Título del Trabajo Fin de Carrera

Fernando Maria Lozano Trillo
“ITI Sistemas”

Don Jodi Bécares Ferrés

Fecha Entrega

Agradecimientos

El presente trabajo no habría sido posible sin las desinteresadas aportaciones de las distintas comunidades en la internet, cuyo esfuerzo por un software libre y abierto han hecho posible que yo, un pobre estudiante, tenga acceso a conocimientos que de otra forma jamás habría podido acceder.

Agradecer especialmente a la comunidad OpenSuSE, y especialmente a la comunidad germana, la ayuda prestada, los esquemas cedidos y el asesoramiento a la hora de tomar decisiones críticas de diseño. Sin su ayuda y estimulo, mi formación no sería todo lo buena que es.

Resumen

Este trabajo de fin de carrera versa sobre la hipotética implantación de un sistema de observación, la parte que se refiere a la recolección de datos, de una presa de agua con un grupo de turbinas generadoras.

Al carecer del asesoramiento de un arquitecto, he tomado algunas decisiones basándome en diversos artículos de divulgación de diversas procedencias.

Las condiciones de mantenimiento las presupongo del protocolo habitual para el mantenimiento de pequeños sistemas electrónicos de monitorización, por ello, y sin cerrar la puerta a ello, se controlan los sensores por áreas, y no por dispositivos.

Esto quiere decir que si un grupo de motas vigila el caudal de entrada, y una mota falla dejando de transmitir, el sistema no reportará error hasta que fallen todas. Así mismo, si una de las motas desciende a un nivel crítico de energía para su alimentación, enviará una alerta general, sin identificar la mota, que podría ser identificada *in situ*, para que todas las baterías de las motas sean revisadas o renovadas.

Sé que esto último puede parecer exagerado, pero está basado en las buenas prácticas de seguridad y fiabilidad de funcionamiento, a fin de evitar averías masivas o

catástrofes. El código y el diseño están planteados para que esto pueda cambiarse si se requiriese.

En este sencillo pero, espero, representativo trabajo serán 4 las motas implicadas, más un PC o sistema basado en ARM.

2 motas basadas en la arquitectura AVR equipadas con un Atmel 1212 y un moderno IBM-PC o una consola de videojuegos Wiz (ARMv9) han sido empleados como hardware para la realización de este trabajo.

Introducción

En el presente trabajo intentaré desarrollar y exhibir mis habilidades y conocimientos sobre programación, planteamiento y viabilidad de una instalación de monitorización basada en microcontroladores y sistemas embebidos.

En la búsqueda de eficiencia, sin que esta implique pérdida de flexibilidad y libertad a la hora de implementar y diseñar el código, se tomó la decisión de utilizar el dialecto nesC y tinyOS a fin de simplificar el desarrollo e incrementar su eficiencia.

Para la correcta comprensión del código del proyecto, es preciso estar en posesión de los siguientes conocimientos:

- Sintaxis AT&T de ensamblador
- Conocimientos a bajo nivel del funcionamiento de la arquitectura AVR
- Conocimientos básicos sobre el bus USB o puerto COM bajo Linux
- Conocimientos de programación en C
- Conocimientos de programación en nesC

si no se está en posesión de tales habilidades, en internet hay una gran cantidad de recursos a disposición para aprender. Yo recomiendo, para comprender la sintaxis AT&T, el tutorial de Bas underwood.

El entorno de software es autoconfigurable si hay conexión a internet.

En el presente trabajo, se ejercitará la lectura de sensores a través de sus puertos e interrupciones generadas y el control del watchdog integrado en el propio controlador.

Se implementa los códigos de 4 motas, un componente watchdog para nesC integrable en tinyOS, un servicio de lectura de la mota para un sistema basado en Linux, que invocará las potenciales acciones para solucionar las posibles incidencias que, en el mundo real, las lecturas pudieran reflejar y un miniservidor web con su página para realizar consultas desde un terminal que sea capaz de ejecutar un navegador web que siga las recomendaciones del W3C.

Justificación

La energía hidráulica es la forma de obtención de recursos energéticos a base de explotación del medio natural mas respetuosa, económica e higiénica que los conocimientos técnicos permiten hoy día.

Las presas no sólo nos ofrecen la posibilidad de aprovechar la energía potencial e incluso cinética del agua, si no que también constituyen una reserva preciosa de agua para tiempos de sequía, una fuente natural de limo que amortiza los gastos de drenaje, y un ecosistema dónde hombre y naturaleza pueden coexistir en armonía y muy apto para el ocio y el turismo.

Si bien es cierto que la presa de las “Tres Gargantas”, en China, es un desastre que ha privado de agua a los agricultores, que se ha convertido en un estanque de aguas sucias, y además, que es lo que me ha motivado a inclinarme por el presente proyecto, estuvo apunto de colapsar por un control deficiente en la observación del estado y actividad de la presa por medios electrónicos, amén de una ausencia total de protocolos de seguridad, que condujeron a un uso inadecuado de los aliviaderos, en España las presas nos han librado de las severas sequías e impedido que la actual crisis que padecemos no se viese agravada, a demás, por una crisis humanitaria de escasez de agua.

Tomar conciencia de la importancia de la ecología y la conservación del medio ambiente es fundamental para que cada vez mas sean los que pongan la técnica al servicio de la naturaleza, y que cada vez sean mas los empresarios y banqueros los que se den cuenta que destrozar el medio ambiente es lo mismo que una gestión deficiente y negligente del medio, y que invertir en mecanismos tecnológicos de control es una inversión, no un gasto superficial en el que hay que racanear todo lo posible.

El siguiente proyecto se ha elaborado tomando cómo base las causas fundamentales que condujeron al borde de la desgracia a la presa de las “Tres Gargantas” en China.

Todo el diseño ha sido pensado para obtener las máxima seguridad al mínimo coste sin renunciar a nada, y, teóricamente, puede implantarse usando sólo la tecnología de

ChinaChips, es decir, microcontroladores de baja gama clónicos AVR y micros RISC basados en la arquitectura MIPS.

Creo que es fundamental pensar en la flexibilidad de componentes a la hora de la futura implantación, ya que la situación actual, de depresión en España y recesión mundial, hacen que el factor económico suponga la traba más importante a la hora de impulsar o promover un proyecto.

Creo que puedo decir, por mi experiencia, que el software no sólo es lo más importante, sino que es la parte que motiva la construcción de los terminales y equipos electrónicos. En la vida real, yo tendría que presentar un software mas o menos terminado, hacer una estimación de los requisitos del hardware para ejecutar dicho software, y, en base del coste del hardware agregado al coste presente e hipotético del software, hacer un estudio serio de viabilidad del proyecto.

Durante el desarrollo he partido de supuestos teóricos de los que soy consciente que pueden cumplirse o no, ya que dependen de factores específicos, como la ubicación de la presa o las decisiones tomadas por los arquitectos y otros ingenieros a la hora de crear las estructuras internas de dicho elemento arquitectónico. Por esta razón, el código está plagado de generalidades y de huecos evidentes; este proyecto es tan solo una plantilla concretable a casi cualquier proyecto, y, como en toda platilla que se precie, el camino fundamental está marcado, pero quedan cosas por hacer.

Otra cosa que he pretendido, no se si con acierto o no, pero si con toda mi intención, es promover promover el uso de programas normalizados, es decir, que para un propósito específico exista un programa que se tome como plantilla, a fin de que el ingeniero no tenga que escribir sus programas desde cero, si que las mejoras realizadas sean por y para terceros.

Porque esto, a parte de beneficioso para la comunidad científica, es sumamente conveniente para la industria, pues los programas serían más fáciles de mantener y los

costes se reducirían, el ciclo de vida del software se prolongaría y los tiempos de amortización se reducirían.

También permitiría un hardware más barato, mas rápido y más ecológico, pues consumiría mucha menos energía al ser más eficiente.

Creo que la tecnología de observación y control de sistemas de explotación de recursos naturales es una herramienta clave para que esta explotación sea segura, eficiente y sostenida y no degenere en devastación del medio, y eso, es lo que justifica mejor mi proyecto y muchos otros planteados por otras personas, que cómo yo, deseamos que la tecnología este al servicio del hombre, no el hombre al servicio de la tecnología.

Descripción

El proyecto consiste en un conjunto de cuatro tipos de motas cuya clasificación viene determinada por el área que controlan y un terminal central ejecutando Linux que ofrecerá la posibilidad de consultas vía WorldWideWeb a través del protocolo de hipertexto.

Para simplificar el modelo, consideraremos sólo cuatro motas, cada una de un tipo, que se comunican entre sí y la última se comunica con el terminal Linux.

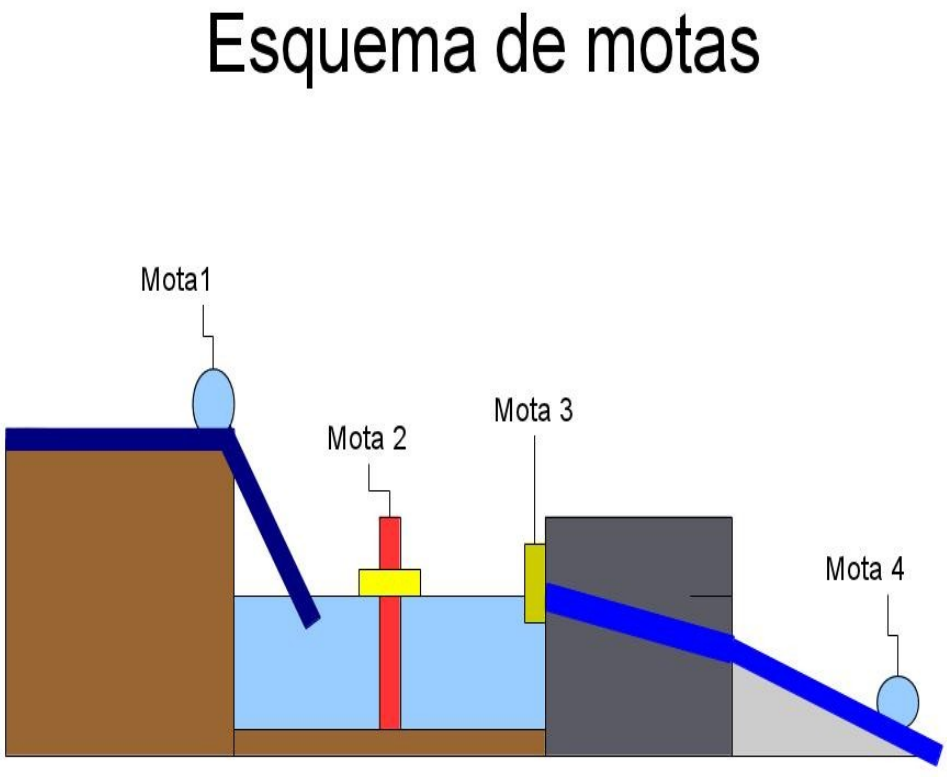


Ilustración 1: Esquema de las ubicaciones de las motas

Los requisitos para la implantación del proyecto son:

- Al menos una mota de clase 1
- Al menos una mota de clase 2
- Al menos una mota de clase 3
- Al menos una mota de clase 4
- Tantos equipos terminales Linux como motas del tipo 4 tengamos[1]
- SO Linux
- Compilador GNU para AVR
- Inyector de código e interprete nesC
- Código fuente del tinyOS
- Una herramienta de reflasheo para micros Atmel, como meshprog o avrprog

[1]O bien, un equipo terminal Linux con tantos puertos como motas del tipo 4 tengamos.

El código implementado está pensado para microcontroladores Atmel y un host terminal que ejecute Linux bajo arquitecturas x86, x86_64, ARM y MIPS[2].

[2]El proyecto ha sido probado y su funcionamiento acreditado como prácticamente viable en arquitecturas x86, x86_64 y ARMv9 a 533Mhz y 40MB de RAM disponibles para datos de usuario. En el caso de los MIPS, sólo teóricamente, sería posible hacerlo funcionar en una distribución Dingux con una cantidad de, al menos, 20MB de RAM disponibles para datos de usuario.

Ilustración 2: Wiz, la videconsola de GPH usada para hacer las pruebas

Objetivos del TFC

- Realizar una síntesis de todas las asignaturas cursadas durante la carrera
- Desarrollar y ejercitar conceptos básicos de diseño de software
- Desarrollar todo lo asimilado en la asignatura Sistemas Operativos
- Desplegar las habilidades adquiridas durante el curo de la asignatura Estructura de Computadores y su correspondiente Ampliación de Estructura de Computadores
- Ejercitar las habilidades propias de todo ingeniero para integrar sus proyectos en otros proyectos de otras ramas de la ingeniería
- Adquirir soltura en el trabajo con microcontroladores y redes de sensores inalambricos
- Estudiar la integración de los sistemas basados en microcontroladores con sistemas basados en microprocesadores, CISC y RISC, y SOCs (sistemas en un chip).
- Demostrar mis habilidades con C, nesC y el ensamblador en sintaxis AT&T
- Exhibir mi capacidad de añadir componentes a un proyecto de programación de controladores basada en tinyOS
- Análisis de protocolos de comunicación usando buferes FIFO
- Estudio de los protocolos en tiempo real y baja latencia y su adecuación al proyecto
- Coordinación de una red inalámbrica de bajo ancho de banda
- Estudios de eficiencia en las transmisiones
- Recolección de datos de los sensores adosados
- Abstracción del tipo de sensor adosado (programación independiente de la naturaleza del sensor)
- Evaluación de la implementación de un programa en base a unos recursos máximos disponibles
- Interactividad Web con los sistemas empotrados
- Diseño básico de servidores de datos para uso industrial

Enfoque

El modelo que se ha seguido para la implementación del presente proyecto es, en el lado del terminal host Linux, el basado en el modelo cliente-servidor según la programación modular estructurada, con posibilidad de aplicar el paradigma de la programación orientada a objetos.

El modelo empleado para el desarrollo del código de las motas se basa en el modelo de componentes e inyección de código mediante el pseudolenguaje nesC.

En el modelo de comunicación se ha enfatizado en la eficiencia cómo medio de ahorrar ancho de banda, ya que con 4Mhz un sistema de compresión adicional sin hardware específico que implemente algún algoritmo bien conocido, es inviable.

La seguridad y el peligro de atentados terroristas mediante bloqueos a las señales de radio tampoco se ha tenido en cuenta, siendo la red vulnerable a ataques de reventadores que inserten datos falsos que anulen o provoquen un comportamiento peligroso del sistema.

La razón, es, al igual que con la compresión, la escasez de recursos disponibles en las motas. Sin un hardware específico es imposible, en un sistema de sólo unos pocos kilobytes de memoria y a penas 4Mhz, implementar un sistema de encriptación fiable, aunque sea mínimamente.

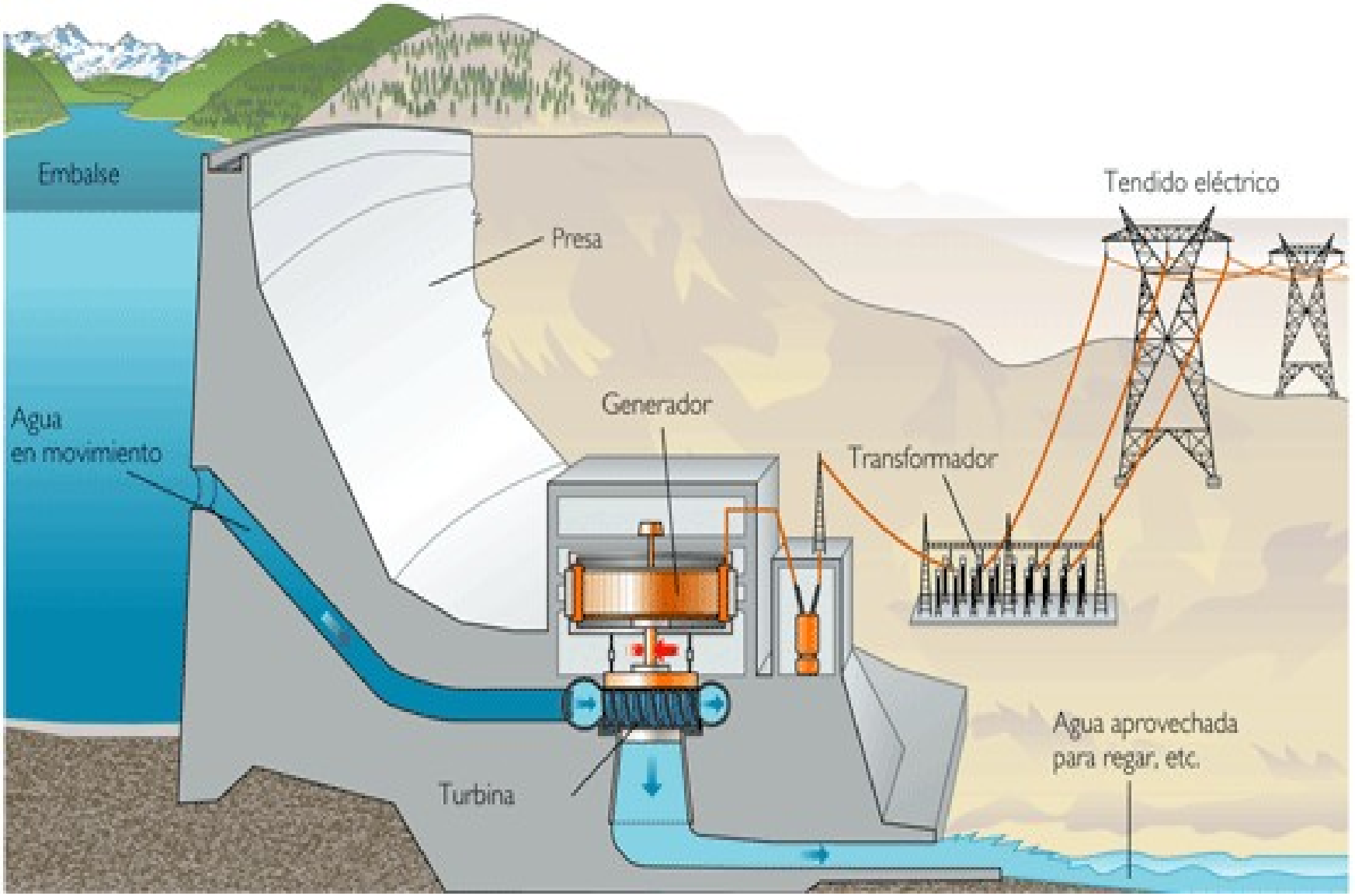
De hecho, el apoyo de un sistema fijo es necesario para poder aislar físicamente el área controlada por las motas, a fin de evitar ataques contra el sistema, y para dotar de potencia suficiente a la hora de gestionar los datos recopilados.

