Programación de Sistemas Embebidos

ADC

Programa analítico de la asignatura

UNIDAD 1: Arquitectura de Sistemas Embebidos: Memorias: RAM, ROM, PROM, EPROM, EEPROM y Flash.

UNIDAD 3: **E/S de bajo nivel:**
ADC: conversor analógico a digital.

UNIDAD 4: **Programación de bajo nivel:**
Lenguaje C. Programación sobre hardware sin sistema (baremetal).

Temario

- Repaso sobre MEMORIAS
- Señales y adquisición de datos
- Sensores analógicos – divisor de tensión
- Metodología y Arquitectura de un driver ADC

Repaso Memorias: RAM, ROM, PROM, EPROM, EEPROM, FLASH

Memory Type	Category	Erasure	Write Mechanism	Volatility
Random-access memory (RAM)	Read-write memory	Electrically, byte-level	Electrically	Volatile
Read-only memory (ROM)	Read-only memory	Not possible	Masks	Nonvolatile
Programmable ROM (PROM)			Electrically	
Erasable PROM (EPROM)	Read-mostly memory	UV light, chip-level		
Electrically Erasable PROM (EEPROM)		Electrically, byte-level		
Flash memory		Electrically, block-level		

Repaso Memorias: RAM, ROM, PROM, EPROM, EEPROM, FLASH

Responda:

6. Match the following with the type of memory they are describing: RAM, ROM, PROM, EPROM, EEPROM, and FLASH.
***Hint**: You might want to read all descriptions first, before starting with the matching.*

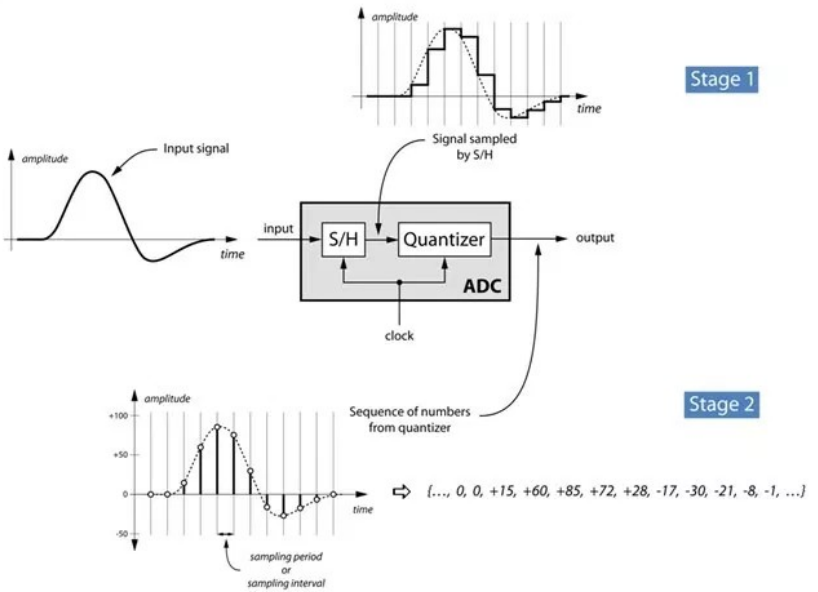
1. This memory can be programmed by the user instead of at the factory, and is read-only
2. This memory is not only nonvolatile, but also can be erased by an electrical signal 1 byte at a time
3. The contents of this memory are programmed one time when manufactured and are nonvolatile
4. This memory can be read from and written to, and is used by microcontrollers for variable data storage
5. This memory is quite similar to the one described in 2. But allows for faster data access in blocks
6. The contents of this memory will persist when the power is removed, but only UV can erase them

ADC: Adquisición de datos

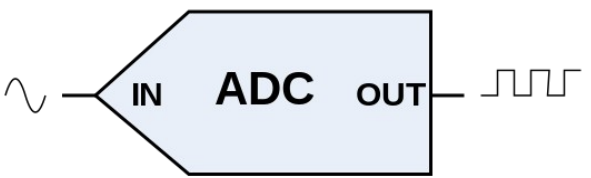
Objetivo: obtener mediciones de variables o fenómenos físicos de interés.

En la naturaleza, el conjunto de señales que percibimos tienen una variación continua. Para poder obtener una señal de cierto tipo suele utilizarse algún sensor (transductor):

- Dispositivo que “capta” magnitudes físicas. Debe tener alguna propiedad sensible a la magnitud que se desea medir (entrada)
- El sensor convierte la magnitud física a una señal eléctrica (transductor).
- La salida del sensor es una señal eléctrica cuyos valores están relacionados con la magnitud física que se quiere medir (señal analógica).



Una señal analógica (muy estudiada/utilizada en ing. eléctrica y electrónica) es una señal continua que varía en el tiempo (matemáticamente puede tomar infinitos valores en un rango continuo), en donde la parte variante en el tiempo es una representación de otra señal variante en el tiempo (es **análoga**).

