

Grado en Ingeniería Informática
2020-2021

Apuntes
Diseño de Sistemas Operativos

Jorge Rodríguez Fraile¹



Esta obra se encuentra sujeta a la licencia Creative Commons
Reconocimiento - No Comercial - Sin Obra Derivada

¹Universidad: 100405951@alumnos.uc3m.es | Personal: jrf1616@gmail.com

ÍNDICE GENERAL

0.1. Información.	1
1. TEMA 1: ARQUITECTURA DE SISTEMAS IOT	2
1.1. En IoT de vehículos hay distintos niveles	3
1.2. Elementos IoT	3
1.3. Evolución	3
1.4. Arquitectura de un Sistema IoT.	4
2. TEMA 2: SENSORES Y ACTUADORES	6
2.1. Sensores.	6
2.2. Actuadores	7
2.3. Factores de selección de Sensores y Actuadores	7
3. TEMA 3: SISTEMAS OPERATIVOS EMBEBIDOS PARA DISPOSITIVOS IOT	8
3.1. ¿Que es un sistema embebido o integrado?	8
3.1.1. Ordenador personal vs. Sistema embebido.	9
3.2. Estructura de un sistema embebido.	10
3.3. Ejemplos de Sistemas embebidos que usamos.	11
3.3.1. Arduino	11
3.3.2. Raspberry Pi	12
3.3.3. Raspberry Pi vs. Arduino	13
3.3.4. ¿Es Raspberry Pi un dispositivo de IoT?.	14
3.4. Sistemas Operativos embebidos.	14
3.4.1. Principales Sistemas Operativos embebidos actuales	14
3.4.2. Requisitos de un Sistema Operativo Embebido	14
3.4.3. Modelo general de un sistema embebido.	15

ÍNDICE DE FIGURAS

2.1	Diagrama de voltaje Sensor Analogico	6
2.2	Diagrama de voltaje Sensor Digital	6
3.1	Estructura genérica de un Sistema Embebido	10
3.2	Estructura Arduino	11
3.3	Estructura Raspberry Pi	12
3.4	Raspberry Pi vs. Arduino	13

ÍNDICE DE TABLAS

0.1. Información

Teorías: Javier García Guzmán

Prácticas: Mat Max Montalvo Martínez

1. TEMA 1: ARQUITECTURA DE SISTEMAS IOT

Definición de Internet de las Cosas

Aplicaciones Actuales de Internet de las Cosas

Arquitectura de Sistemas de Internet de las Cosas

IoT - Internet de las Cosas: Consiste en conectar a internet cualquier dispositivo, vehículo, edificios o en general objetos a los que se les haya dotado de sensores, actuadores y conexión a la red. Lo que les permite obtener e intercambiar información. Red de objetos conectados a internet que aporta valor añadidos a los usuarios que interactúan con ellos.

Consiste en añadir **inteligencia computacional** a dispositivos para mejorar las funcionalidades.

Permite que los dispositivos puedan intercambiar información.

Se busca que sean pequeños y tengan un chip que les permita **conectarse a la red** y operar en ella. Lo estandarizo IBM.

El **top 6 áreas** de aplicación de IoT:

1. **Industrial/Fabricación:** Automatizar, controlar la distribución, gestión de instalaciones.
2. **Transporte/Movilidad:** Coches, tráfico en ciudad, vehículos de transporte masivo y transporte industrial.
3. **Energía/Gestión eléctrica:** Predecir consumo, personalizar, bien estar de los ocupantes, monitorizar el consumo detallado, instalaciones con sensores.
4. **Gestión de inventarios/Comercial:** Saber cuánto queda y pedir, facilitar compras.
5. **Ciudad:** Recoge muchas áreas; basura, vigilancia, trafico, etc.
6. **Salud:** Investigación, la forma de tratar, las emergencias, distribución de información médica y dispositivos.

Agricultura de precisión: Según las previsiones ambientales y meteorológicas, plagas y demás se calcula el mejor momento para sembrar. Además, cuando ya está plantado, las condiciones del suelo, cuando regar y abonar las tierras, así como la recolecta.

