROM:** Se usa cuando se tienen **varios esclavos** conectados al bus y quieres direccionar uno sólo para realizar alguna función con él. Se transmite el Match ROM command junto con el id del esclavo, así sólo este dispositivo se queda a la espera de recibir un comando de memoria, el resto se quedarán a la espera de un reset.

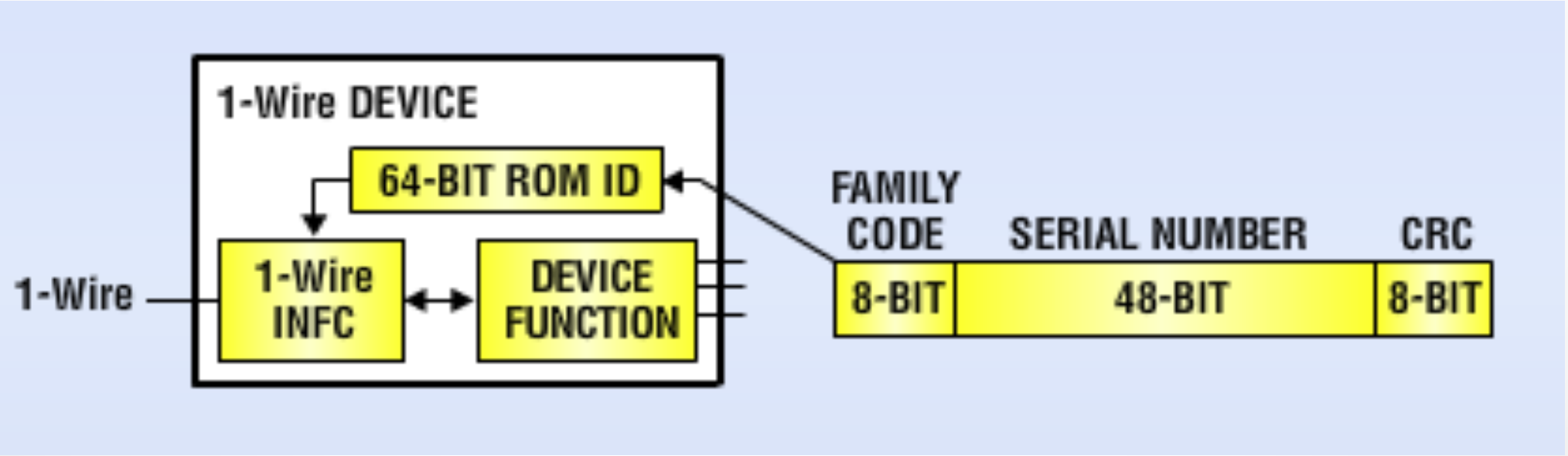
- **Search ROM:** Se usa cuando tenemos **varios esclavos** conectados en la línea y sus identificadores son desconocidos. Este comando permite descubrir los identificadores de todos los dispositivos conectados en el bus.

- **Alarm Search ROM:** Se usa cuando tenemos **varios esclavos** conectados en la línea. Identifica y direcciona todos los dispositivos cuya temperatura leída está fuera de los límites de alarma establecidos.

2. Protocolo One Wire (1-Wire)

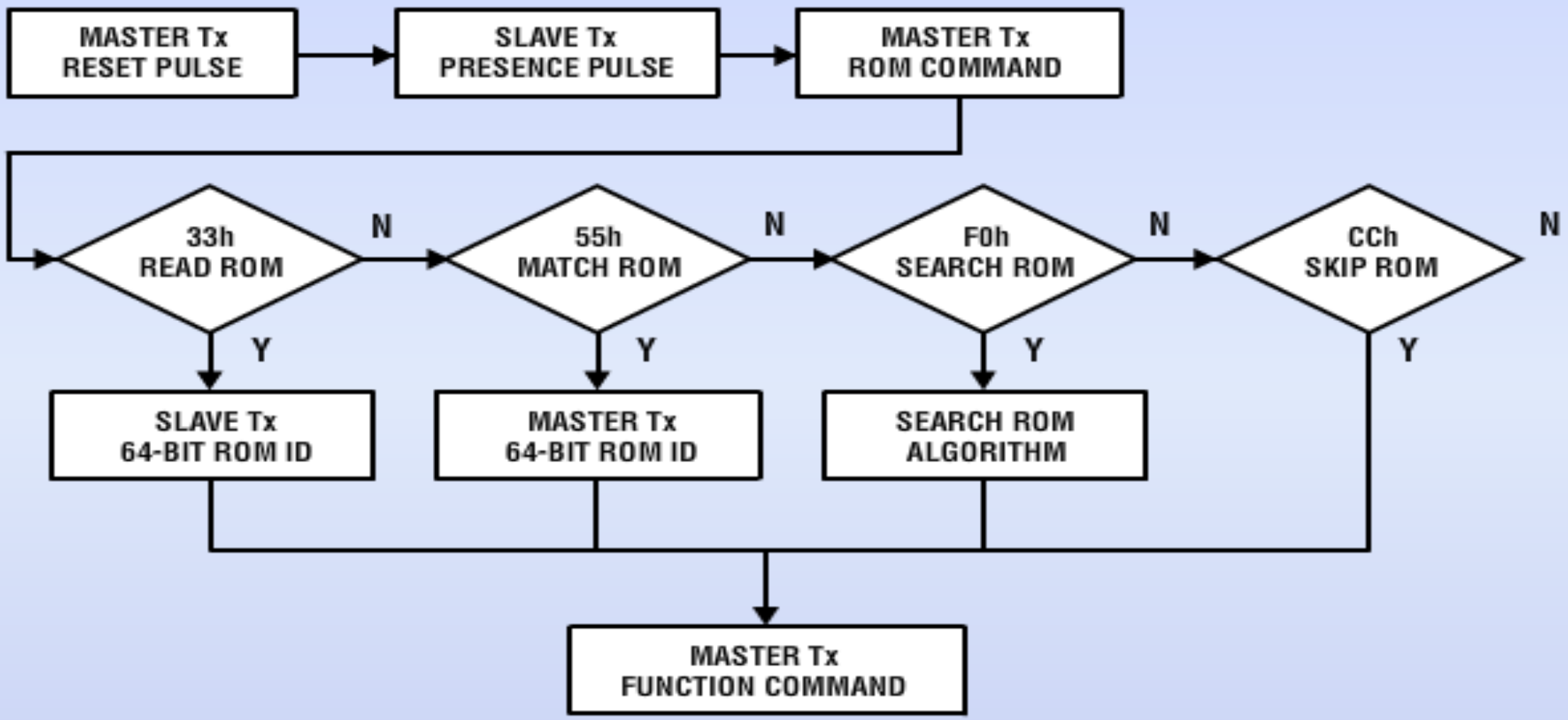
Comandos de ROM:

