Sistemas con Microprocesadores

Teoría

Este es un resumen del contenido del temario de teoría de Sistemas con Microprocesadores que puede utilizarse para ampliar la información de las diapositivas. No incluye todo el contenido del temario, pero si toca gran parte de los puntos principales.

Para la mayoría de los puntos se puede encontrar información más detallada en el libro: *El Microcontrolador ATMega328P de Microchip: Programación en Ensamblador, Lenguaje C y un enlace con Arduino*. Santiago Espinosa, Felipe. 2021. Disponible gratuitamente en: http://repositorio.utm.mx/handle/123456789/388
En este documento las referencias a este libro aparecen como: [Nº de sección. Título]

Este documento está estructurado siguiendo mismo orden que el temario de la asignatura.

Contenido

Tema 1: Introducción

1.1 Microprocesador vs microcontrolador:

Los microprocesadores y microcontroladores a menudo se confunden, pero tienen diferencias clave. Un microprocesador contiene solo una CPU, mientras que un microcontrolador incluye CPU, memoria, temporizadores y más circuitería en un solo circuito integrado. Aunque los microcontroladores tienen más recursos integrados, su capacidad de procesamiento suele ser limitada en comparación con los microprocesadores. Los microcontroladores son adecuados para sistemas de propósito específico, como electrodomésticos y equipos industriales, donde no se necesita alta velocidad o manejo masivo de datos. En contraste, los microprocesadores se utilizan en computadoras y dispositivos que requieren multitarea y procesamiento intensivo, como videoconsolas de alto rendimiento. Hay otras diferencias notables, como la interfaz que tienen con el exterior, es decir, el patillaje (los procesadores se suelen conectar con un bus y los microcontroladores tienen puestos de entrada/salida que permiten conectarse directamente con sensores y actuadores), también el consumo (los procesadores suelen consumir más), el software (los procesadores suelen utilizar sistema operativo para poder ejecutar una aplicación u otra, mientras que en los pequeños microcontroladores suelen ejecutar una misma aplicación que accede directamente al hardware: programación *bare metal*).
[1.3. Microprocesadores y Microcontroladores]

- **Componentes de un computador:**
    - **CPU (Central Processing Unit):** El cerebro del sistema que ejecuta instrucciones.
    - **Memoria:**
        - **Volátil (RAM):** Utilizada para almacenar datos temporales (como las variables del programa).
        - **No volátil (ROM, EEPROM, Flash):** Almacena el firmware y datos que no se borran al interrumpirse la alimentación.
    - **Periféricos:** Dispositivos para entrada y salida de datos (teclados, pantallas, sensores, actuadores).
- **Buses de Comunicación:** Interconectan la CPU con memoria y periféricos.

1.2 Evolución de la tecnología:

Ser conscientes de la evolución desde los primeros procesadores comerciales (como el 4004 de 4 bits de Intel) hasta los actuales (como el Intel i7 de 64 bits) para conocer dónde se enmarcan los microcontroladores de 8 bits en los que se centra esta asignatura.

Ley de Moore: Establece que el número de transistores en un microchip se duplica aproximadamente cada dos años, lo que resulta en un aumento exponencial del rendimiento computacional y una reducción en el costo por transistor.

- Aplicaciones en las que se usan microcontroladores y en las que se usan microprocesadores
- Clasificación de los procesadores según su ámbito de aplicación interna (microcontroladores, microprocesadores, PLD, DSP, procesadores multimedia, procesadores de red, GPU, s*ystem on chip*)
- Clasificación de los procesadores según su arquitectura (nº de bits, repertorio de instrucciones, taxonomía de Flynn, arquitectura de memoria)
  [1.5.1. La Unidad Central de Procesamiento (CPU): Tipos de CPU] [1.5.2. Sistema de Memoria]

1.3 Diseño e Implementación de Sistemas con Microprocesadores:

Pasos del desarrollo:

- **Definición de Requisitos:** Identificación de necesidades y especificaciones del sistema.
- **Diseño del Hardware:** Selección de componentes, diseño de esquemáticos y PCB.
- **Desarrollo del Software:** Programación del microcontrolador, desarrollo de firmware y software de control.
- **Simulación del circuito y el programa:** Mediante software de simulación**.**
- **Pruebas:** Ensamblaje del sistema, pruebas funcionales y de rendimiento.

