dit
UPM

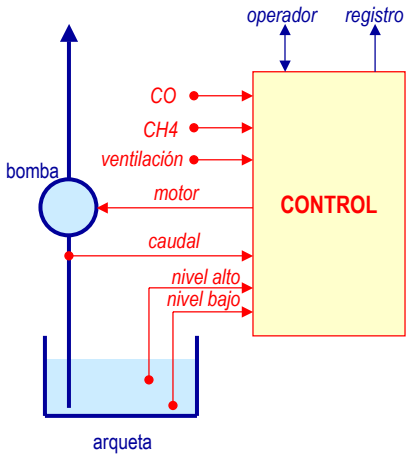
Control de drenaje de una mina

Juan Antonio de la Puente
DIT/UPM

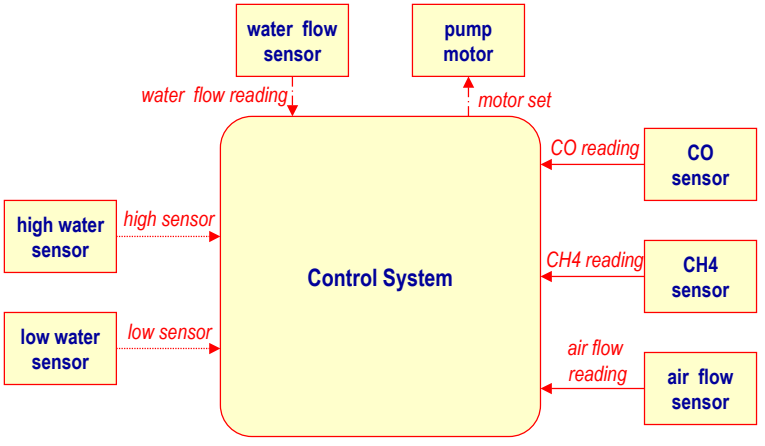
Índice

Planteamiento

- El agua que brota en el pozo de una mina se recoge en una arqueta
- Se trata de diseñar un sistema de control que mantenga el nivel de la arqueta entre unos límites, accionando una bomba
- El sistema supervisa otros parámetros ambientales
- La bomba no debe funcionar con niveles de metano altos por riesgo de explosión



Dispositivos de entrada y salida



Requisitos funcionales

- **Bomba**
  - se pone en marcha cuando el nivel de agua está alto
  - se para cuando el nivel está bajo
  - no puede funcionar si la concentración de metano es muy alta
  - el operador puede arrancar y parar manualmente la bomba
  - si no circula agua con la bomba en marcha, se activa una alarma
- **Parámetros ambientales**
  - se mide la concentración de metano y monóxido de carbono
  - se detecta si la ventilación funciona
  - si algún valor es crítico, se activa una alarma
- **Operador**
  - el operador puede dar órdenes al sistema, y recibe las alarmas
- **Registro**
  - se almacenan secuencialmente todos los sucesos significativos

Requisitos temporales: Períodos y plazos de los sensores

- **CO, CH4 y ventilación**
  - período nominal: 100 ms
  - para los manejadores de los sensores de CO y CH4 se usa desplazamiento de períodos, y la lectura dura 40 ms en total: $D \leq T - S$ = 60 ms
- **Caudal de agua**
  - período nominal: 1 s
  - se usan dos lecturas consecutivas; para ajustar el intervalo entre ambas se hace D = 40ms (960 ≤ Δt ≤ 1040)
- **Nivel de agua**
  - los sensores interrumpen cuando se activan
  - separación entre interrupciones: al menos 6 s
  - respuesta del sistema: D = 200 ms

Requisitos temporales: Plazo de desactivación de la bomba

- Cuando la concentración de metano sobrepasa el valor seguro hay que detener la bomba dentro de un plazo que asegure un margen de seguridad
  - con lectura directa
    $R\,(T - D) < M$
  - con desplazamiento de períodos
    $R\,(2T - D) < M$

    donde
    - *R*: tasa de acumulación de metano (c/s)
    - *T*: período de muestreo
    - *D*: plazo de desactivación de la bomba
    - *M*: margen de seguridad
  - suponemos $R = 5$; $M = 1000$, y hacemos,
    T = 80 ms; D = 30 ms
    para el sensor de CH4 (quedan 50 ms para hacer la lectura)