Planificación

Las fases de ejecución de la practica han sido las siguientes:

- Evaluación de medios
- Documentación sobre las posibilidades de aplicación
- Pruebas de rendimiento de los elementos disponibles
- Planteamiento general del hipotético escenario, posible en la vida real
- Análisis de los requisitos y protocolos empleados en otros proyectos similares ya realizados
- Planteamiento de objetivos mínimos
- Planteamiento de objetivos máximos
- Planteamiento de la priorización de objetivos no esenciales pero si deseables
- Planificación de tareas
- Reunión de la documentación técnica sobre el hardware a emplear
- Análisis económico superficial en base a los precios contractuales publicitados en comercios mayoristas y minoritarios
- Plan de trabajo
- Revisión del calendario
- Ejecución del 50% del proyecto
- Ejecución del 50% restante del proyecto
- Recopilación de notas del diario de desarrollo
- Resumen
- Creación de la memoria

Ilustración 3: Esquema genérico de una presa común



Sobre estas líneas, el esquema fundamental que cualquier presa hidroeléctrica debe cumplir.

Existen diversos tipos de presas, las hay de gravedad:

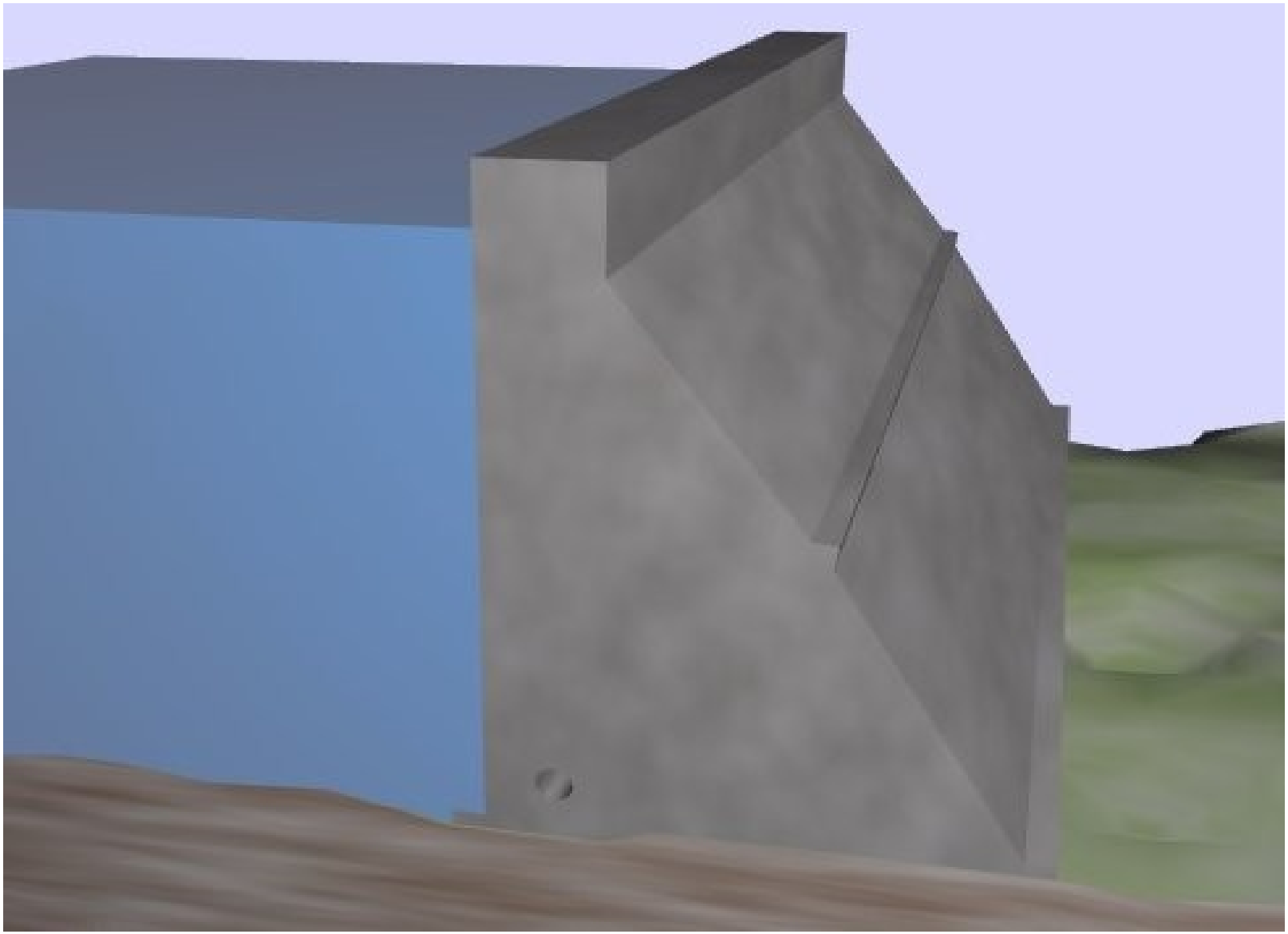
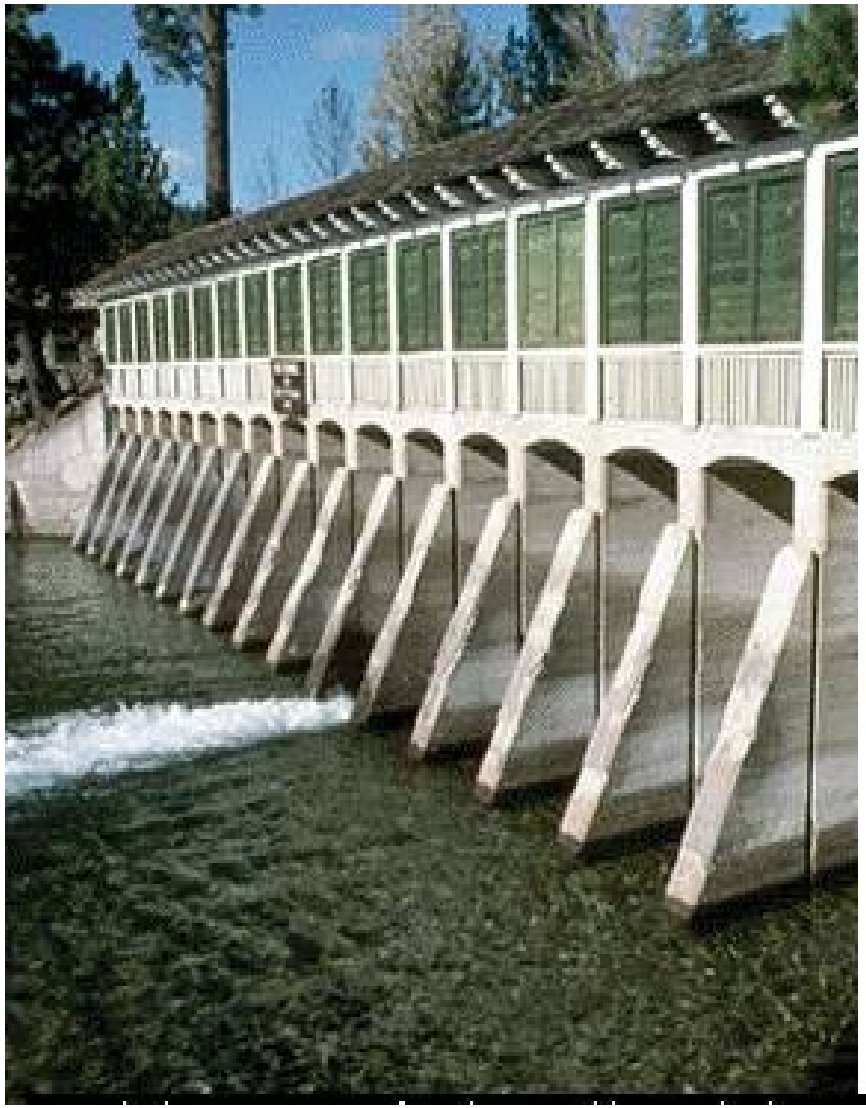
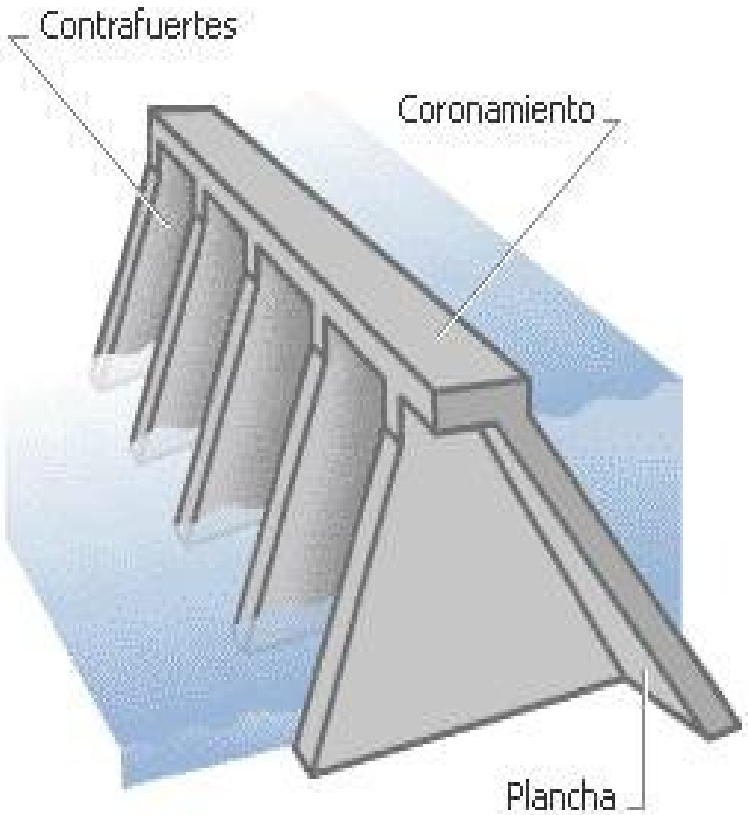


Ilustración 4: Presa de tipo gravedad

de contrafuertes:

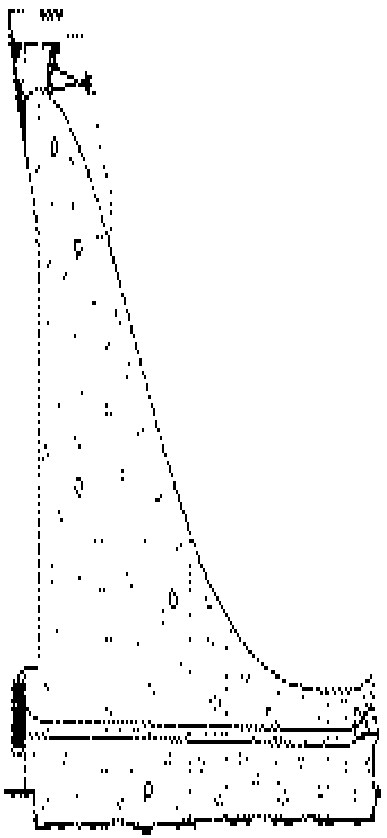




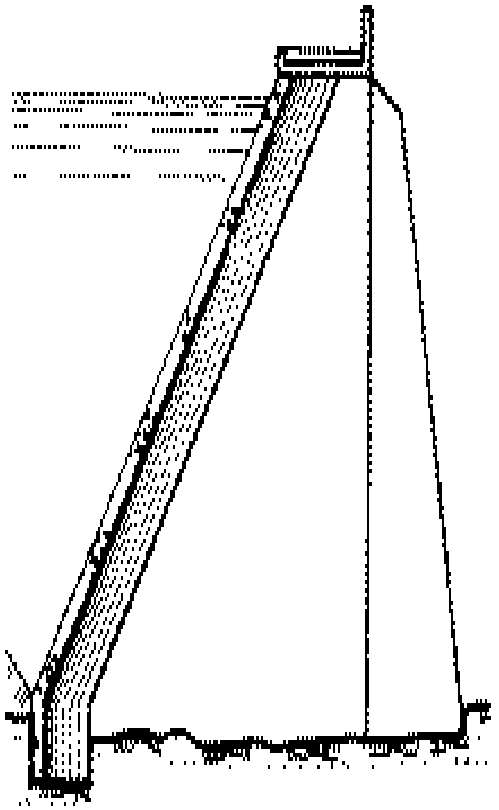
Presa de contrafuertes

Ilustración 5: Presa de contrafuertes

de arco:



Presa de tipo arco-gravedad

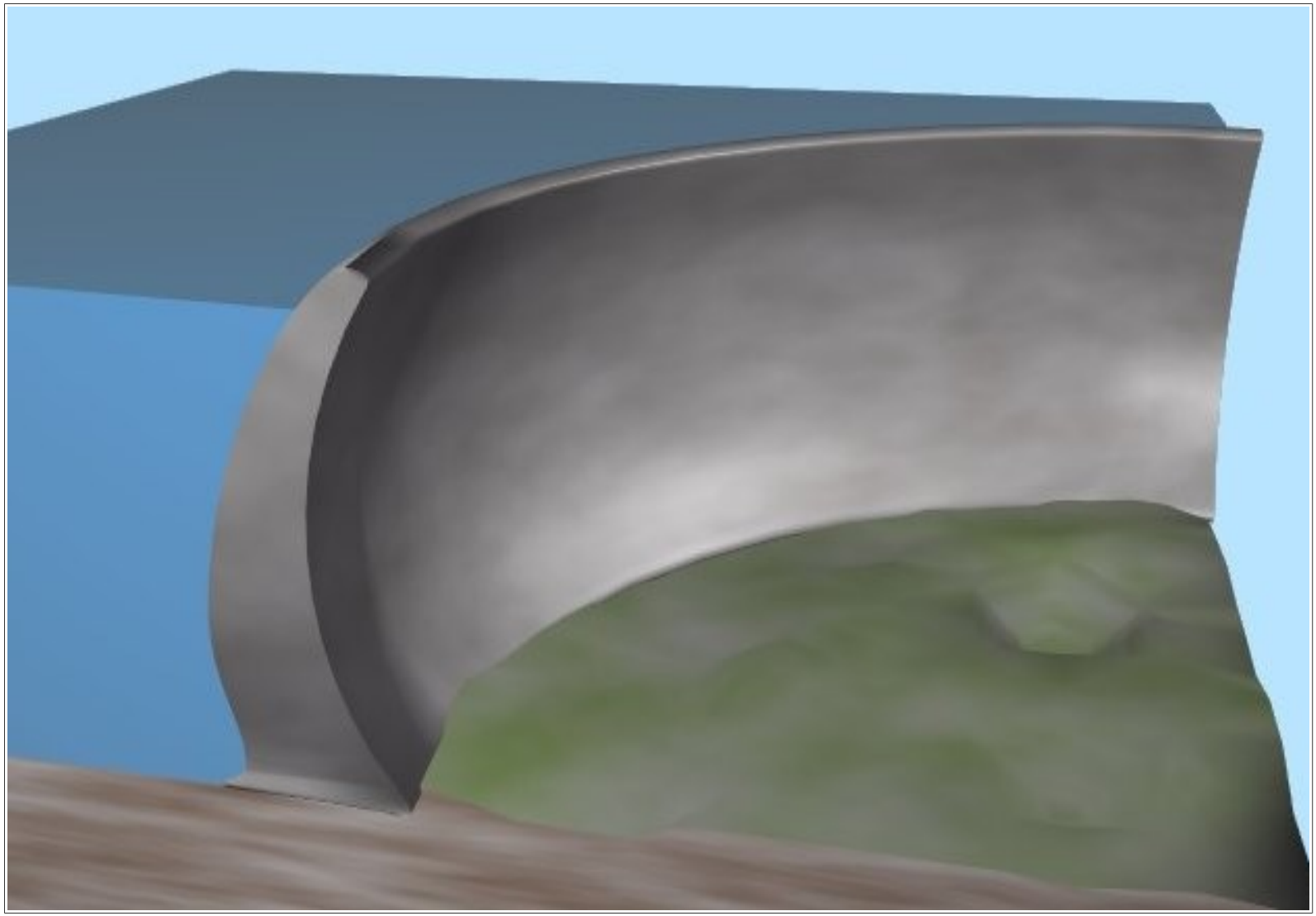


Presa de arcos multiples

Ilustración 6: Presas de tipo arco

de bóvedas o doble curvatura:

Ilustración 7: Presa de bóveda o de doble curvatura



o mixta, como la presa Hoover en los Estados Unidos de Norteamerica:

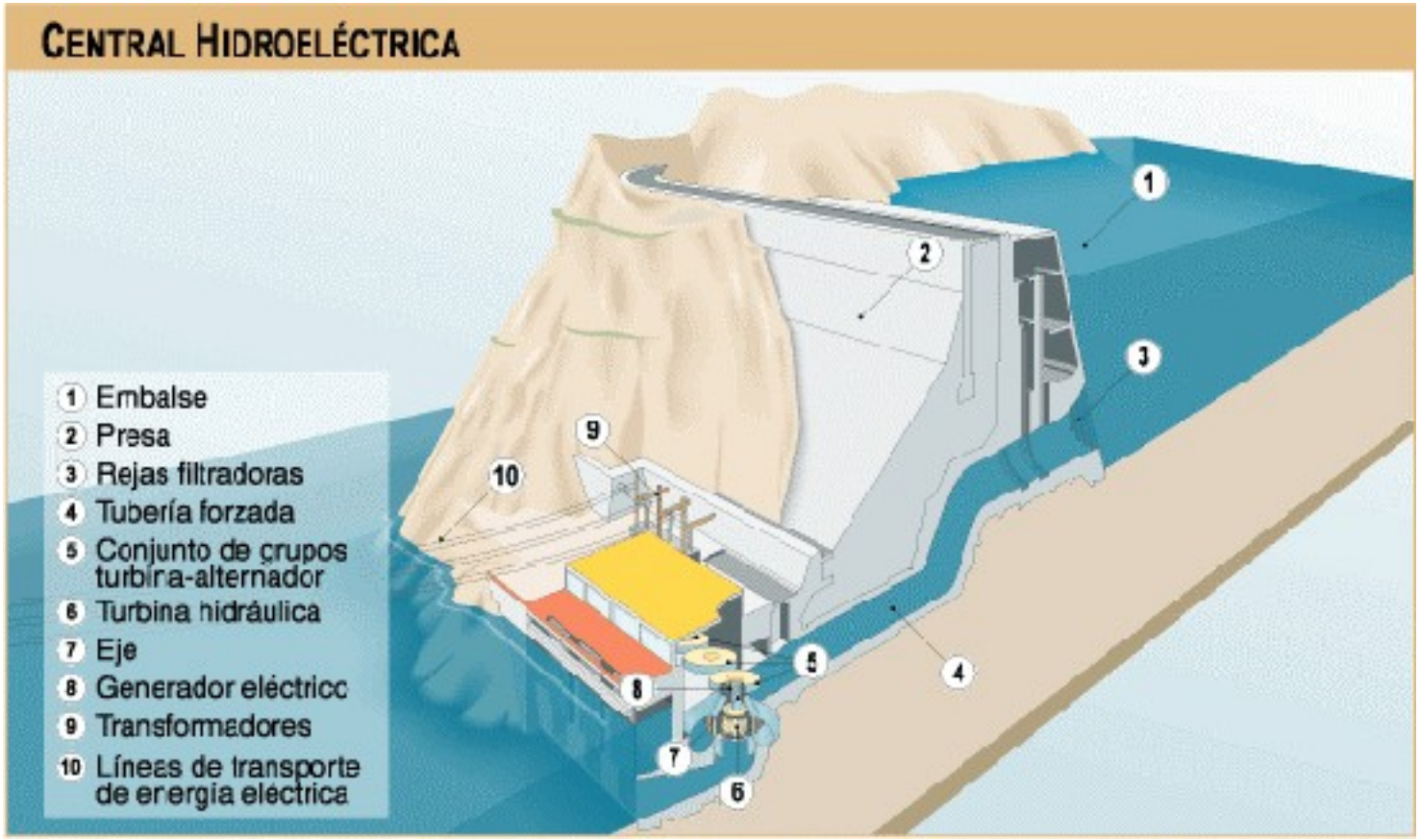
Ilustración 8: Presa Hoover del tipo mixto arco-gravedad

Para evitar que la presa se desborde se utilizan aliviaderos:

Ilustración 9: Aliviadero sin exclusas

Existen aliviaderos pasivos, es decir, sin exclusas por los que el agua discurre libremente en cuanto se alcanza un determinado nivel, o bien con exclusa, que se emplean especialmente en situaciones de un incremento anómalo del caudal, para facilitar un desagüe adicional en caso de un incremento muy excesivo del caudal.

Este es el esquema general de una central hidroeléctrica:



Las motas se situarán las de tipo 1 en la entrada del embalse (1), a fin de medir el caudal de entrada, las de tipo 2 en el propio embalse (1), las de tipo 3 tendrán una sonda en el aliviadero suplementario con exclusa y se situarán en la presa (2).

Las exclusas de los aliviaderos adicionales sólo son necesarios cuando se desea un alto rendimiento en embalses cuyo flujo de entrada puede sufrir grandes fluctuaciónes.

Las motas de tipo 4 se alojarán en la sala de turbinas (5), donde sondearán los parámetros de funcionamiento y condiciones de la sala de turbinas.

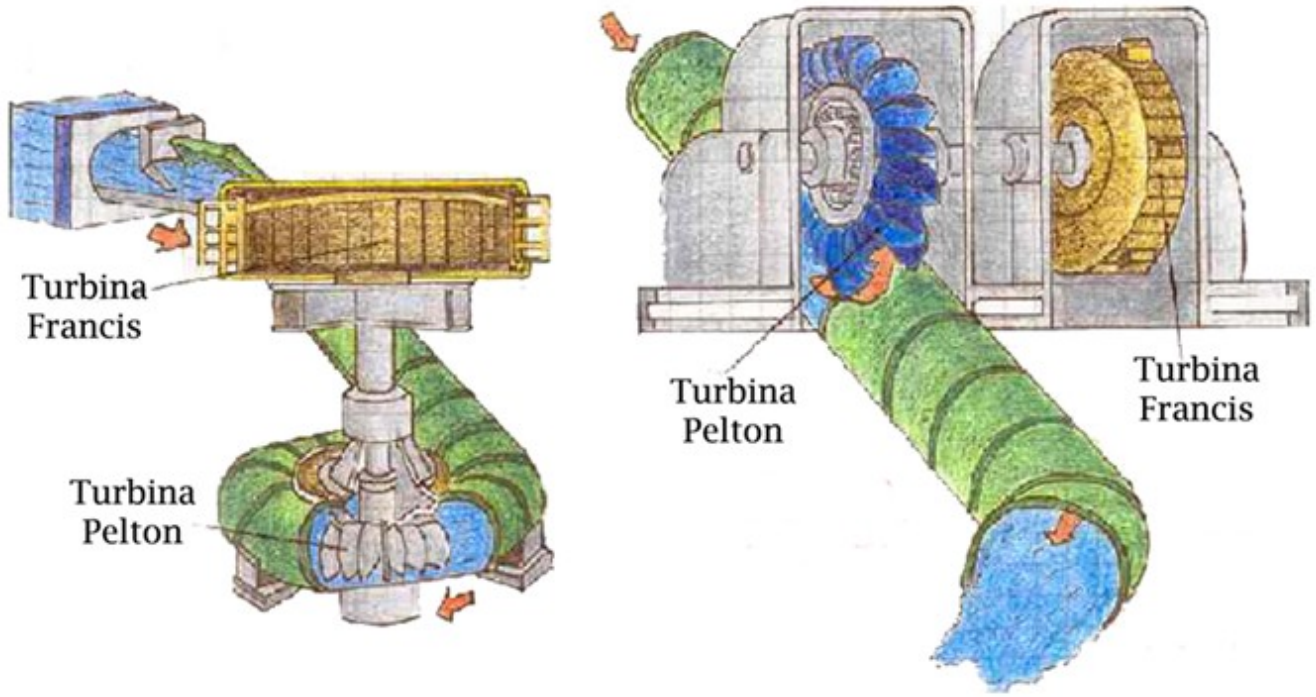
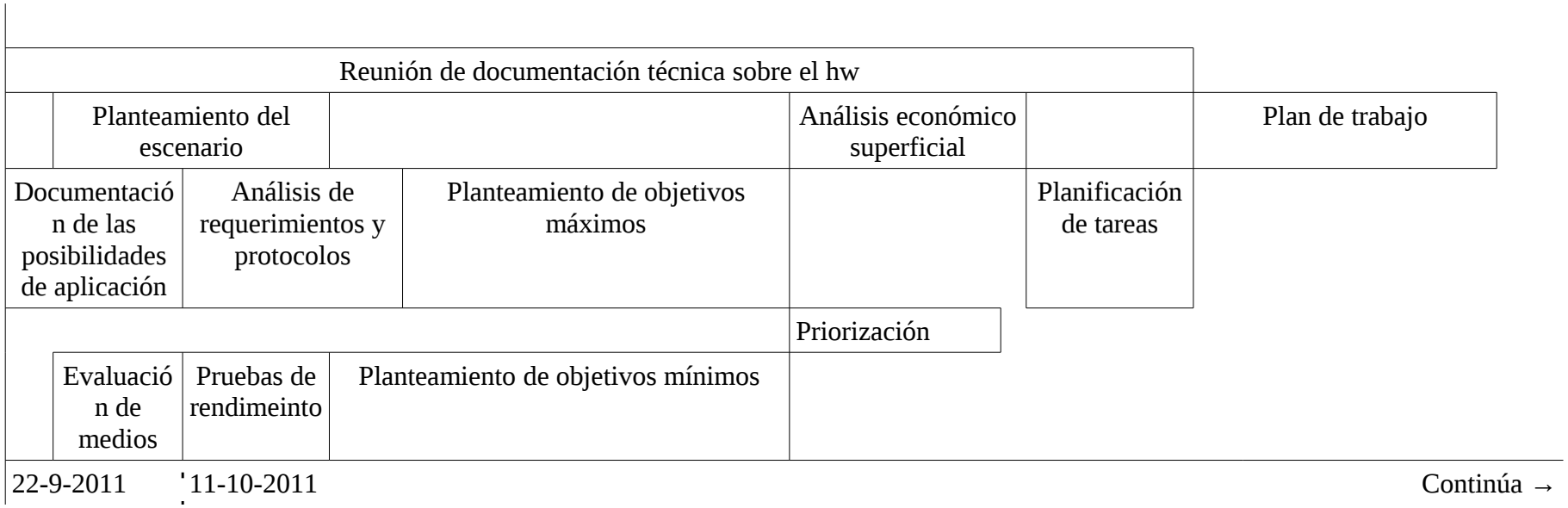
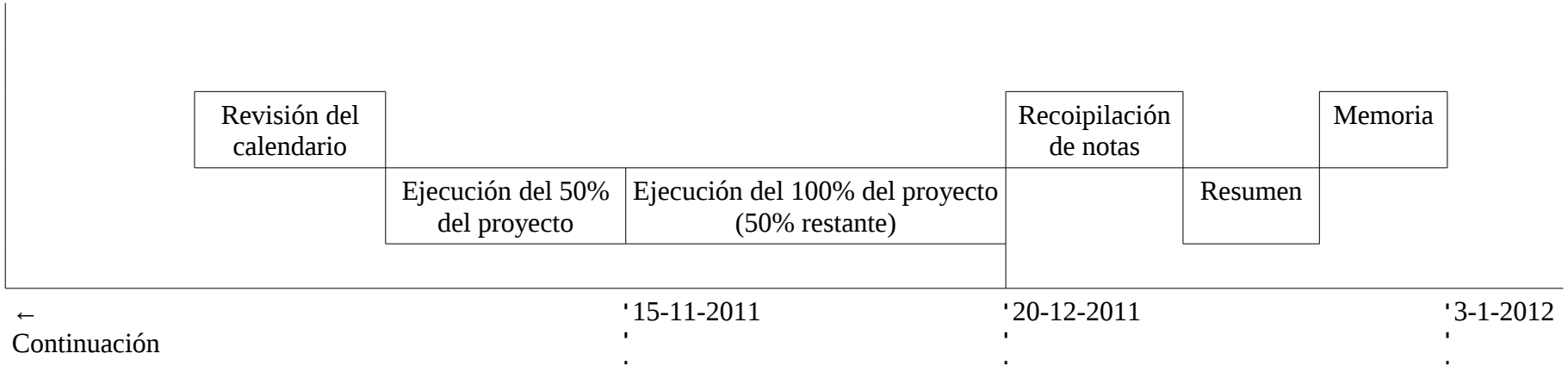


Ilustración 10: Turbinas

Las sondas de las motas se ubican en los ejes. En los ejes se instala un imán en su parte exterior junto al sensor de efecto Hall, el cual leerá una mota de tipo 4.

Diagrama de tiempos





El factor que más a determinado las primeras fases, que son aquellas que preceden a la ejecución del proyecto, ha sido la disponibilidad y el tiempo invertido a la hora de recopilar toda la información a través de la internet.

Una planificación correcta es el preludio de un desarrollo sin agobios ni problemas que lo hagan inviable a corto, medio o largo plazo. Para ello, es imprescindible conocer detalles funcionales a bajo nivel de aquello que vas a programar.

No importa si estas usando clases o componentes, todo ello será transformado en códigos operaciones que son totalmente ajenos a los paradigmas de la informática de gestión. No es necesario conocer el funcionamiento a nivel eléctrico o electrónico, aunque es aconsejable, mas si es esencial conocer su respuesta lógica, que es lo que va a determinar la naturaleza de nuestro programa.

Así, al plantear mi programa, he concebido motas montadas con distintos sensores de los actuales, ya que cosas como el nivel del agua se miden con otro tipo de sensor cuya respuesta puede ser idéntica a la de un sensor ya presente en las motas suministradas.

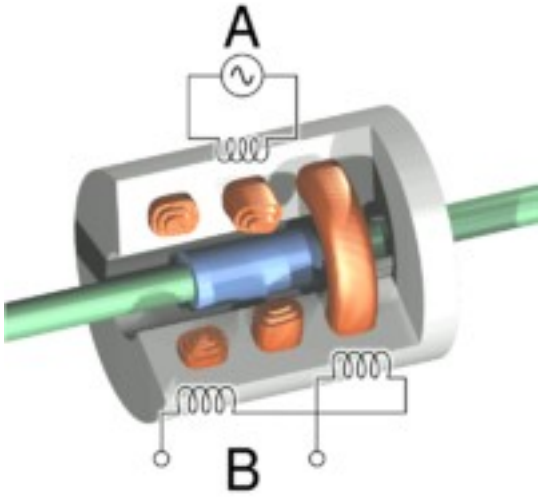


Ilustración 11: Sensor de proximidad por corrientes Foucault

Por ejemplo, en las motas tipo 2, en lugar de un forosensor, se puede emplear un sensor piezoeléctrico con un resorte de presión o un montaje sensible a la variación presión atmosférica.

Ilustración 12: Sensores piezoeléctricos

Lógicamente, al planificar las tareas, el hecho de sólo disponer 2 motas condicionó el itinerario de desarrollo, siendo lo primero en ser implementado una comunicación simple entre dos motas vía rádio, después, una lectura simple a través de los distintos sensores, y finalmente, salida vía COM. Una vez exploradas las distintas formas de programación, se procedió a hacer un ensayo de rendimientos ejecutando programas de pruebas para observar el comportamiento de las motas.

De dichas pruebas, se observo una limitación en el ancho de banda manejado por las motas, hecho que me condujo a diseñar el sistema como si fuera de baja latencia, en lugar de como si fuera en tiempo real. Las pruebas de recepción realizadas sobre el puerto de serie revelaron un sobrecarga en la transmisión que hacía casi imposible que un MIPS o un ARMv9 a menos de 600Mhz o con más de 50MB de RAM, pudiese garantizar un correcto manejo del buffer FIFO. Pruebas hechas con libusb para comunicar directamente con la mota, revelan la imposibilidad de no provocar un cuello de botella si se prescinde de un buffer FIFO, lo cual hace absurdo la implementación de un módulo integrado en el Kernel.

Es necesario implementar bajo nivel un servicio de gestión del buffer FIFO que permita que un hardware similar a cualquier teléfono móvil sea capaz de gestionar dicho buffer.

Las pruebas sobre un ARMv9 a 150Mhz con un consumo monitorizado de 5MB de datos de usuario lo ha revelado como capaz de ejecutar el servicio de gestión del puerto COM sin necesidad de que el buffer FIFO crezca exageradamente y sin perdidas de datos en condiciones extremas, así como de gestionar el servicio de consulta web con una carga moderada.

