Nombre de la asignatura: Laboratorio de Programación de Avanzada

Maestro en Ciencias: Moisés Agustín Martínez Hernández

Nombre de la práctica: Convertidor Analógico Digital (ADC)

Integrantes:

- Zuñiga Fragoso Diego Joel
- Manríquez Navarro Daniela del Carmen

Número de práctica: 4

Total de horas: 3 Hrs.

Objetivos

- Comprender el concepto de la conversión analógica a digital, las variables que influyen en el proceso como la frecuencia de muestreo, la resolución del dispositivo, los niveles de tensión y acondicionamiento de la señal.
- Aprender a configurar pines de salida para leer señales de tipo Analógicas en el microcontrolador.
- Implementar la lectura de 1 canal analógico para realizar una conversión de nivel de tensión con el propósito de controlar el giro de un motor a pasos.

Descripción de la práctica

Se utilizará una entrada analógica para leer un voltaje aplicado al microcontrolador, el nivel de voltaje de entrada dependerá del ajuste hecho a un potenciómetro conectado a la entrada analógica. La variación del voltaje se verá reflejada en la velocidad de giro de un motor a pasos unipolar controlado por un puerto de salida del microcontrolador, dicho motor realizara secuencias de giro que mediante una banda o acoplamiento que permitirán visualizar un desplazamiento, el desplazamiento será medido en cm.

Marco teórico

Conversión Analógica a Digital

La salida de los sensores, que permiten al equipo electrónico interaccionar con el entorno, es normalmente una señal analógica, continua en el tiempo. En consecuencia, esta información debe convertirse a binaria (cada dato analógico decimal codificado a una palabra formada por unos y ceros) con el fin de adaptarla a los circuitos procesadores y de presentación. Un convertidor analógico-digital (CAD) es un circuito electrónico integrado cuya salida es la palabra digital resultado de convertir la señal

analógica de entrada. La conversión a digital se realiza en dos fases: cuantificación y codificación. Durante la primera se muestrea la entrada y a cada valor analógico obtenido se asigna un valor o estado, que depende del número de bits del CAD. El valor cuantificado se codifica en binario en una palabra digital, cuyo número de bits depende de las líneas de salida del CAD. Estos dos procesos determinan el diseño del circuito integrado.

En un CAD de N bits hay 2N estados de salida y su resolución (porción más pequeña de señal que produce un cambio apreciable en la salida) se expresa como 1/2N (una parte en el número de estados). Con frecuencia la resolución se expresa a partir del margen de entrada del convertidor para definir el intervalo de cuantización o espacio de 1 LSB (Least Significant Bit; bit menos significativo).

La figura 1 representa la respuesta de un convertidor A/D de 3 bits a una entrada analógica sinodal de 1 kHz de frecuencia, valor medio 5 V y valor cresta a cresta de 10 V, coincidentes con el margen de entrada. En ella se observan los 23=8 estados de la salida, correspondientes a los códigos binarios desde el 000 al 111. Cada intervalo de cuantización tiene una anchura de 10 (V)/8 (estados)=1,25 V.

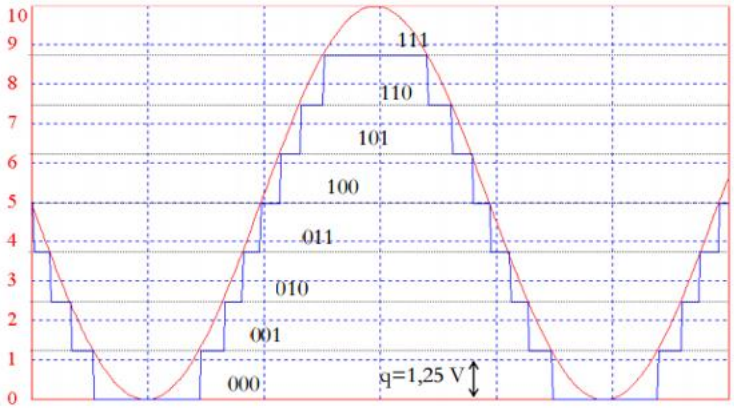


Figura. 1. Digitalización de una señal analógica por un convertidor A/D de 3 bits.

Motor a pasos

Un motor paso a paso, Figura 2, como todo motor, es en esencia un conversor electromecánico, que transforma energía eléctrica en mecánica. Mientras que un motor convencional gira libremente al aplicarle una tensión, el motor paso a paso gira un determinado ángulo de forma incremental (transforma impulsos eléctricos en movimientos de giro controlados), lo que le permite realizar desplazamientos angulares fijos muy precisos (pueden variar desde 1,80° hasta unos 90°).