1.4 Criterios de Selección en Función de la Aplicación:

- **Potencia de Cálculo:** Medida en MIPS (*Millions of Instructions Per Second*) o MFLOPS (*Million Floating-Point Operations Per Second*).
- **Interfaces y Periféricos:** Necesidad de puertos de comunicación (UART, I^2C, SPI), convertidores A/D y D/A, y otros periféricos.
- **Consumo de Energía:** Importante para dispositivos portátiles y alimentados por baterías. Considerar modos de bajo consumo y eficiencia energética.
- **Facilidad de Actualización:** Preferencia por memoria flash o EEPROM para facilitar la reprogramación y actualizaciones de *firmware*.
- **Requisitos Ambientales:** Tolerancia a temperaturas extremas, resistencia a interferencias electromagnéticas, robustez en entornos industriales.

- **Encapsulado del microcontrolador o del microprocesador**

- ***Time to market***

Tema 2: Desarrollo con microcontroladores y Arduino

Fabricantes populares de microcontroladores son: ST microelectronics, National Semiconductors, NXP (antes Philips, ha absorbido a FreeScale (antes Motorola)).
8051: es uno de los microcontroladores (µC) de 8 bits más influyentes de la historia. Fue desarrollado en 1980 por Intel. Fue (y todavía lo es en parte) ampliamente utilizado en sistemas embebidos y aplicaciones de control debido a su simplicidad, versatilidad y popularidad.

2.1 Hardware y Software de Desarrollo:

- **Lenguajes de programación para microcontroladores:** Principalmente ensamblador y C, aunque también se usa Basic, Python, Javascript…

- **Memorias para almacenar el programa en microcontroladores:** ROM, OTP (*One-Time-Programmable*), EPROM, EEPROM, flash

- **Herramientas de Software:**
    - **Compiladores y Ensambladores:** Transforman el código fuente en código máquina ejecutable.

    - **Simuladores:** Herramientas para probar y corregir errores en el código sin necesidad de hardware físico.
    - **Entornos de Desarrollo Integrados (IDE):** Software que combina editor de código, compilador y herramientas de depuración en una sola aplicación.
- **Herramientas de Hardware:**
    - **Tarjetas de Desarrollo:** Placas como Arduino, que facilitan la conexión y programación de microcontroladores.
    - **Programadores ICSP (*In-Circuit Serial Programming*):** Dispositivos de interfaz con el computador que permiten la programación de microcontroladores directamente en el circuito.
    - **Programadores con función de depuración (ejemplo: ICE: *In Circuit Emulator*).**

2.2 Arduino y Otras Plataformas:

- **Componentes de una placa Arduino UNO**: 1 microcontrolador ATmega328P, 1 microcontrolador que hace de pasarela entre el USB y el puerto serie del ATmega328P, reguladores de tensión, cristales de cuarzo…

Shield: Placa de expansión diseñada para conectarse a una plataforma de desarrollo Arduino para añadir funcionalidades adicionales como comunicación, sensores o control de motores.

Bootloader (cargador de arranque) para autoprogramación: El *bootloader* es un programa ubicado en la sección de arranque de la memoria de programa del microcontrolador. En el caso del ATmega328P en Arduino permite la reprogramación del microcontrolador a través de su interfaz USART (la cual está conectada al puerto USB en la placa Arduino UNO), sin necesidad de hardware adicional, es decir, sin necesidad de un programador.
[9.3. Sección de Arranque en la Memoria Flash]

2.3 Arquitectura de Microcontroladores:

Arquitectura AVR de 8 bits:

- **RISC (*Reduced Instruction Set Computing*):** Implementa un conjunto reducido y eficiente de instrucciones.
- **Harvard Architecture:** Separación de memorias de programa y datos, permitiendo accesos simultáneos.

Características del ATmega328P: cantidad de memoria, frecuencia máxima de funcionamiento.

Perro guardián (*Watchdog Timer*): El WDT es un temporizador especial diseñado para mejorar la confiabilidad del sistema. Su función principal es reiniciar automáticamente el microcontrolador (o generar una interrupción) si este se cuelga o queda atrapado en un bucle sin fin. Para ello, el programa debe reiniciar periódicamente el temporizador del perro guardián dentro de un intervalo de tiempo predefinido. Si el programa no lo hace —por ejemplo, debido a un error de software o a un fallo inesperado— el temporizador vence y el microcontrolador se reinicia, lo que permite que el programa comience y recupere el funcionamiento normal sin intervención externa. Este mecanismo es comúnmente utilizado en sistemas embebidos donde la estabilidad y la disponibilidad continua son importantes.
[1.5.5. Perro Guardián (WDT, watchdog timer)] [9.2. El perro guardián (WDT, watchdog timer)]