1.1. En IoT de vehículos hay distintos niveles

0: Sin automatización.

1: Asistencia en la conducción.

2: Control de carril, lateral y longitudinal.

3: Conducción autónoma, pero el conductor en su puesto, para riesgos.

4: Conducción autónoma, pero el conductor en su puesto, supervisar.

5: Conducción totalmente autónoma, sin conductor.

1.2. Elementos IoT

Colector: Recogen información, sensores, o se activan, actuadores, que se intercambian en internet.

Transmisor: Puertas de enlace, pasarelas, pasan los datos a la red desde los dispositivos.

Agregación + Distribución: Calculo y procesamiento de la información.

Consumidor: Los usuarios/clientes acceden a los datos.

1.3. Evolución

3 generaciones.

1. RFID y sensores

Tecnología de detección por radiofrecuencia.

Las cosas contengan información, las etiquetas (como NFC, se estandarizó para indicar que datos contiene) y tener un dispositivo que al acercarlo podamos leerla.

2. Web services e inter-networking (2004-2012): Interconexión completa de las cosas y la red de las cosas.

IPv4, HTTP, Bluetooth, TCP, UDP, etc.

Pasan a tener una manera fácil de conectarse los dispositivos entre sí o con internet.

3. Social, Cloud & ICN: La era de la computación en la nube y la Internet del futuro.

En esta generación la lógica pasa a estar en la nube, no en el dispositivo.

Gestión de grandes cantidades de información.

Seguridad, evitar accesos fraudulentos.

1.4. Arquitectura de un Sistema IoT

Dispositivos (Devices) Sensores y actuadores FÍSICOS, que normalmente tienen un microprocesador, que mide el medio físico y transforma las mediciones a señales digitales. Su función es tomar medidas y procesarlas, pero su función no puede ser solo transmitir la información.

Actuador, Sensores, LED, LCD, Beacon (la parte dispositivo), Termostato, RFID, Trampa para ratones inteligente, Dispositivos embebidos, etc.

Pasarela (Gateway) Dispositivo o protocolo con la capacidad de comunicar con internet los dispositivos, para transmitir los datos tomados. Su única función es transmitir.

Router, Wifi, GSM, Bluetooth, Zigbee, Raspberry a veces, AMQP, CoAP, LoRa-WAN (sistema de radio), Wimax.

Un móvil está entre Device y Gateway.

Plataforma IoT (IoT Platform) Conjunto de servicios orquestados para gestionar una gran red de dispositivos interconectados y que proporcionan información a aplicaciones u otros tipos de sistemas de información. Gestiona y almacena grandes cantidades de datos y las redirige. Funciona como middleware.

Es una nube de servidores que dan servicios:

1. **Message broker y Message bus:** Se encarga de conectar los dispositivos físicos con los distintos procesos que forman parte de la red de IoT. Manda los datos a todos los que estén conectados a su bus, suscritos por API Rest.
2. **Message router:** Está suscrito al message broker, los mensajes que recibe los enriquece; dando información semántica, de contexto, de estado y los reenvía a aquellos componentes que van a gestionar la lógica de las aplicaciones relacionadas con la nube IoT. Otra cosa que hace es transformar datos, descomprimir y decodificar datos para hacerlos más fáciles de procesar y tratar.
3. **Rest API:** Interfaz que usan otros programas para obtener los servicios o las funcionalidades de un componente. Esta API se caracteriza por ser accesible por http e independiente del estado del sistema.
4. **Data Management:** Para almacenar y gestionar los datos tomados en la red.
5. **Rule engine:** Permite monitorear los mensajes recibidos desde el router y permite lanzar distintas acciones en distintos elementos. Decide que acción tomar. Ejem: Si se abre la puerta, entonces avisar de intruso.
6. **Microservicios:** Proporciona funcionalidades muy específicas a través de una interfaz API Rest bien definidas mediante un contrato de datos. Muchos los coordina el rule engine. Se busca que este muy cohesionado y poco acoplado. Ejem: El que actualiza la estación meteorológica en el móvil, es un proceso muy concreto.

