PRÁCTICA 2. CURSO 2014/15
INFORMÁTICA INDUSTRIAL
Grado Ingeniería Electrónica Industrial y Automática.

Se pretende controlar la temperatura en reactor industrial (*figura 1*) de forma que esta permanezca constante en torno a una referencia dada. La temperatura en el reactor disminuye en 0.1ºC cada segundo cuando no existe aporte de energía (cuando la válvula de agua caliente está cerrada). Para realizar el control se dispone de un sensor que captura la temperatura del proceso y que envía al sistema de control. Este es el encargado de activar o desactivar el paso de agua caliente a través de una válvula. El paso de agua caliente incrementa la temperatura en reactor en 0.3ºC cada segundo.

La política que sigue el sistema de control es en base a la señal de error obtenida:

```
error = tempRef – temp.
```

El control solo activa la apertura o cierre cuando detecta una desviación porcentual superior al 10% de la referencia. Es decir:

- `Si (error > 0.1*tempRef) ABRIR Valvula`
- `Si (error < -0.1*tempRef) CERRAR Valvula`

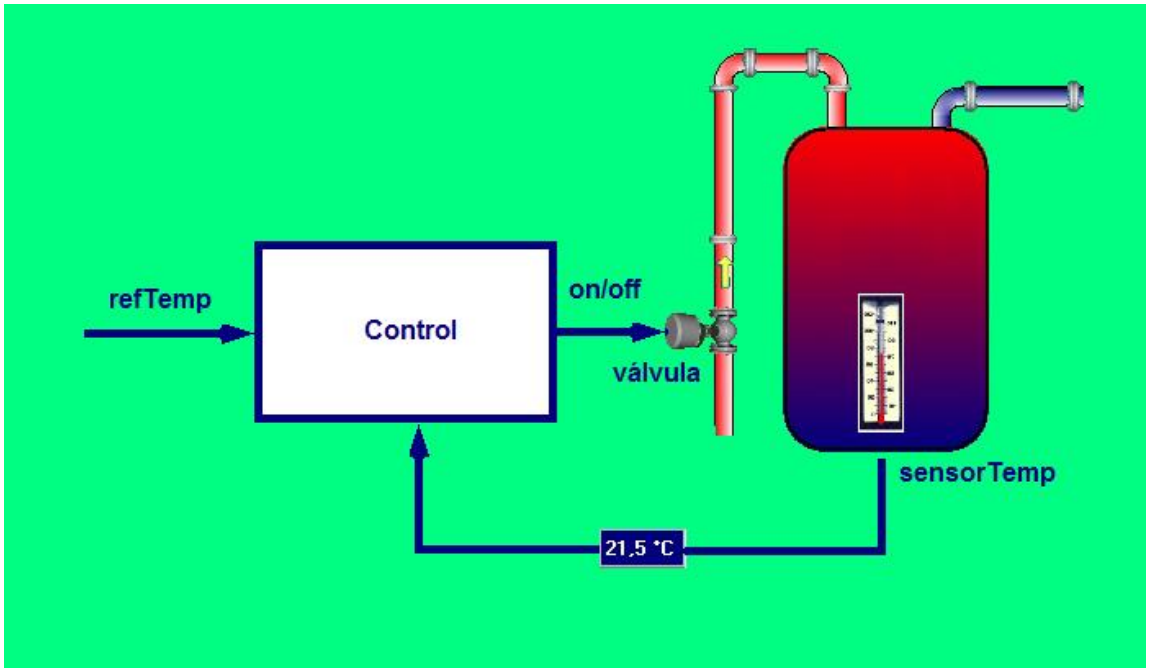

Figura 1

Se pide programar los cuatro procesos siguientes:

1. Control
2. Sensor
3. Válvula
4. Temperatura del reactor

para simular el sistema planteado. Se emplearán colas de mensajes y memoria compartida POSIX para su comunicación según se indica en la figura 2.