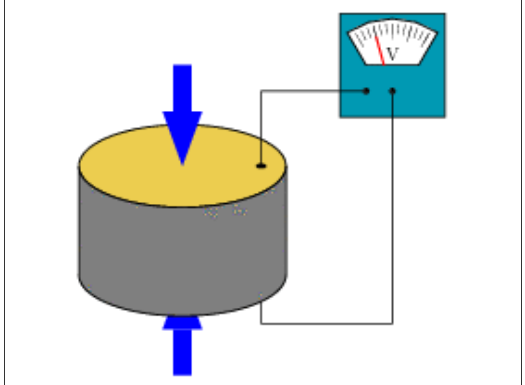


Ilustración 14: Funcionamiento de un sensor piezoeléctrico

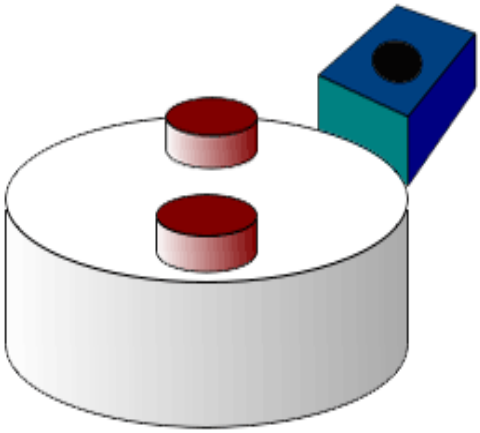


Ilustración 13: Funcionamiento de un sensor de efecto Hall

Recursos empleados

El primer y más valioso recurso, son las fuentes de información. Y ninguna tan amplia y rápida como la internet. El segundo recurso necesario son las herramientas de desarrollo, disponibles todas ellas en los repositorios de Linux y en los sitios web de universidades.

Y por último, una vez hecho el programa desde el conocimiento, es necesario herramientas que permitan reprogramar la mota. Estas herramientas pueden encontrarse en SourceForge o en la web del fabricante del microcontrolador.

Este es el resumen de todos los recursos utilizados:

Software:

- Sistema OpenSUSE 12.1 (Linux RPM)
- Compilador GNU C para AVR
- Inyector de código y dialecto nesC
- Código fuente del tinyOS (válido para microcontroladores de TI y Atmel)
- Una herramienta de subida de código máquina a la mota. (por ejemplo, meshprog)
- Las GNU binutils para automatizar el proceso de construcción

Hardware:

- 2 motas basadas en un microcontrolador Atmel 1212 a 4Mhz
- 1 Notebook basado en un Pentium 4 a 2.8Ghz con una disro Linux OpenSUSE 12.1
- 1 videoconsola Wiz
- 1 netbook eCafé de Hercules
- 1 PC basado en Pentium i5 650 a 3.2Ghz
- Un péndulo imantado y una luz led

La compilación y el desarrollo se han llevado a cabo en el PC y en el laptop, las pruebas de rendimiento se han llevado a cabo usando cómo servidor la consola Wiz, a fin de optimizar el máximo posible, ya que en un PC los recursos van holgadísimos y las ralentizaciónes por culpa de una mal coding pasan inadvertidas.

Ilustración 15: Mota suministrada por la UOC

Las pruebas funcionales se hicieron usando un PC como servidor contra 300 clientes, WIZ (1), laptop (199) y netbook (100). Cómo era de esperar, el PC demostró su capacidad para procesar miles de peticiones por segundo, ya que el tiempo de proceso es despreciable y sólo en ancho de banda de la red supone una limitación.

La misma prueba con el notebook sirviendo a 300 clientes y el tiempo de procesado fue despreciable igualmente.

La misma prueba con el eCafé dio como resultado que el hardware presente en dicha máquina es mas que suficiente como para manejar todas las peticiones recibidas.

La Wiz es una máquina sin un navegador web portado, por ello, cree un programa específico que pedía los datos como si fuera un cliente. La Wiz a 533Mhz, su velocidad nativa, sólo fue capaz de procesar bien 100 peticiones por segundo, en lugar de las 300 por segundo de las que eran capaces el i5 el Pentium IV y el ARMv8 a 800Mhz.

La Wiz como servidor logró procesar hasta 200 peticiones por segundo de manera fluida al subir la frecuencia a 850Mhz, durando unas 4 horas las baterías. El eCafé logro unas 16 horas de autonomía.

Las implemetaciones realizadas admiten incluso mas optimización, pero dado que las lecturas de las motas no van a ser continuas, si no periódicas, un mayor desempeño no tiene sentido. De hecho, según mis cálculos, una consola Dingoo con un MIPS a 400Mhz y 32MB de RAM (total) sería lo suficientemente potente.

Y dada la naturaleza de la instalación, con suerte se superaran las 5 lecturas simultáneas por segundo, por lo que los costes de hardware para la implantación son nímios.

Ilustración 16: El eCafé del integrador Hercules

Productos obtenidos

- Un componente watchdog reultilizable con tinyOS y nesC
- 1 mota capaz de expresar, mediante el sensor de efecto Hall, una velocidad angular
- 1 mota capaz de medir del agua como una tensión procedente del sensor de luz
- 1 mota capaz de medir el grado de apertura como una tensión procedente del sensor de luz, y detectar una apertura total gracias al sensor de efecto Hall
- 1 mota capaz de medir la temperatura, la energía generada como una tensión leída del sensor de luz, y las revoluciones de la turbina mediante el sensor de efecto Hall.
- 1 servicio de lectura puerto COM capaz de invocar rutinas de usuario y suministrar los datos según la sintaxis Java Script
- 1 mini servidor web capaz de ejecutarse en máquinas con 10 MB de RAM libres o incluso menos

Estado del arte

Actualmente, en el mercado de los microcontroladores las opciones más extendidas son Texas Instruments, mas cara y mejor, Microchip, con series muy diversas de precio y calidad, y Atmel, con una excelente relación calidad-precio.

Los microcontroladores pueden encontrarse en electrodomésticos tales como los hornos, los microondas, los lavavajillas, equipos de aire acondicionado, climatizadores, domótica... así como también en los automóviles, en los sistemas de frenado antibloqueo, en automatizaciones industriales, en relojes digitales de última generación, en el hardware de los ordenadores personales, cargadores de baterias, videocámaras, gadgets...

Según su construcción y características existen 4 gamas de microcontoladores:

- Mini: Encapsulado pequeño, de 8 pines. Su régimen de alimentación oscila entre los 2,2V y los 5.5V. La velocidad de su unidad de proceso interna es de 4Mhz, y su consumo es de 2mA a 5V, 10mW, cuando funcionan a 4Mhz. Su formato de instrucciones oscila entre los 12 o 14 bits y su número de instrucciones suele ser de entre un mínimo de 33 a un máximo de 35. Bajo consumo y un ratio muy bajo de instrucciones ejecutadas por segundo. Sólo 2 pines del encapsulado se destinan a la alimentación, los otros, hasta 6, se pueden conectar a periféricos como puertos E/S. Disponen de un oscilador interno R-C. Algunos de estos modelos incluyen conversores de analógico a digital integrados.

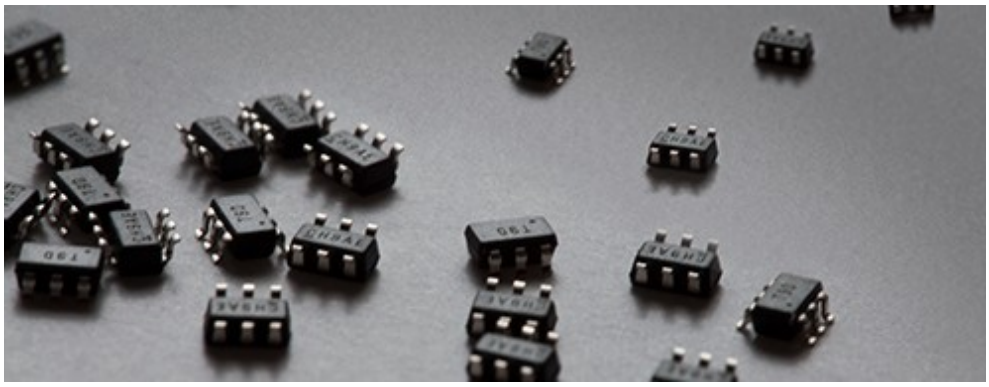


Ilustración 17: Microcontoladores AVR de gama mini

- Baja: Representan la mejor opción relación calidad-precio. Poseen un encapsulamiento que va desde 18 a 28 pines, y un régimen de alimentación de 2,5V y menos de 2mA a 4Mhz, lo que los convierte en la mejor opción para un sistema alimentado con pilas. Su repertorio de instrucciones es de 12 bits y consta de 33 diferentes. Estos microcontroladores destacan tambien sobre los de gama mini por poseer mecanismos anticuelgue, como el perro guardián (watchdog), sistema *power on reset*, código de protección....

Ilustración 18: Microcontrolador Atmel de gama baja

- Media: Esta es la gama más extendida y variada de los microcontroladores. Su patillaje oscila desde los 18 hasta 68 pines. Un microcontolador de esta gama puede integrar abundantes y diversos periféricos. Poseen comparadores de magnitudes analógicas, convertidores de analógico a digital, puertos de serie y diversos temporizadores. Sus 35 instrucciones de 14 bits son compatibles con los de la gama baja. Poseen todos los recursos necesarios para ejecutar cualquier programa de 8bits y disponen de una pila de 8 niveles para el almacenaje de subrutinas y servicio de interrupciones. Están preparados para ser alimentados por cualquier tipo de baterías, tanto alcalinas como recargables basadas en Ni, todas, o Li-ion.

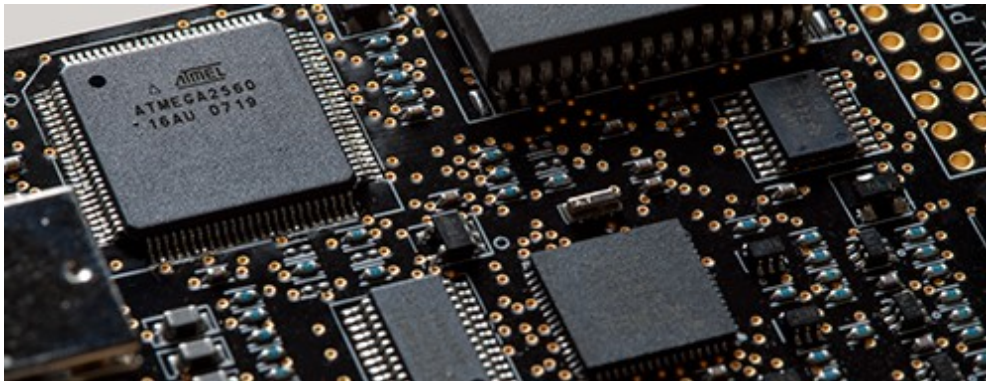


Ilustración 19: Microcontrolador de gama media

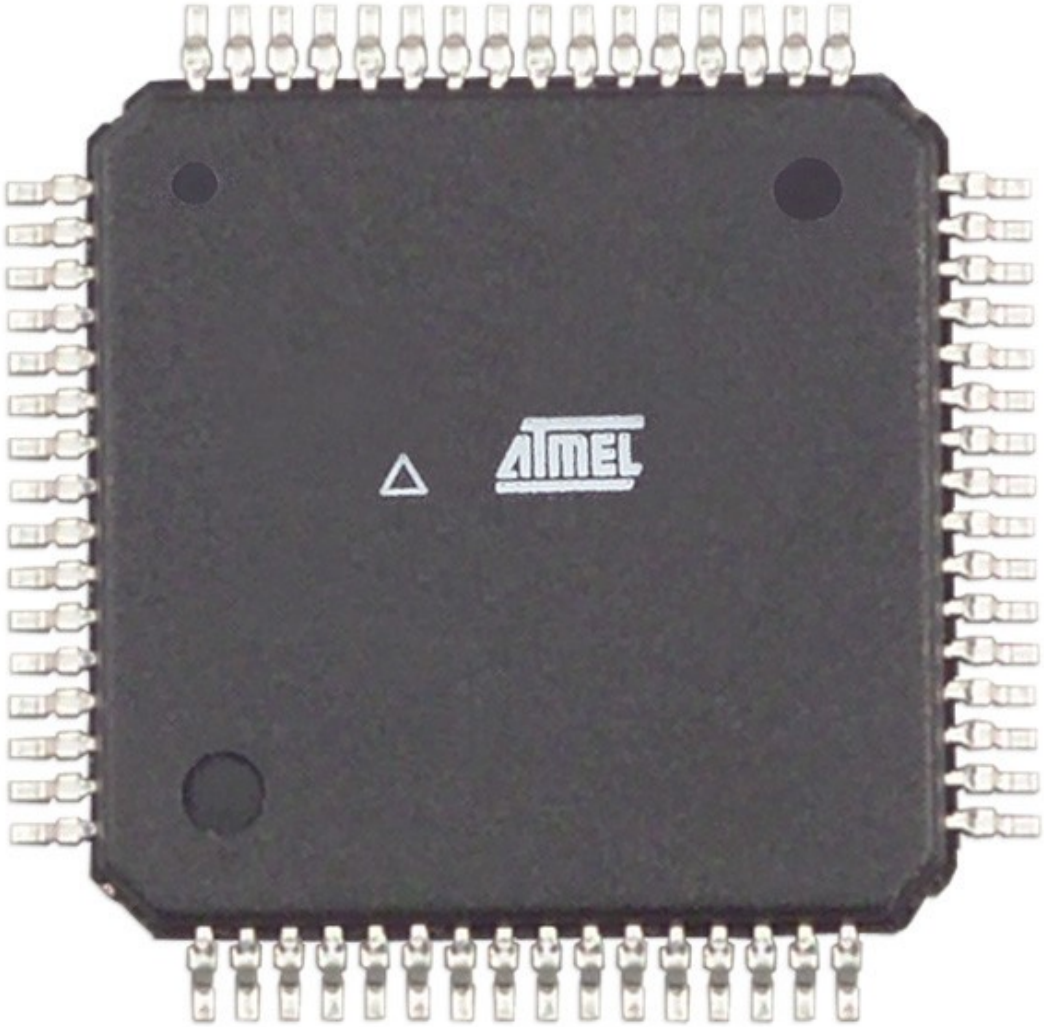


Ilustración 20: Detalle de un microcontrolador de gama media

- Alta: Los microcontroladores clasificados en eesta gama, disponen de un potente juego de 58 instrucciones de 16 bits y sistema de gestión vectorizado de interrupciones, así como controladores de periféricos, posibilidad de adosarles puertos de comunicación en serie y en paralelo, un multiplicador de frecuencia, hasta 8K de palabras en memoria y 454 bytes en la memoria de datos. Su arquitectura es totalmente abierta, admitiendo una gran variedad de perifericos adosables, gracias a que algunos pines comunican directamente con los buses de direcciones, datos y control del microcontolador, razón por la cual su numero de pines en el encapsulado va desde los 40 a los 44. Este filosofía de construcción es más bien cercana a un microprocesador, razón por la cual los microcontroladores de esta gama no gozan de un uso muy extendido.

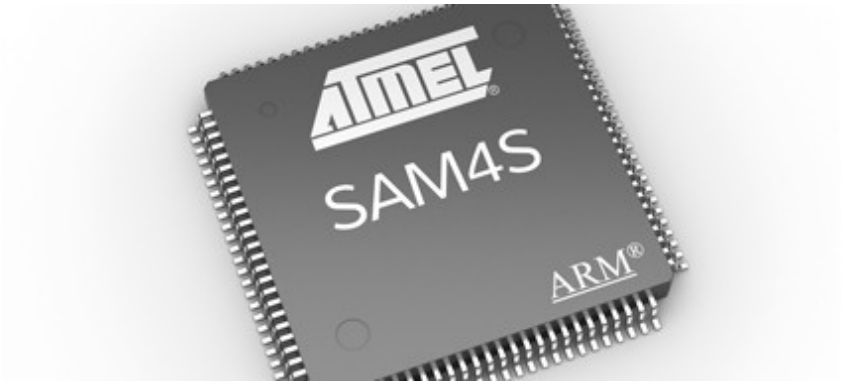


Ilustración 21: Microcontolador con un Cortex-M4

Mucho ha pasado desde que Intel produjese el primer microprocesador 4004, hemos visto como el PIC16F84 de Microchip, de gama media, se convertía en el microcontrolador más utilizado, para quedarse obsoleto y dejar paso a nuevas generaciones de microcontroladores de nuevos y antiguos fabricantes.