- **Match ROM:** Se usa cuando se tienen **varios esclavos** conectados al bus y quieres direccionar uno sólo para realizar alguna función con él. Se transmite el Match ROM command junto con el id del esclavo, así sólo este dispositivo se queda a la espera de recibir un comando de memoria, el resto se quedarán a la espera de un reset.



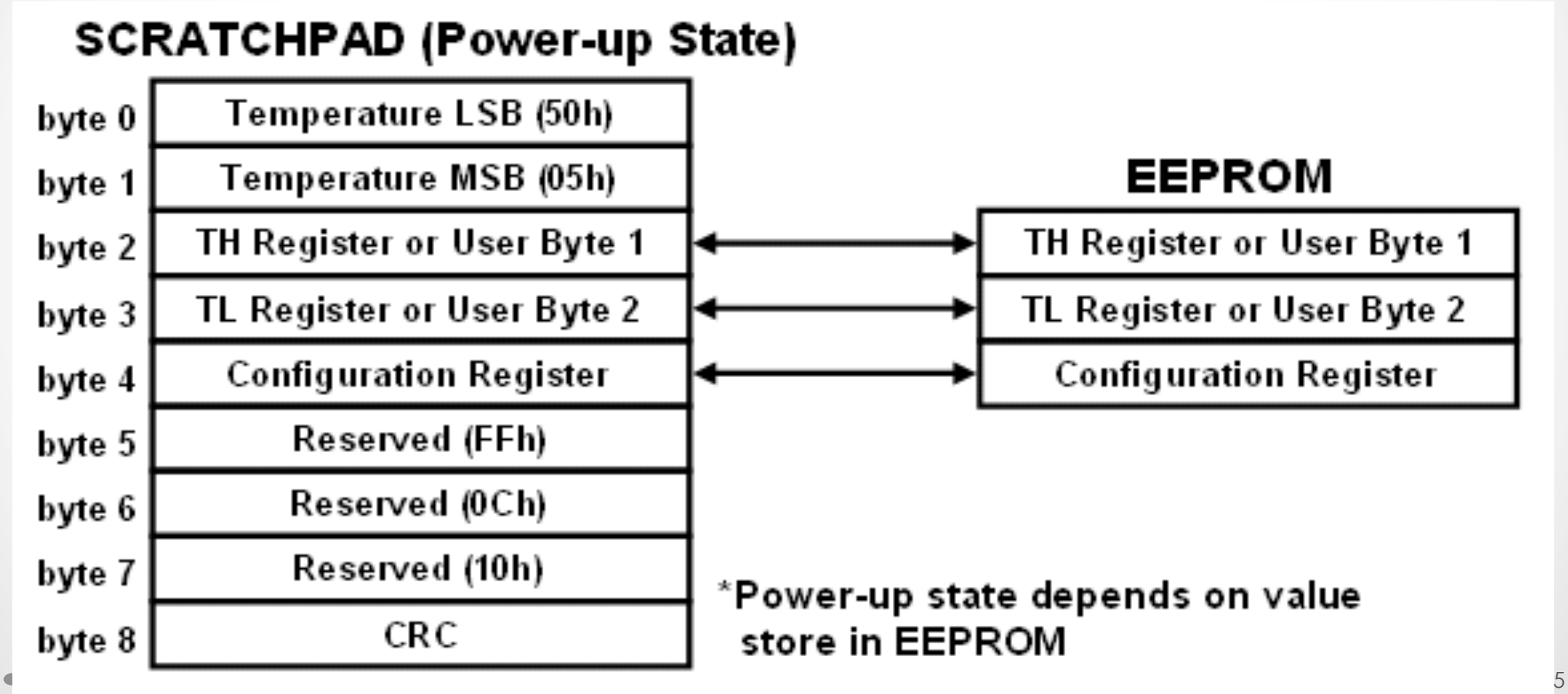
2. Protocolo One Wire (1-Wire).

Secuencia



2. Protocolo One Wire (1-Wire). Comandos Memoria

Memoria del sensor.



2. Protocolo One Wire (1-Wire). Comandos Memoria

- Estos comandos son específicos para cada tipo de sensor, por tanto, pueden variar según el tipo.

- En este trabajo hemos modelado los principales tipos de comando de memoria para un sensor de temperatura, como podría ser el DS18B20.

- Estos comando leen o escriben **la memoria interna de los dispositivos y sus registros**.

- En este trabajo se ha modelado la lectura de la temperatura del sensor y la escritura de los umbrales máximos y mínimos de alarma (**Read Memory y Write Memory**).

ÍNDICE

3. PTCPNs y CPN tools

Una Red Petri (PN) es **un gráfico bipartito** dirigido con nodos de dos tipos: **lugares y transiciones**.

Un arco puede conectar un lugar con una transición (pt-arc) o una transición con un lugar (tp-arc).
En una CPN (Colored Petri Net), los lugares tienen un conjunto de colores asociados (tipo de datos).

