

Nombre: _____ Grupo: _____

Dr. Enrique García Trinidad
Universidad Tecnológica Fidel Velázquez
<https://sites.google.com/site/mysillyrobots>
phd.enrique.garcia@ieee.org

Práctica 6

Convertidor Analógico Digital

6.1. Material

El material enlistado es necesario para la realización de la práctica 6.

Ct	Dispositivo	Descripción	Eti.
1	ATmega328P-PU	Microcontrolador AVR RISC 8-bit 20Mhz	U1
1	Regulador L7805CV	Regulador de voltaje 5V 1 A	IC1
1	Capacitor cerámico de $0.1\mu\text{F}$ 50V	Código: 104	C1
1	Capacitor electrolítico de $470\mu\text{F}$ 25V	Tolerancia $\pm 20\%$	C2
1	Capacitor electrolítico de $220\mu\text{F}$ 25V	Tolerancia $\pm 20\%$	C3
3	Resistencia de 330Ω 1/4W	Código: Naranja, naranja, café, oro	R1, R2, R3
1	Resistencia de $10\text{k}\Omega$ 1/4W	Código: Café, negro, naranja, oro	R4
1	Led 5mm difuso	Color rojo	LED1
1	Push button (Microswitch)	Tipo push, 4 o 2 terminales	S1
1	Display de 7 segmentos	Ánodo común	D1
1	Potenciómetro de 10k	Tipo preset horizontal de 10 mm	
2	Metro de alambre para protoboard		
1	Protoboard		
1	Grabador Usbasp	Grabador microcontroladores AVR 8-bit	J3
1	Fuente de alimentación de 12V 2A	Eliminador de voltaje	J2
1	Computadora con puerto USB		

6.2. Conexión de los componentes

- Conecte el diagrama de la práctica 6 de acuerdo a la Figura 6.1:

6.3. Ejercicio 1

- Inicie el software Codevision AVR. Cree un nuevo proyecto dando click en el menú [New>Project](#).
- Cuando el software pregunte si queremos usar el asistente [CodeWizardAVR](#) le indicamos que [Si](#).

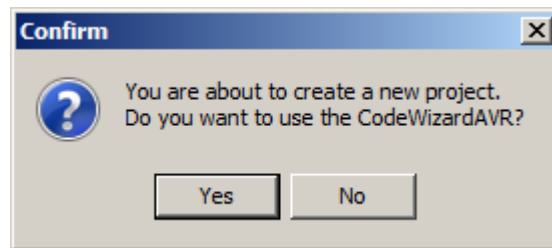


Figura 6.2: Elegir [Yes](#).

- Seleccionar en el tipo de microcontrolador a utilizar [AT90](#), [ATtiny](#), [ATmega](#)

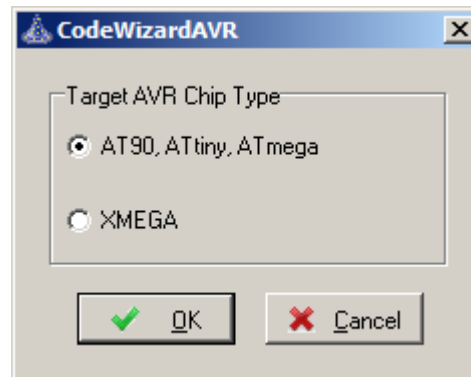


Figura 6.3: Al finalizar seleccionar [OK](#)

- En el asistente, en la ficha [Chip](#) dejar la configuración como sigue:



Figura 6.1: Conexión de la práctica 6.