Osciladores: Los osciladores proporcionan la señal de reloj para el microcontrolador, determinando su velocidad de operación. Pueden ser internos o externos, con configuraciones de baja o alta frecuencia según las necesidades del sistema.
[1.5.3. Oscilador]

Power Supervision: ***Brown-Out Detector* (BOD)**: La supervisión de energía se encarga de monitorear los niveles de voltaje del microcontrolador, asegurando que opere dentro de los parámetros correctos y tomando acciones como resetear el dispositivo en caso de bajadas de tensión muy pronunciadas.
[2.7. Inicialización del Sistema (reset): Brown out]

Debug Wire: La interfaz debugWIRE permite la depuración de microcontroladores usando un solo cable bidireccional. Se utiliza principalmente para controlar la ejecución de instrucciones y requiere la activación del fusible DWEN (*DebugWIRE Enable Fuse*).
[9.5. La interfaz debugWire]

Comparador analógico: El comparador analógico compara dos señales analógicas externas y genera una señal digital basada en la relación entre ellas, indicando si

una señal es mayor o menor que la otra.
[6.2. Comparador Analógico]

Entrada/Salida Digital: Los puertos de entrada/salida permiten al microcontrolador comunicarse con dispositivos externos. Pueden configurarse como entradas o salidas y manejar señales digitales a través de registros específicos.
[1.5.7. Entradas y Salidas Digitales]

Interrupciones: Las interrupciones permiten que el microcontrolador responda a eventos asíncronos, suspendiendo temporalmente la ejecución del programa principal para atender el evento a través de una rutina de atención a interrupción (ISR).
[2.6. Sistema de Interrupciones]

Memoria de Datos: La memoria de datos hace referencia principalmente a la memoria SRAM (2 kB), la cual es un almacenamiento volátil. También se puede incluir en este apartado la EEPROM, que es un espacio no volátil donde el programa del microcontrolador almacena datos si lo necesita para la aplicación. La EEPROM del ATMega328P tiene 1024 bytes y es gestionada a través de registros de entrada/salida específicos.
[2.4. Memoria de Datos]

Memoria de programa: Contiene palabras de 16 bits, en concreto 16 kpalabras, es decir, 32 KiB. Una instrucción suele ocupar una palabra.
[2.3. Memoria de Programa]

Banco de Registros de propósito general: El *General Purpose Register File* contiene 32 registros de propósito general de 8 bits en el núcleo AVR. Algunos se pueden combinar para formar registros de 16 bits. El programa puede acceder a ellos rápidamente mediante instrucciones en un ciclo de reloj, optimizando el rendimiento del microcontrolador.
[2.2.2. El Archivo de Registros]

Vectores de interrupción: Tabla en la memoria de programa donde normalmente se codifican las posiciones de memoria donde se encuentran las ISR. En el ATmega328P estos vectores contienen instrucciones (normalmente instrucciones de salto a donde realmente están las ISR), mientras que en otros procesadores son simplemente punteros a las ISR. En este microcontrolador cada vector ocupa 2 palabras de 16 bits, pudiendo almacenar hasta 2 instrucciones pequeñas. En este

microcontrolador, si la ISR fuera muy corta (2 instrucciones o menos), se podría almacenar directamente en el vector de interrupción.

Registros de entrada/salida (*I/O registers*): Permiten acceder y configurar los módulos hardware que se incluyen en el microcontrolador, por ejemplo, los módulos que configuran las patillas del microcontrolador somo entradas o salidas digitales. A un registro de este tipo se puede acceder a través del programa del microcontrolador, usando una instrucción de entrada/salida (IN/OUT), pero en este microcontrolador, y al contrario de otros procesadores, también se puede acceder con instrucciones de lectura escritura de memoria (LD/ST) cuando se accede a ciertas posiciones del espacio de memoria de datos.
[2.4.1. Espacio de SRAM: Registros I/O y Registros I/O Extendidos]

Registro de estado: Almacena los bits de estado (*flags*) del procesador: desbordamiento, acarreo…
[2.4.1. Espacio de SRAM: Registro de Estado]

Segmentación de cauce: divide el procesamiento en etapas secuenciales, permitiendo que varias instrucciones se procesen simultáneamente y permite ejecutar aproximadamente tantas instrucciones como su frecuencia de reloj.
[2.2.1. Ejecución de Instrucciones]