Requisitos temporales: Información al operador

- **Alarmas por exceso de metano o monóxido de carbono**
  - plazo: 1 s
- **Alarmas por ventilación insuficiente**
  - plazo: 2 s
- **Alarmas por fallos de funcionamiento de la bomba**
  - plazo: 3 s

Estos requisitos son menos restrictivos que los anteriores

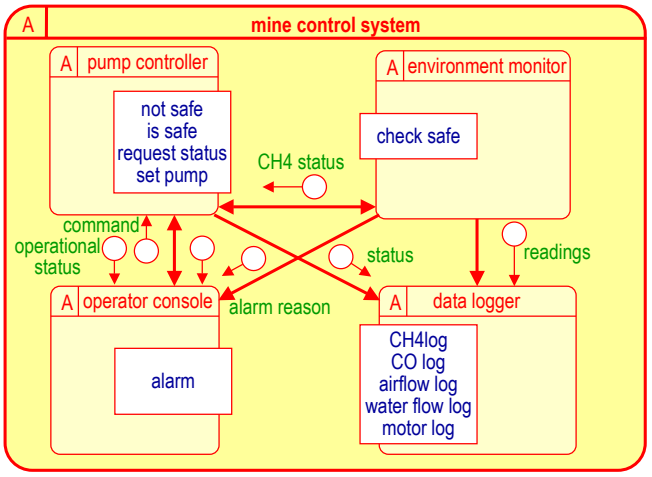
Resumen de los requisitos temporales

Sensor	Tipo	T	D
CH4	P	80	30
CO	P	100	60
Ventilación	P	100	100
Caudal de agua	P	1000	40

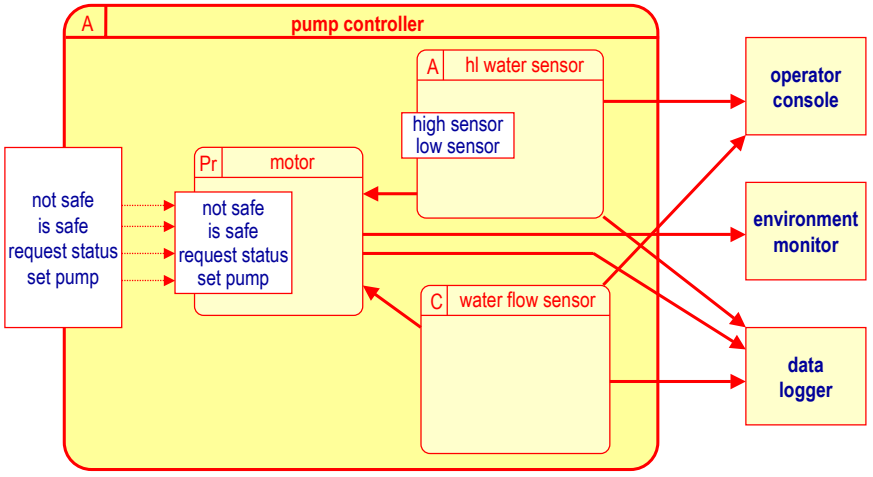
Diseño de la arquitectura lógica

- El primer paso es la identificación de objetos o clases de objetos con los que se pueda construir el sistema
- Tomando como punto de partida los requisitos funcionales descomponemos el sistema en cuatro subsistemas:
  - **controlador de la bomba**
    - » detección de nivel, arranque y parada, funcionamiento de la bomba etc.
  - **monitor ambiental**
    - » medida de metano, monóxido de carbono y ventilación
  - **consola de operador**
    - » órdenes del operador y alarmas
  - **registro de datos**
    - » almacenamiento de datos y sucesos