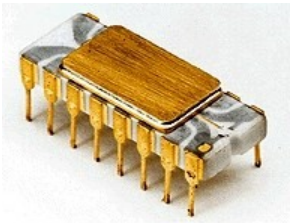


Ilustración 22: El Intel 4004, el primer microprocesador

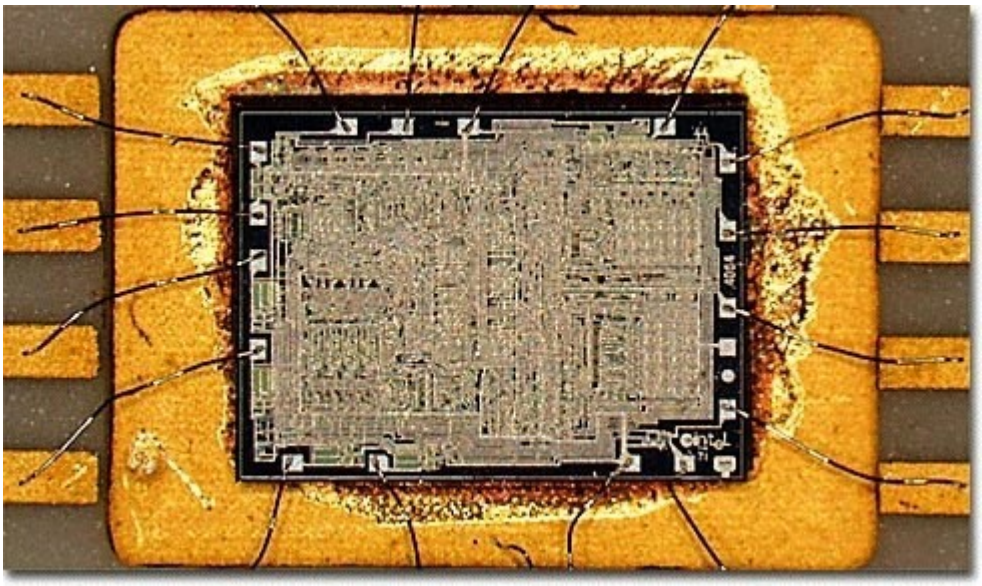


Ilustración 23: Detalle del interior del Intel 4004, el abuelo de los microprocesadores

La programación de microcontroladores es un hobby que muchos tenemos y que nos hace pasar muy gratos momentos ya sea reparando averías en electrodomésticos, o creando nuestros propios sistemas para iluminación, domótica, etc...

Existen muchas comunidades en internet dedicadas a la programación y montajes con microcontroladores dónde se encuentra abundante información y gente dispuesta a resolver dudas.

Actualmente, los microcontoladores, junto con los SOC, están mas presentes que nunca, siendo el cerebro de muchos aparatos de uso diario y ofreciendo posibilidades hasta ahora insospechadas, como la posibilidad de recargar las pilas alcalinas, automatización de procesos industriales, etc...

Y la industria que más y mejores microcontroladores demanda es la automotriz, pues los microcontroladores montados en automóviles son los que tienen que soportar las condiciones más extremas de vibración y temperatura y seguir siendo 100% fiables.

Fujitsu

Family	Core [Bit]	&RAM [Bytes]	Flash/ROM [KBytes]	Package [Pins]
F²MC 16LX	16	1K...30K	32...512	48...120
F²MC 16FX	16	12K...24K	288...576	64...120
FR-Series	32	2K...96K	2...2112	64...256

Microchip

Family	Core [Bit]	&RAM [Bytes]	Flash/ROM [KBytes]	Package [Pins]
PIC16	8	25...368	0,75...14	14...44
PIC18	8	256...4K	4...256	18...100

Renesas

Family	Core [Bit]	&RAM [Bytes]	Flash/ROM [KBytes]	Package [Pins]
R8C (R5FXX)	8	512...3K	4...128	20...80
M16C (M306XX)	16	1K...31K	0...512	32...145
M32C/R32C	32	8K...48K	0...1024	100...144

ST

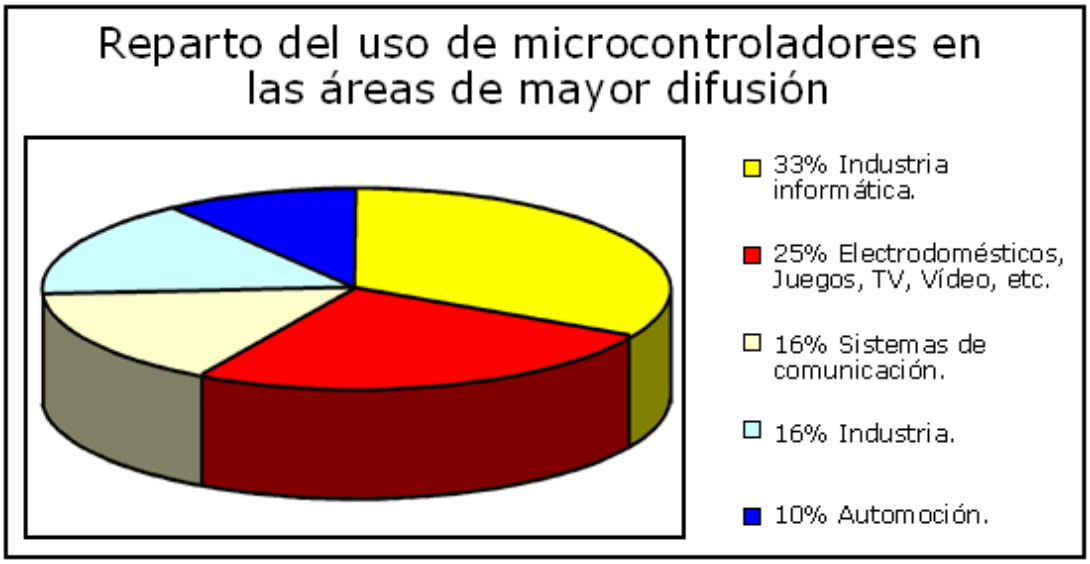
Family	Core [Bit]	&RAM [Bytes]	Flash/ROM [KBytes]	Package [Pins]
ST7	8	128...2K	1...60	8...80
ST10	16	1K...68K	128...832	100...144
STR7	32	16K	64...256	64...144
STR9	32	64K...96K	256...512	80...144
STM32	32	6...20	32...128	48...100

Ilustración 24: Cuadro de produccion y cartacteísticas generales de los principales fabricantes

Estudio de mercado

El control de presas y otros sistemas fijos mediante microcontroladores es ya una práctica habitual en todos los ámbitos en los paises desarrollados, y cada vez lo va siendo más en los paises emergentes.

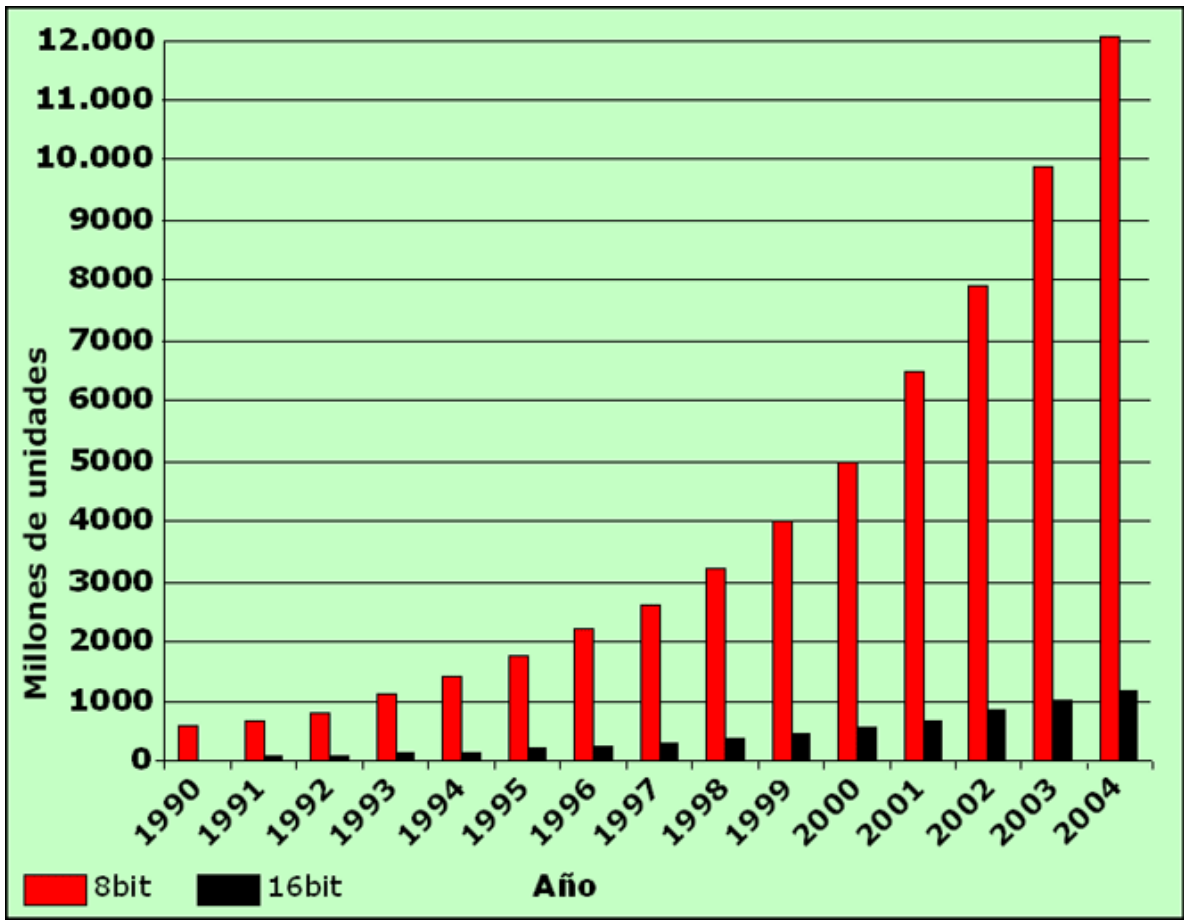
Sólo hay que ver las cifras de ventas de microcontroladores para darse cuenta de que es un mercado en plena expansión y que, a día de hoy, aún no a alcanzado su apoteosis.



El proyecto de un sistema de control para presas es específico de la presa a la cual va dirigido, es decir, se hace siempre sobre un proyecto diseñado por un arquitecto o siguiendo las indicaciones de este. No obstante, es posible encontrar cosas muy comunes, generalidades, que forman parte de cualquier presa que se precie.

Mi objetivo es crear un marco de trabajo, un esqueleto, que posibilite la implementación lo más eficientemente posible, de un sistema de control y monitorización de una instalación de contención de aguas sea cual sea su filosofía de construcción, de manera que el sistema pueda adaptarse al diseño del arquitecto, porque de esta manera los costes se reducirían: los arquitectos no tendrían la necesidad de rehacer sus cálculos o empezar el proyecto desde cero, los sistemas podrían mejorarse gracias a la herencia recivida de la experiencia de uso, y los costes se amortizarían más rápidamente al ser posible una normalización cumplible.

En el mercado no existe a nivel de norma, tanto de *iure* como de *facto*, un marco de trabajo de código abierto que cubra objetivos muy específicos, si bien es cierto que existe tinyOS, este es sólo, de momento, para arquitecturas AVR, los Atmel, y MPS430, los Texas Instruments, y cubre sólo lo que es la programción estricta de los recusos del microcontrolador.



Si existe, de hecho, el que cada industria, como practica habitual, tenga plantillas y marcos de trabajo para su uso particular y de fuente cerrada.

Los costes de hardware iniciales no superarían los 6000€, ya que el servidor puede ser, sin problemas, una máquina cuyo precio ronde los 150€ y no supere los 400€, más un mínimo de cuatro motas.

Obviamente, los costes variaran dependiendo del tamaño de la presa, si tiene generador hidroeléctrico o no, etc...

Mis cálculos se basan en una presa equipada con turbinas Pelton a 75rpm, generando 135MW cada una, lo que hace un total de 1890MW con 14 turbinas con control independiente, al estilo de la presa de Salto Grande, lo que harían:

- 14 motas tipo 4
- Un único terminal de control con 14 motas adosadas (mediante un hub USB)
- 2 motas (una de respaldo) tipo 3
- 2 motas (una de respaldo) tipo 2
- 2 motas (una de respaldo) tipo 1

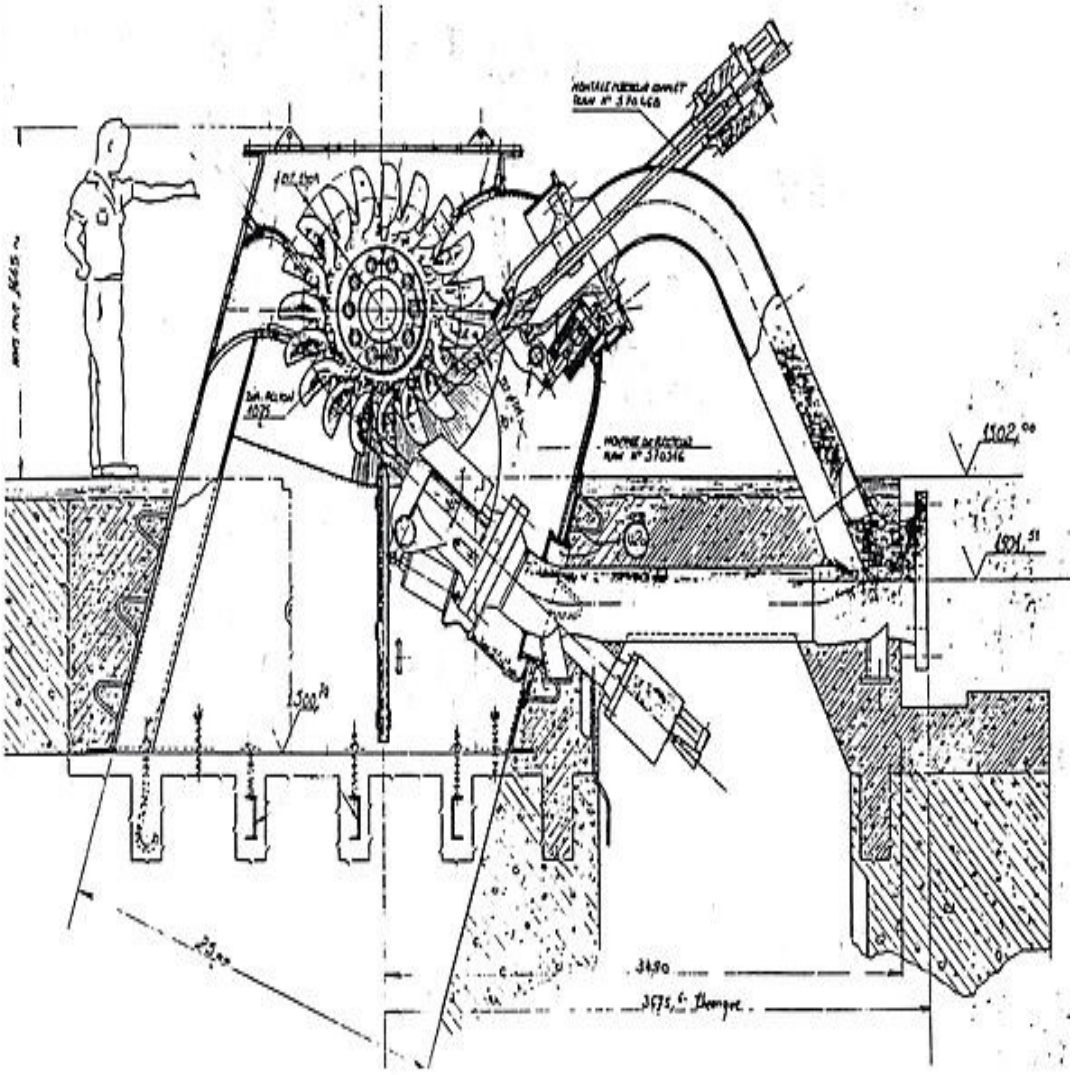


Ilustración 25: Esquema de una turbina Pelton

Existen terminales, vendidos al por menor, por 150€, todo incluido, cuyas especificaciones son sumamente satisfactorias, por ello, nos quedaremos siempre con la opción más económica.

Ilustración 26: El eCafé de Hercules

Cada mota puede realizarse a un coste máximo de 100€ cada unidad, así pues, teniendo en cuenta esto, una primera aproximación económica de costes serían unos 2150€.

Los cálculos los he realizado asumiendo que un microcontrolador apropiado para estas tareas tiene un coste de unos 15€, lo cual es elevadísimo ya que un microcontrolador PIC16f628a de la Wii vale apenas 5€, unos costes de componentes pasivos de 15€ y un valor de los únicos componentes activos del montaje de 10€, lo cual es carísimo, pues el integrado 222 no vale ni 1€ y el SILABS CP2102 para montaje lo tienes por unos 2€ como mucho. El resto, 60€, irían para el integrador encaso de externalizar el montaje. También un coste base infladísimo, pero dado el enorme abanico de pequeñas y grandes empresas dedicadas al negocio de la integración, todo dependería del volumen de negocio del momento, para así tomar partido por una oferta o por otra.

A esto habría que añadirle los costes de programación. Por mi experiencia, lo que las empresas suelen hacer es seguir la técnica de cargar un 36% o un 37% sobre el coste base del hardware y luego inflar costes en las interfaces web. Así, la programación de las motas y el sistema de lectura de datos y miniservidor web sería un 37% de 2150€, unos 800€, ya que mas de 15 días de desarrollo no lleva, a parte de los desplazamientos, que se suelen cobrar por horas, al lugar de instalación para depurar y afinar algunos aspectos que no hallan podido ser probados en el estudio.