Modos de direccionamiento: Directo a registro (de propósito general), Directo a registro de entrada/salida…
[3.2. Modos de Direccionamiento]

Tema 3: Actuadores y Periféricos

3.1 Actuadores:

Los motores de corriente continua (C.C.): Convierten energía eléctrica en mecánica mediante la rotación de un eje. Su velocidad puede controlarse variando la tensión de alimentación o mediante técnicas de modulación por ancho de pulso (PWM), y se utilizan comúnmente en aplicaciones que requieren considerable velocidad de giro con un control sencillo. Una aplicación típica sería proporcionar tracción a las ruedas de un microrobot móvil.
[10.5.1. Motores de Corriente Directa]

PWM (*Pulse Width Modulation*): La modulación por ancho de pulso (PWM) permite generar señales cuadradas en las que la señal está en alta (1 lógico) un porcentaje configurable de tiempo del periodo de la señal. Es utilizada para controlar la velocidad de los motores y la intensidad luminosa de las lámparas. Arduino las llama “analógicas” ya que suelen poder hacer la función de una señal analógica, pero en realidad no son analógicas (siempre valen 1 o 0).
[5.5. Modulación por Ancho de Pulso (PWM)]
Servomotores: Los servomotores son actuadores que permiten un control preciso de la posición de su eje a través de una señal PWM. Incluyen un motor de CC, un potenciómetro y un conjunto de engranajes, proporcionando movimientos limitados a un rango de 180 grados.
[10.5.3. Servomotores]

Motores de Pasos: Los motores paso a paso convierten pulsos eléctricos en movimientos angulares discretos, permitiendo control preciso de posición. Pueden ser bipolares o unipolares, requiriendo controladores hardware específicos para su operación.
[10.5.2. Motores Paso a Paso]

Driver: Un driver es un circuito o componente utilizado para controlar dispositivos como motores que requieren mayor corriente de la que un microcontrolador puede proporcionar directamente.

Relé: Un relé es un interruptor electromecánico que permite controlar un circuito de alta potencia con una señal de baja potencia. Se utiliza para aislar y controlar cargas grandes con seguridad, pero su conmutación es lenta en comparación con los transistores o circuitos integrados.

Puente H: Es un circuito (habitualmente integrado) que permite conectar y controlar motores de pasos o de corriente continua al microcontrolador. Un ejemplo es el L293, el cual proporciona la corriente necesaria para la alimentación del motor (es un *driver*) y cuenta con la función *push-pull* en sus salidas.

Fuentes de Alimentación: Las fuentes de alimentación se suelen utilizar para convertir la energía de la red eléctrica a niveles de voltaje utilizables para circuitos electrónicos, normalmente en corriente continua (C.C.). Se pueden construir utilizando transformadores, rectificadores y reguladores de voltaje.

Condensadores usados en la alimentación: De aluminio (llamados electrolíticos) se utilizan para filtrar las variaciones de tensión que no sean muy rápidas (gracias a su gran capacidad en microfaradios). Los cerámicos se utilizan para filtrar las variaciones muy rápidas en la tensión (ruidos) (gracias a su baja impedancia a bajas frecuencias) pero suelen tener menos capacidad. Es por esto que ambos se suelen encontrar en paralelo en el mismo circuito de alimentación, como por ejemplo en la alimentación de la placa Arduino UNO.

Reguladores de Tensión: Los reguladores de voltaje son circuitos (normalmente circuitos integrados) que mantienen una salida de tensión constante a pesar de variaciones en la tensión de entrada o de variaciones en cantidad de carga conectada. Aseguran un nivel de tensión de alimentación estable para los componentes electrónicos. Los más conocidos son los circuitos integrados de 3 patillas 78XX, donde XX es la tensión que dan. Son baratos, tienen el problema de desperdiciar energía a través de calor y solo sirven para bajar la tensión.

Conversores C.C./C.C.: Convierten un nivel de tensión en corriente continua a otro y se usan para alimentación. Unos permiten bajar la tensión, otros subirla, y otros permiten ambas operaciones. No disipan tanto calor como los reguladores de tensión, pero son más caros y aparatosos.

3.2 Recursos internos del ATmega328P / Adaptación e interfaces con sensores:

- **Registros de entrada/salida para acceder o configurar las entradas/salida del propósito general (GPIO): DDRx, PINx, PORTx.** La mayoría de las patillas del microcontrolador se pueden configurar para funcionar como GPIO.