Cada token tiene un valor de datos adjunto (token color), que pertenece al conjunto de colores del lugar.

Una TCPN (Timed Colored Petri Net) es una **extensión temporizada de CPNs**, donde los tokens tienen un timestamp asociado. Un reloj global discreto representa el tiempo total transcurrido en el sistema.

CPN Tools es una herramienta de software utilizada para **editar, simular y analizar redes de Petri Coloreadas**. Con esta herramienta se realiza **el análisis de rendimiento** basado en la simulación y la exploración explícita del espacio de estado para la verificación de modelos.

3. PTCPNs y CPN tools

Principales características de las TCPN utilizadas:

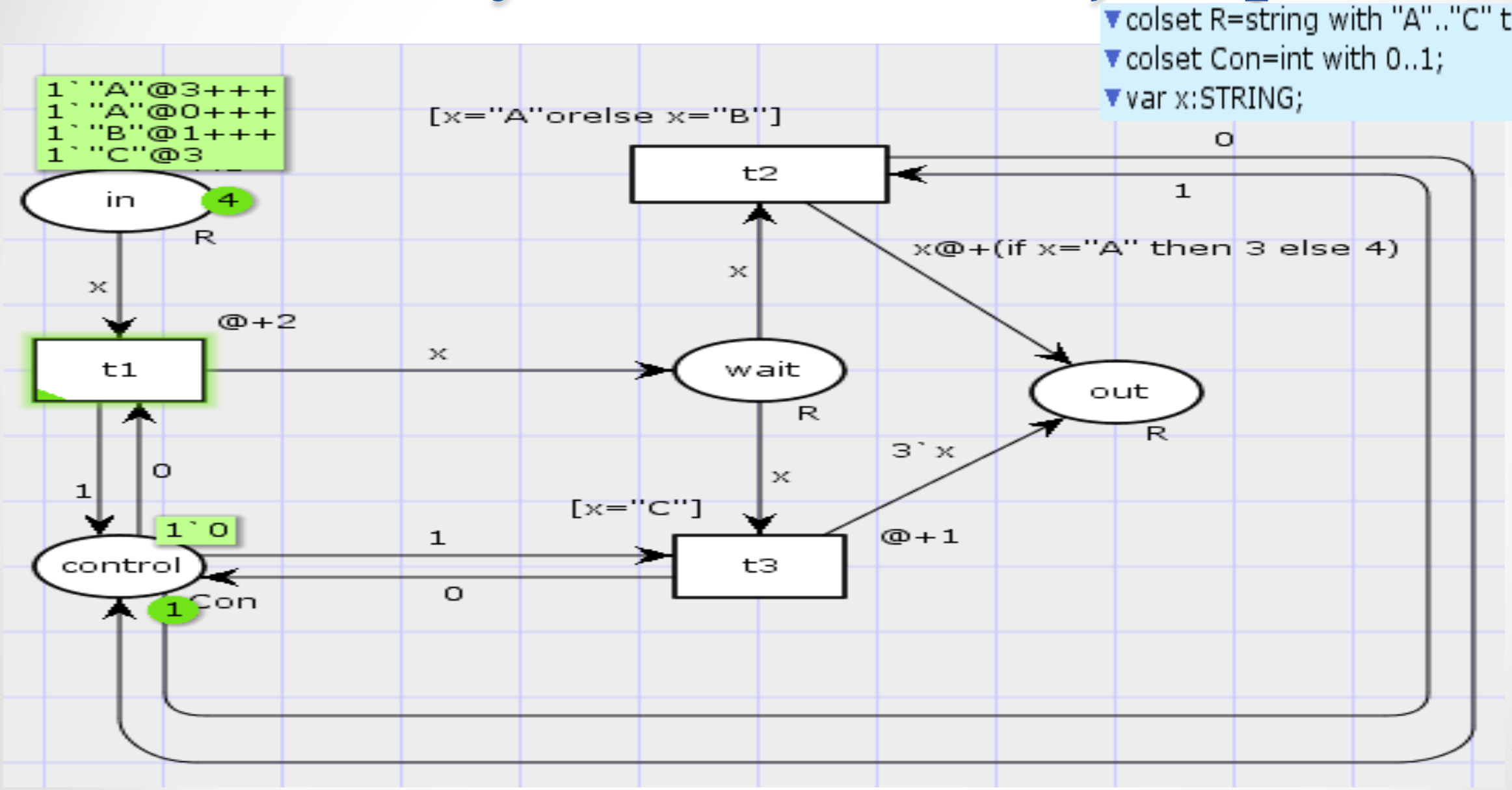
- Los arcos pueden tener inscripciones. Para evaluar una expresión de arco hay que asignar un valor a las variables que aparecen en la inscripción del arco. Estos valores se utilizan para seleccionar los colores del token.

- Las expresiones de arco también pueden tener información de tiempo asociada tanto para pt-arcs como para tp-arcs, pero solo los usamos en tp-arcs.

- Se puede asociar tiempos a las transiciones, lo que significa que los tokens producidos con su disparo serán retrasados por ese tiempo.