Figura 2. Motor a pasos.

Este tipo de motores son ideales cuando lo que queremos es posicionamiento con un elevado grado de exactitud y/o una muy buena regulación de la velocidad. Están constituidos esencialmente por dos partes:

- Estator: parte fija construida a base de cavidades en las que van depositadas las bobinas.
- Rotor: parte móvil construida mediante un imán permanente. Este conjunto va montado sobre un eje soportado por dos cojinetes que le permiten girar libremente.

Al número de grados que gira el rotor, cuando se efectúa un cambio de polaridad en las bobinas del estator, se le denomina "ángulo de paso". Existe la posibilidad de conseguir una rotación de medio paso con el control electrónico apropiado, aunque el giro se hará con menor precisión. Los motores son fabricados para trabajar en un rango de frecuencias determinado por el fabricante, y rebasado dicho rango, provocaremos la pérdida de sincronización.

EQUIPO Y MATERIALES

- Software para programar microcontrolador.
- Software de simulación.
- Microcontrolador (sistema mínimo).
- Resistencia variable (potenciómetro, trimpot, etc.).
- 4 push buttons.
- 4 resistencias de 220 o 330 ohms.
- 1 motor a pasos unipolar.
- Semiconductores para disparo de fases en motor.

DESARROLLO DE LA PRÁCTICA

1. Configurar el microcontrolador. Figura 4, considerando:
   a) Frecuencia de reloj 4 MHz.
   b) Habilitar el ADC con la mayor resolución posible del microcontrolador (10 bits).
   c) Configurar un canal analógico del microcontrolador para conectar la señal de la resistencia variable.
   d) Configurar 4 pines como entradas digitales, mismas que se conectaran a 4 interruptores S1, S2, S3 y S4 (figura 4).
   e) Configurar las salidas digitales necesarias para controlar la activación de las fases del motor a pasos, estas se conectarán a los dispositivos electrónicos que se emplee.
2. Escribir un código para que efectué las siguientes instrucciones:
   a) Al inicio del programa el microcontrolador estará esperando la instrucción de usuario para establecer una distancia a recorrer y el motor se encontrará en su posición inicial.
   b) Cuando el usuario presione alguno de los 4 botones (S1, S2, S3 o S4), el motor comenzara a girar causando que la banda acoplada a la flechdesplace un objeto indicador y mediante una regla se aprecie la distancia recorrida (tabla 1).

c) Cuando el motor realice las secuencias necesarias para lograr la distancia requerida esperará 2 segundos en la distancia requerida y posteriormente invertirá la secuencia realizada para regresar a la posición inicial.
d) Al regresar a la posición inicial el programa volverá al estado inicial del inciso a.
e) El valor de lectura del ADC alterara el retardo entre los pasos del motor. El valor que se asigne será mediante una relación de conversión

3. En el código deberá considerar los siguientes escenarios:
   a) Si son presionados dos o más botones de selección al mismo tiempo, el motor no deberá ejecutar ninguna instrucción.
   b) Si es presionado cualquier botón mientras el motor está ejecutando alguna rutina, este deberá culminar la secuencia de manera normal (inciso C punto 2), al llegar a la posición inicial deberá esperar 2 segundos y ejecutar automáticamente la secuencia solicitada mientras ejecutaba la rutina inicial. Esta condición puede pasar n cantidad de veces.
   c) En cualquier momento de la ejecución del código se podrá modificar el valor del potenciómetro para variar la velocidad de giro del motor. La relación de cambio en la velocidad deberá ser notable conforme se varia el valor de la resistencia.
4. Realizar la simulación adecuada para validar el código.
5. Programar el microcontrolador e implementar el circuito necesario para cumplir los requisitos de los puntos 1, 2 y 3.
6. Construir una maqueta que acople una banda o mecanismo que desplace un objeto con el movimiento de la flecha del motor para cumplir los desplazamientos en cada caso de la tabla 1.

Tabla 1. Relación de distancias respecto a la selección.