7. **Device manager:** Permite monitorizar algunos elementos de los sensores físicos como si está activo, la batería o si está conectado a la red.
8. **App y User management:** Sistema de permisos que identifica y gestiona el acceso de usuarios y aplicaciones.

Aplicación (Application) La interfaz que el usuario utiliza para controlar el sistema.

iBeacon es un ejemplo de protocolo y **Beacon** es el dispositivo. Ambos están relacionados con dispositivos.

Small Data: Que solo proporcione la información de valor añadido. Es un conjunto de datos con un volumen y un formato que hacen que los datos sean accesibles, informativos y procesables.

Wimax: Conjunto de tecnologías y protocolos para aumentar el alcance de las redes inalámbricas (en vez de 30 metros, 40 kilómetros).

2. TEMA 2: SENSORES Y ACTUADORES

Introducción a Sensores y Actuadores

Hasta la sección 'Up to Functional examples (putting it all together)' incluida.

2.1. Sensores

Conjunto de componentes electrónicos capaces de detectar cambios físicos en el entorno y enviar información a otros componentes electrónicos, generalmente un procesador de computadora.

Ejemplos: Sensor de luz (LDR), sensor de ultrasonidos, giroscopio, fototransistor, Reed switch, ...

Los sensores se pueden clasificar en tipos según lo que miden: Gases, velocidad, flujo, fugas, movimiento, electricidad, ...

Según la señal que produce:

- **Analógico:** Produce voltaje analógico constante de lo medido. El grafico de voltaje sobre el tiempo debe ser continuo y suave.

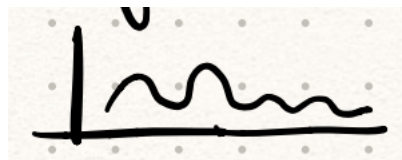


Fig. 2.1. Diagrama de voltaje Sensor Analógico

Sensor de presión, sensor de luz, sensor de temperatura, acelerómetro, sensor de sonido.

- **Digital:** Produce un voltaje discreto, por lo general tendrá uno u otro de dos valores, 0V (apagado) a 5V (encendido). Gracias a la miniaturización hay más dado que se puede introducir un conversor.

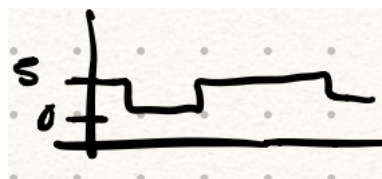


Fig. 2.2. Diagrama de voltaje Sensor Digital

Sensor de ultrasonidos, sensor de infrarrojos, acelerómetro, sensor de sonido (suele ser analógico), sensor de temperatura.

Según si necesitan energía:

- **Sensor activo:** Siempre **necesitan** su propia fuente de energía.
 - Sensor de ultrasonidos, radar, LiDAR, sensor de humedad, cámara infrarroja.
- **Sensor pasivo:** **No necesitan** una fuente de energía, usan factores externos para alimentarse.
 - Sensor infrarrojo (fotodiodo infrarrojo), sensor PIR, sensor de luz (LDR)

Sensor piezoeléctrico:

1. Un cristal piezoeléctrico se coloca entre dos placas de metal que están en perfecto equilibrio y conduce ninguna corriente eléctrica.
2. Las placas de metal aplican tensión o fuerza mecánica sobre el material que hace que las cargas eléctricas del cristal se desequilibren.
3. Las placas de metal recogen esas cargas y producen un voltaje y envía una corriente eléctrica a través de un circuito.

2.2. Actuadores

Cualquier dispositivo capaz de intervenir para cambiar las condiciones físicas del entorno generando los datos.

Ejemplos: Display, LED, servomotor, motor de paso a paso, Relay, solenoide, actuadores lineales, ...

