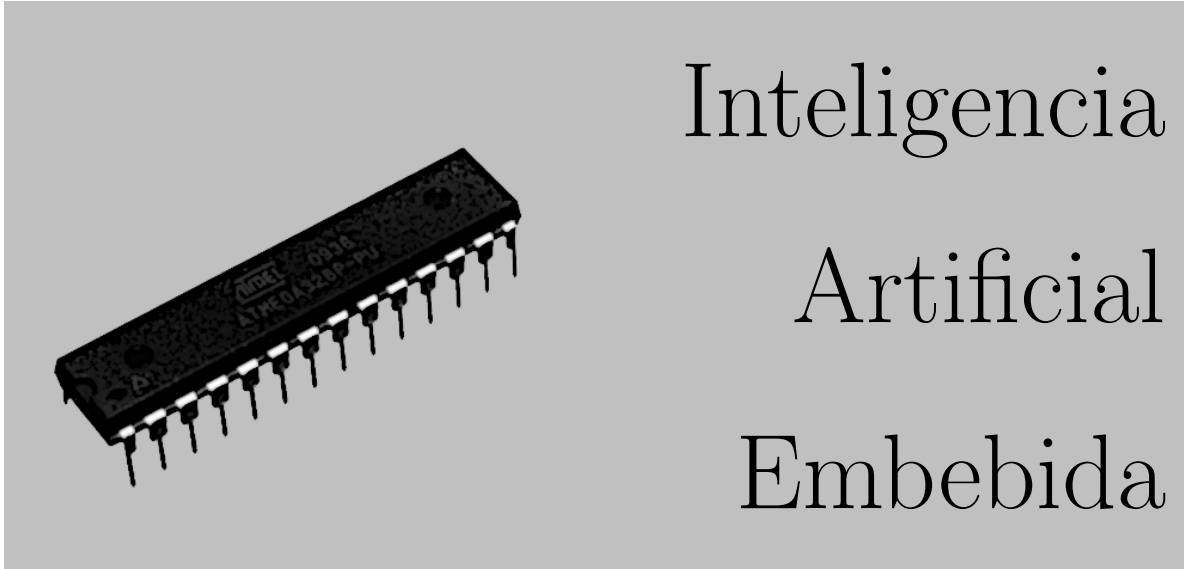




Inteligencia Artificial Embebida

Nombre: _____ Grupo: _____

Dr. Enrique García Trinidad
Tecnológico de Estudios Superiores de Huixquilucan
<https://enriquegarcia.xyz>
enrique.g.t@huixquilucan.tecnm.mx

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 ??:

6.3. 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](#).

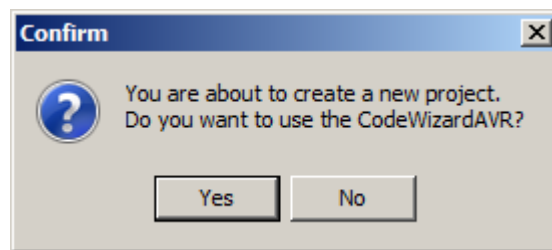


Figura 6.2: Elegir [Yes](#).

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

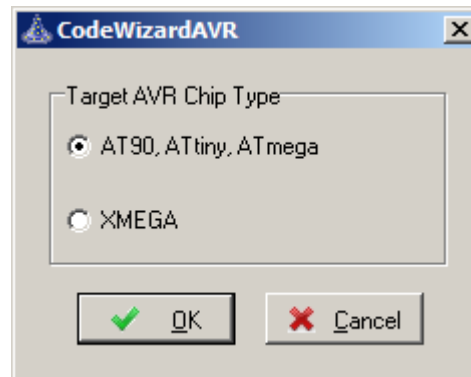


Figura 6.3: Al finalizar seleccionar [OK](#)