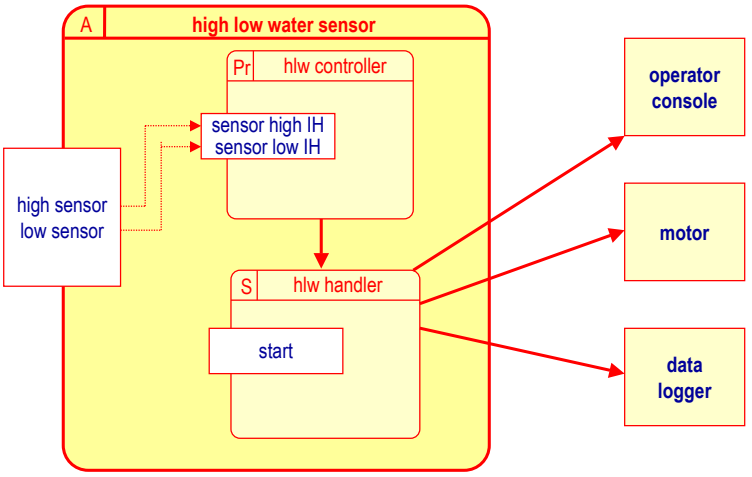
Descomposición de primer nivel



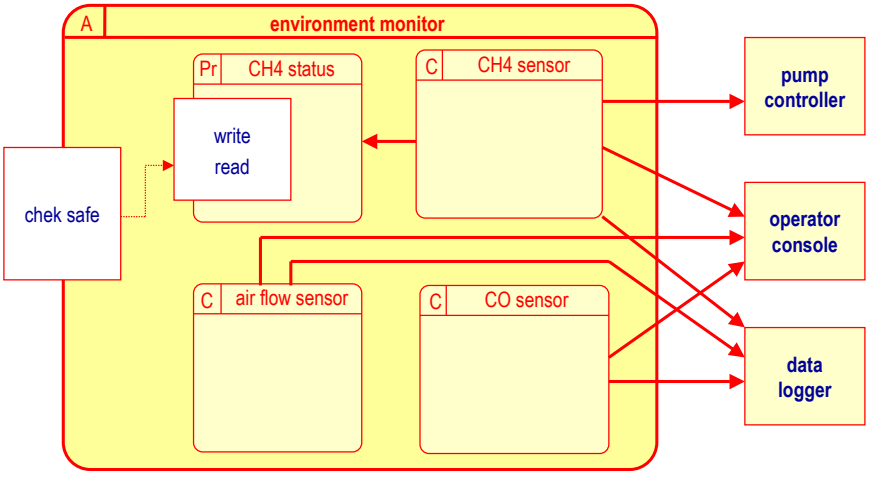
Controlador de la bomba



Sensor de nivel



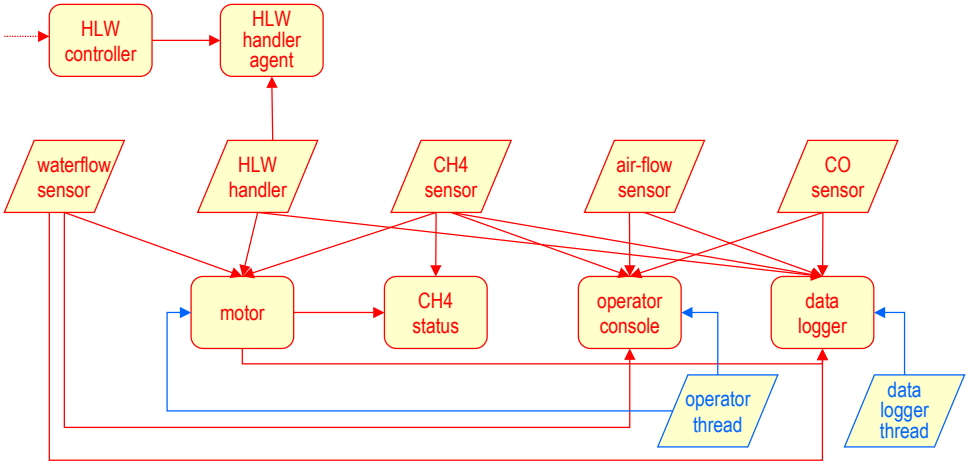
Monitor ambiental



Diseño de la arquitectura física

- El primer paso consiste en asociar atributos temporales a los objetos
  - hay que tener una estimación de los tiempos de ejecución de las operaciones y de las tareas
- A continuación se analizan los tiempos de respuesta de los objetos terminales
- Suponemos un procesador, planificación con prioridades fijas y acceso a los objetos protegidos mediante el protocolo del techo inmediato de prioridad
- Tendremos también en cuenta las características del entorno de ejecución

Diagrama de procesos



Atributos temporales de los objetos

Objeto	Tipo	T	D	P
CH4 sensor	P	80	30	10
CO sensor	P	100	60	8
Air-flow sensor	P	100	100	7
Water-flow sensor	P	1000	40	9
HLW handler	S	6000	200	6
Motor	Pr			10
HLW controller	Pr			30
CH4 status	Pr			10
Operator console	Pr			10