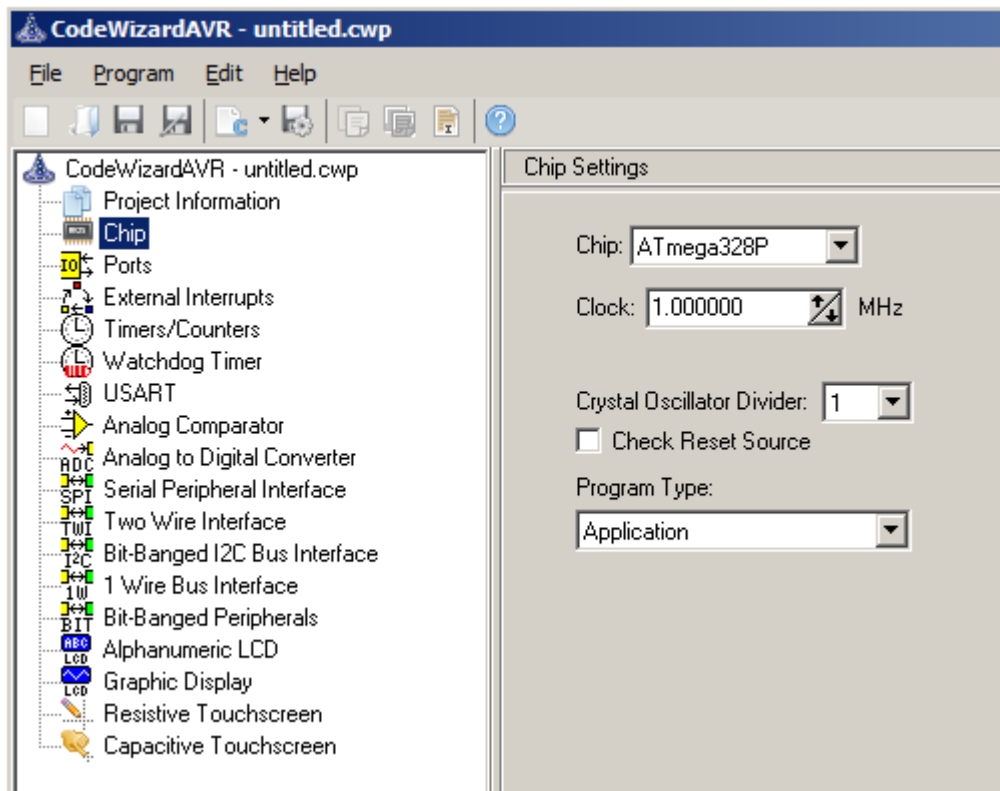


Figura 6.4: Configuración ficha **Chip**

- En el asistente, en la ficha **Ports**, en el Puerto C dejar la configuración como sigue:

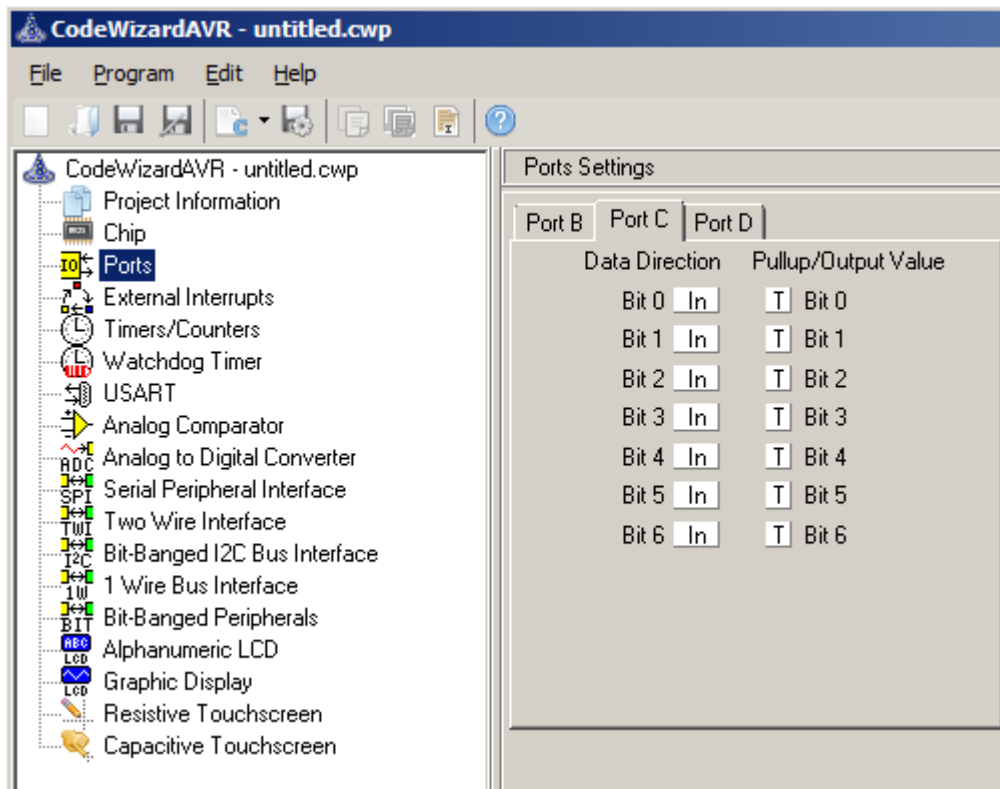


Figura 6.5: Configuración E/S Port C

- En el asistente, en la ficha **Ports**, en el Puerto D dejar la configuración como sigue:

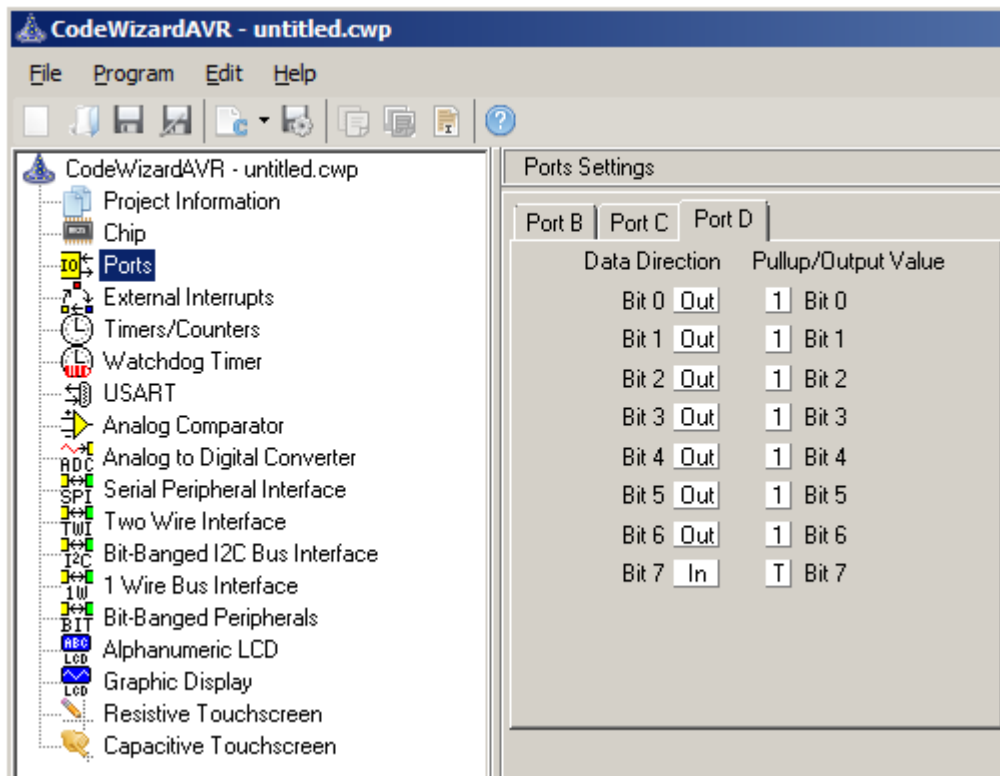


Figura 6.6: Configuración E/S Port D

- En el asistente, en la ficha [Analog to Digital Converter](#), dejar la configuración como sigue:

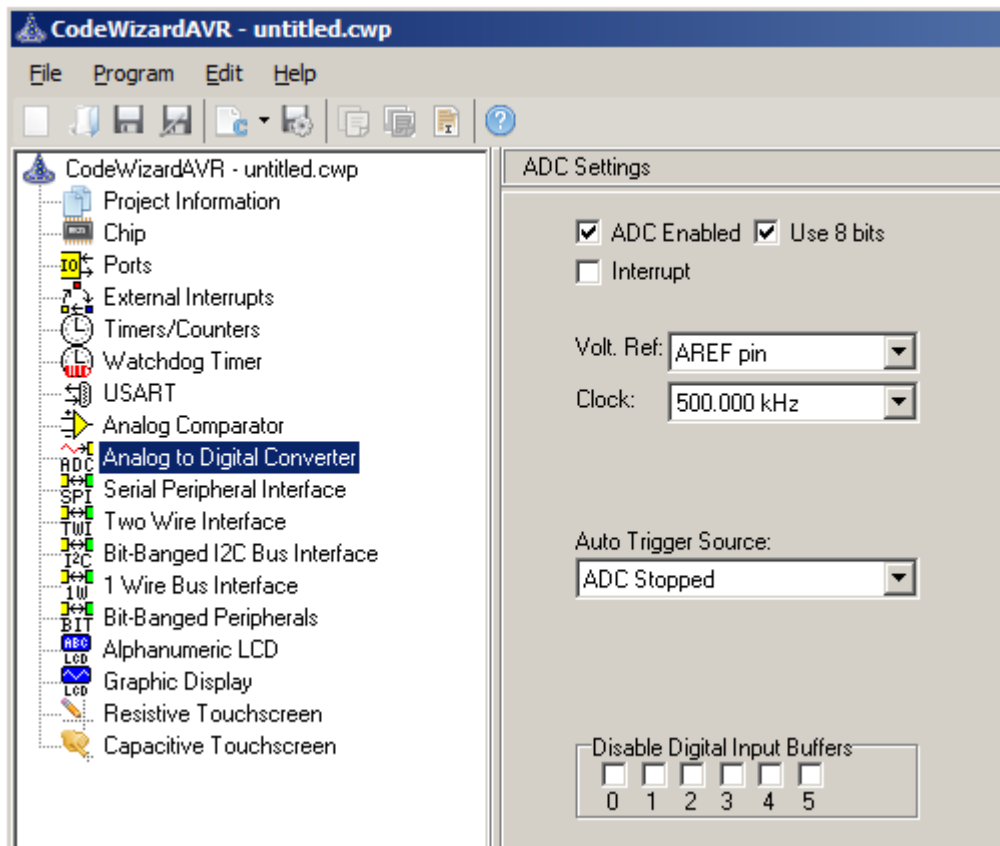


Figura 6.7: Configuración del ADC

- Posteriormente debemos de generar el proyecto y el código, eligiendo **Program>Generate**, **Save and Exit**. Guarde el asistente, código y proyecto con el mismo nombre **prac06**.