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

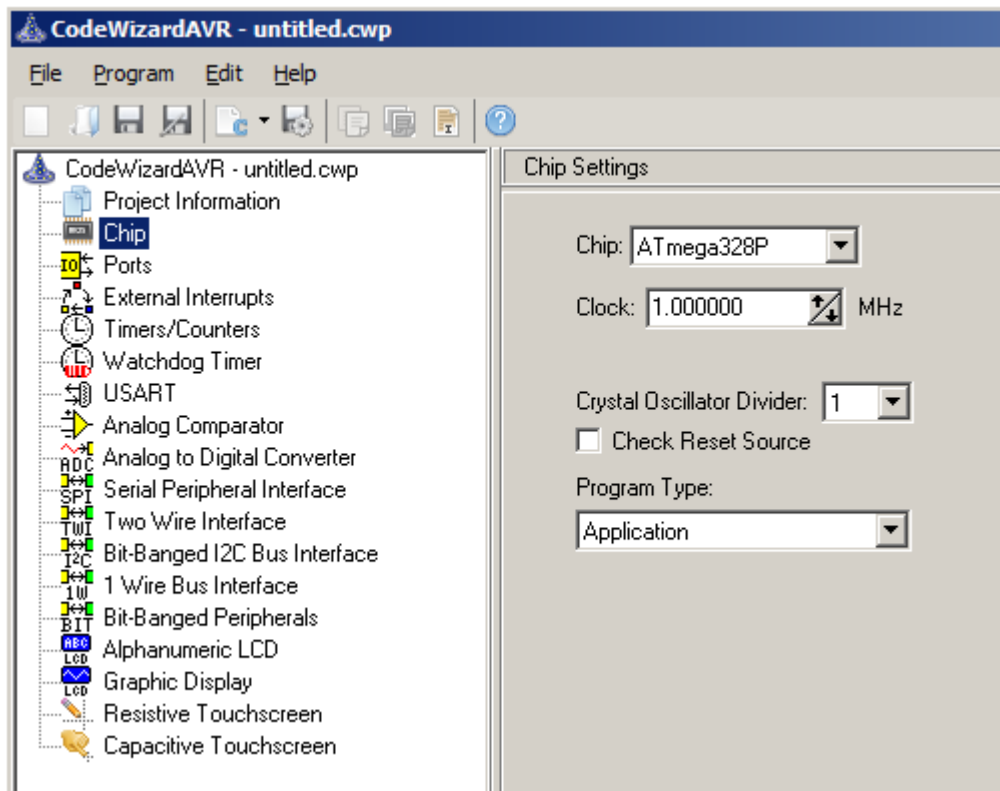


Figura 6.4: Configuración ficha **Chip**

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

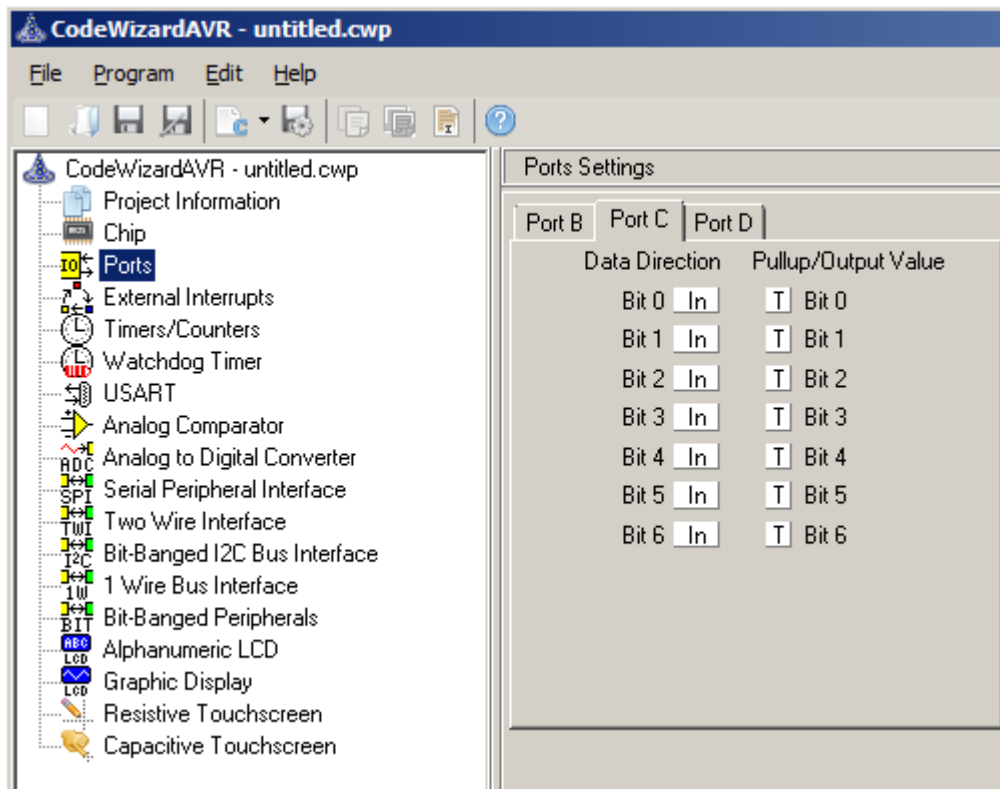


Figura 6.5: Configuración E/S Port C

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