La tarifa por la instalación fija de los elementos que conforman el sistema de control suele ser una cantidad fija por tipo de elemento si las instalaciones no ofrecen excesiva dificultad y más o menos se sabe de antemano cuanto tiempo habrá que invertir en la instalación de cada elemento.

En nuestro caso, los soportes para las motas forman parte del edificio, así que el coste por mota sería de unos 5€ por elemento, 100€ en total, ya que no tomaría más de una hora dejarlas perfectamente instaladas.

La configuración e instalación del servidor puede hacerse por 5€, ya que sólo consistiría en poner una máquina sobre una mesa y conectarle las motas a sus puertos.

Esto elevaría el coste inicial a unos 3055€ en total por la instalación y puesta en marcha de todo el sistema.

El mayor volumen de negocio está en las interfaces web y la programación JavaScript, así como en el mantenimiento e inspección de las motas. Por una maquetación web pueden obtenerse unos beneficios de 400€ a 500€ por página, si el momento económico es propicio, ya que esto es, en sí, un desarrollo a parte.

Este proyecto se dirige a compañías productoras del sector energético cuya facturación sea de unos 18000 millones de Euros y tengan en su mix energético un 20% o más de hidráulica para satisfacer la demanda energética.

Para competir en el mercado de instalaciones automatizadas tendría que ser a través de subcontrata con una empresa grande o a través de empresas de outsourcing, ya que el panorama económico actual no deja excesivo margen de maniobra.

Las cifras de E.ON y Endesa

Datos de 2005 — En millones de euros

E.ON[1]

Ventas	56.399
EBITDA ajustado	10.272
Cashflow operativo	6.601
Ventas de electricidad (en TWh)	404,3
Ventas de gas (en TWh)	924,4
Empleados	79.947

Endesa[2]

Ingresos	18.229
EBITDA	6.020
Cashflow operativo	3.362
Ventas de electricidad (en TWh*)	203,3
Ventas de gas (en TWh*)	21,1
Empleados	27.204

[1] Cifras preliminares y sin auditar
[2] Resultados 2005 (IFRS)
* TWh: teravatios hora

Fuente : Endesa; E.ON — elmundo.es

Ilustración 27: Cifras de beneficios de EON y Endesa

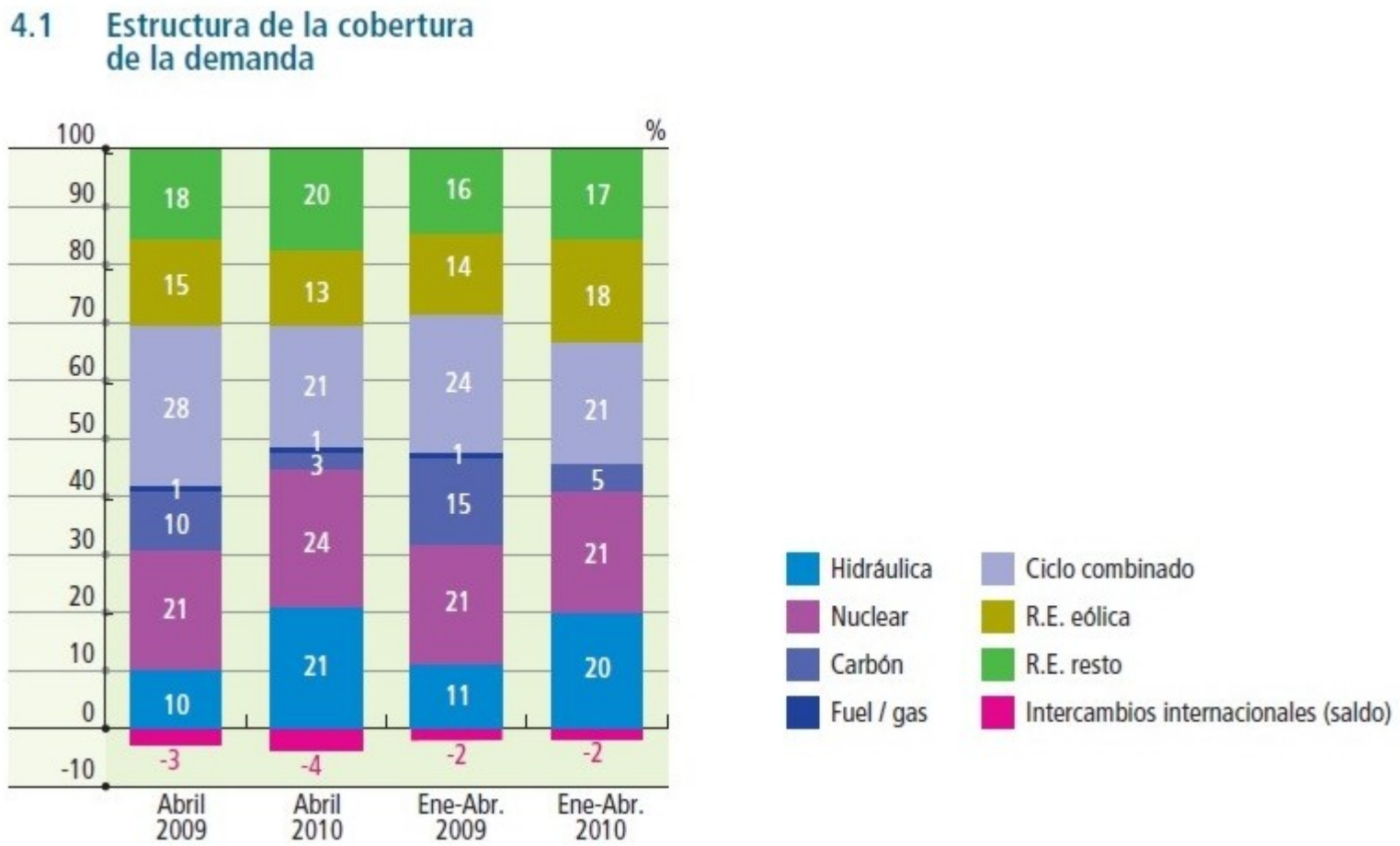


Ilustración 28: Cobertura de la demanda enegética

Descripción funcional

Se trata de un sistema de monitorización activa del caudal entrante, nivel de llenado, grado de apertura de exclusas, caudal saliente y sala de turbinas de bajo coste, alta fiabilidad, y alta disponibilidad, de fácil mantenimiento totalmente automatizable y operable tanto en local, cómo en remoto.

La implantación del software de gestión de los componentes del sistema y del sistema en sí mismo, es otro proyecto a parte y por ello, no lo incluyo en el presente trabajo.

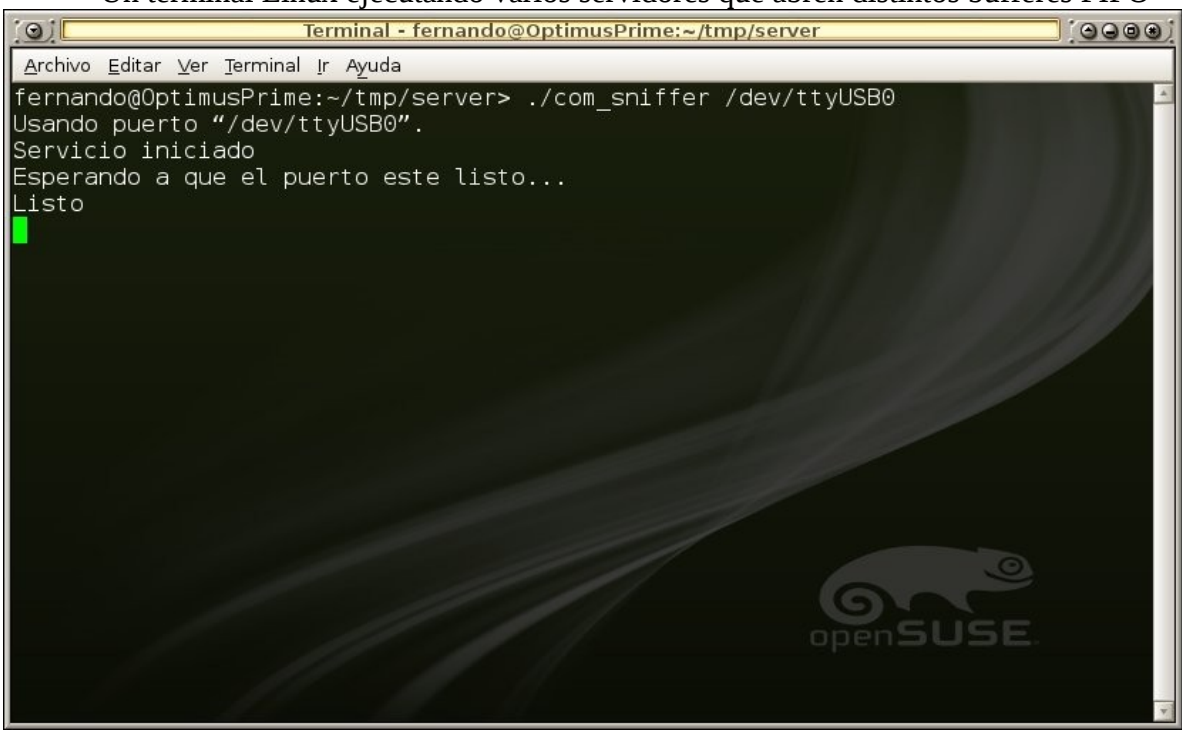
Sistema total

De acuerdo con esta situación:

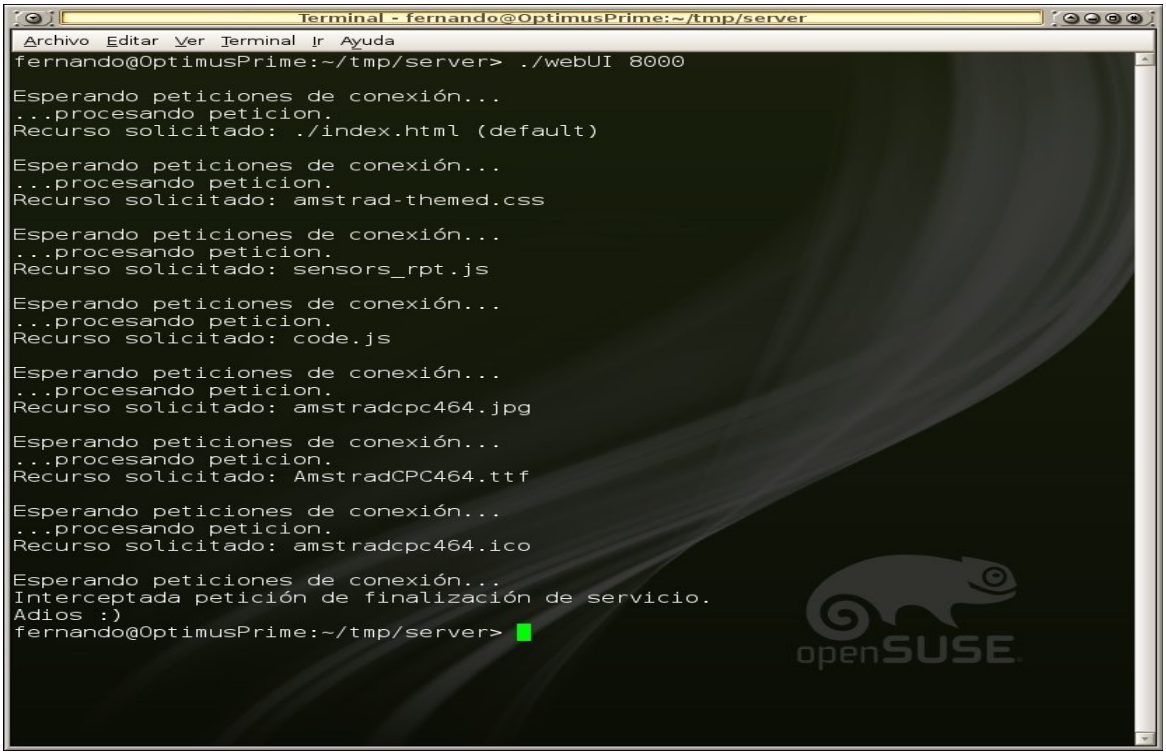
- El sistema funcionará como una red de terminales inalámbricos cuyas direcciones se asignarán por el protocolo MAC **IEEE 802.15.4**, del cual se encarga el propio tinyOS, y cuya comunicación funcional se hará a cabo siguiendo este método:
  - Las motas mandan, mediante una señal de difusión masiva, una señal de solicitud de ingreso a la red
  - Las motas ya están programadas para seguir el esquema de comunicación siguiente:
  - Una mota del tipo 1 o del tipo 2 sólo se comunican con una del tipo 3 y con ninguna más
  - Una mota del tipo 3 sólo transmite hacia una mota del tipo 4 y recibe de las motas tipo 1 y 2
  - Una mota del tipo 4 recibe resúmenes y avisos de una mota del tipo 3
- Siguiendo los anteriores axiomas, las motas responden, si el tipo de mota que solicita el ingreso constituye una fuente de datos de entrada, una señal de respuesta a esa mota con su dirección, quedando así la mota integrada en la red
- Las motas, una vez integradas, sólo transmiten a destinatario específicos, no mandando nunca mensajes de difusión masiva.
- Si una mota no responde durante mucho tiempo, se asume avería
- No importa que algunas transmisiones se pierdan, siempre y cuando su número no exceda
- Las transmisiones perdidas no se recuperan, pues sólo tienen importancia en el momento de su lectura.

El esquema de explotación del sistema es:

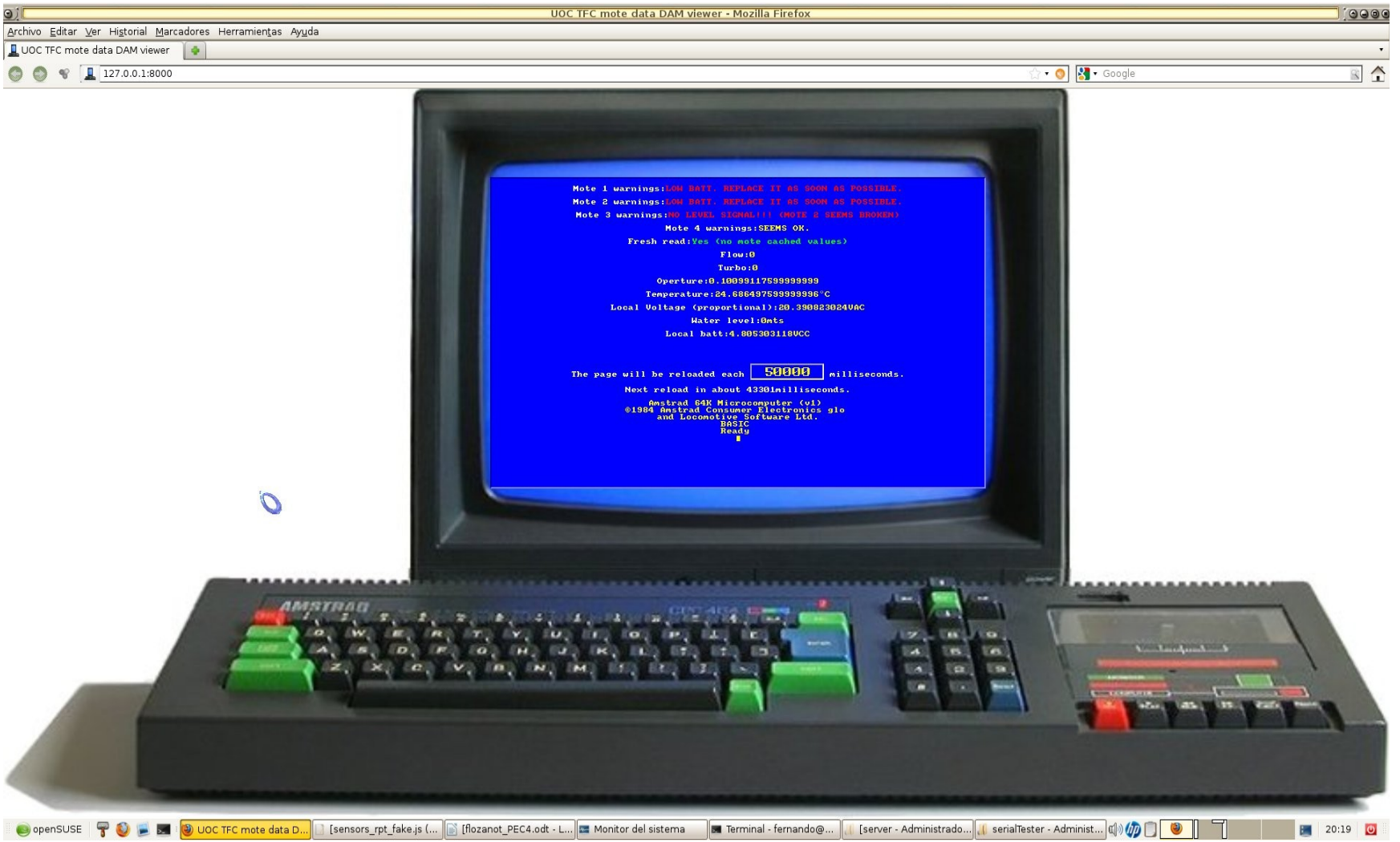
- Un terminal Linux ejecutando varios servidores que abren distintos bufferes FIFO