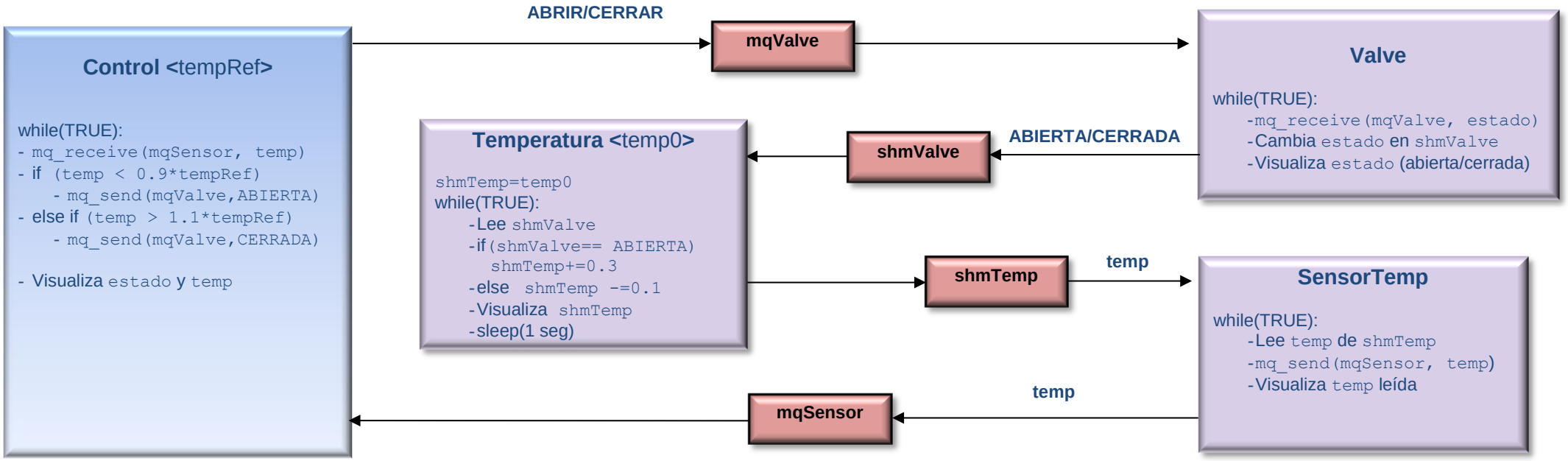
ABRIR/CERRAR
mqValve
Control <tempRef>
while(TRUE):
- mq_receive(mqSensor, temp)
- if (temp < 0.9*tempRef)
- mq_send(mqValve,ABIERTA)
- else if (temp > 1.1*tempRef)
- mq_send(mqValve,CERRADA)
- Visualiza estado y temp
Valve
while(TRUE):
-mq_receive(mqValve, estado)
-Cambia estado en shmValve
-Visualiza estado (abierta/cerrada)
ABIERTA/CERRADA
shmValve
Temperatura <temp0>
shmTemp=temp0
while(TRUE):
-Lee shmValve
-if(shmValve== ABIERTA)
shmTemp+=0.3
-else shmTemp -=0.1
-Visualiza shmTemp
-sleep(1 seg)
shmTemp
temp
SensorTemp
while(TRUE):
-Lee temp de shmTemp
-mq_send(mqSensor, temp)
-Visualiza temp leída
temp
mqSensor

Figura 2

Esquema de trabajo propuesto.

Realizar la comunicación con **colas de mensajes** entre Control-Válvula y Control-Sensor

1. Programar proceso **Control**.
   Crea colas `mqValve` y `mqSensor`.
   while(TRUE)
   mq_receive (mqdSensor, msgTemp, TAM_MSG, 0);
   Pide `accion` (abrir/cerrar) por teclado
   mq_send (`mqValve`, `accion`, TAM_MSG, 0);
   Visualiza `accion` y `temp`

2. Programar proceso **Valve**.
   Abre cola `mqValve`
   while(TRUE):
   `mq_receive(mqValve, estado)`
   Escribe `estado` (abierta/cerrada) en `shmValve`
   Visualiza `estado` (abierta/cerrada)

3. Programar proceso **SensorTemp**
   Abre cola `mqSensor`
   while(TRUE):
   Pide `temp` por teclado (float)
   `mq_send(mqSensor, temp)`
   Visualiza `temp` leída

Realizar la **comunicación por memoria compartida** entre Temperatura-Sensor y Temperatura-Válvula

4. Programar proceso **Temperatura**
   Crea `shmValve` y `shmTemp`
   Inicializa `shmTemp=temp0`
   Inicializa `shmValve=0` (off)
   while(TRUE):
      Lee `shmValve`
      if(`shmValve`==abierta) `shmTemp`+=0.3
      else `shmTemp` -=0.1
      Visualiza `shmTemp`
      Sleep(1)