Botón	Distancia a recorrer
S1	1 cm
S2	3 cm
S3	5 cm
S4	7 cm

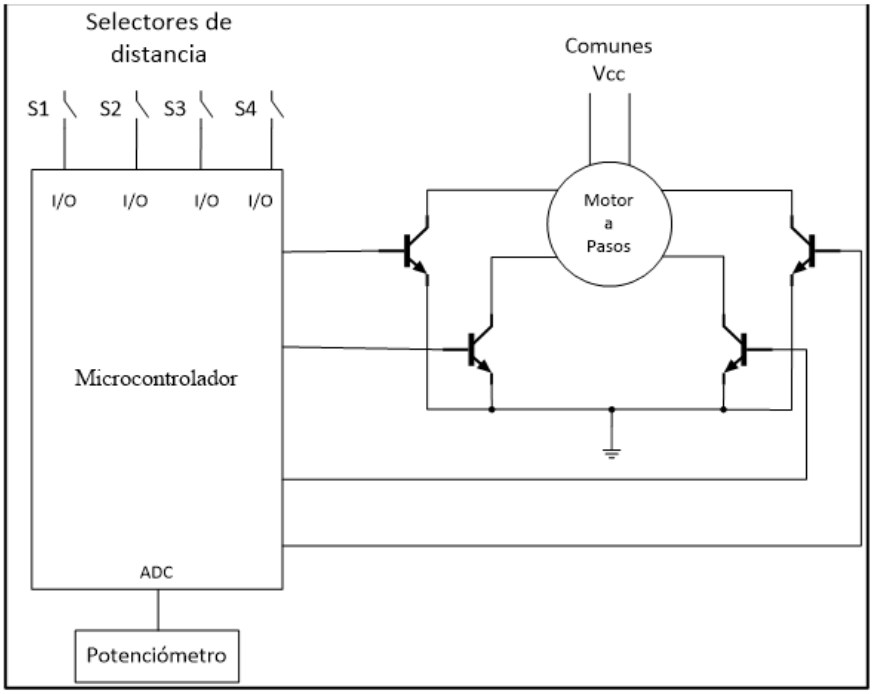
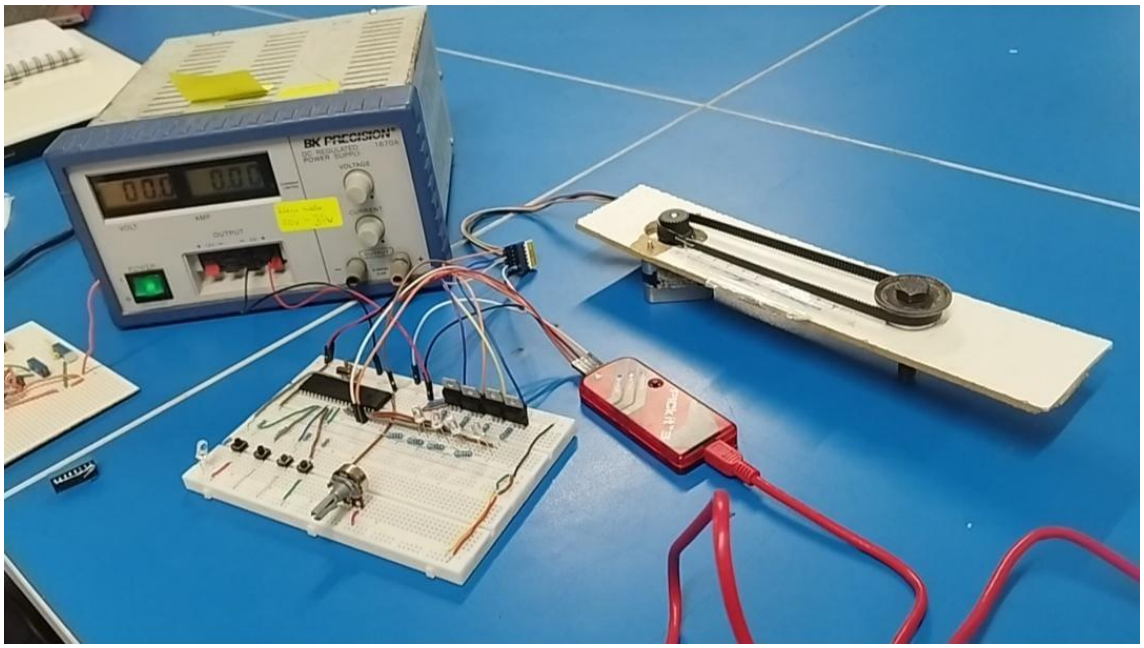


Figura 4. Diagrama de bloques para el circuito *Control de velocidad de motor a pasos con ADC.*