- Un miniservidor web capaz de leer bufferes FIFO como si de ficheros normales se tratara



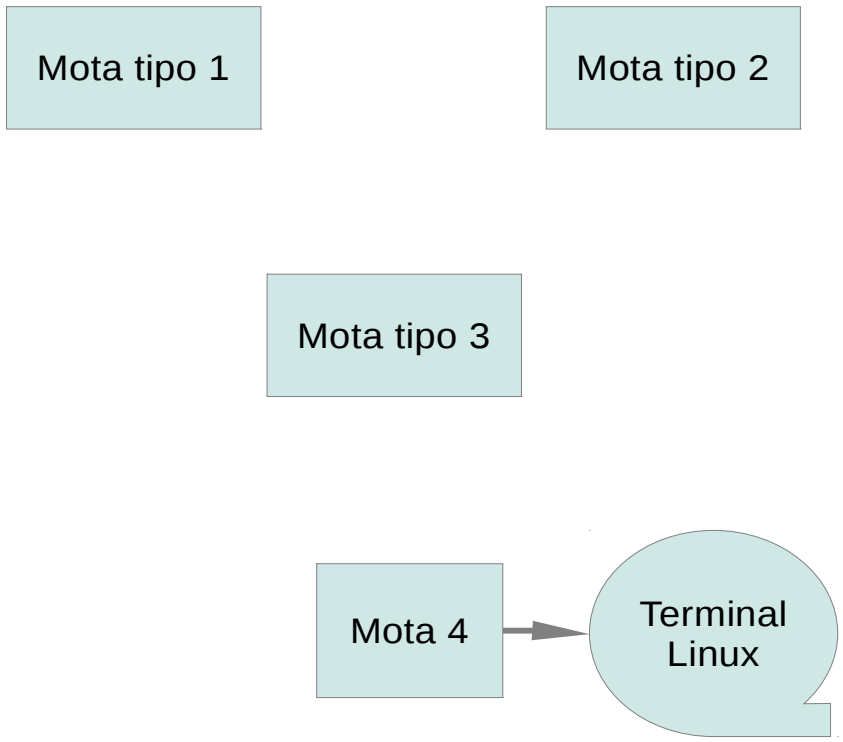
- Un página con HTML, CSS, JavaScript y todos sus recursos alojada en el servidor web que puede ser específica para una turbina o de propósito generalista
- Un navegador web manda una petición HTML estándard al miniservidor web
- El miniservidor web lee el buffer FIFO del servidor de recepción de datos para transmitirlos como variables de un programa JavaScript



Descripción detallada

Escenario de partida

Las motas resuelven sus direcciones mediante el protocolo MAC para redes inalambricas de baja velocidad

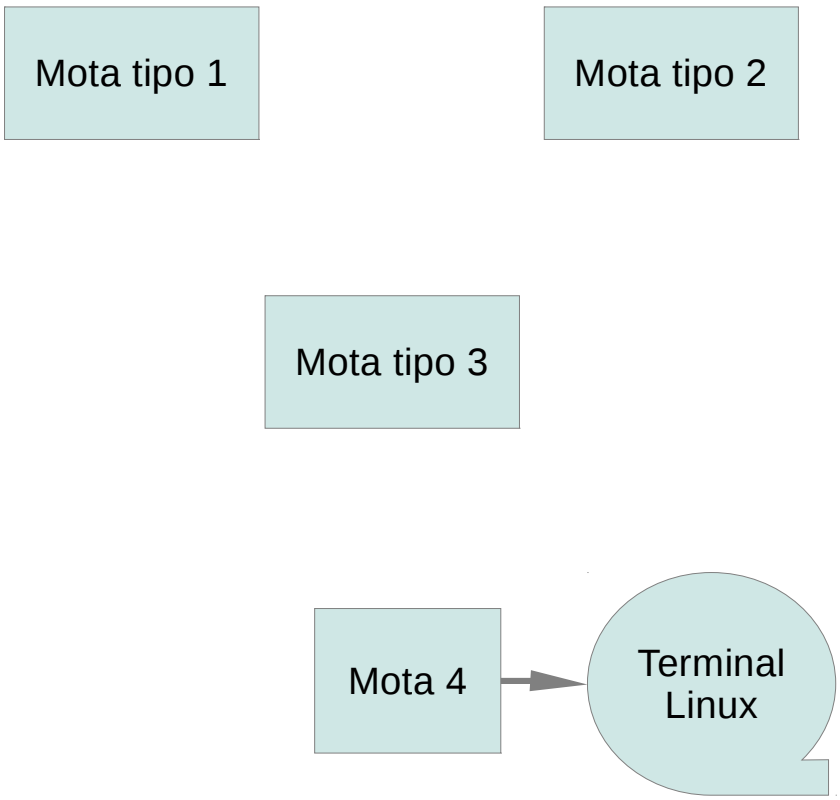


Ya sólo queda implementar un protocolo de red.
El protocolo de nivel de Transporte y superiores no son viables ni tienen sentido, dada la naturaleza del proyecto

Primera fase de constitución de la red

Las normas son las siguientes:

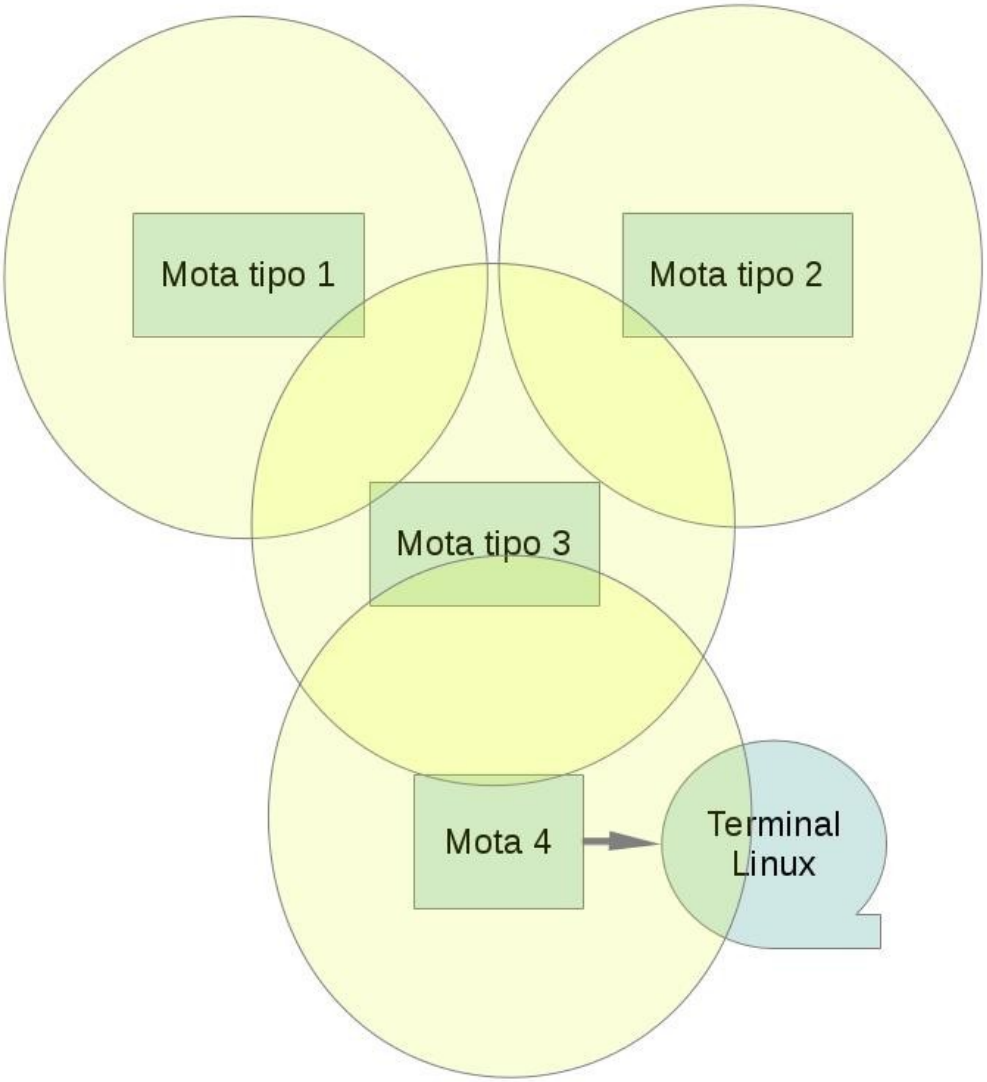
- Una mota tipo 1 o 2 siempre enviará hacia una mota tipo 3
- Las motas tipo 3 son colectores de información o repetidores si los mensajes de las motas 1 o 2 son avisos de estado
- Una mota tipo 3 sólo envía a una mota tipo 4 y sólo recibe de las motas tipo 1 y 2
- Una mota tipo 4 sólo recibe de una mota tipo 3 y envía, vía puerto de comunicación en serie, al terminal Linux



Mensaje de difusión masiva

La primera acción que llevan a cabo cualquier mota de cualquier tipo es mandar un mensaje de difusión masiva con su dirección de red y su MAC.

Todas las motas recibirán el mensaje con la MAC, dirección y tipo del emisor.