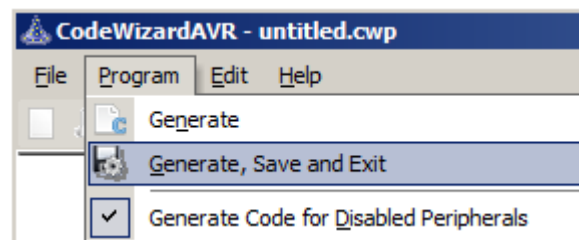


Figura 6.8: Generar el proyecto y el código.

- Mucho del código generado por el asistente no se utiliza en esta práctica. Modifique el código borrando algunas líneas para que sólo quede lo necesario para usar el ADC. El código resultante se muestra a continuación:

```
#include <mega328p.h>
#include <delay.h>

// Voltage Reference: AREF pin
#define ADC_VREF_TYPE ((0<<REFS1) | (0<<REFS0) | (1<<ADLAR))

// Read the 8 most significant bits
// of the AD conversion result
unsigned char read_adc(unsigned char adc_input)
{
    ADMUX = adc_input | ADC_VREF_TYPE;
    // Delay needed for the stabilization of the ADC input voltage
    delay_us(10);
    // Start the AD conversion
    ADCSRA |= (1 << ADSC);
    // Wait for the AD conversion to complete
    while ((ADCSRA & (1 << ADIF)) == 0);
    ADCSRA |= (1 << ADIF);
    return ADCH;
}

void main(void)
{
    // ADC initialization
    // ADC Clock frequency: 500.000 kHz
    // ADC Voltage Reference: AREF pin
    // ADC Auto Trigger Source: ADC Stopped
    // Only the 8 most significant bits of
    // the AD conversion result are used
    // Digital input buffers on ADC0: 0n, ADC1: 0n, ADC2: 0n, ADC3
    // : 0n
    // ADC4: 0n, ADC5: 0n
    DIDR0 = (0 << ADC5D) | (0 << ADC4D) | (0 << ADC3D) | (0 <<
        ADC2D) | (0 << ADC1D) | (0 << ADC0D);
    ADMUX = ADC_VREF_TYPE;
    ADCSRA = (1 << ADEN) | (0 << ADSC) | (0 << ADIF) | (0 << ADIF
        ) | (0 << ADIF) | (0 << ADIF) | (0 << ADIF) | (1 << ADIF);
    ADCSRB = (0 << ADTS2) | (0 << ADTS1) | (0 << ADTS0);

    while (1)
```

```
{
}
}
```