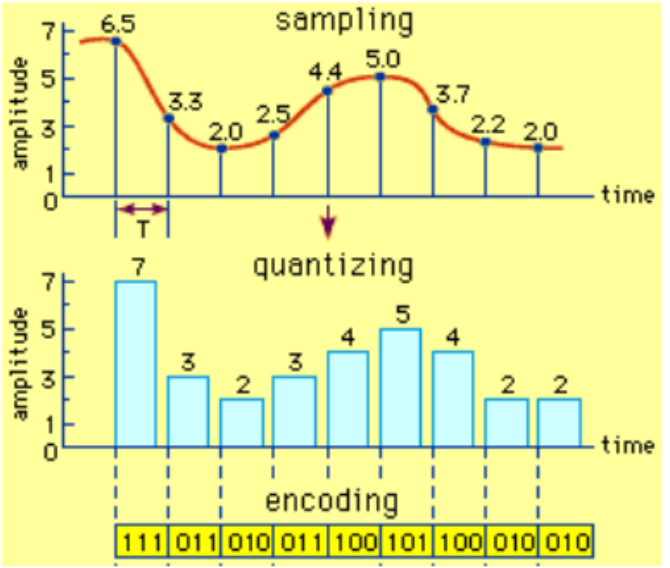


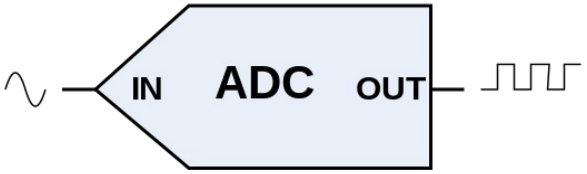
Ejemplo de señal analógica: una señal de audio; porque está compuesta por un voltage que varía con valores contínuos en el tiempo, y representa a la presión que ejerce el sonido. El micrófono es un sensor que puede ser utilizado para capturar el sonido (fenómeno físico) y convertirlo en una señal de audio (señal analógica).

ADC: Adquisición de datos

Una señal analógica puede tomar cualquier valor real.

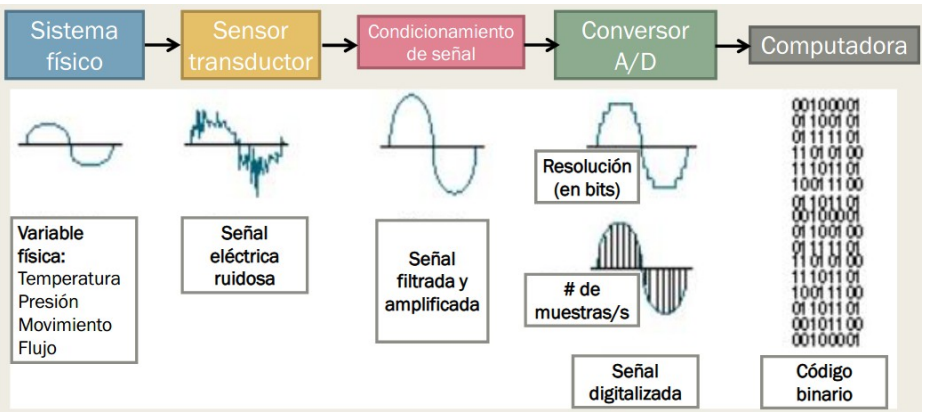
Una señal digital toma valores discretos de un conjunto predeterminado.





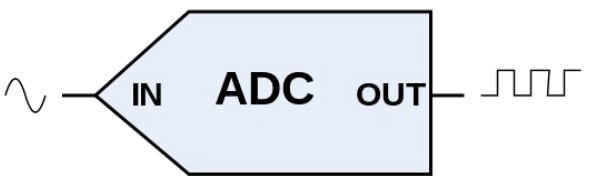
ADC: Adquisición de datos

Al convertir una señal analógica a digital se deben definir dos variables:



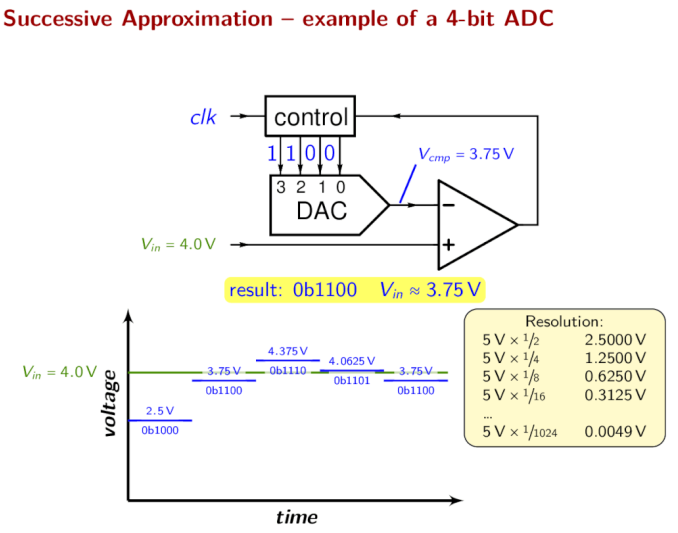
- Resolución: Cantidad de bits que representarán digitalmente los niveles de la señal analógica de entrada. Ej. 10 bits: 0 a 1023.