Data logger	Pr			10

Tiempos de ejecución

Objeto	Tipo	WCET
CH4 sensor	P	12
CO sensor	P	10
Air-flow sensor	P	10
Water-flow sensor	P	10
HLW handler	S	20
Motor	Pr	3

Parámetros del núcleo

Parámetro	Símbolo	Valor
Período del reloj	*Tclk*	20
Rutina de reloj	*CTc*	2
Encolamiento de 1 tarea	*CTs*	1
Rutina de interrupción	*Cih*	2

Análisis del tiempo de respuesta

Tarea	Tipo	P	T	C	B	D	R
CH4 sensor	P	10	80	12	3	30	25
CO sensor	P	8	100	10	3	60	47
Air-flow sensor	P	7	100	10	3	100	57
Water-flow sensor	P	9	1000	10	3	40	35
HLW handler	S	6	6000	20	3	200	79

- Se garantizan todos los plazos
- Para calcular *R* se ha tenido en cuenta el efecto del núcleo de ejecución

Realización en Ada

- Cada objeto se realiza mediante un paquete
- La descomposición de un objeto se realiza mediante paquetes hijos
- Los atributos temporales se declaran en paquetes hijos
- Hay un paquete adicional con definiciones de tipos para registros de E/S

Diagrama de módulos (1)

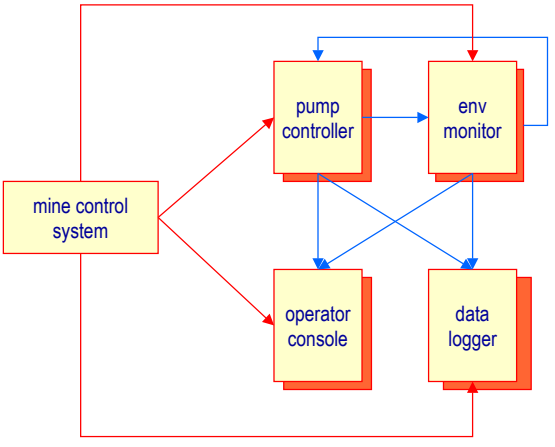


Diagrama de módulos (2)