Resultados de la práctica



Circuito armando

Código explicado

```c
#include <18f4550.h>   // Libreria del Microcontrolador
#device adc = 10
#fuses XT, NOWDT, NOPROTECT, NOLVP, CPUDIV1, PLL1, NOMCLR // Fusibles (Configuraciones del microcontrolador)
#use delay(clock = 4M) // 4 Megahertz

#define S1 pin_d4
#define S2 pin_d5
#define S3 pin_d6
#define S4 pin_d7
#define LED pin_c0

// Intervalo de MS
const int16 MAX = 1925;
const int16 MIN = 50;

void ROTAR(int16);
int SEC_I(); // Secuencia Inicial (No genera cola, ni respeta delay)
int SEC_R(int16, int16 *); // Secuencia con Retraso (Se utiliza para respetar delay)

int orden[25] = {0}; // Arreglo con orden de ejecucion de secuencias
int pos = 0; // Contador de secuencias en cola

void main()
{
   int posE = 0; // Contador de secuencias ejecutadas
   int16 i;

   setup_adc(adc_clock_div_2);
   setup_adc_ports(AN0); // Hay un sensor en AN0
   set_tris_a(0b00000001); // 1 entrada 0 salida
   set_tris_d(0b11110000);
   set_adc_channel(0);
   delay_us(50);

   while(true)
   {
      if(pos == posE) // Ya no hay botones en la cola
      {
         output_low(LED); // Se apaga el led de la cola

         do
         {
            orden[pos] = SEC_I(); // Recibimos un posible boton
         }
         while(orden[pos] == 0);

         //if(orden[pos] != 0) // Si se recibio un boton presionado
            pos++;
      }
      else // Hay botones en cola, esperamos 2 segundos antes de reproducir el siguiente
      {
         // Delay de 2000 ms esperando posible presion de boton
         for(i = 1; i <= 2000 ; i++)
         {
            delay_ms(1);

            orden[pos] = SEC_R(2000-i, &i); // Le envio los ms maximos que se puede atrapar el codigo. (ms totales - ms que han pasado)

            if(orden[pos] != 0) // Si se detecto un boton presionado
               pos++;
         }
      }

       // Ejecucion, va de uno en uno en el arreglo

      switch(orden[posE++])
      {
         case 1: // S1
            ROTAR(17);
         break;
         case 2: // S2
            ROTAR(50);
         break;
         case 3: // S3
            ROTAR(86);
         break;
         case 4: // S4
            ROTAR(122);
         break;
         default: // 0, no hace nada ni avanza al siguiente posE
         posE--; // Revertimos el incremento anterior pues no se ejecuto ningun caso
         break;
      }
   }
}
```

```
86
87  // Funcion para Rotacion de motor
88
89  // Problema cuando N es 1 (No es muy relevante)
90  void ROTAR(int16 N)
91  {
92     int cont=0, pmotor = 1;
93     int16 MS, i;
94
95     // Hacer las N rotaciones
96
97     while(cont < N)
98     {
99        // read_adc() al ser de 10 bits, recibe numeros en el intervalo (0:1023)
100       MS = read_adc() * 2 + MIN; // Obtiene el valor del potenciometro
101
102       output_b(pmotor); // Mueve el motor 1 vez
103
104       pmotor *= 2; // Pasamos al siguiente pin para la sig rotacion
105       cont++; // Contador de rotaciones
106
107       if(pmotor > 8)
108          pmotor = 1;
109
110       // Haremos el delay detectando cada milisegundo si se presiono un boton
111       for(i = 1; i <= MS ; i++)
112       {
113          delay_ms(1);
114
115          orden[pos] = SEC_R(MS-i, &i); // Le envio los ms maximos que se puede atrapar el codigo. (ms totales - ms que han pasado)
116
117          if(orden[pos] != 0) // Si se detecto un boton y se guardo una posicion en la cola avanzamos
118          {
119             pos++;
120             output_high(LED);
121          }
122       }
123    }
124    output_b(0);
125
126    // Delay de 2 segundos
127
128    for(i = 1; i <= 2000 ; i++)
129    {
130       delay_ms(1);
131
132       orden[pos] = SEC_R(2000-i, &i); // Le envio los ms maximos que se puede atrapar el codigo. (ms totales - ms que han pasado)
133
134       if(orden[pos] != 0) // Si se detecto un boton y se guardo una posicion en la cola avanzamos
135       {
136          pos++;
137          output_high(LED);
138       }
139    }
140
141    // Regresamos una posicion extra que avanzo
142
143    if(pmotor == 1)
144       pmotor = 8;
145    else
146       pmotor /= 2;
147
148    // Volvemos la origen
149
150    cont = 0;
151
152    while(cont < N)
153    {
154       // read_adc() al ser de 10 bits, recibe numeros en el intervalo (0:1023)
155       MS = read_adc() * 2 + MIN; // Obtiene el valor del potenciometro
156
157
158       output_b(pmotor); // Mueve el motor 1 vez
159
160       pmotor /= 2; // Pasamos al siguiente pin para la sig rotacion
161       cont++; // Contador de rotaciones
162