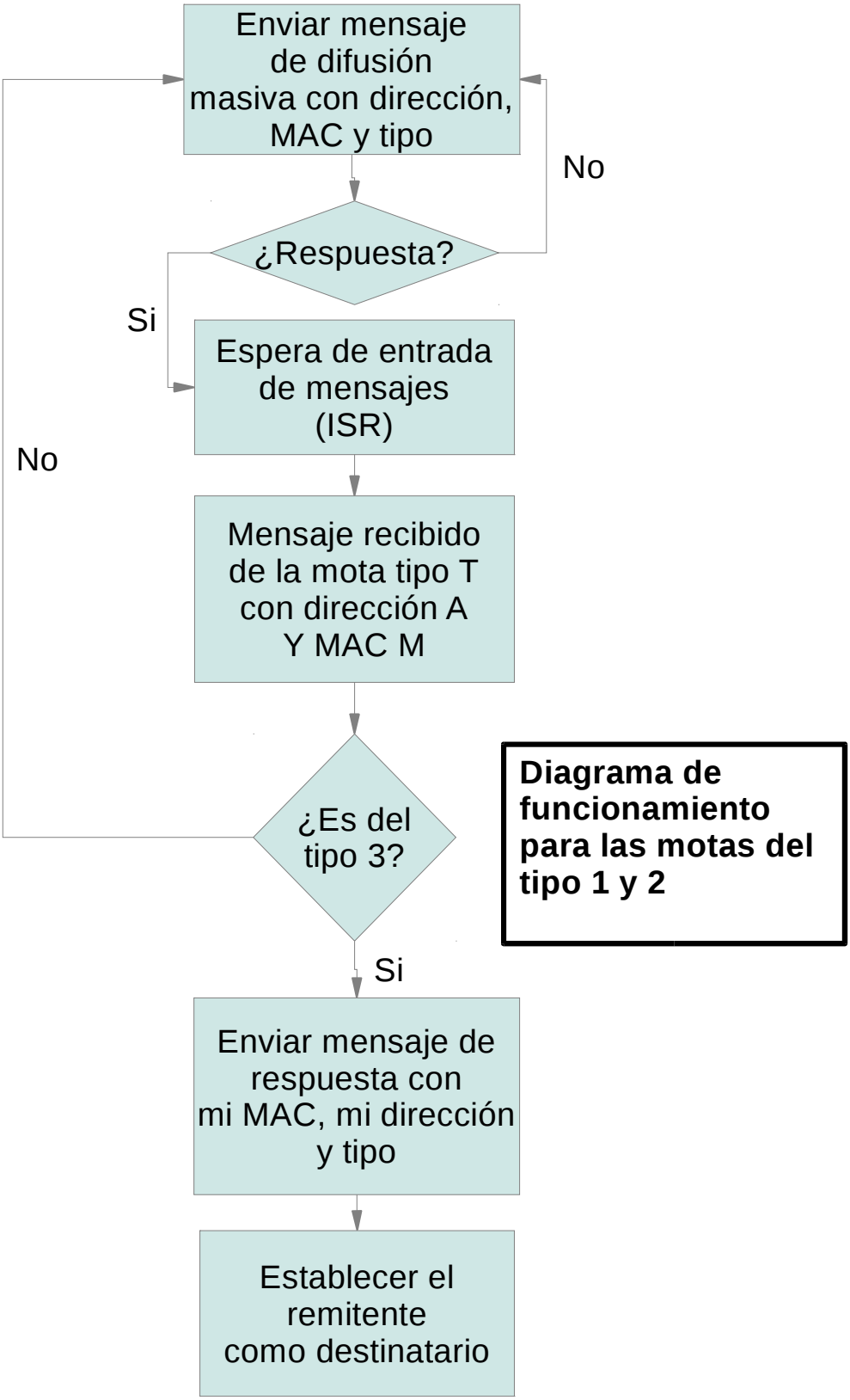


Diagrama de funcionamiento para las motas del tipo 1 y 2

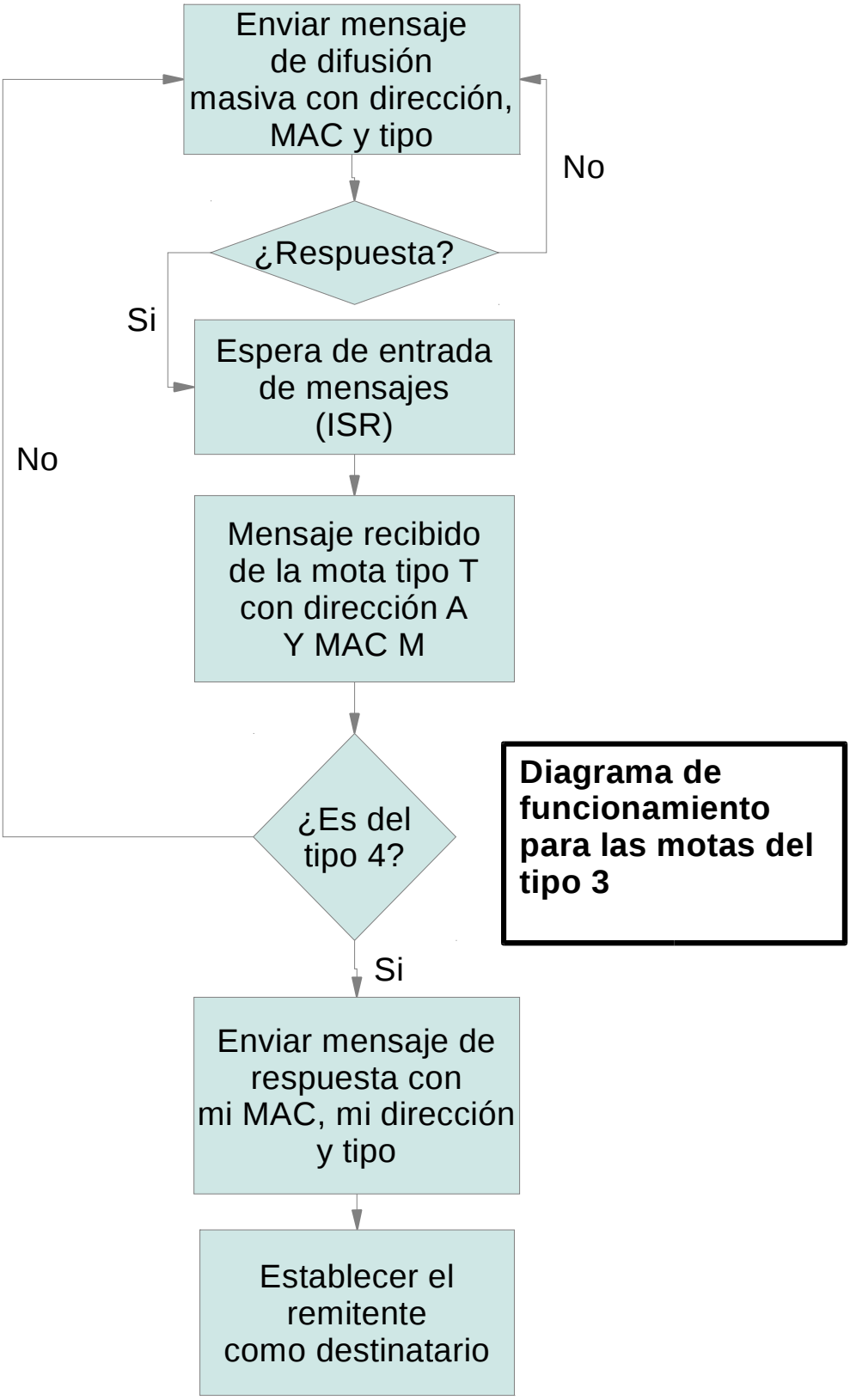


Diagrama de funcionamiento para las motas del tipo 3

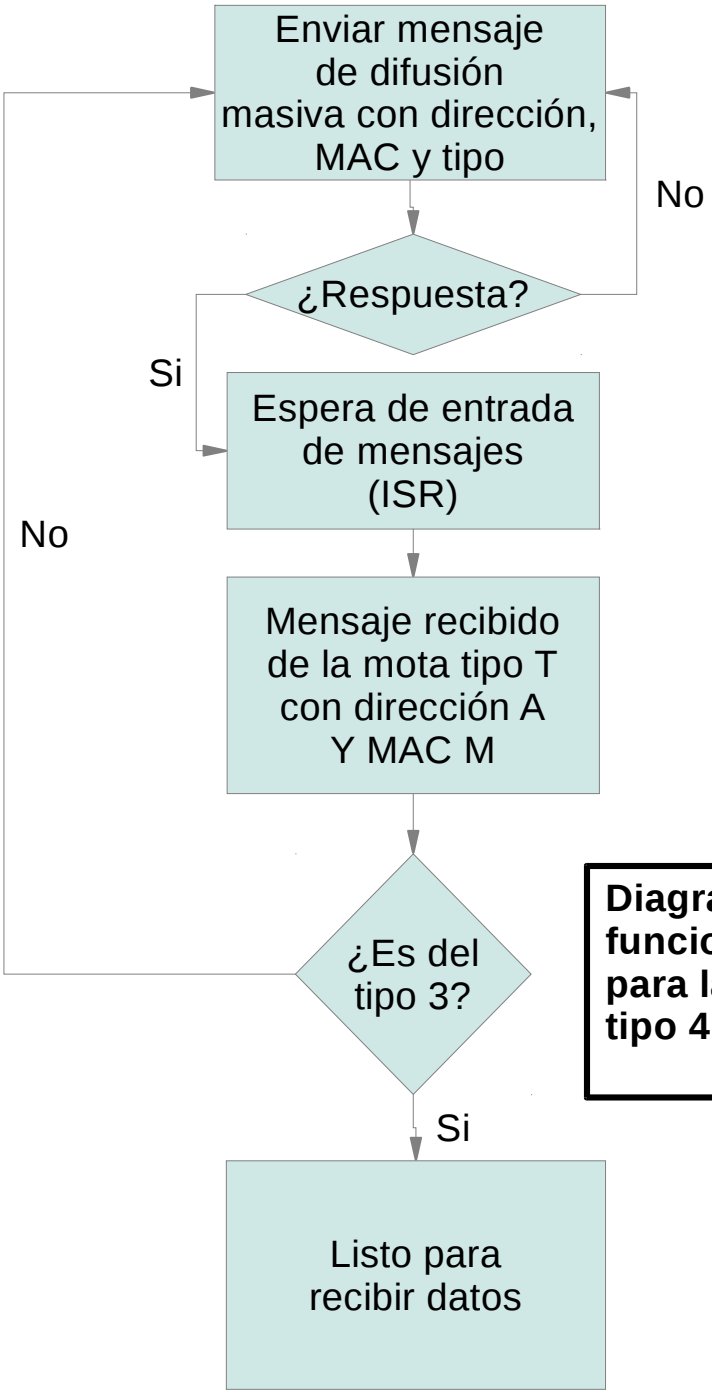
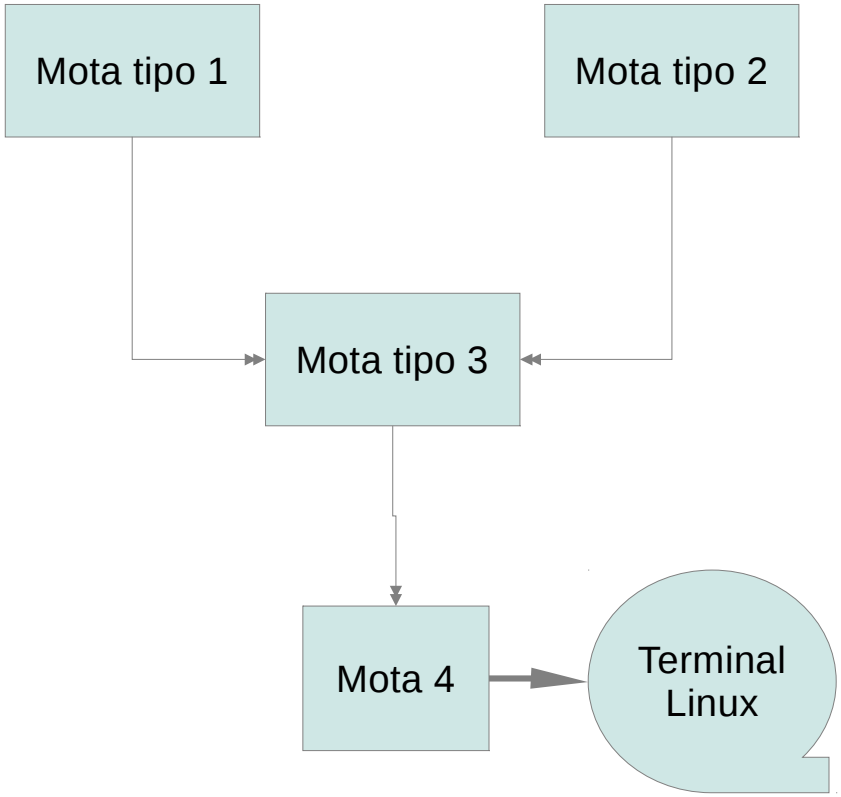


Diagrama de funcionamiento para las motas del tipo 4

Esquema final de la red

- Se establece comunicación unidireccional entre las motasdel tipo 1 y 2 hacia las de tipo 3, y las motas de tipo 3 se comunican unidireccionalmente hacia las de tipo 4.
- Las de tipo 4 estan conectadas vía comunicación en serie con el equipo gestor de la información que estas recopilan.
- La mota tipo 3 manda resúmenes o repite avisos hacia la mota 4. El ancho de banda de las motas no da para mucho más si se pretende una fiabilidad sumamente satisfactoria.
- Si un mensaje se pierde, no importa. Los mesajes sólo tienen sentido en el momento de su lectura.
- Si es un aviso, se repetirá ciclicamente. Si una mota está demasiado tiempo sin transmitir, la mota receptora mandará mensajes de aviso.



Valoración económica

Concepto	Cantidad	Coste unitario	Coste total
Terminal Linux	1	150€	150€
Mota Hall	1	26€	26€
Mota Hall+Sensor de nivel de presa	2	27€	54€
Mota Hall+Sesnsor de temperetura+Medidor de tensión	1	28€	28€
Integrador			60€
Programación mota T1	1	409,2€	409,2€
Programación mota T2	1	426,25€	426,25€
Programación mota T3	1	426,25€	426,25€
Programación mota T4	1	443,3€	443,3€
Programación del servidor central	1	30€	30€
Programación del miniservidor	1	30€	30€
Instalación y configuración del sistema			200€
Total:			2283€
Proveedores de componentes: Nanyang Xinlijia Electronics Co., Ltd. (China continental), Atmel y Silabs			

Desarrollo del software y producción del hardware necesario en un plazo aproximado de dos meses y medio. Después de este periodo, se procederá a la implantación e instalación de los dispositivos en las estructuras destinadas para albergar los elementos electrónicos necesarios.

Se estima que en un plazo máximo de 72 horas el sistema quedará instalado.

Conclusiones

En el desarrollo del TFC he alcanzado todos mis objetivos iniciales: he ejercitado mis habilidades con C y sus dialecto nesC, así mismo he tenido la ocasión de practicar ingeniería inversa descifrando formatos de tramas de datos, he sacado partido de mis conocimientos de programación en ensamblador AT&T, he aprendido a trabajar con autodisciplina, he buscado proveedores de componentes, he visto las tarifas de los distintos integradores y he podido ver las dificultades añadidas al desarrollo de un proyecto.

He tenido que analizar situaciones tales como decidir si ser desarrollador e integrador o externalizar la integración, y en el caso de externalizar, ver el catálogo de compañías de outsourcing o subcontratas para la realización de las motas.

He analizado los costes de amortización de un taller de integración y montaje de circuitos electrónicos, llegando a la conclusión que sólo si el volumen de negocio es grande, merece la pena ser tambien integrador. Tambien me he apoyado en hechos históricos, como el desarrollo de los Amstrad CPC, en los que el constructor era la empresa Amstrad y la desarrolladora del software fue Locomotiv Software, gracias a la cual el Amstrad CPC cambio a un Zilog Z80 por tener ya implementado un núcleo para este.

Generalmente las compañías integradoras usan sus propios suministradores de componentes, pero hay empresas que si que admiten componentes ya comprados. Buscando, me he dado cuenta de que la mayoría de los componentes proceden, sobretodo, de una empresa de la China continental llamada Nanyang Xinlijia Electronics Co., Ltd. que goza, quien lo iba a decir, de una excelente fama.

Gracias a este trabajo de fin de carrera he podido contemplar el panorama tecnológico actual, dónde el grueso de la producción tecnológica de gama mini o baja se centra en China, los de gama media en Alemania o Francia, al igual que los principales integradores, como Hércules, y los mejores microcontroladores se ensamblan en Costa Rica, así como los mejores condensadores se fabrican en Alemania, Japón y EEUU y los de gama media en Taiwan y China.

Por mi experiencia, los retrasos debidos a los proveedores ocurren cuando el volumen de pedidos es muy bajo, pues siempre dan prioridad al cliente que mas volumen encargue. Esto se recrudece con las chapuzas perpetradas por las fabricas chinas, cuyas endebles soldaduras hacen de sus compañías integradoras estandartes del bajo coste.

Si lo que se busca es seriedad y mínimos retrasos, lo mejor son las compañías Alemanas y algunas Francesas.

En España, me he dado cuenta de que montar una empresa que dure más de un proyecto, es decir, cuya actividad sea sostenible, si se dedica al sector productivo, es una verdadera heroicidad. Honestamente, llego a la conclusión de que los mejores lugares para montar una industria es la zona del BENLUX (Bélgica, Holanda y Luxemburgo, sobre todo Holanda) y Alemania por excelencia.

El mercado de desarrollo de software, a todos los niveles, actualmente es un partido de ping-pong entre EEUU y Francia, ya que casi todo el software tiene patente francesa o estadounidense. España sólo se dedica a desarrollar software de 3er nivel, meras

modificaciones del software ya realizado por las grandes compañías norteamericanas y francesas.

A España se le atragantaron los 16bits y se ha quedo estancada en los 8bits. Los integradores que hay son de risa: compran productos chinos ya integrados, les cambian las etiquetas chavales y chavalas que a duras penas llegan a la veintena de años y a vender con el precio duplicado.

El software y hardware de fuente abierta son oportunidades de oro para que países atrasados o estancados, como España, puedan ponerse al día.

Por otra parte, es imposible en España iniciar un negocio sin endeudarte, cosa que no pasaría si fuese Alemán en lugar de Español.

Respecto al proyecto, quedan mejoras por hacer, se podría mejorar la gestión de la red añadiendo la posibilidad de conocer qué mota está fallando, en lugar del tipo (sería sumamente sencillo y no tomaría mas de media hora tenerlo listo) , controlar la sobrecarga de envíos a una mota en concreto (sería tan fácil como poner un contador y no responder si el contador ya es lo suficientemente alto, dejando que sean otras las que respondan), plantear el proyecto de desarrollo de un interfaz web profesional, optimizar los servidores, escribir los plugins de servicio, hacer que sea un plugin el que rellene el buffer FIFO en lugar del propio servidor, cosa que en un minuto estaría hecha, ya que nos es una tarea más compleja que copiar y pegar y escribir un par de líneas de código. Otra trivialidad sería mejorar el sistema de control de errores de operación en las motas, pues el actual esquema se basa en las trasmisiones masivas de un tipo de mota a otro, sin importar si la principal cayó o no. Conforme a lo que hay implementado, simplemente sería cuestión de que el servidor en el terminal Linux pasase también en el FIFO la dirección de la mota, lo cual es una tarea trivial que sólo requeriría tocar el código del servidor.

Y otras cosas no tan triviales, como un sistema mas racional de soporte para motas caídas, ya que si una se avería no se advertirá su ausencia hasta una revisión *in situ* o hasta que todas las motas de ese tipo caigan.

Soy sumamente consciente de las deficiencias que mi implementación tiene. No obstante, se me dejó claro desde un primer momento, que el objetivo de este TFC era sobre todo, demostrar una buena habilidad a la hora de gestionar los recursos que nos ofrece la mota y describir un escenario viable de aplicación, cosas que creo haber cumplido bien.

Anexos

Compilación

Simplemente ir a la carpeta raíz del proyecto y desde un terminal Linux, con las binutils instaladas y acceso a la internet, escribir “make” y pulsar enter.

Si hay algo que configurar, los guiones bash shell se encargarán de ayudarle a configurarlo mediante procesos automatizados guiados mediante diálogos en consola.

Instalación del software

En la carpeta “release/server/servidor/servidor.txt” estan las instrucciones de instalación detalladas para las motas y el terminal.

En la carpeta “release/client/cliente/cliente.txt” estan las instrucciones de instalación detalladas para conectar vía web.

Formato de la trama enviada por puerto paralelo

Esto son notas tomadas durante el desarrollo. Por falta de tiempo, no puedo elaborar el tutorial prometido.

Notas tomadas 19/11/2011:

```
Con tipo=8 y 25 bytes conteniendo "Hola Mundo\n"
(0x7E)(0x45)(0x0)(0xFF)(0xFF)(0x0)(0x0)(0x19)(0x0)(0x8)(0x48)(0x6F)(0x6C)(0x61)(0x20)(0x4D)(0x75)(0x6E)(0x64)(0x6F)(0xA)(0x0)(0x0)(0x0)(0x0)
(0x0)(0x0)(0x0)(0x0)(0x0)(0x0)(0x0)(0x0)(0x0)

con tipo=255 y 25 bytes conteniendo "Hola Mundo\n"
(0x7E)(0x45)(0x0)(0xFF)(0xFF)(0x0)(0x0)(0x19)(0x0)(0xFF)(0x48)(0x6F)(0x6C)(0x61)(0x20)(0x4D)(0x75)(0x6E)(0x64)(0x6F)(0xA)(0x0)(0x0)(0x0)
(0x0)(0x0)(0x0)(0x0)(0x0)(0x0)(0x0)(0x0)(0x0)(0x0)


(0x7E)(0x45)(0x0)(0xFF)(0xFF)(0x0)(0x0)(0x19)(0x0)(0x8)(0x48)(0x6F)(0x6C)(0x61)(0x20)(0x4D)(0x75)(0x6E)(0x64)(0x6F)(0xA)(0x0)(0x0)(0x0)(0x0)
(0x0)(0x0)(0x0)(0x0)(0x0)(0x0)(0x0)(0x0)(0x0) ... (0xC8)(0x88)(0x7E)
(0x7E)(0x45)(0x0)(0xFF)(0xFF)(0x0)(0x0)(0x19)(0x0)(0xFF)(0x48)(0x6F)(0x6C)(0x61)(0x20)(0x4D)(0x75)(0x6E)(0x64)(0x6F)(0xA)(0x0)(0x0)(0x0)
(0x0)(0x0)(0x0)(0x0)(0x0)(0x0)(0x0)(0x0)(0x0)(0x0) .. (0xC8)(0x88)(0x7E)
                                          +-----SEC-------------------+-PAYLOAD                                                    SIZE-+-GRP-
+TYPE+-------------------------------------------------------------------------------------------DATA AREA---------------------...     +---
CR-----++STOP+
(0x7E)(0x45)(0x0)(0xFF)(0xFF)
START+PROTO+-SEQ+-ADDRDEST--+


con tipo=255 y mensaje "Adios mundo\n"
(0x7E)(0x45)(0x0)(0xFF)(0xFF)(0x0)(0x0)(0x19)(0x0)(0xFF)(0x41)(0x64)(0x69)(0x6F)(0x73)(0x20)(0x4D)(0x75)(0x6E)(0x64)(0x6F)(0xA)(0x0)(0x0)
(0x0)(0x0)(0x0)(0x0)(0x0)(0x0)(0x0)(0x0)(0x0)(0x0)(0x0)(0x0)(0x0)(0xA6)(0x3F)


Protocolo de comunicación:
Secuencia inicio:
+Leer del puerto hasta encontrar 0x7E.
+Leer un byte mas para verificar que el procolo es 0x45.
+Leer un byte mas para verificar que el byte de secuencia es 0.
+Leer una palabra para dirección de destino (será 0xFFFF, pero podría ser otra si así lo pongo en la mota...)
+Leer 3 bytes para el tamaño del payload. (Como nuesytra estructura esta entre 12bytes y 24bytes, sólo miraremos la última lectura)
+Leer un byte para el grupo (ignorar, debería ser 0)
+Leer un byte mas para el tipo (ignorar, debería ser 8)
+Leer tantos bytes como se especifica en "payload" (entre 12bytes y 24bytes)
+Leer una palabra (secuencia de cheksum, ignorar)
+Leer un byte mas. Es el byte de parada, y es igual al del inicio.
```

Esto se hizo programando una mota para que enviase los mensajes “Hola Mundo\n”y “Adios mundo\n”. Se ve mejor en el “gedit” maximizado, sin ajuste de línea y con una fuente monoespaciada de, al menos, 8 pts.