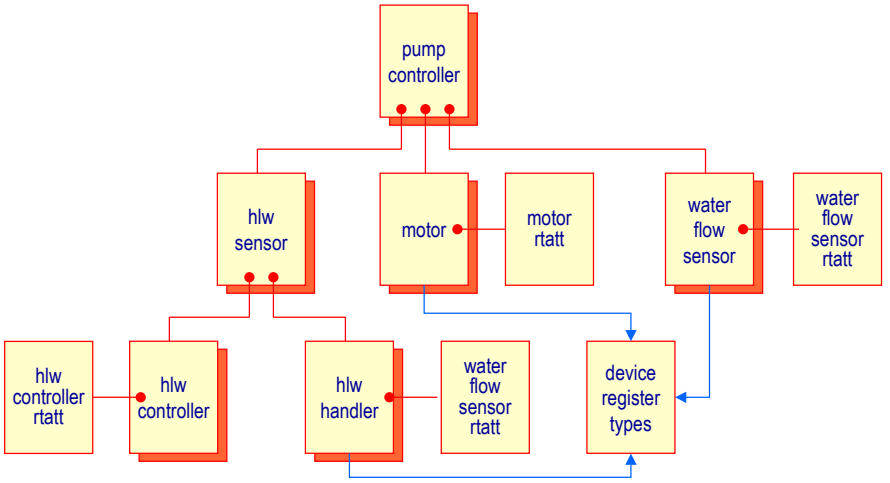
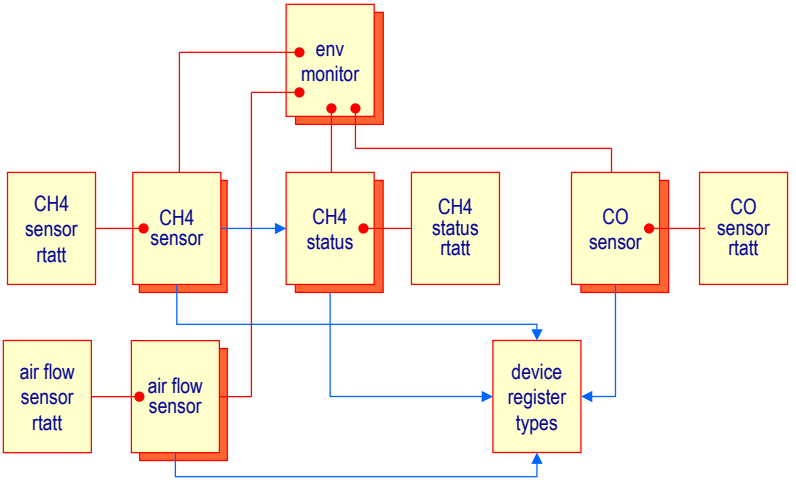


Diagrama de módulos (3)



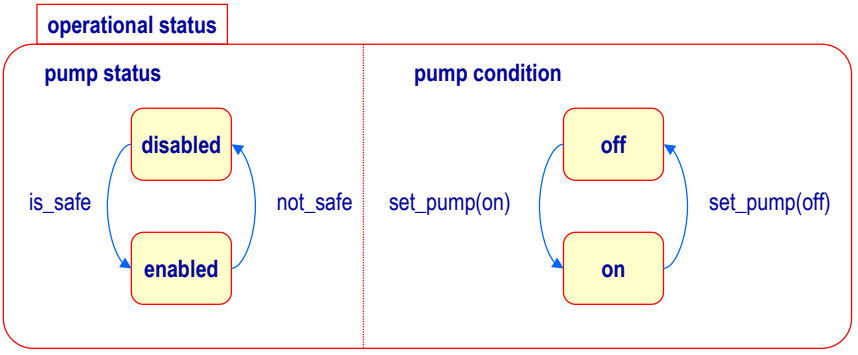
Controlador de la bomba

```
package Pump_Controller is -- active
   type Pump_Status    is  (On, Off, Disabled);
   type Pump_Condition is  (Enabled, Disabled);
   type Motor_State_Changes is
    (Motor_Started, Motor_Stopped, Motor_Safe, Motor_Unsafe);
   type Operational_Status is
     record
        Status    : Pump_Status;
        Condition : Pump_Condition;
     end record;
   type Water_Mark is (High, Low);
   type Water_Flow is (Yes, No);

   Pump_Not_Safe : exception;

   procedure Is_Safe;
   procedure Not_Safe;
   function  Request_Status return Operational_Status;
   procedure Set_Pump (To : Pump_Status);
end Pump_Controller;
```