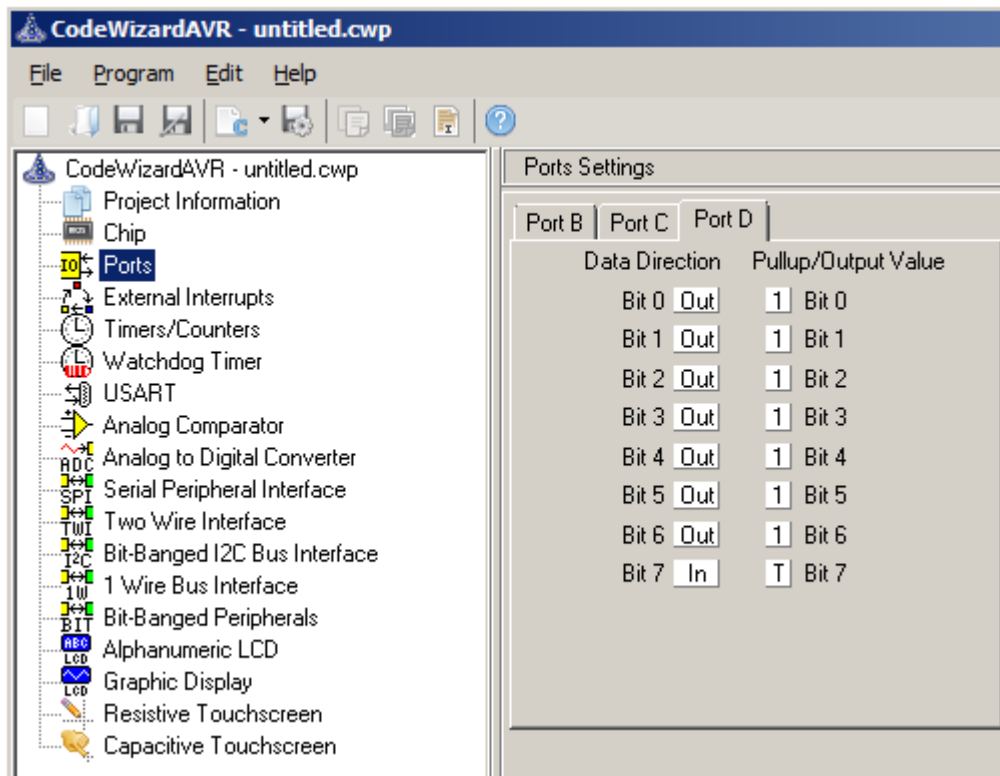


Figura 6.6: Configuración E/S Port D

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

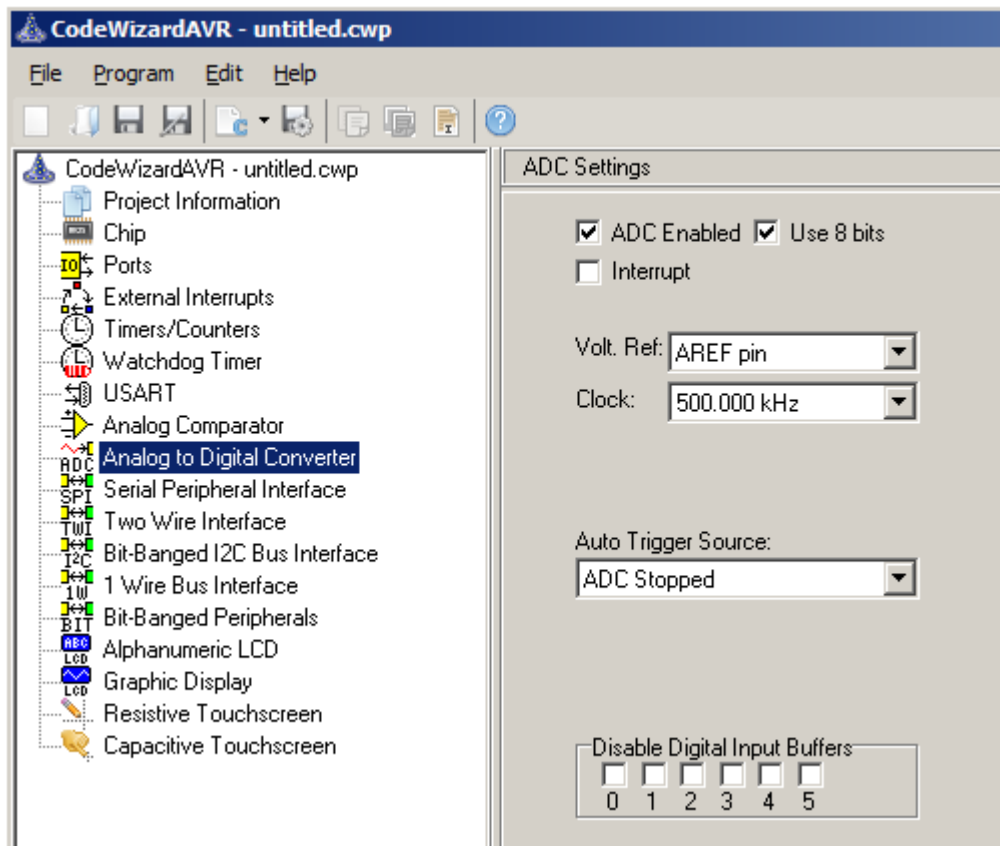


Figura 6.7: Configuración del ADC

- 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**.

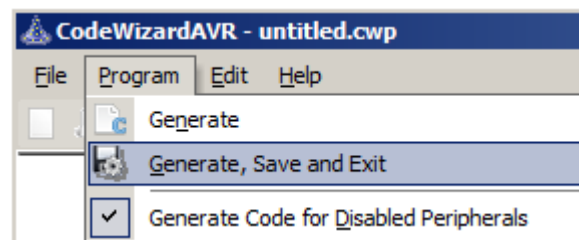


Figura 6.8: Generar el proyecto y el código.

- 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 << ADIE) | (0 << ADPS2) | (0 << ADPS1) | (1 << ADPS0
        );
    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 << ADATE) | (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.