- Frecuencia: la frecuencia de muestreo determina cada cuánto se toma una muestra (nro de muestras por segundos).



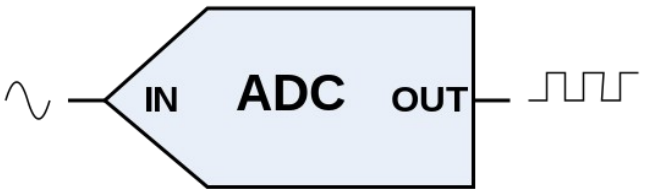
ADC: Analog to Digital Converter

Varios tipos de microcontroladores AVR contienen un conversor de señales analógicas a digital (ADC).



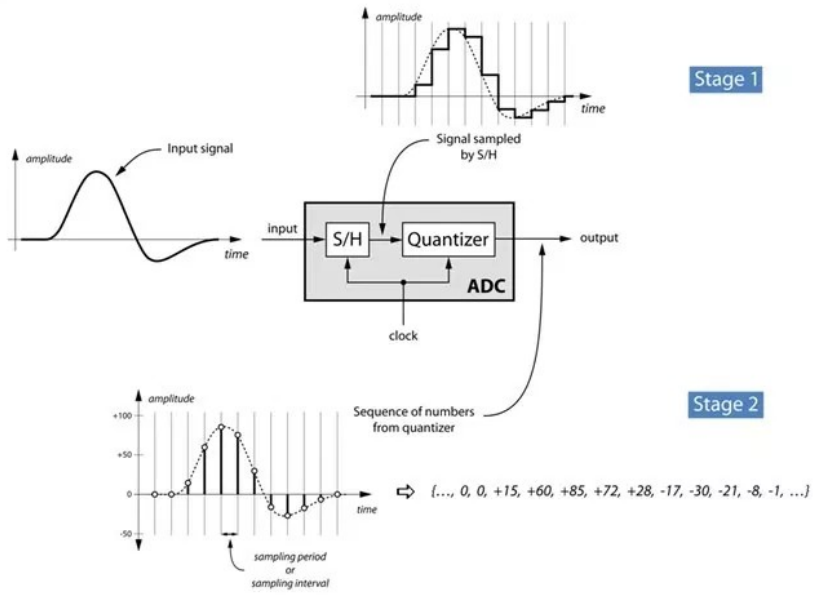
Animación de un con ADC Aproximación sucesiva

- Internamente utilizan el mecanismo de aproximación sucesiva.



ADC: Analog to Digital Converter

Varios tipos de microcontroladores AVR contienen un conversor de señales analógicas a digital (ADC).



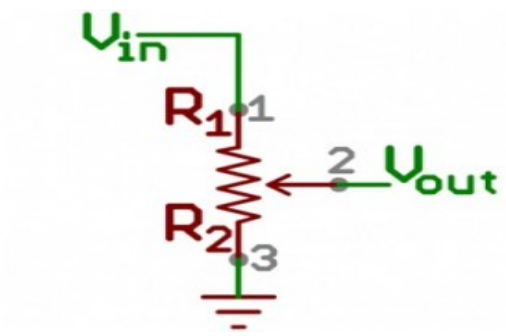
Sensores analógicos

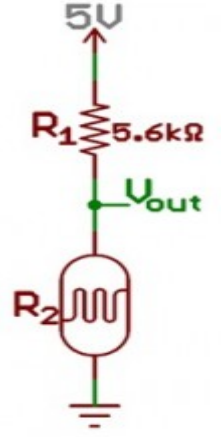
- Temperatura
- Humedad
- Luz
- Presión
- Niveles de líquidos
- Magnéticos
- Alcoholímetros
- Corriente
- Acústico
- Acidez
- Proximidad
- Caudal
- Aceleración
- Velocidad
- Potenciómetro
- Magnetómetro
- Acelerómetro