- Compile el proyecto eligiendo desde el menú **Project>Build All**. Una compilación correcta arrojará la información *No errors, No warnings*
- Modifique de nuevo el código como se muestra a continuación:

```
#include <mega328p.h>
#include <delay.h>
#define xtal 1000000L

/*
  Codificación de cada dígito para display de 7 segmentos
  Dig_0=0x40, Dig_1=0x79, Dig_2=0x24, Dig_3=0x30, Dig_4=0x19
  Dig_5=0x12, Dig_6=0x02, Dig_7=0x78, Dig_8=0x00, Dig_9=0x18
*/
const unsigned char tabla7segmentos[10] = {0x40, 0x79, 0x24, 0
    x30, 0x19, 0x12, 0x02, 0x78, 0x00, 0x18};

// Voltage Reference: AREF pin
#define ADC_VREF_TYPE ((0<<REFS1) | (0<<REFS0) | (1<<ADLAR))

// Read the 8 most significant bits
// of the AD conversion result
unsigned char read_adc(unsigned char adc_input)
{
    ADMUX = adc_input | ADC_VREF_TYPE;
    // Delay needed for the stabilization of the ADC input voltage
    delay_us(10);
    // Start the AD conversion
    ADCSRA |= (1 << ADSC);
    // Wait for the AD conversion to complete
    while ((ADCSRA & (1 << ADIF)) == 0);
    ADCSRA |= (1 << ADIF);
    return ADCH;
}

void main(void)
{
    unsigned char lectura;

    /* Configuración E/S
       Puerto D:
```

```

    PD7=Entrada, Pull-down
    PD6=Salida, Estado Inicial= Vcc
    PD5=Salida, Estado Inicial= Vcc
    PD4=Salida, Estado Inicial= Vcc
    PD3=Salida, Estado Inicial= Vcc
    PD2=Salida, Estado Inicial= Vcc
    PD1=Salida, Estado Inicial= Vcc
    PD0=Salida, Estado Inicial= Vcc
    Puerto C:
    PC6 a PC0= Entrada, Tri-estado
*/

DDRD = 0x7F;
PORTD = 0x7F;

DDRC = 0x00;
PORTC = 0x00;

// ADC initialization
// ADC Clock frequency: 500.000 kHz
// ADC Voltage Reference: AREF pin
// ADC Auto Trigger Source: ADC Stopped
// Only the 8 most significant bits of
// the AD conversion result are used
// Digital input buffers on ADC0: On, ADC1: On, ADC2: On, ADC3
: On
// ADC4: On, ADC5: On
DIDR0 = (0 << ADC5D) | (0 << ADC4D) | (0 << ADC3D) | (0 <<
    ADC2D) | (0 << ADC1D) | (0 << ADC0D);
ADMUX = ADC_VREF_TYPE;
ADCSRA = (1 << ADEN) | (0 << ADSC) | (0 << ADIF) | (0 << ADIF
    ) | (0 << ADIE) | (0 << ADPS2) | (0 << ADPS1) | (1 << ADPS0
    );
ADCSRB = (0 << ADTS2) | (0 << ADTS1) | (0 << ADTS0);

while (1)
{
    lectura = read_adc(0);
    if((lectura >= 0) && (lectura <= 25))
    {
        PORTD = tabla7segmentos[0];
    }
    else if((lectura >= 26) && (lectura <= 50))
    {
        PORTD = tabla7segmentos[1];
    }
}

```

```

else if((lectura >= 51) && (lectura <= 75))
{
    PORTD = tabla7segmentos[2];
}
else if((lectura >= 76) && (lectura <= 100))
{
    PORTD = tabla7segmentos[3];
}
else if((lectura >= 101) && (lectura <= 125))
{
    PORTD = tabla7segmentos[4];
}
else if((lectura >= 126) && (lectura <= 150))
{
    PORTD = tabla7segmentos[5];
}
else if((lectura >= 151) && (lectura <= 175))
{
    PORTD = tabla7segmentos[6];
}
else if((lectura >= 176) && (lectura <= 200))
{
    PORTD = tabla7segmentos[7];
}
else if((lectura >= 201) && (lectura <= 225))
{
    PORTD = tabla7segmentos[8];
}
else if((lectura >= 226) && (lectura <= 250))
{
    PORTD = tabla7segmentos[9];
}
else
{
    PORTD = tabla7segmentos[9];
}
}
}

```

- Compile el proyecto eligiendo desde el menú **Project>Build All**. Una compilación correcta arrojará la información *No errors, No warnings*
- Descargue el archivo que resulto de la compilación **prac06.hex** en la memoria Flash del microcontrolador.