2.3. Factores de selección de Sensores y Actuadores

Factores ambientales: Temperatura, Humedad, Corrosión, Interferencia electromagnética, Tamaño, Rudeza y Consumo de energía.

Factores económicos: Coste, Disponibilidad y Tiempo de vida.

Factores característicos del sensor: Sensibilidad, Rango, Estabilidad, Repetibilidad, Rango de error, Tiempo de respuesta y Linealidad.

3. TEMA 3: SISTEMAS OPERATIVOS EMBEBIDOS PARA DISPOSITIVOS IOT

[Introducción a los Sistemas Embebidos](#) Solo capítulo 1

[Sistemas Operativos Embebidos](#) Capítulo 9

[Drivers de Dispositivos](#) Hasta el encabezado "8.1 Example 1: Device Drivers for Interrupt Handling"

[Middleware](#) Secciones 10.1 y 10.2

3.1. ¿Que es un sistema embebido o integrado?

Sistemas basados en computadora que no parecen ser computadoras, la complejidad esta oculta al usuario.

Son sistemas que integran uno o más sensores y que son capaces de comunicarse con la red, con capacidades limitadas, por lo que están entre la capa de Dispositivos y Pasarelas.

Se aplican sobre cosas cotidianas para mejorarla, pero no proporciona una mayor complejidad del sistema, permite realizar la mismas funciones o alguna más pero mejor.

Todo dispositivo IoT es un sistema embebido, pero no todo sistema embebido es IoT. Los sistemas IoT son accesibles a través de internet y puede enviar la información que registra en tiempo real por internet.

Los sistema embebidos o integrados son aquellos capaces de interactuar con el usuario (a través de una interfaz simple) o con otra herramienta invisible para el usuario. Es decir, no tiene por qué haber una interacción directa con el usuario (un pendrive se enchufa al ordenador, no al usuario)

- Ejem: Memoria flash, pendrive, sistema antibloqueo de ruedas.

Un sistema IoT es aquel con el que podemos interactuar directamente, acceder a sus datos o que nos los muestre, y tiene capacidad de internet. Hoy en día es muy barato transformar un sistema embebido a IoT.

Factor clave de los sistemas embebidos:

- La **eficiencia**, velocidad a la que responde o realiza la tarea específica. Para alcanzar la eficiencia **se cambia el enfoque de la programación**, no hay recursos ilimitados y hay que adaptarlo para que consuma poca energía y memoria.
- El **consumo de energía**, si se encuentra en algún lugar remoto y tiene una batería debe durar mucho.

- El **uso de memoria**, ya que afecta al rendimiento y son caras.
- **Precio**, ya que ante productos similares se elige el más barato.
- **Sistema crítico**, aquel del que el tiempo de respuesta es clave, que si falla puede correr riesgo alguna vida humana.

Fuertes restricciones: Coste de fabricación, Coste de diseño, Rendimiento, Energía, Tiempo de comercialización.

No podemos aprovechar la Ley de Moore, nos tenemos que ajustar al sistema como está actualmente, no podemos esperar a que pase el tiempo suficiente para que compremos otro que de mejor rendimiento. Hay que diseñar sistemas que sean rápidos con la tecnología actual y pueda durar en el un largo periodo de tiempo.

Del cuestionario:

- Se dice que un **sistema es en tiempo real si el tiempo de respuesta es crítico**. Como el sistema ABS o de detección de colisión.
- Es cierto que la mayoría de los sistemas informáticos integrados están diseñados por equipos pequeños con plazo ajustados.
- Un sistema en tiempo real se define como un sistema cuya corrección de la puntualidad de su respuesta.
- Es cierto que un sistema integrado puede definirse como un sistema de control o un sistema informático diseñado para realizar una tarea específica.

3.1.1. Ordenador personal vs. Sistema embebido

Sistema embebido: Son específicos de una aplicación, se focalizan en una tarea o conjunto de tareas relacionadas en todo momento.