El máximo rango de voltaje que puede recibir una patilla del microcontrolador es entre -0,5 V y la alimentación del microcontrolador + 0,5 V. La máxima corriente que puede dar una patilla es de 40 mA.

- **Correspondencia entre la numeración usada por Arduino para los pines y la usada por Atmel para las patillas del microcontrolador:** por ejemplo, el pin 13 es la patilla PB5.

- **Acceso y configuración de los GPIO mediante código C y código Arduino.**

[2.5. Puertos de Entrada/Salida]

Rebotes e Interruptores: Los rebotes en interruptores son múltiples transiciones rápidas entre estados al presionar o soltar un botón. Esto puede causar rápidas conmutaciones no deseadas en la señal, solucionándose mediante software con técnicas de *debounce*, como el uso de temporizadores para ignorar pulsos repetitivos, o bien con hardware que elimina estos rebotes: filtro RC y un disparador de Schmitt (S*chmitt trigger*) como el circuito integrado 40106.
[10.1. Interruptores y Botones]

Teclado Matricial: Un teclado matricial es una matriz de botones en filas y columnas, permitiendo reducir el número de patillas necesarias para la interfaz. Cada botón se identifica por su fila y columna, y su detección se realiza mediante sondeo secuencial de las filas. Hay biblioteca Arduino para su uso.
[10.2. Teclado Matricial]

Sensores Ópticos: Los sensores ópticos detectan cambios en la luz para medir parámetros como distancia, presencia o color. Algunos funcionan emitiendo luz infrarroja y midiendo la cantidad reflejada (sensores de reflexión, como el CNY70) o interrumpida por un objeto (somo el sensor de herradura o como el sensor de barrera), utilizados en aplicaciones como *encoders* de posición o barreras fotoeléctricas.

Optoacopladores: Para transmitir señales entre 2 circuitos sin que haya una conexión eléctrica entre ellos. Funcionan con tecnología similar a los sensores ópticos de herradura.

Entradas analógicas del microcontrolador: Módulo conversor A/D. Se puede usar con funciones Arduino.
[6.1. Convertidor Analógico a Digital]

Teclado Analógico: Un teclado analógico utiliza una red de resistencias para permitir la lectura de múltiples botones con una sola entrada analógica. Al presionar un botón, cambia la resistencia total y, por tanto, el voltaje leído, permitiendo identificar qué botón fue presionado.

Resistencias: Las resistencias son componentes que limitan el flujo de corriente en un circuito. Se pueden utilizan, por ejemplo, para establecer un valor por defecto en una patilla de entrada del microcontrolador (resistencias *pull-up* o *pull-down*), limitar corrientes (protegiendo componentes) y dividir voltajes en un circuito (divisores de tensión).

Fotoresistencia: Mide el nivel de luz variando su resistencia.

Cómo Programar Señales PWM: Las señales PWM se pueden generar utilizando los módulos temporizadores del microcontrolador, produciendo una señal de onda cuadrada en la que se configura el tiempo en que está en alta y en baja en cada periodo. Este ciclo determina el voltaje promedio de la señal, controlando dispositivos como motores y led.

Aplicaciones PWM: La modulación por ancho de pulso (PWM) se utiliza para controlar la velocidad de motores, la intensidad de luces, posición del eje de un servomotor, y otros dispositivos mediante la variación del porcentaje de tiempo en el que la señal está en alta durante cada periodo de la señal. Proporciona una forma de regular la energía entregada a un dispositivo de forma eficiente.

Temporizadores/contadores (*timers*/*counters*): Son unas de las circuiterías/módulos que más típicamente se encuentran en un microcontrolador. Incluyen un conjunto de registros con autoincremento. Se configura la velocidad a la que se incrementan. Pueden generar señales PWM por hardware (mientras el procesador ejecuta otras instrucciones) en unas patillas específicas del microcontrolador (no en todas), por tanto, se puede configurar el ciclo de trabajo deseado de estas señales. También pueden generar interrupciones periódicas.
[Capítulo 5. Temporizadores]

Acceso a módulos internos (circuitería) del ATmega328P: Se accede a ellos a través de los puertos de entrada salida. En Arduino se puede acceder usando bibliotecas de funciones. Memoria EEPROM, perro guardián, 3 temporizadores (*timers*)…

Funcionamiento de las interrupciones: Existe un bit global de habilitación de interrupciones. También se pueden activar para módulos concretos mediante registros de entrada/salida. Hay funciones Arduino para utilizarlas.