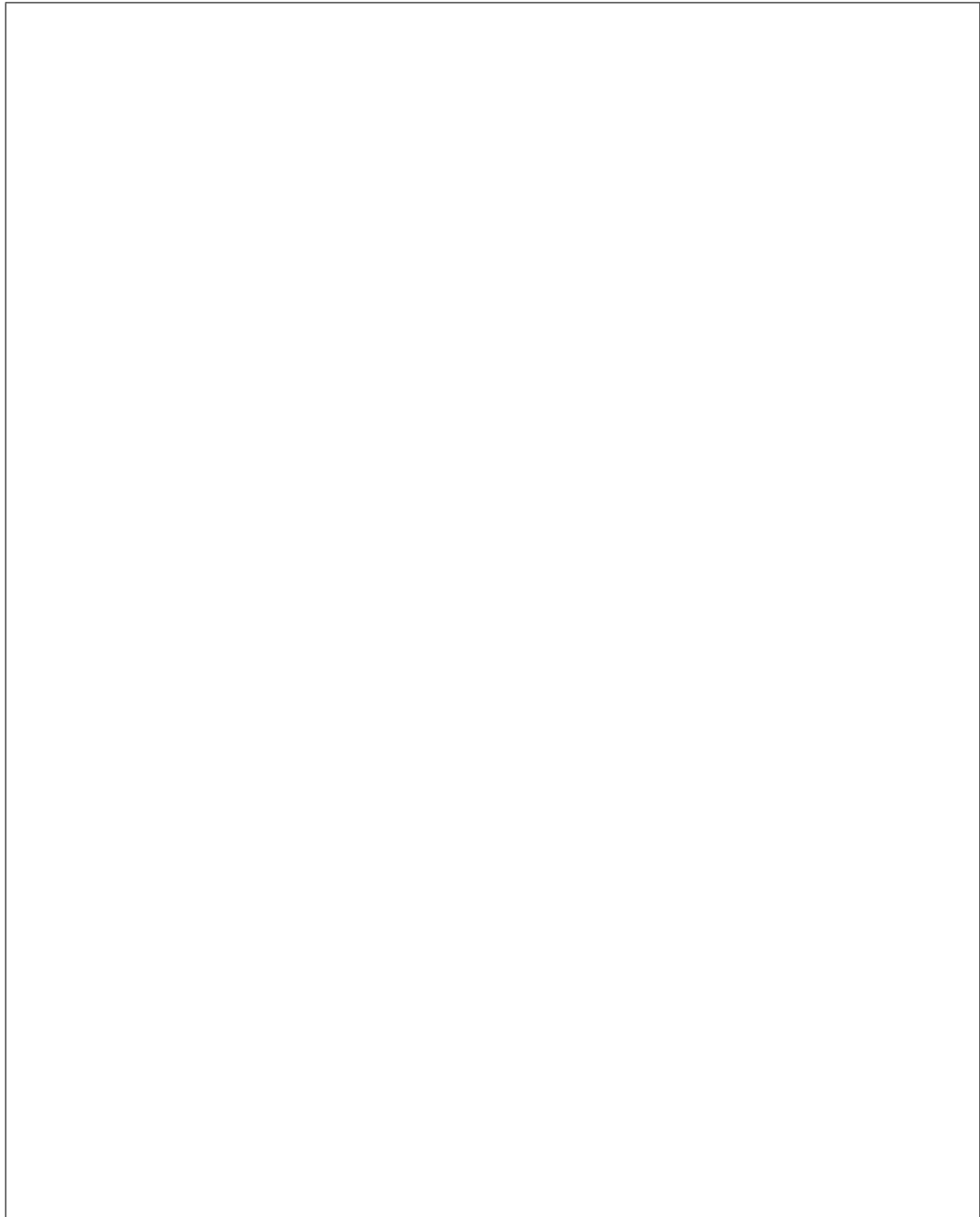
- Muestre el circuito funcionando al profesor, para que le sea tomado en cuenta. En el display deberá aparecer el dígito 0 hasta 9, dependiendo del voltaje que le llega a PC0 el cual esta en función de la posición de la perilla del potenciómetro.

6.4. Ejercicio 2.

- Lea el documento adjunto a esta práctica *Registros de configuración del ADC* y responda a las siguientes preguntas:

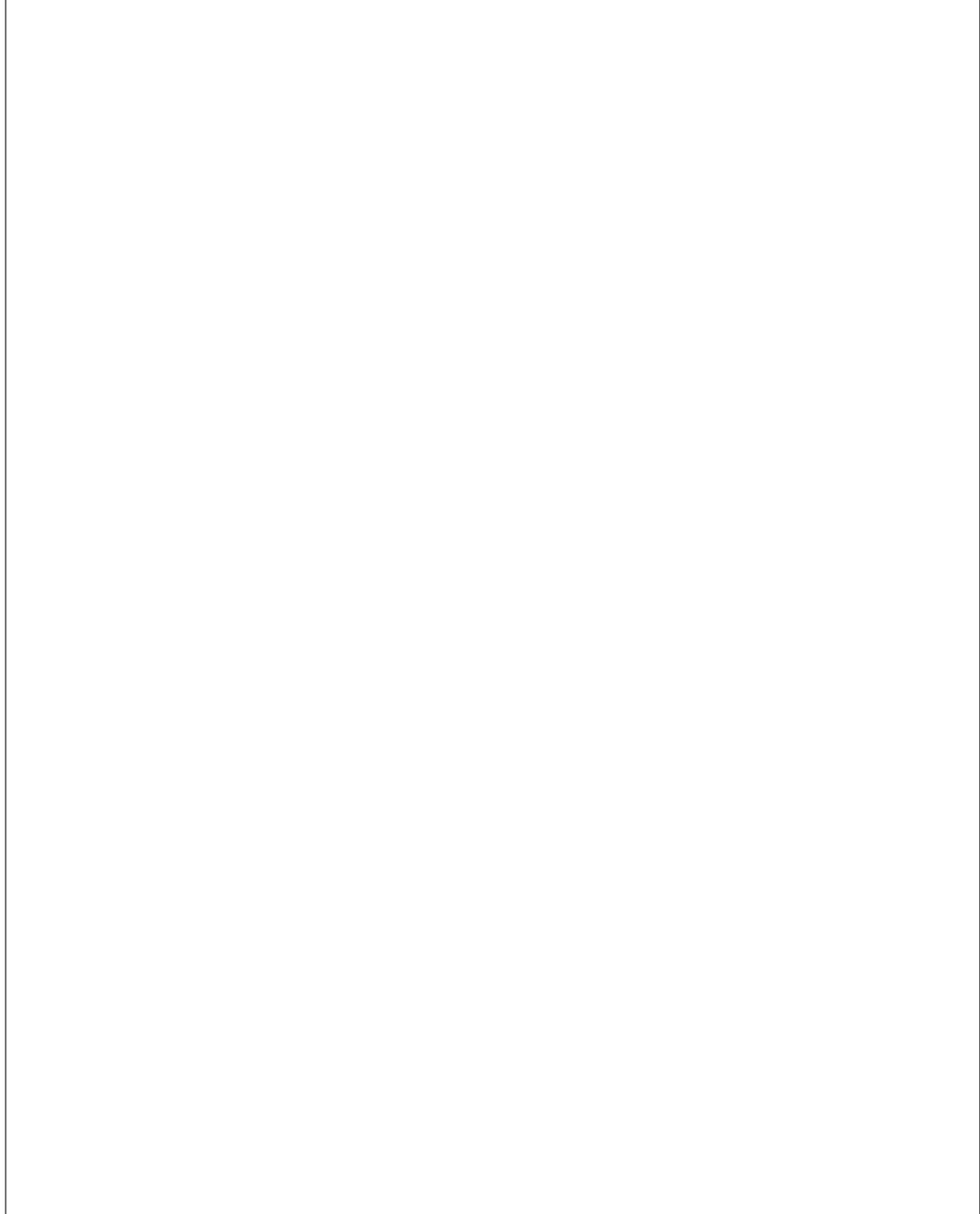
1. ¿Qué esta configurando la definición `ADC_VREF_TYPE`?