163       if(pmotor == 0)
164          pmotor = 8;
165
166       // Haremos el delay detectando cada milisegundo si se presiono un boton
167       for(i = 1; i <= MS ; i++)
168       {
169          delay_ms(1);
```

```
            orden[pos] = SEC_R(MS-i, &i); // Le envio los ms maximos que se puede atrapar el codigo. (ms totales - ms que han pasado)

            if(orden[pos] != 0) // Si se detecto un boton y se guardo una posicion en la cola avanzamos
            {
               pos++;
               output_high(LED);
            }
         }
      }
      output_b(0);
   }


   // Funcion para Secuencia con delays (No envia el resultado de las comparaciones hasta que se suelte el boton)

   int SEC_R(int16 MS, int16 *i)
   {
      int r = 0;
      int16 time = 0;

      if(input(S1) && !input(S2) && !input(S3) && !input(S4)) // S1
      {
         while(input(S1) && (time < MS)) // Atrapamos el codigo mientras se presione el boton y no exceda los ms maximos
         {
            delay_ms(1);
            time++;
         }

         if(time < MS) // No se excedieron los ms maximos
            r = 1;
      }
      else if(!input(S1) && input(S2) && !input(S3) && !input(S4)) // S2
      {
         while(input(S2) && (time < MS)) // Atrapamos el codigo mientras se presione el boton y no exceda los ms maximos
         {
            time++;
            delay_ms(1);
         }

         if(time < MS)
            r = 2;
      }
      else if(!input(S1) && !input(S2) && input(S3) && !input(S4)) // S3
      {
         while(input(S3) && (time < MS)) // Atrapamos el codigo mientras se presione el boton y no exceda los ms maximos
         {
            time++;
            delay_ms(1);
         }

         if(time < MS)
            r = 3;
      }
      else if(!input(S1) && !input(S2) && !input(S3) && input(S4)) // S4
      {
         while(input(S4) && (time < MS)) // Atrapamos el codigo mientras se presione el boton y no exceda los ms maximos
         {
            time++;
            delay_ms(1);
         }

         if(time < MS)
            r = 4;
      }

      *i += time;

      return r;
   }


   // Funcion para Secuencia sin delays (No se genera cola)

   int SEC_I()
   {
      int r = 0, B = 1;

      if(input(S1)) // S1
      {
         while(input(S1))
         {
            if(input(S2) || input(S3) || input(S4))
            {
```

```
            while(input(S2) || input(S3) || input(S4));
            B = 0;
         }
      }

      if(B)
         r = 1;
   }
   else if(input(S2)) // S2
   {
      while(input(S2))
      {
         if(input(S1) || input(S3) || input(S4))
         {
            while(input(S1) || input(S3) || input(S4));
            B = 0;
         }
      }

      if(B)
         r = 2;
   }
   else if(input(S3)) // S3
   {
      while(input(S3))
      {
         if(input(S1) || input(S2) || input(S4))
         {
            while(input(S1) || input(S2) || input(S4));
            B = 0;
         }
      }

      if(B)
         r = 3;
   }
   else if(input(S4)) // S4
   {
      while(input(S4))
      {
         if(input(S1) || input(S2) || input(S3))
         {
            while(input(S1) || input(S2) || input(S3));
            B = 0;
         }
      }

      if(B)
         r = 4;
   }

   delay_ms(200);
   return r;
}
```

Conclusiones de la práctica

Joel Zúñiga:

Durante la realización de esta práctica, adquirí conocimientos sobre cómo el microcontrolador transforma información analógica, en este caso, la proveniente de un potenciómetro, en datos digitales de 10 bits que representan la posición del potenciómetro. Asimismo, aprendí a controlar la velocidad de un motor a pasos, y a comprender mejor la función de un transistor, que no solo actúa como un interruptor, sino que también puede amplificar el voltaje y la corriente para los dispositivos conectados a su emisor.

Daniela Manríquez:

En esta práctica conocí como utilizar el microcontrolador para recibir e interpretar las señales analógicas que se recibe por el puerto ADC. Así como también aprendí como es que funciona un motor a pasos, con esto pude saber manipularlo para que realizara las secuencias pedidas, así como controlar la velocidad a través de un potenciómetro.

Bibliografía

González J. (2001). Instrumentación Electrónica. Universidad de Cadiz: Boixareu Editores.

Grindling G. & Weiss B. (2007). Introduction to Microcontrollers. Vienna University of Technology.

(2014). Optoacoplador 4N28 Salida Transistor. 19/07/2016, de Carrod Electronica Sitio web: http://www.carrod.mx/products/optoacoplador-4n28-salida-transistor.