Tema 4: Comunicaciones

Buses de comunicadores:

- **Corta distancia:** I^2C, SPI, One Wire, (USB) …
- **Media distancia:** CAN, RS-232…

USART (puerto serie): En Arduino, el puerto serie del microcontrolador (USART) se usa para recibir nuevos programas, grabándolos en la memoria de programa, y también para enviar cadenas de texto que se pueden visualizar en una consola en el computador o para

recibir en el programa las cadenas que el usuario teclea en la consola. La placa Arduino UNO utiliza un microcontrolador adicional que hace de pasarela entre el puerto USB de la placa Arduino y el puerto serie del microcontrolador ATmega328P, permitiendo así la comunicación entre este microcontrolador principal y el computador. Para esto la placa Arduino debe estar conecta al computador con el cable USB. Hay 2 estándares eléctricos para el puerto serie: el original, es decir, el RS232, (que utiliza unos -/+12 V para codificar las señales, los unos y ceros) y el usado por los microcontroladores, es decir, el TTL (que utiliza 0 V / 5 V).
En general este puerto serie se utiliza para la comunicación serie entre el microcontrolador y otros dispositivos que soportan esta interfaz.
[7.1. Comunicación Serial a través de la USART]

SPI (*Serial Peripheral Interface*): La interfaz de periféricos serie (SPI) permite la comunicación sincrónica de alta velocidad entre un microcontrolador y otros dispositivos. Permite configurar la polaridad y la fase del reloj para compatibilidad con diferentes dispositivos.
[7.2. Comunicación Serial por SPI]

I^2C: El bus I2C (TWI para Atmel) es una interfaz que permite la comunicación entre múltiples dispositivos y que requiere solamente conectar 2 cables (más la masa): las líneas de datos (SDA) y reloj (SCL). Soporta hasta 127 dispositivos direccionables, siendo ideal para sensores y módulos que no requieren una velocidad muy elevada. Los dispositivos que se conectan a este bus usan una interfaz en **colector abierto**.
[Capítulo 8. Comunicación Serial (Parte II)]

Bus 1-Wire: El bus 1-Wire es una interfaz de comunicación que utiliza una única línea (más la masa) para datos y alimentación. Es popular en aplicaciones de identificación y sensores de temperatura, permitiendo la comunicación con múltiples dispositivos en una sola línea.

Bus CAN: El bus CAN es una interfaz de comunicación robusta y eficiente utilizada principalmente en aplicaciones automotrices e industriales. Permite la comunicación entre múltiples microcontroladores y dispositivos sin un controlador maestro, soportando altas velocidades de transmisión y detección de errores.

Bibliotecas de funciones para acceder: USART, I2C, SPI, 1-Wire y CAN.
Ejemplos de aplicaciones de estos buses.

Tema 5: Desarrollo de Aplicaciones

Archivos del proceso de construcción de una aplicación para microcontrolador:

Archivos .map: Un archivo .map es un archivo de texto generado durante el proceso de enlazado de un programa (por el *linker*), que contiene información detallada sobre la disposición de memoria y las direcciones de los símbolos en el código compilado. Es útil para el análisis y la optimización del uso de la memoria.

Archivos .hex: Un archivo .hex es un archivo de texto que contiene el código máquina del programa y de sus datos codificado en un formato hexadecimal. Es utilizado en el momento de grabar el programa en la memoria del microcontrolador (o en otros dispositivos embebidos como memorias), generalmente a través de un dispositivo programador o de un *bootloader*.

Programación del microcontrolador: Cuando se programa una placa Arduino mediante el *bootloader*, se puede utilizar el programa avrdude desde el PC. avrdude es ejecutado internamente por el IDE de Arduino al programar.

Bits de configuración (*fuse bits*): 3 bytes. Se almacenan de forma no volátil y configuran opciones que el programa del microcontrolador no se suele cambiar.
Bytes de firme (*signature bits*): Identifican el modelo de microcontrolador.
Otros bytes programables: *lock bits*, *calibration byte*.
[9.4. Bits de Configuración y Seguridad]

Tema 6: Microcontroladores avanzados

El microcontrolador ATmega2560: Diferencias con respecto al ATmega328P: Mayor cantidad de memoria, mayor número de patillas, mayor cantidad de módulos de los que dispone.