5. Incorporar memoria compartida a proceso **Valve**

```
Abrir shmValve=shm_open(nombreMemoValve,O_RDWR, 0);
while(1)
{
     mq_receive(mqdValve,msgValveOnOff,TAM_MSG, 0);
     *valveOnOff  =  atoi(msgValveOnOff);
      Visualizar estado valvula
 }
```

6. Incorporar memoria compartida a proceso **sensorTemp**

```
Abrir shmTemp=shm_open(nombreMemoTemp,O_RDONLY, 0);
while(1)
 {
     sprintf(msgTemp, "%2.1f",*temp);
     mq_send(mqdSensor, msgTemp, TAM_MSG, 0);
     sleep(1);
 }
```

IMPORTANTE: Para facilitar esta primera codificación se han propuesto bucles infinitos en los programas. La parada de los procesos se hará con **`Crtl-C`** e **inmediatamente** se procederá a borrar los ficheros que hayan quedado en el directorio **/dev/shm** y **/dev/mqueue** como consecuencia de no haber destruido los mecanismos de IPCs en el programa.

Esquema mejorado empleando señales

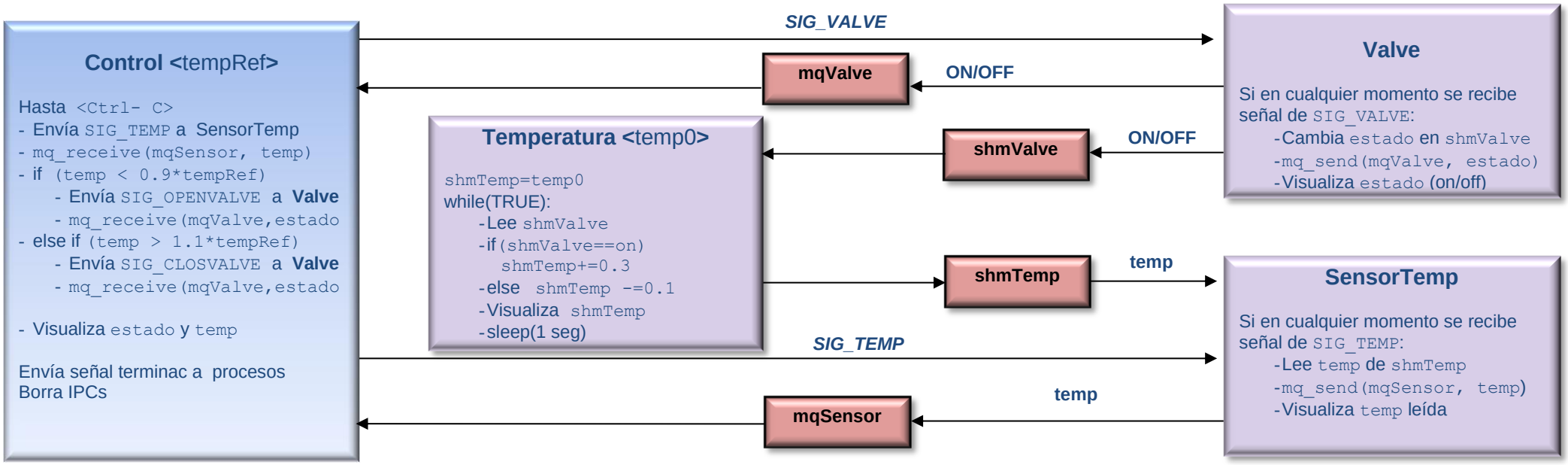

Figura 1

Este esquema permite que, al recibir el proceso **Control** la señal Ctrl-C, envíe una señal de terminación a los procesos **Valve** y **SensorTemp** y además eliminar desde el programa los mecanismos de IPCs empleados. Para envíar la señal a dichos procesos estos deberán enviar su `pid` a Control como primer mensaje.
El empleo de señales permite, por otro lado, detectar si el proceso **Valve** o **SensorTemp** están perfectamente operativos. En caso contrario, ante la llegada de la señal a estos no se producirá respuesta por la cola de mensajes, situación que puede ser detectada estableciendo un tiempo límite para la llegada de los mensajes a Control desde que fue enviada la señal de solicitud.