- Todos los recursos están dirigidos a realizar esa tarea, por lo que la realiza de manera eficiente, pero no le sobran recursos y realizar alguna otra tarea es muy difícil o imposible. El software y hardware lo diseñan juntos por lo que es más eficiente y fiable, se adaptan al hardware perfectamente.
- Utilizan arquitecturas muy variadas, con diferentes CPU, periféricos, SO y prioridades de diseño.
- El tiempo de arranque es casi instantáneo, medido en segundos.

Computadora de escritorio: Puede ejecutar cualquier clase de aplicación según las necesidades del usuario.

- Está listo para cualquier tarea por lo que consume más energía y recursos. El diseño de hardware lo desarrollan empresas distintas, por lo que sobran recursos o se requiere más de los que hay, sobreestima. Además, se pueden ampliar fácil y económicamente si es necesario.
- Usan una arquitectura muy similar todos y ejecutan software en sistemas idénticos.
- El tiempo de inicio se puede medir en minutos cuando se carga desde disco.

Del cuestionario:

- Un sistema embebido no necesita interacción humana para realizar tareas.
- Un sistema embebido necesita menos potencia operativa que una computadora.
- Los ordenadores se pueden reprogramar para un nuevo propósito.
- Los ordenadores son difíciles cuando se usan, en comparación con un sistema embebido.
- Los ordenadores pueden realizar muchas tareas.

3.2. Estructura de un sistema embebido

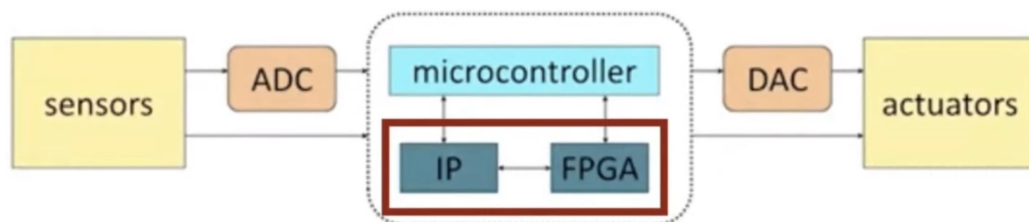


Fig. 3.1. Estructura genérica de un Sistema Embebido

Microcontrolador Es el componente que ejecuta un programa en un sistema integrado, y se encuentra en la zona central, se encarga de realizar el procesamiento. Recibe los datos de los sensores, procesa y envía señal a los actuadores.

Consta de CPU, RAM, ROM, puertos E/S y temporizadores.

Más lento que un microprocesador, menos memoria y menos funciones.

Requiere ser programado, se debe escribir el código del programa que va a realizar. El código se escribe en el host, un ordenador de sobremesa o laptop, y después se transfiere el programa del host al microcontrolador.

Cajas negras Realizan parte del trabajo del microprocesador para reducir la carga, algunas tareas específicas. Algunos ejemplos son:

IP Core Circuito integrado que desempeña una función que interactúa con el microcontrolador, y son baratos en un volumen alto. Muy útiles para tareas comunes como controlador de network (Ethernet, CAN) o de audio/video (Audio Códec, Controlador VGA). Necesita interactuar con el microprocesador y se siguen unos protocolos de comunicación.

FPGA - Field Programmable Gate Array Matriz de puertas lógicas programable en campo, es un chip con una red de puertas de memoria RAM que se pueden configurar (establecer las conexiones entre chips o puertas), para realizar una serie de tareas, como puede ser filtrar una señal o comprimir video. Es más rápido que software, pero más lento que ASIC.

DSP Procesadores y compresores de señales digitales, tanto de audio como de video. Son más económicos que los procesadores, pero tiene capacidades más limitadas.

Conversores Analógico-Digital sale de sensor, izquierda y el Digital-Analógico va al actuador, derecha. Los analógico a digital son muy comunes, porque los sensores son analógicos y el microcontrolador es digital.

Sensores y actuadores Toma medidas del medio que rodea al dispositivo y puede alterar el entorno. Sensores a la izquierda y actuadores a la derecha.