Divisor de tensión (voltage)

$$V_f = \frac{Z_i}{\sum Z_n} \cdot V_i$$

$$V_{\text{out}} = \frac{R_2}{R_1 + R_2} \cdot V_{\text{in}}$$

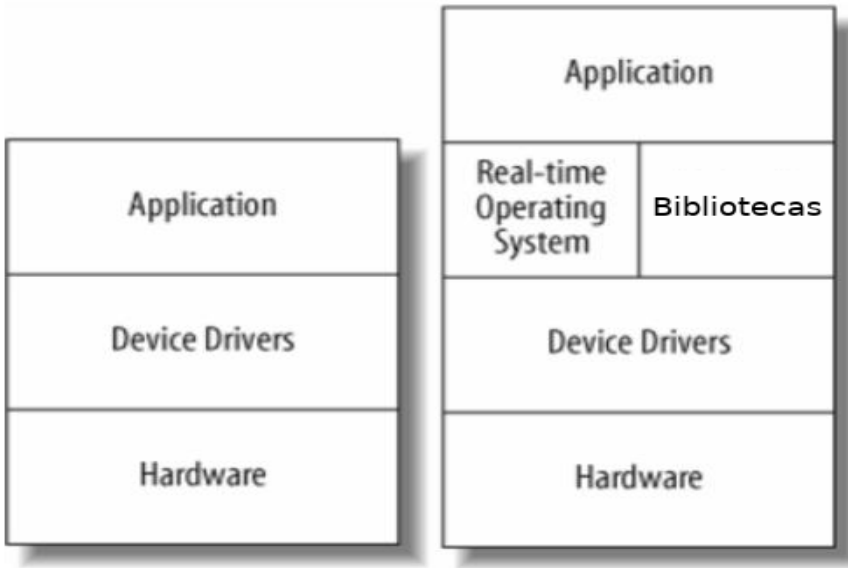




ADC: Metodología para el desarrollo del driver

Metodología para el desarrollo del driver (la misma que para el UART):

1) Observar el diagrama de bloques general del AVR, encontrar a qué bus se encuentra el periférico (ADC).
2) Realizar una lectura de funcionamiento general del dispositivo de E/S. Detallar sus características.
3) Observar el diagrama de bloques del periférico (ADC). Enumerar los registros de E/S conectados al bus.
4) Obtener una descripción de cada registro de E/S del periférico.
5) Definir una estructura en C que represente los registros de estado, control y datos del periférico
6) Crear la rutina de inicialización (init)
7) Escribir la rutina de ENTRADA
   - Con E/S programada
   - Con E/S via interrupciones (rutina de atención de interrupciones)
8) Escribir una aplicación que verifique (utilice) el driver de manera básica (main.c)

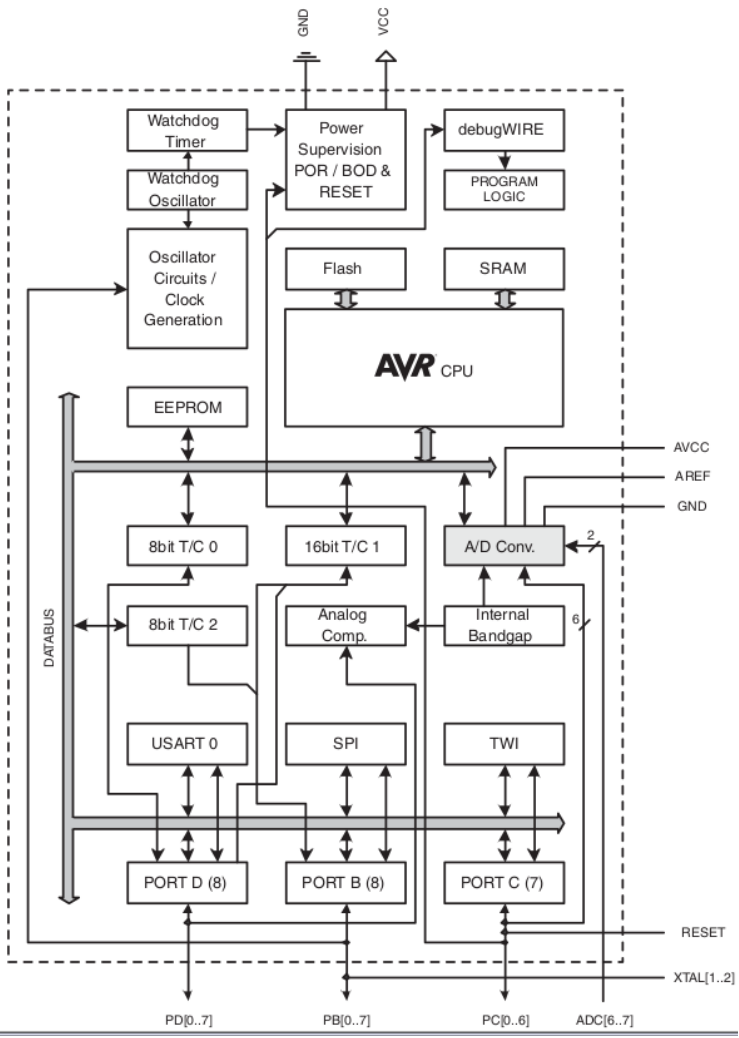


- IDEA CLAVE: Ocultar el Hardware completamente (main no debería conocer detalles de los registros ni del funcionamiento interno del periférico). Permite portar la aplicación a otras plataformas.