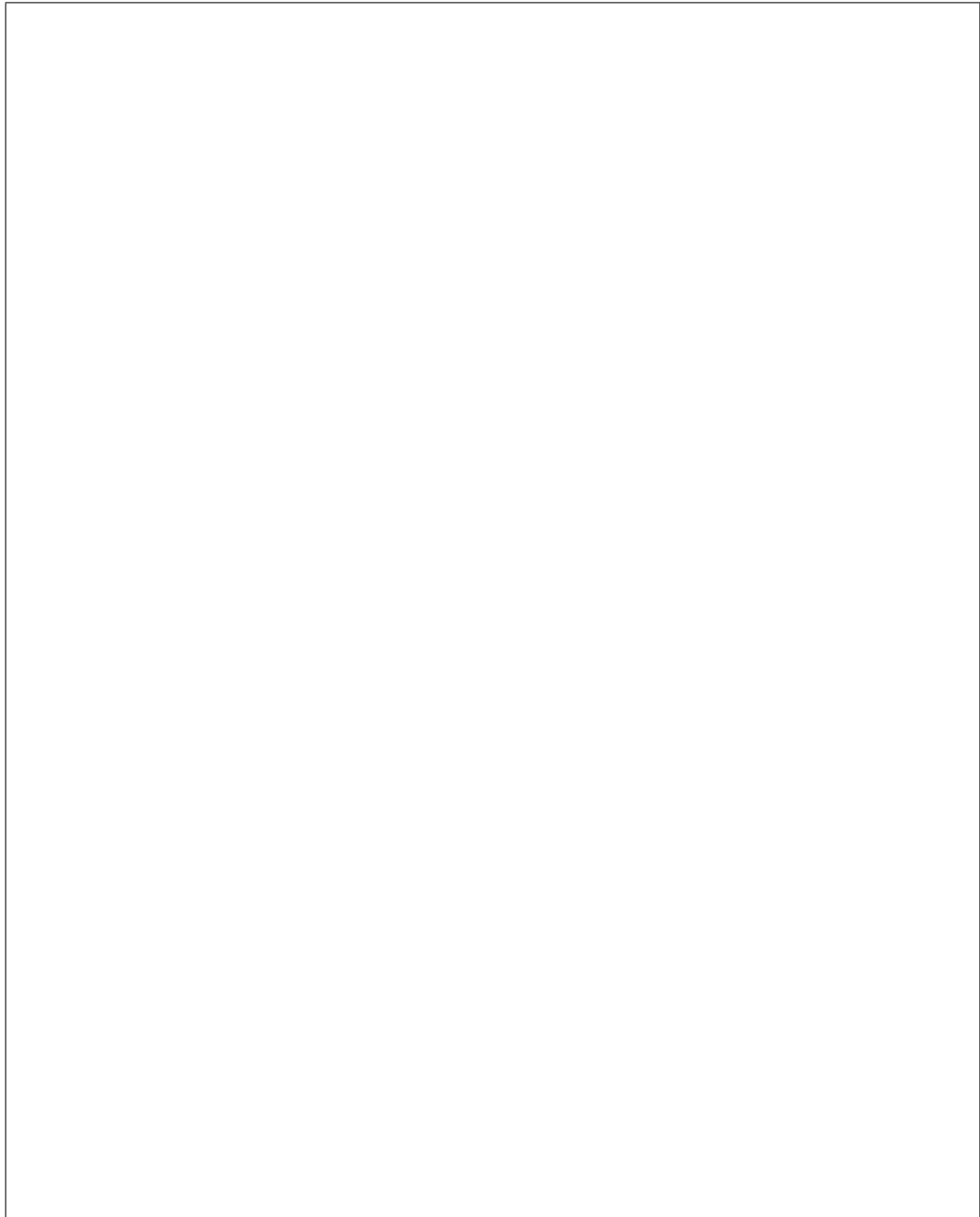
- Muestre el circuito funcionando al profesor, para que le sea tomado en cuenta. En el display deberá aparecer el dígito 0 hasta 9, dependiendo del voltaje que le llega a PC0 el cual esta en función de la posición de la perilla del potenciómetro.

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.