3.3. Ejemplos de Sistemas embebidos que usamos

3.3.1. Arduino



Fig. 3.2. Estructura Arduino

Es una plataforma de código abierto sed para construir proyecto de electrónica. Consta de una placa de circuito programable física (microcontrolador) y una pieza de software, o IDE (entorno de desarrollo) que se ejecuta en una computadora portátil o computadora

personal utilizada para escribir y cargar código de computadora en la física. Tiene pines analógicos y digitales.

No necesita una pieza software separada para cargar un nuevo código en la placa.

Utiliza una versión simplificada de C++, lo que lo hace más fácil de aprender.

Razones por la que se usa: Es de código abierto, económico, multiplataforma, prototipos rápidos, entorno programable más simple y claro. Algunas también cuentan con conexión a Internet incorporada o la posibilidad de conectividad externa.

Razones para no utilizarlos: Poca memoria para datos y RAM, procesamiento, arquitectura basada en 8 bits, caro cuando se necesita a gran escala, mala calidad/precio, las empresas no les gusta que sea open source, no está aliado con proveedores y no da garantías.

3.3.2. Raspberry Pi

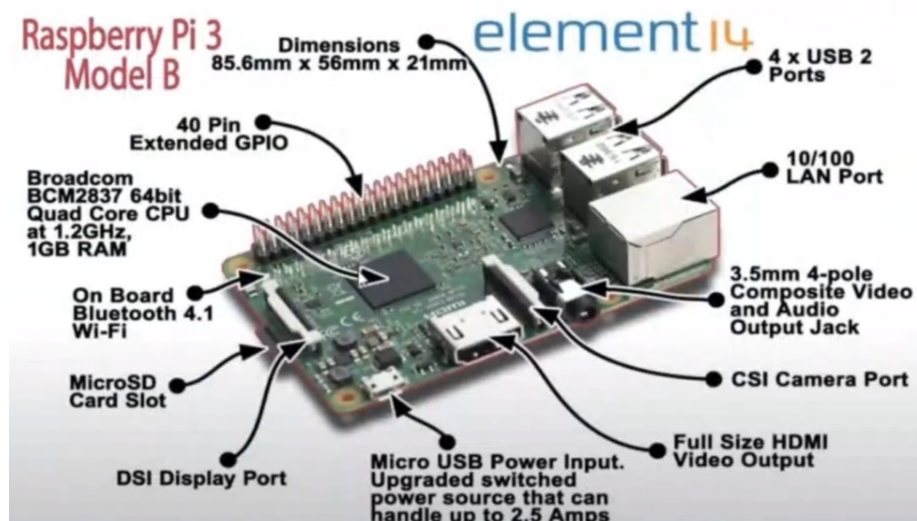


Fig. 3.3. Estructura Raspberry Pi

Es una computadora de bajo costo del tamaño de una tarjeta de crédito que se conecta a un monitor de computadora o televisor y utiliza un teclado y un mouse estándar.

La Raspberry PI tiene una conexión GPIO de 40 pines, lo que facilita la conexión con el mundo exterior.

GPIO Significa entrada/salida de uso general. Estos pines son una interfaz física entre Raspberry Pi y el mundo exterior.

El Pi puede recibir información de sensores y actuadores, y controlar actuadores como LED, ejecutar motores y muchas otras cosas.

Hay 40 pines y proporcionan varias funciones diferentes.

- Los pines 27 y 28, ID SD e ID SC, son para conectar una EEPROM.
- Pines 8 y 10, GPIO 14 y 15, son para comunicación UART puerto serie que transmite poca información (baja frecuencia), como USB, RFID, Bluetooth, GPS, GSM o GPRS.
- GPIO 2 y 3 (clock) se usan para I2C, comunicación entre placas, como un LCD, otras placas base (Raspberry, Arduino, ...) o un reloj de tiempo real.
- GPIO 7, 8, 9, 10 y 11 implementan otro mecanismo de comunicación el SPI, que es la evolución del protocolo I2C, su función es comunicar full duplex, como SSD card, IP Core o Memoria Flash.