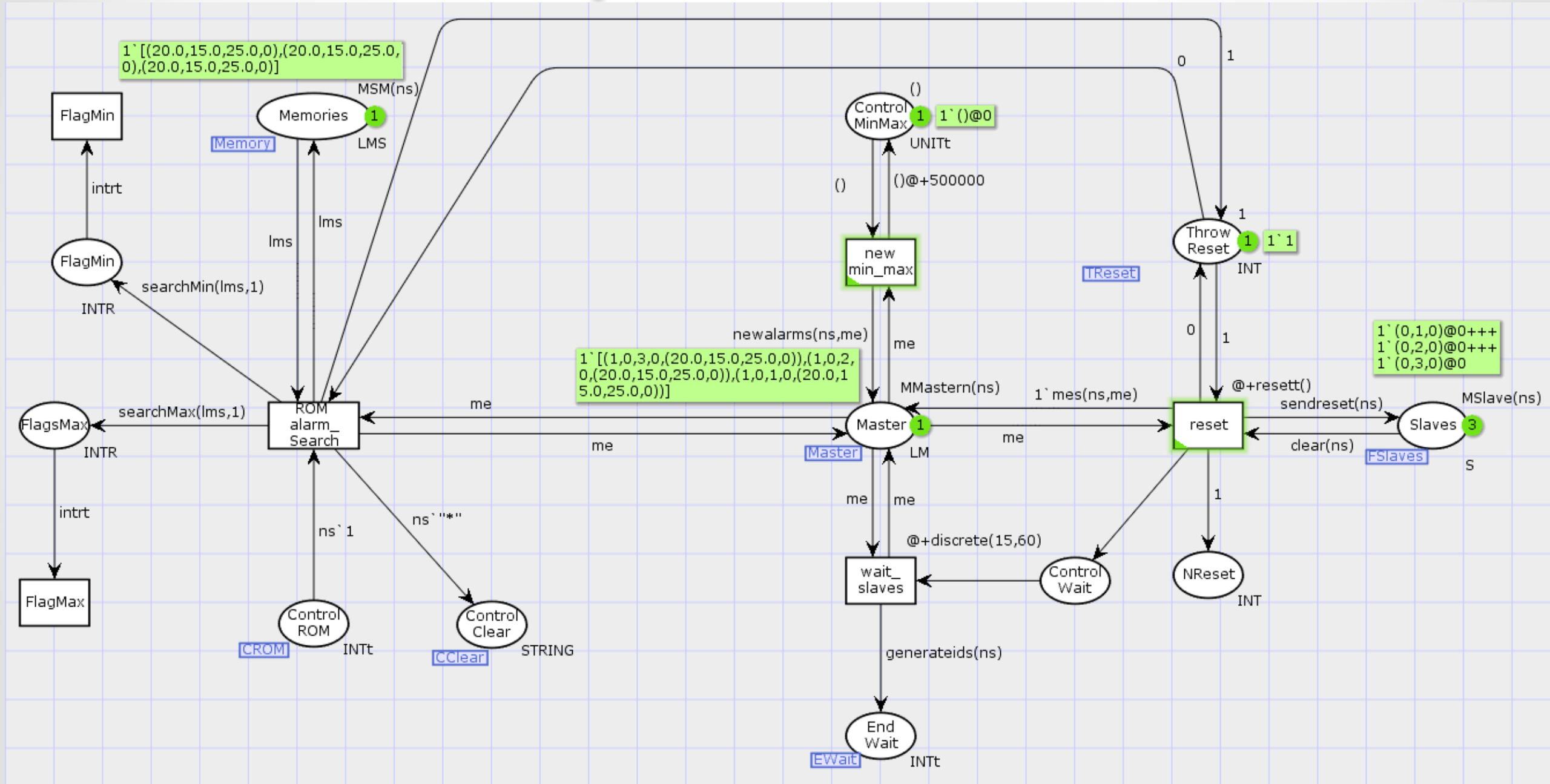
- Las transiciones pueden tener guardas asociadas, que son expresiones booleanas que pueden impedir que se disparen. Por lo tanto, una transición que tiene una guardia se debe evaluar a verdadero para que se pueda disparar.

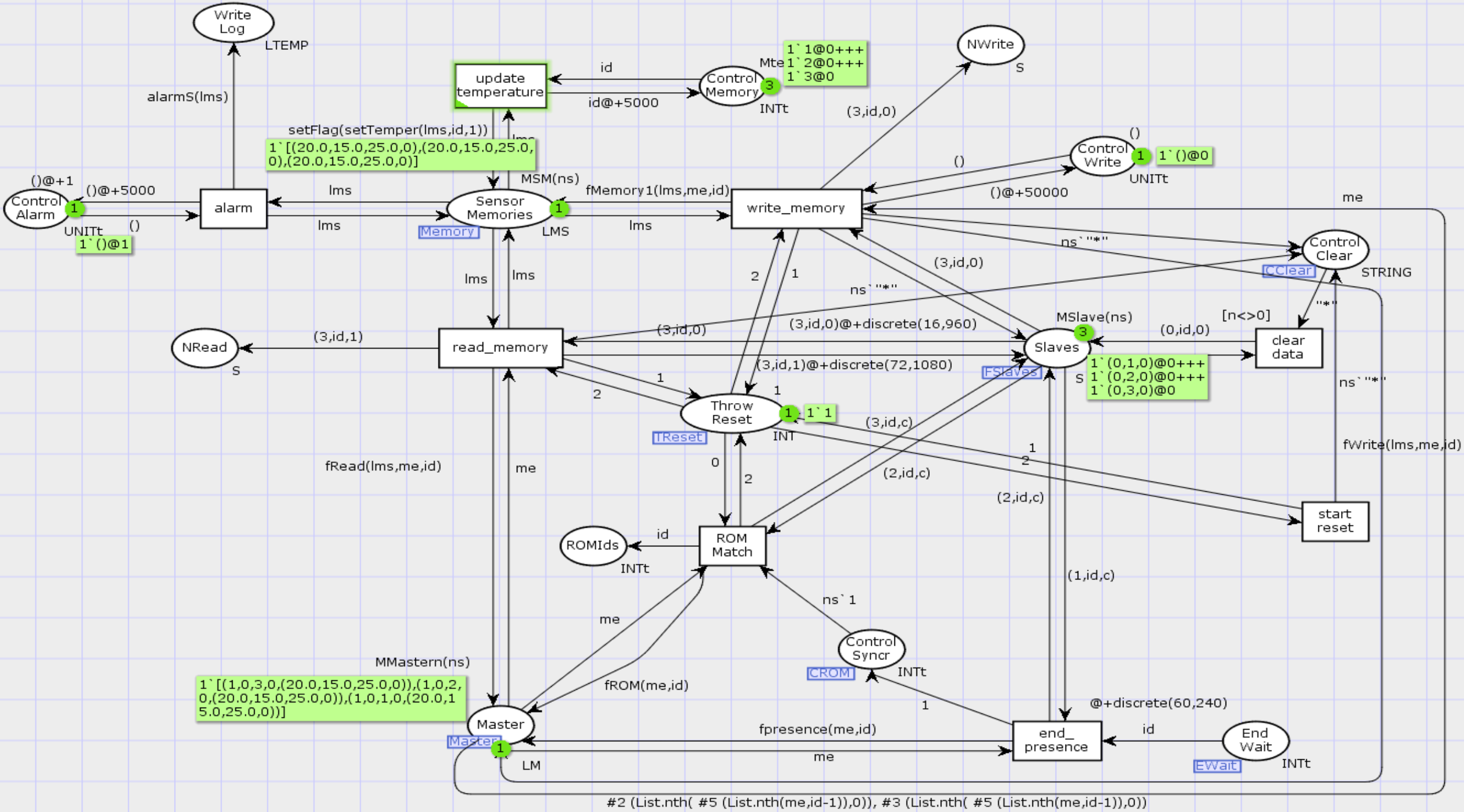
3. PTCPNs y CPN tools. Ejemplo:



ÍNDICE

4. PTCPNs para One Wire. Master



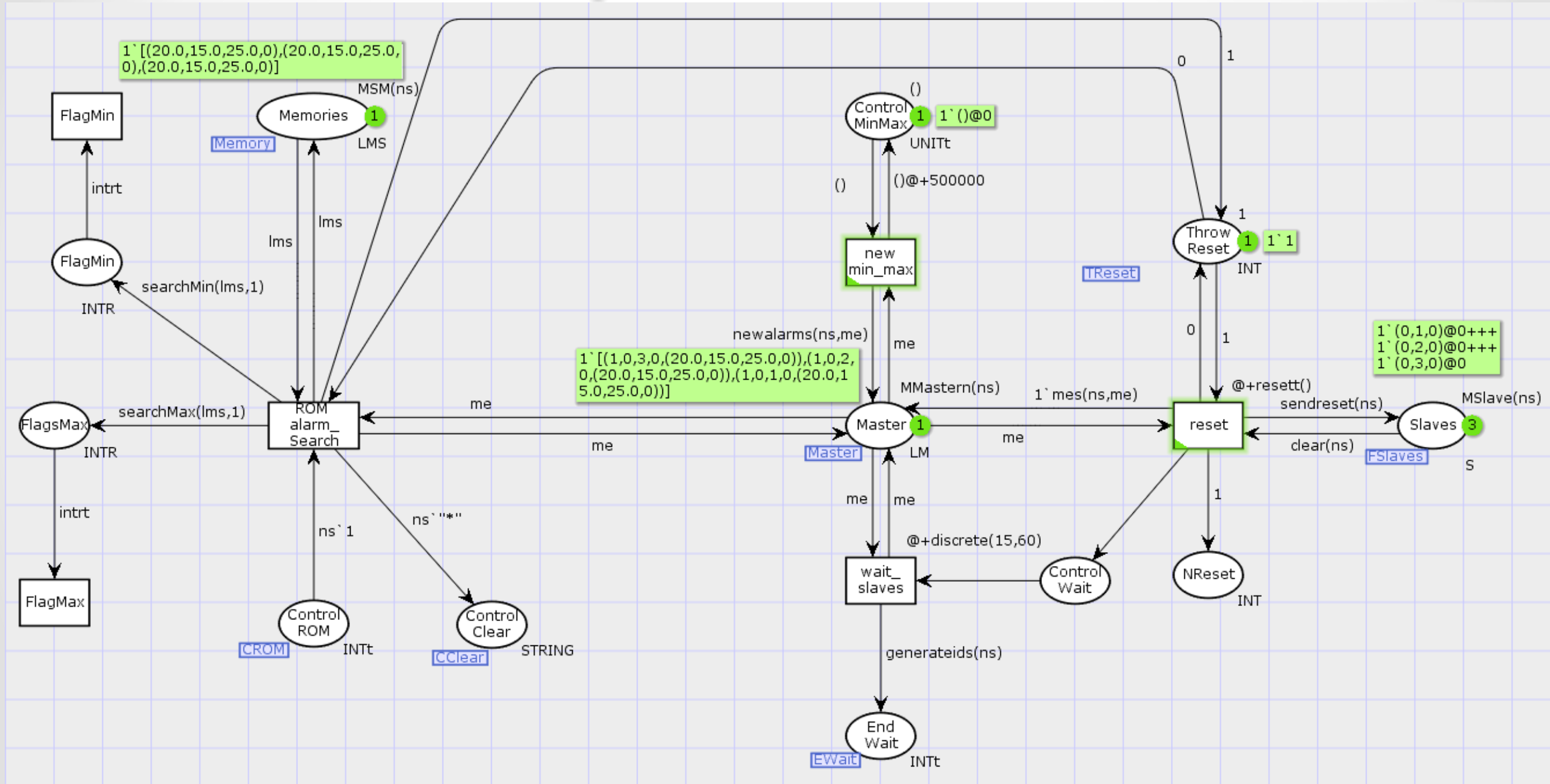

Write Log
LTEMP
alarmS(lms)
update temperature
id
id@+5000
Control Memory
Mte
1`1@0+++
1`2@0+++
1`3@0
INTt
NWrite
S
(3,id,0)
setFlag(setTemper(lms,id,1))
1`[(20.0,15.0,25.0,0),(20.0,15.0,25.0,0),(20.0,15.0,25.0,0)]
MSM(ns)
Control Write
()
1`()@0
UNITt
()@+50000
()@+1
()@+5000
Control Alarm
1`()@1
alarm
lms
Sensor Memories
Memory
LMS
fMemory1(lms,me,id)
write_memory
me
ns`"*"
Control Clear
CClear
STRING
(3,id,0)
MSlave(ns)
[n<>0]
(0,id,0)
NRead
(3,id,1)
read_memory
(3,id,0)@+discrete(16,960)
Slaves
clear data
1`(0,1,0)@0+++
1`(0,2,0)@0+++
1`(0,3,0)@0
FSlaves
(3,id,1)@+discrete(72,1080)
Throw Reset
1`1
TReset
(3,id,c)
fWrite(lms,me,id)
fRead(lms,me,id)
(2,id,c)
start reset
ROMIds
ROM Match
(1,id,c)
ns`1
Control Syncr
CROM
fROM(me,id)
MMastern(ns)