2. Explique con **detalle** qué esta haciendo la función `read_adc`, mencionando para qué sirve cada uno de los registros `ADMUX`, `ADCSRA`, `ADIF` y `ADCH` y como interactuan entre estos. **¡Experimente!**, cambie sus valores y vea qué pasa con el comportamiento del circuito.



3. Explique con **detalle** qué esta haciendo la sección de [ADC initialization](#), mencionando para qué sirve cada uno de los registros [DIDRO](#), [ADC5D](#), [ADC4D](#), [ADC3D](#), [ADC2D](#), [ADC1D](#), [ADCOD](#), [ADEN](#), [ADSC](#), [ADATE](#), [ADIE](#), [ADPS2](#), [ADPS1](#),

ADPS0, ADCSRB, ADTS2, ADTS1 y ADTS0 y como interactúan entre estos. **¡Experimente!**, cambie sus valores y vea qué pasa con el comportamiento del circuito.



28.9.1. ADC Multiplexer Selection Register

Name: ADMUX

Offset: 0x7C

Reset: 0x00

Property: -

Bit	7	6	5	4	3	2	1	0
	REFS1	REFS0	ADLAR		MUX3	MUX2	MUX1	MUX0
Access	R/W	R/W	R/W		R/W	R/W	R/W	R/W
Reset	0	0	0		0	0	0	0

Bits 7:6 – REFSn: Reference Selection [n = 1:0]

These bits select the voltage reference for the ADC. 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 28-3. ADC Voltage Reference Selection

REFS[1:0]	Voltage Reference Selection
00	AREF, Internal V_{ref} turned off
01	AV_{CC} with external capacitor at AREF pin
10	Reserved
11	Internal 1.1V Voltage Reference with external capacitor at AREF pin

Bit 5 – ADLAR: ADC Left Adjust Result

The ADLAR bit affects the presentation of the ADC conversion result in the ADC Data Register. Write one to ADLAR to left adjust the result. Otherwise, the result is right adjusted. Changing the ADLAR bit will affect the ADC Data Register immediately, regardless of any ongoing conversions. For a complete description of this bit, see [ADCL](#) and [ADCH](#).

Bits 3:0 – MUXn: Analog Channel Selection [n = 3:0]

The value of these bits selects which analog inputs are connected to the ADC. 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).

Table 28-4. Input Channel Selection

MUX[3:0]	Single Ended Input
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5

MUX[3:0]	Single Ended Input
0110	ADC6
0111	ADC7
1000	Temperature sensor
1001	Reserved
1010	Reserved
1011	Reserved
1100	Reserved
1101	Reserved
1110	1.1V (V_{BG})
1111	0V (GND)

28.9.2. ADC Control and Status Register A

Name: ADCSRA

Offset: 0x7A

Reset: 0x00

Property: -

Bit	7	6	5	4	3	2	1	0
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	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.