ADC: Arquitectura del hardware en AVR

Utilizado mayormente para la medición (lectura) de sensores

Figure 2-1. Block Diagram

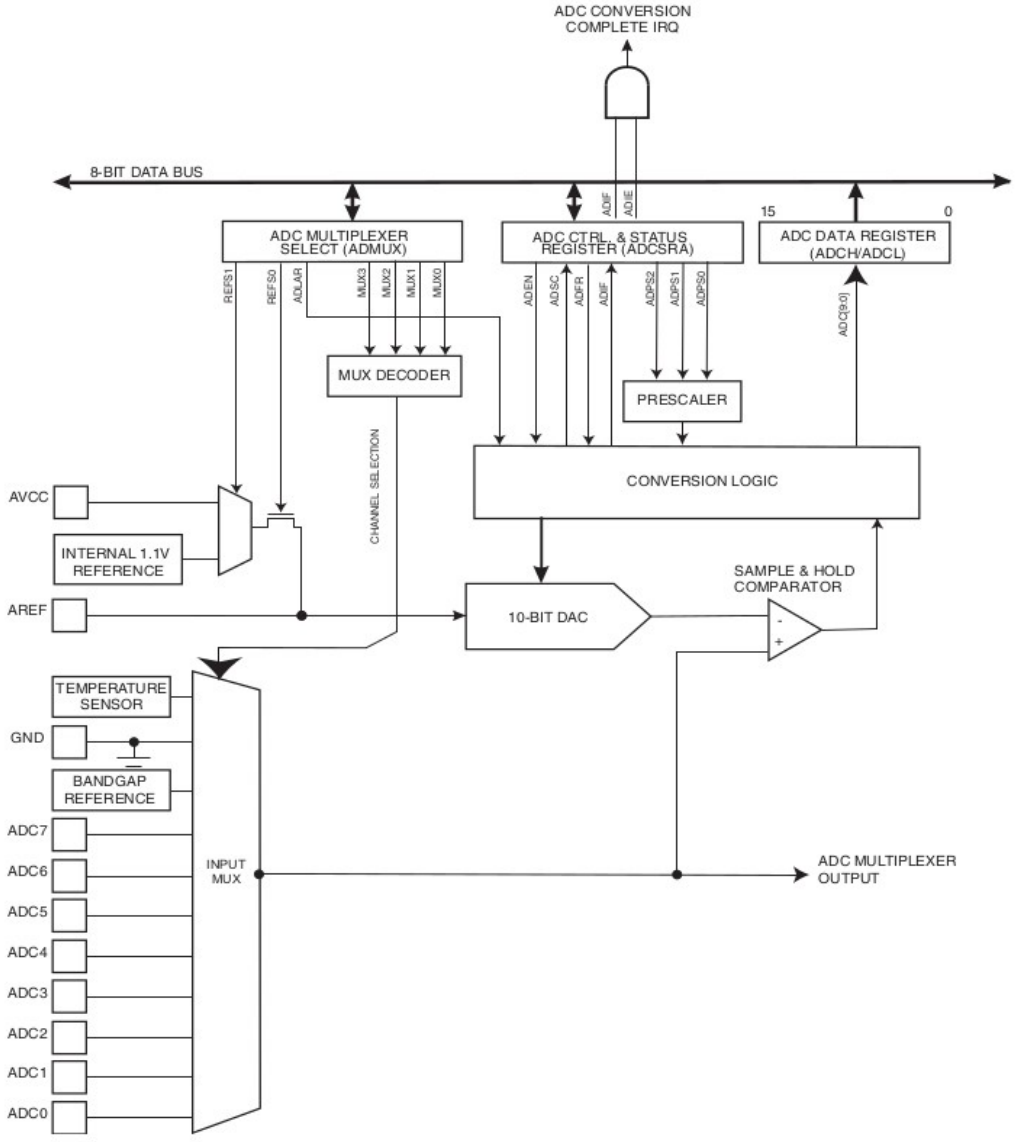


ADC en AVR: características

- reloj 16Mhz = 16 millones de ciclos por segundo
- 1 segundo (s) = 1.000.000.000 nanosegundos (ns)
- 50khz-200khz (reloj necesario)
- Cada 80 ciclos de reloj MCU => 1 ciclo de reloj del ADC => Prescalar 128
- 1 conversión toma 13 ciclos de reloj ADC (excepción: 25 ciclos toma la primera conversión)
- 1 ciclo de reloj MCU es 62 ns. 4960 ns un ciclo de reloj ADC
- 4960 ns * 13 = 64480 ns