1`[(1,0,3,0,(20.0,15.0,25.0,0)),(1,0,2,0,(20.0,15.0,25.0,0)),(1,0,1,0,(20.0,15.0,25.0,0))]
Master
LM
fpresence(me,id)
end_ presence
@+discrete(60,240)
End Wait
EWait
#2 (List.nth(#5 (List.nth(me,id-1)),0)), #3 (List.nth(#5 (List.nth(me,id-1)),0))

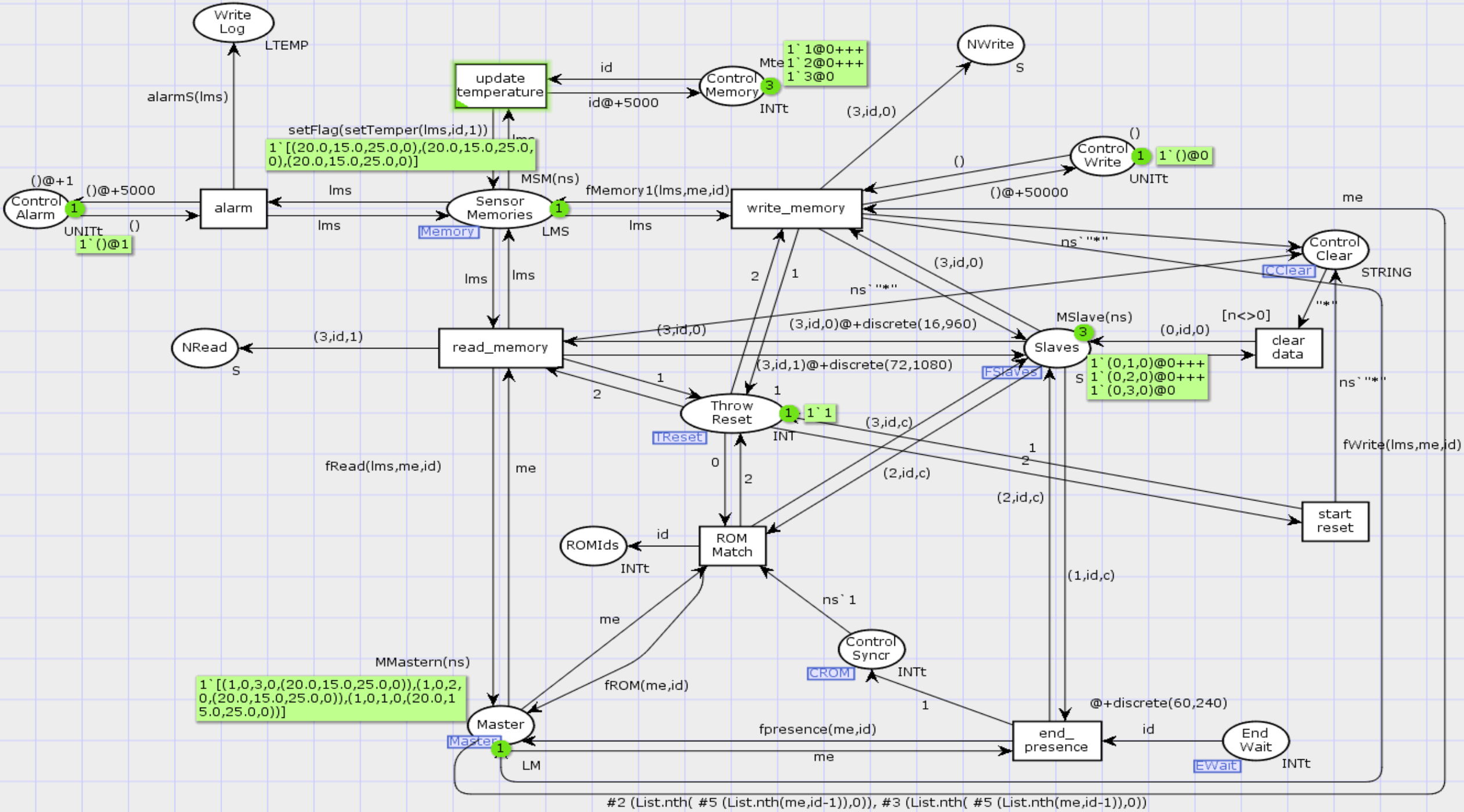
4. PTCPNs para One Wire

Declaraciones de Colsets:

```
declarations colset
  colset B = int with 0..1;
  colset D= int with 0..2;
  colset C = int with 0..3;
  colset S=product C*INT*B timed;
  colset MS=product REAL*REAL*REAL*D timed;
  colset LMS=list MS ;
  colset INTt=INT timed;
  colset TEMP= product  REAL*STRING*STRING timed;
  colset LTEMP=list TEMP timed;
  colset M=product B*INT*INT*B*MS timed;
  colset LM=list M ;
  colset UNITt=UNIT timed;
  colset REAL3=product REAL*REAL*REAL*D;
  colset INTR=product INT*REAL timed;
```