Del cuestionario:

- Cual de los siguientes NO es un beneficio de usar un sistema operativo?

La frecuencia del reloj del microprocesador se puede aumentar significativamente.

3.3.3. Raspberry Pi vs. Arduino

- Raspberry usa un procesador de proposito general y Arduino un microprocesador.
- Raspberry Pi es más rápido (1.4GHz vs 16MHz).
- Raspberry Pi puede administrar un espacio de direcciones más grande (procesador de 64 bits, frente a 8 bits).
- Raspberry Pi tiene más memoria: Arduino tiene 32K Flash, 2K RAM, 1K EPROM; y Raspberry Pi SRAM de 1 Gb, Micro SD.
- Raspberry Pi tiene niveles de voltaje de E/S más bajos (3.3v frente a 5v)



 <p>Un Arduino es un microcontrolador. Son un chip autónomo. Por lo general, se pueden utilizar con muy pocos circuitos de soporte.</p>	 <p>Una placa Raspberry Pi tiene muchos chips diferentes, incluido un microprocesador, para crear una computadora funcional.</p>	<p>No ejecutan un sistema operativo, sino que revisan el código que se les proporciona.</p>	<p>Puede ejecutarse como una computadora independiente y el código se puede escribir en su interior.</p>
<p>Los sensores y otro hardware se pueden controlar directamente a través de los pines de E / S.</p>	<p>Para controlar los pines de E / S para sensores, motores, etc., debe escribir un código que controle el software de nivel inferior.</p>	<p>Diseñado para circuitos / hardware de bajo nivel y programación directa.</p>	<p>Diseñado para interacción de software y hardware de alto nivel.</p>

Fig. 3.4. Raspberry Pi vs. Arduino

3.3.4. ¿Es Raspberry Pi un dispositivo de IoT?

Similitudes:

- Conectividad de red e inteligencia computacional.
- Pequeño y barato (en relación con una PC).
- Puede interactuar directamente con sensores/actuadores a través de pines.

Diferencias:

- La interfaz puede ser exactamente la misma que la de una PC con Linux: Las complejidades del sistema pueden ser visibles.

3.4. Sistemas Operativos embebidos

Proporcionar una capa de abstracción para el software sobre el sistema operativo.

Administrar los diversos recursos de hardware y software del sistema para garantizar que todo el sistema funcione de manera eficiente y confiable.

FOTO DE CAPAS Y PARTES

3.4.1. Principales Sistemas Operativos embebidos actuales

Zephyr, Micrium, RTOS, RIOT, VxWorks, ThreadX, MicroEJ, TinyOS, APACHE, ARMmbed, Contiki, Nucleus, Windows IoT, snappy, android things, Mongoose o mynewt.

3.4.2. Requisitos de un Sistema Operativo Embebido

- Requisitos de CPU
- Requisito de Memoria, suelen tener poca memoria.
- Características limitadas

Autonomía

- Soporte a Plataformas
- Eficiencia energética, como suspenderse bajo inactividad o protocolos de comunicación de bajo consumo.
- Stack de Red Adaptativo

- Fiabilidad

Programabilidad: Gestionar procesos, threads, sockets...

- API estandar
- Lenguajes de Programación estandar

3.4.3. Modelo general de un sistema embebido

FOTO

Controladores de Dispositivos - Drivers

Son las bibliotecas de software que sirven para interactuar con los distintos elementos hardware. Inician el hardware y administran el acceso al mismo mediante capas superiores de software.

Acciones:

Hardware Startup Inicialización del hardware tras el encendido o el restablecimiento.

Hardware Shutdown Configurando el hardware en su estado PowerOFF.

Hardware Disable Permitir que otro software desactive el hardware sobre la marcha.

Hardware Acquire Permitir que otro software obtenga acceso singular (bloqueo) al hardware.

Hardware Write Permitir que otro software escriba datos en el hardware.