- 10-bit Resolution
- 0.5 LSB Integral Non-linearity
- ± 2 LSB Absolute Accuracy
- 13 - 260µs Conversion Time
- Up to 76.9kSPS (Up to 15kSPS at Maximum
- 6 Multiplexed Single Ended Input Channels
- 2 Additional Multiplexed Single Ended Input
- Temperature Sensor Input Channel
- Optional Left Adjustment for ADC Result Re
- 0 - V_{CC} ADC Input Voltage Range
- Selectable 1.1V ADC Reference Voltage
- Free Running or Single Conversion Mode
- Interrupt on ADC Conversion Complete
- Sleep Mode Noise Canceler

Figure 24-1. Analog to Digital Converter Block Schematic Operation,



ADC en AVR: tiempos

Table 24-1. ADC Conversion Time

Condition	Sample & Hold (Cycles from Start of Conversion)	Conversion Time (Cycles)
First conversion	13.5	25
Normal conversions, single ended	1.5	13
Auto Triggered conversions	2	13.5

ADC: Registros

- ADMUX. Selector de entrada (multiplexor)
- ADCSRA. Status y Control
- ADC(H:L): 10bits/8bits. Datos

ADC: Registros

- Registro ADMUX. Selector de entrada (multiplexor)

24.9.1 ADMUX – ADC Multiplexer Selection Register

Bit	7	6	5	4	3	2	1	0	
(0x7C)	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – REFS[1:0]: Reference Selection Bits**

These bits select the voltage reference for the ADC, as shown in Table 24-3. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set). The internal voltage reference options may not be used if an external reference voltage is being applied to the AREF pin.

Table 24-3. Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal V_{ref} turned off
0	1	AV_{CC} with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 1.1V Voltage Reference with external capacitor at AREF pin

- **Bit 5 – ADLAR: ADC Left Adjust Result**

Table 24-4. Input Channel Selections

MUX3...0	Single Ended Input
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5
0110	ADC6
0111	ADC7
1000	ADC8(1)
1001	(reserved)
1010	(reserved)
1011	(reserved)
1100	(reserved)
1101	(reserved)
1110	1.1V (V_{BG})
1111	0V (GND)

ADC: Registros

- Registros: ADCSRA. Status y Control

24.9.2 ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ADEN: ADC Enable**

Writing this bit to one enables the ADC. By writing it to zero, the ADC is turned off. Turning the ADC off while a conversion is in progress, will terminate this conversion.

- **Bit 6 – ADSC: ADC Start Conversion**

In Single Conversion mode, write this bit to one to start each conversion. In Free Running mode, write this bit to one to start the first conversion. The first conversion after ADSC has been written after the ADC has been enabled, or if ADSC is written at the same time as the ADC is enabled, will take 25 ADC clock cycles instead of the normal 13. This first conversion performs initialization of the ADC.

ADSC will read as one as long as a conversion is in progress. When the conversion is complete, it returns to zero. Writing zero to this bit has no effect.

ADC: Registros

- Registros: ADCSRA. Status y Control

  - **Bit 5 – ADATE: ADC Auto Trigger Enable**

  When this bit is written to one, Auto Triggering of the ADC is enabled. The ADC will start a conversion on a positive edge of the selected trigger signal. The trigger source is selected by setting the ADC Trigger Select bits, ADTS in ADCSRB.

  - **Bit 4 – ADIF: ADC Interrupt Flag**

  This bit is set when an ADC conversion completes and the Data Registers are updated. The ADC Conversion Complete Interrupt is executed if the ADIE bit and the I-bit in SREG are set. ADIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ADIF is cleared by writing a logical one to the flag. Beware that if doing a Read-Modify-Write on ADCSRA, a pending interrupt can be disabled. This also applies if the SBI and CBI instructions are used.

  - **Bit 3 – ADIE: ADC Interrupt Enable**

  When this bit is written to one and the I-bit in SREG is set, the ADC Conversion Complete Interrupt is activated.

  - **Bits 2:0 – ADPS[2:0]: ADC Prescaler Select Bits**

  These bits determine the division factor between the system clock frequency and the input clock to the ADC.

Table 24-5. ADC Prescaler Selections

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

ADC: Arquitectura del hardware en AVR

- Registros: ADC(H:L): 10bits/8bits. Datos

24.9.3 ADCL and ADCH – The ADC Data Register

24.9.3.1 ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
(0x79)	–	–	–	–	–	–	ADC9	ADC8	ADCH
(0x78)	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

24.9.3.2 ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
(0x79)	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
(0x78)	ADC1	ADC0	–	–	–	–	–	–	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

When an ADC conversion is complete, the result is found in these two registers.