4. PTCPNs para One Wire. Master




Write Log
LTEMP
alarmS(lms)
update temperature
id
id@+5000
Control Memory
Mte
1`1@0+++
1`2@0+++
1`3@0
INTt
NWrite
S
(3,id,0)
setFlag(setTemper(lms,id,1))
1`[(20.0,15.0,25.0,0),(20.0,15.0,25.0,0),(20.0,15.0,25.0,0)]
MSM(ns)
Control Write
()
1`()@0
UNITt
()@+50000
()@+1
()@+5000
Control Alarm
1`()@1
alarm
lms
Sensor Memories
Memory
LMS
fMemory1(lms,me,id)
write_memory
me
ns`"*"
Control Clear
CClear
STRING
(3,id,0)
MSlave(ns)
[n<>0]
(0,id,0)
NRead
(3,id,1)
read_memory
(3,id,0)@+discrete(16,960)
Slaves
clear data
1`(0,1,0)@0+++
1`(0,2,0)@0+++
1`(0,3,0)@0
FSlaves
(3,id,1)@+discrete(72,1080)
Throw Reset
1`1
TReset
(3,id,c)
fWrite(lms,me,id)
fRead(lms,me,id)
(2,id,c)
start reset
ROMIds
ROM Match
(1,id,c)
ns`1
Control Syncr
CROM
fROM(me,id)
MMastern(ns)
1`[(1,0,3,0,(20.0,15.0,25.0,0)),(1,0,2,0,(20.0,15.0,25.0,0)),(1,0,1,0,(20.0,15.0,25.0,0))]
Master
LM
fpresence(me,id)
end_ presence
@+discrete(60,240)
End Wait
EWait
#2 (List.nth(#5 (List.nth(me,id-1)),0)), #3 (List.nth(#5 (List.nth(me,id-1)),0))

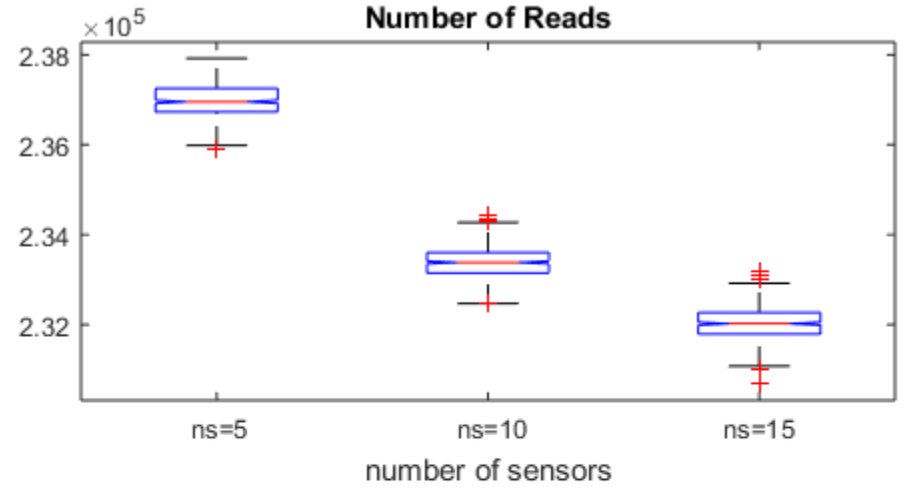
ÍNDICE

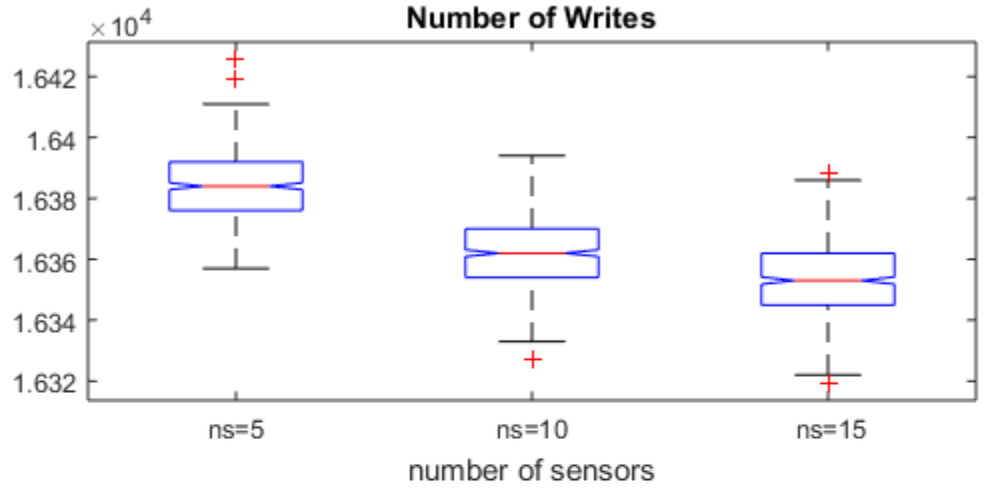
5. Análisis de One Wire

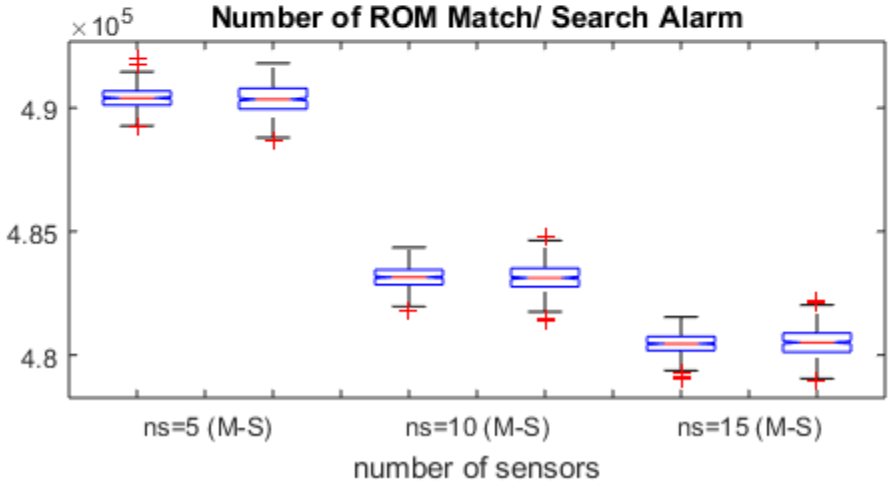
Escenario de las Simulaciones:

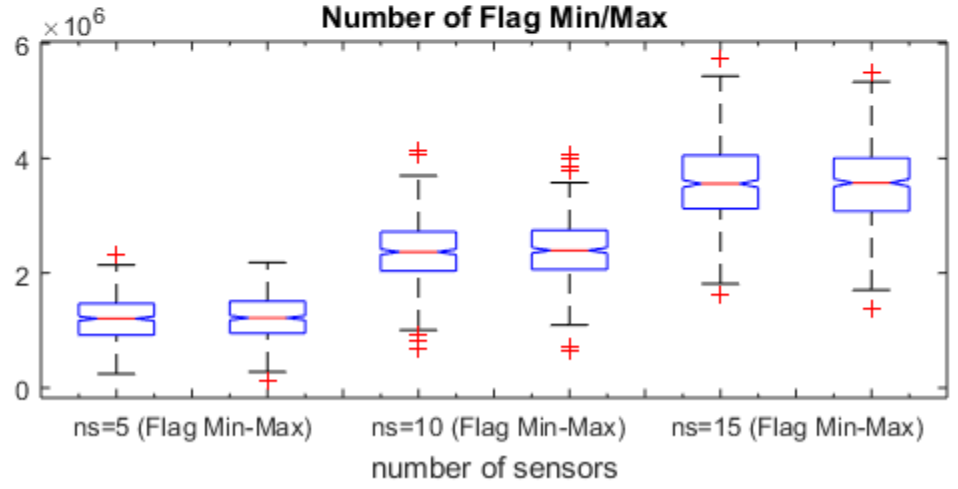
```
val  ti=(20.0,15.0,25.0,0);(*initial marking temperature*)
val mtn=0.0;(*mean of normal to modify temperature*)
val sdtn=3.0;(*dt of normal to modify temperature*)
val man=0.0;(*mean of normal to modify alarm*)
val sdam=2.0;(*dt of normal to modify alarm*)
```

5. Análisis de One Wire









5. Análisis de One Wire

	Avrg	90% Half Length	95% Half Length	99% Half Length	StD
ns=5					
Flag Max	1223699.952000	29130.358380	34816.042787	46082.121148	392394.346665
Flag Min	1200924.342000	29122.163385	34806.248287	46069.157259	392283.957716
ROM Alarm Search	490355.414000	42.492317	50.785998	67.219774	572.383792
ROM Match	490388.492000	31.548062	37.705635	49.906753	424.961512
Read	236990.154000	26.819306	32.053917	42.426203	361.263800
Write	16384.248000	0.830840	0.993003	1.314328	11.191649
ns=10					
Flag Max	2402746.714000	39845.349191	47622.393251	63032.462034	536728.369755
Flag Min	2375051.258000	39827.833196	47601.458471	63004.752996	536492.424237
ROM Alarm Search	483128.520000	42.808570	51.163977	67.720062	576.643805
ROM Match	483143.128000	32.893598	39.313794	52.035294	443.086270
Read	233379.308000	25.749902	30.775787	40.734484	346.858627
Write	16361.996000	0.836792	1.000117	1.323744	11.271827
ns=15					
Flag Max	3558448.274000	49956.849496	59707.463494	79028.124564	672933.201289
Flag Min	3571859.452000	49948.599168	59697.602861	79015.073141	672822.066988
ROM Alarm Search	480514.368000	42.410314	50.687989	67.090050	571.279183
ROM Match	480447.290000	31.474399	37.617595	49.790224	423.969251
Read	232028.416000	27.089389	32.376715	42.85345	364.901896
Write	16353.432000	0.876669	1.047778	1.386828	11.808992

5. Análisis de One Wire

Las carácterísticas de los PCs son:

- Core i7, 16 Gb RAM…

Tiempos para realizar una simulación con 500 replicaciones:

- Para 5 esclavos: 220 sg/simulación.
- Para 10 esclavos: 474-481 sg/simulación.
- Para 15 esclavos: 824-830 sg/simulación.

ÍNDICE

6. Conclusiones

Se ha presentado **un modelo formal genérico del protocolo de comunicación de sensores 1-Wire**.

El modelo es escalable, por lo que se puede tener el número de sensores que se desee, con sólo modificar el valor de una variable entera.

El modelo permite **simular y analizar el comportamiento** de una red de sensores de temperatura utilizando el protocolo 1-Wire.

El análisis nos permite comprobar el funcionamiento del protocolo, así como los tiempos de ejecución de las distintas acciones definidas en el mismo.

6. Trabajo Futuro

Realizar un análisis más completo del protocolo con **mayor número de sensores.**

Aplicar el protocolo modelado a **varios casos de estudio reales**.

Modelar el comando de **ROM Search** que permite obtener los identificadores de todos los dispositivos conectados en el bus, bit por bit.

Modelado y Análisis Formal del Protocolo de Comunicación de Sensores One Wire

María Emilia Cambronero Piqueras
Escuela Superior de Ingeniería Informática de Albacete
Universidad de Castilla-La Mancha