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 – ADPSn: ADC Prescaler Select [n = 2:0]

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

Table 28-5. Input Channel Selection

ADPS[2:0]	Division Factor
000	2
001	2

ADPS[2:0]	Division Factor
010	4
011	8
100	16
101	32
110	64
111	128

28.9.3. ADC Data Register Low (ADLAR=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 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.

Name: ADCL

Offset: 0x78

Reset: 0x00

Property: ADLAR = 0

Bit	7	6	5	4	3	2	1	0
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – ADCn: ADC Conversion Result [n = 7:0]

These bits represent the result from the conversion. Refer to [ADC Conversion Result](#) for details.

28.9.4. ADC Data Register High (ADLAR=0)

Name: ADCH
Offset: 0x79
Reset: 0x00
Property: ADLAR = 0

Bit	7	6	5	4	3	2	1	0
							ADC9	ADC8
Access							R	R
Reset							0	0

Bit 1 – ADC9: ADC Conversion Result
Refer to [ADCL](#).

Bit 0 – ADC8: ADC Conversion Result

28.9.5. ADC Data Register Low (ADLAR=1)

Name: ADCL
Offset: 0x78
Reset: 0x00
Property: ADLAR = 1

Bit	7	6	5	4	3	2	1	0
	ADC1	ADC0						
Access	R	R						
Reset	0	0						

Bit 7 – ADC1: ADC Conversion Result
Refer to [ADCL](#).

Bit 6 – ADC0: ADC Conversion Result

28.9.6. ADC Data Register High (ADLAR=1)

Name: ADCH

Offset: 0x79

Reset: 0x00

Property: ADLAR = 1

Bit	7	6	5	4	3	2	1	0
	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2
Access	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit 7 – ADC9: ADC Conversion Result 9

Refer to [ADCL](#).

Bit 6 – ADC8: ADC Conversion Result 8

Refer to [ADCL](#).

Bit 5 – ADC7: ADC Conversion Result 7

Refer to [ADCL](#).

Bit 4 – ADC6: ADC Conversion Result 6

Refer to [ADCL](#).

Bit 3 – ADC5: ADC Conversion Result 5

Refer to [ADCL](#).

Bit 2 – ADC4: ADC Conversion Result 4

Refer to [ADCL](#).

Bit 1 – ADC3: ADC Conversion Result 3

Refer to [ADCL](#).

Bit 0 – ADC2: ADC Conversion Result 2

Refer to [ADCL](#).

28.9.7. ADC Control and Status Register B

Name: ADCSRB

Offset: 0x7B

Reset: 0x00

Property: -

Bit	7	6	5	4	3	2	1	0
		ACME				ADTS2	ADTS1	ADTS0
Access		R/W				R/W	R/W	R/W
Reset		0				0	0	0

Bit 6 – ACME: Analog Comparator Multiplexer Enable

When this bit is written logic one and the ADC is switched off (ADEN in ADCSRA is zero), the ADC multiplexer selects the negative input to the Analog Comparator. When this bit is written logic zero, AIN1 is applied to the negative input of the Analog Comparator. For a detailed description of this bit, see *Analog Comparator Multiplexed Input..*

Bits 2:0 – ADTSn: ADC Auto Trigger Source [n = 2:0]

If ADATE in ADCSRA is written to one, the value of these bits selects which source will trigger an ADC conversion. If ADATE is cleared, the ADTS[2:0] settings will have no effect. A conversion will be triggered by the rising edge of the selected Interrupt Flag. Note that switching from a trigger source that is cleared to a trigger source that is set, will generate a positive edge on the trigger signal. If ADEN in ADCSRA is set, this will start a conversion. Switching to Free Running mode (ADTS[2:0]=0) will not cause a trigger event, even if the ADC Interrupt Flag is set.

Table 28-6. ADC Auto Trigger Source Selection

ADTS[2:0]	Trigger Source
000	Free Running mode
001	Analog Comparator
010	External Interrupt Request 0
011	Timer/Counter0 Compare Match A
100	Timer/Counter0 Overflow
101	Timer/Counter1 Compare Match B
110	Timer/Counter1 Overflow
111	Timer/Counter1 Capture Event

28.9.8. Digital Input Disable Register 0

When the respective bits are written to logic one, the digital input buffer on the corresponding ADC pin is disabled. The corresponding PIN Register bit will always read as zero when this bit is set. When an analog signal is applied to the ADC7...0 pin and the digital input from this pin is not needed, this bit should be written logic one to reduce power consumption in the digital input buffer.

Name: DIDR0

Offset: 0x7E

Reset: 0x00

Property: -

Bit	7	6	5	4	3	2	1	0
	ADC7D	ADC6D	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7 – ADC0D, ADC1D, ADC2D, ADC3D, ADC4D, ADC5D, ADC6D, ADC7D: ADC Digital Input Disable