When ADCL is read, the ADC Data Register is not updated until ADCH is read. Consequently, if the result is left adjusted and no more than 8-bit precision is required, it is sufficient to read ADCH. Otherwise, ADCL must be read first, then ADCH.

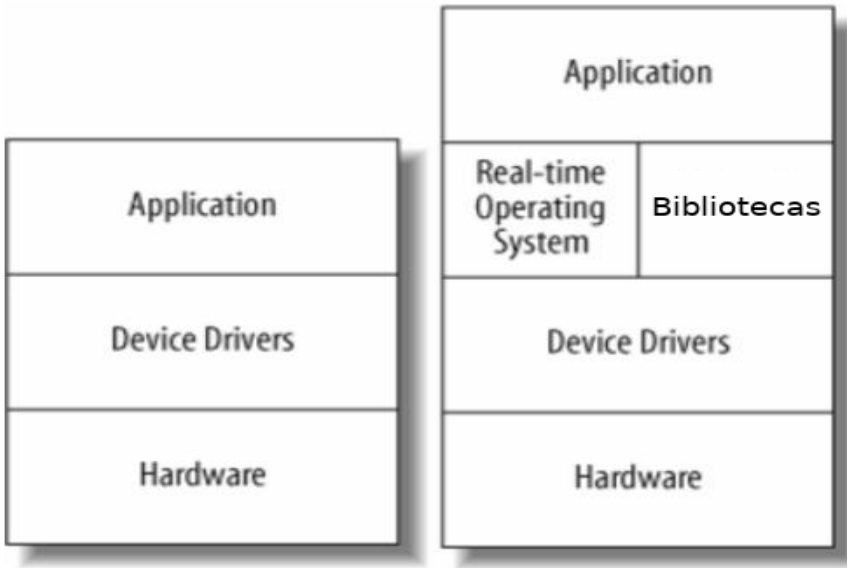
The ADLAR bit in ADMUX, and the MUXn bits in ADMUX affect the way the result is read from the registers. If ADLAR is set, the result is left adjusted. If ADLAR is cleared (default), the result is right adjusted.

ADC: Arquitectura de un driver para sistemas embebidos (software)

adc.c
adc.h

- Definir una estructura que represente los registros de estado, control y datos del periférico

```
typedef struct
{
    uint8_t adcl;
    uint8_t adch;
    uint8_t adcsra; /* ADC Control and Status Register A */
    uint8_t adcsrb;
    uint8_t admux; /* ADC Multiplexer Selection Register */
    uint8_t reserved;
    uint8_t didr0;


} volatile adc_t;


/* puntero a la estructura de los registros del periferico */
volatile adc * adc = (adc_t *) (0x78);
```



ADC: Arquitectura de un driver para sistemas embebidos (software)

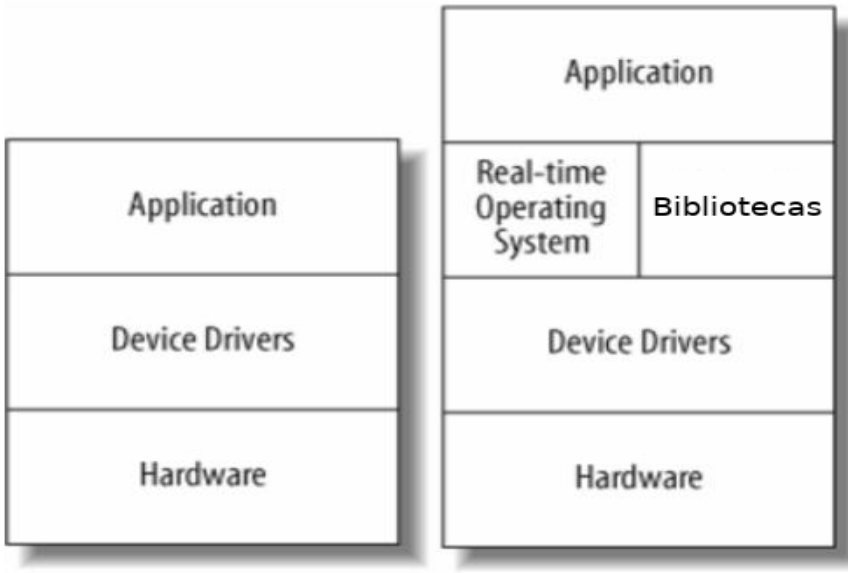
adc.c

adc.h

- Crear la rutina de inicialización (init)

  Ejemplo: Establecer los registros de control para usar la referencia interna y habilitar el periférico. Establecer el prescalar (divisor).

```
void adc_init() {

        /* Escribir una rutina de inicializacion */

        /* Configurar los registros ADMUX y ADCSRA para utilizar el voltage de referencia interno y encender (habilitar) el periférico */

        /* Establecer el prescalar (divisor) */

        adc->    =       ;
       ....

}
```