Hardware Install Permitir que otro software instale nuevo hardware sobre la marcha.

Hardware Release Permitir que otro software libere (desbloquee) hardware.

Hardware Unmapping Permitiendo eliminar bloques de datos de dispositivos de almacenamiento de hardware.

Kernel

Process Management Threads: Crear hilos, controlarlos, terminarlos o sincronizarlos...

Semaphores: Primitiva de sincronización.

Priority scheduling:

Real-time signal extension: Interrupciones y señales, para activar notificaciones de aplicaciones asincrónicamente.

Timers: Mecanismo de notificación de cuando un ha ocurrido un evento como la escritura de un dato.

.PCI.

Memory Management El proceso de

Process memory locking

Memory mapped files

Shared memory object

Kernel vs user mode?

I/O System Management Los dispositivos de E/S también deben compartirse entre los diversos procesos y, por tanto, al igual que con la memoria, el acceso y la asignación de un dispositivo de E/S deben administrarse.

Middleware

Cualquier software del sistema, que no es kernel del sistema operativo, controladores de dispositivos o las aplicaciones que se van a desarrollar.

Controladores de red.

Virtualización...

Middleware básico - core: Middleware de controlador de red.

Middleware complejo: Middleware específico del mercado, mensajería y comunicaciones complejas, mensajería distribuida y orientada a mensajes, y transacciones distribuidas.

Gestión de multitareas y de procesos en sistemas embebidos No se pueden hacer múltiples tareas a la vez, por lo que tenemos que tener un mecanismo que administre y planifique el orden de ejecución de los distintos procesos. Indicar cuando entran y salen, y el paso de datos.

La Rpi como tiene 4 núcleos se considera nanoordenador, no sistema embebido.

Implementación de procesos en Sistemas Operativos embebidos Las tareas están estructuradas como una jerarquía de tareas principales y secundarias, y cuando se inicia el kernel embebido, solo existe una tarea.

Creación de tareas La creación tipo “fork” crea una copia del espacio de memoria de la tarea principal en lo que se asigna para la tarea secundaria, lo que permite que la tarea secundaria herede varias propiedades, como el código del programa y las variables, de la tarea principal.

La creación tipo “exec” se utiliza para eliminar explícitamente del espacio de memoria de la tarea secundaria cualquier referencia al programa principal y establece el nuevo código de programa que pertenece a la tarea secundaria para que se ejecute.

La creación tipo “spawn model” crea un espacio de direcciones completamente nuevo para la tarea secundaria. La llamada al "modelo de generación" permite definir el nuevo programa y los argumentos para la tarea secundaria.

Spawn model: No tiene espacios duplicados, no gasta tanta memoria como fork/exec.

fork/exec: Nos permite coordinar de una manera mas sencilla, pero es mas lenta por que tiene que copiar y eliminar cuando hace el exec.

Scheduler: Es el que nos dice cuando y cuanto tiene que ejecutar cada uno de los procesos, va metiendo y sacando de la CPU.

Factores clave del scheduling Reponse Time: Tiempo para que el programador haga que el contexto cambie a una tarea lista e incluye el tiempo de espera de la tarea en cola lista.

Turnaround time: El tiempo que tarda un proceso en completarse

Overhead: El tiempo y los datos necesarios para determinar que tareas se ejecutan a continuación.

Fairness: ¿Cuáles son los factores...

XXX

Starvation:

Preventivo vs. No preventivo Preventivo - Apropiativo: Las tareas reciben el control de la CPU maestra hasta que finalizan su ejecución, independientemente.

No Preventivo - No apropiativo: Se les puede expulsar de la CPU mientras ejecutan, se divide los procesos en rodajas.

Planificación e Raspberry Pi OS Opera en todos los componentes.

Solo se ejecuta una tarea a la vez.

El planificador es un componente independiente.

El programador predeterminado es una cola FIFO simple.

El planificación es consciente de la energía, y pone el procesador en suspensión cuando no hay ninguna tarea en la cola.