Estados de la bomba



Motor

```
private package Pump_Controller.Motor is -- protected
   procedure Not_Safe;
   procedure Is_Safe;
   function  Request_Status return Operational_Status;
   procedure Set_Pump (To : Pump_Status);
end Pump_Controller.Motor;
```

```
with System; use System;
private package Pump_Controller.Motor.RTATT is
   Ceiling_Priority : constant Priority := 10;
end Pump_Controller.Motor.RTATT;
```


Sensor de caudal

```
private package Pump_Controller.Water_Flow_Sensor is -- cyclic
   pragma Elaborate_Body;
end Pump_Controller.Water_Flow_Sensor;
```

```
with Ada.Real_Time; use Ada.Real_Time;
with System;        use System;
private package Pump_Controller.Water_Flow_Sensor.RTATT is
   Period          :          Time_Span := Milliseconds(1000);
   Thread_Priority : constant Priority  := 9;
end Pump_Controller.Water_Flow_Sensor.RTATT;
```

Sensor de nivel

```
private package Pump_Controller.High_Low_Water_Sensor is
   -- active
   pragma Elaborate_Body;
end Pump_Controller.High_Low_Water_Sensor;
```

```
private package Pump_Controller.High_Low_Water_Sensor.Controller is
   --  protected
   pragma Elaborate_Body;
end Pump_Controller.High_Low_Water_Sensor.Controller;
```

```
private package Pump_Controller.High_Low_Water_Sensor.Handler is
   --  sporadic
   procedure Start(Int : Water_Mark);
end Pump_Controller.High_Low_Water_Sensor.Handler;
```


Realización del controlador de la bomba

```
with Pump_Controller.Motor;
with Pump_Controller.Water_Flow_Sensor;
with Pump_Controller.High_Low_Water_Sensor;
package body Pump_Controller is

   procedure Is_Safe   renames Motor.Is_Safe;
   procedure Not_Safe  renames Motor.Not_Safe;
   function Request_Status return Operational_Status
     renames Motor.Request_Status;
   procedure Set_Pump (To : Pump_Status)
     renames Motor.Set_Pump;

end Pump_Controller;
```

Monitor ambiental

```
package  Environment_Monitor is -- active

   type CH4_Reading is range 0 .. 1023;
   CH4_High : constant CH4_Reading := 400;

   type CO_Reading  is range 0 .. 1023;
   CO_High  : constant CO_Reading :=  600;

   type Methane_Status is (Motor_Safe, Motor_Unsafe);

   type Air_Flow_Status is (Air_Flow, No_Air_Flow);

   function Check_Safe return Methane_Status;

end  Environment_Monitor;
```


Consola de operador

```
package Operator_Console is -- active

   type Alarm_Reason is
     (High_Methane,
      High_CO,
      No_Air_Flow,
      CH4_Device_Error,
      CO_Device_Error,
      Pump_Dead,
      Unknown_Error);

   procedure Alarm (Reason  : Alarm_Reason;
                    Name    : String := "Unknown";
                    Details : String := "");

end Operator_Console;
```

Registro de datos

```
with Environment_Monitor; use Environment_Monitor;
with Pump_Controller;     use Pump_Controller;
package Data_Logger is

   procedure CO_Log             (Reading : CO_Reading);
   procedure CH4_Log            (Reading : CH4_Reading);
   procedure Air_Flow_Log       (Reading : Air_Flow_Status);
   procedure High_Low_Water_Log (Mark    : Water_Mark);
   procedure Water_Flow_Log     (Reading : Water_Flow);
   procedure Motor_Log          (State   : Motor_State_Changes);

end Data_Logger;
```


Procedimiento principal

```
with Pump_Controller;
with Environment_Monitor;
with Data_Logger;
with Operator_Console;

procedure Mine_Control_System is
begin
   null;
end Mine_Control_System;
```

Otros objetos

- El código completo está en el libro de texto y también en

  ftp://ftp.dit.upm.es/str/software/mine.tar.gz