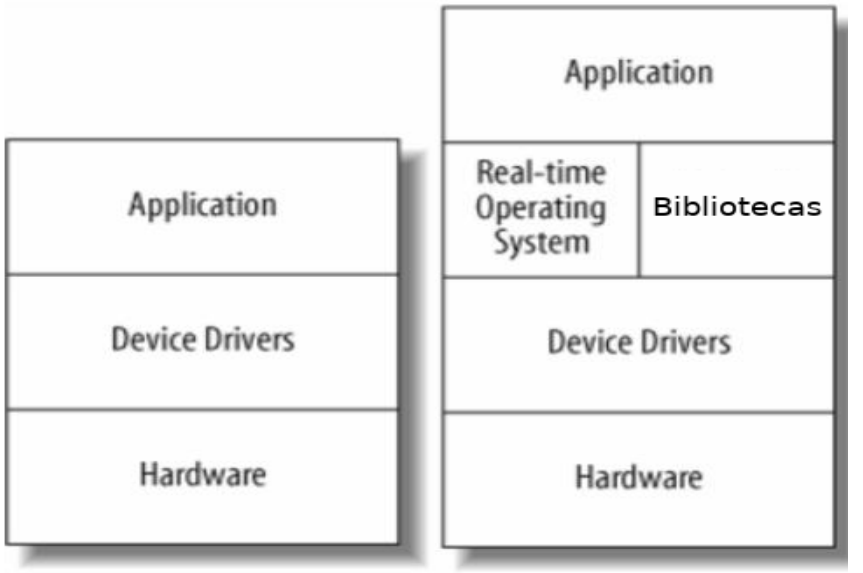


UART: Arquitectura de un driver para sistemas embebidos (software)

adc.c
adc.h

- Escribir la rutina de ENTRADA
  - Con E/S programada
  - Con E/S via interrupciones (rutina de atención de interrupciones)

```
/* devuelve un valor de 10bits de la conversión ADC entrada input */
int adc_get(char input)
{
    int val;
   /* 1. establecer el multiplexor a la entrada input */

   /* 2. iniciar una conversion ADC */

    /* 3. completar con E/S programada */
    while(!((adc-> ...) & (... )));

    /* 4. devolver el valor del registro de datos del ADC (Low y High)*/
    /* IMPORTANTE: hay que leer el registro L antes del H !!! */
     …..
     return val;

}
```

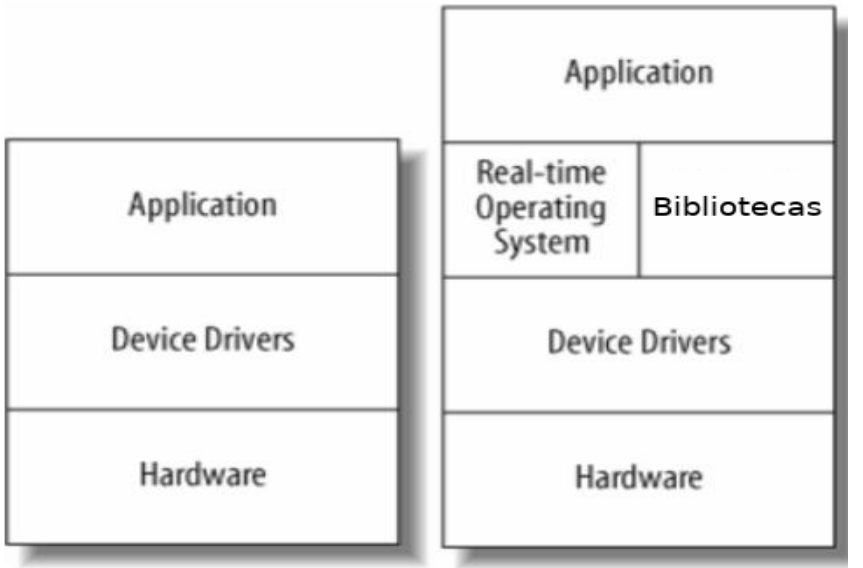


ADC: Arquitectura de un driver para sistemas embebidos (software)

main.c

- Escribir una aplicación que utilice el driver de manera básica (main.c)

```
#include “adc.h”

main()
{
    int val;

    adc_init()
    val = adc_read(2);  /* obtener una conversión ADC del pin de entrada ADC 2 */

    /* realizar alguna acción con val */

}
```



Bibliografía

- Programming Embedded SystemS in C and C++, Michael Barr, O'Reilly, 1999, ISBN: 1-56593-354-5
- Designing Embedded Hardware, John Catsoulis, O'Reilly, 2003, ISBN: 1-596-00362-5
- Hoja de datos atmega328p, Atmel.
- Esquemático arduino pro mini
- Apunte: Periféricos