Nombre de la asignatura: Laboratorio de programación de avanzada

Maestro en Ciencias: Moisés Agustín Martínez Hernández

Nombre de la práctica: RS-232

Integrantes:

- Zúñiga Fragoso Diego Joel
- Manríquez Navarro Daniela del Carmen

Número de práctica: 8

Total de horas: 4 Hrs.

Objetivo

- Comprender el concepto de comunicación serial.
- Aprender a configurar pines de salida del microcontrolador como puerto serial para envío y recepción de datos.
- Configurar una PC con el propósito tener una comunicación bidireccional con el microcontrolador.
- Comunicar el microcontrolador con la PC por medio del protocolo RS-232 con la finalidad de controlar las funciones del microcontrolador.

Descripción de la práctica

Enviar y recibir información del microcontrolador a la PC a través de comunicación serial, usando el protocolo RS-232. Las instrucciones del microcontrolador serán mostradas en un monitor de puerto serial. Las tareas que se podrán realizar son las siguientes: lectura de ADC, mostrar en dos displays de 7 segmentos un valor de 2 dígitos como máximo que será ingresado desde a PC, controlar la velocidad de giro de un motor DC y controlar el sentido y velocidad de giro de motor a pasos.

Marco teórico

Los modos de transmisión de datos se dividen en cuatro tipos:

- *Simplex*. Se dice a la transmisión que puede ocurrir en un sólo sentido, sea sólo para recibir o sólo para transmitir.
- *Half-duplex.* Se refiere a la transmisión que puede ocurrir en ambos sentidos pero no al mismo tiempo, en donde una ubicación puede ser un transmisor y un receptor, pero no los dos al mismo tiempo.
- *Full-duplex*. Se dice a la transmisión que puede ocurrir en ambos sentidos y al mismo tiempo.
- *Full/full-duplex.* Con este modo de transmisión es posible transmitir y recibir simultáneamente, pero no necesariamente entre las dos ubicaciones, es decir una estación puede transmitir a una segunda estación y recibir de una tercera estación al mismo tiempo.

La norma RS-232 es la más habitual en la comunicación serie. Básicamente comunica un equipo terminal de datos (DTE o Data Terminal Equipment) y el equipo de comunicación de datos (DCE o Data Communications Equipment). Las características eléctricas de la señal en esta norma establecen que la longitud máxima entre el DTE y el DCE no puede ser superior a 15 metros y la velocidad máxima de transmisión es de 20,000 bps. Los noveles lógicos no son compatibles TTL (Transistor-Transistor Logic), deben situarse dentro de los siguientes rangos: 1 lógico entre -3V y -15V y 0 lógico entre +3V y +15V. Se utilizan conectores de 25 patillas (DB25) o de 9 patillas (DB9) siendo asignado el conector macho al DTE y el conector hembra al DCE.
Para la comunicación full dúplex, se debe de conectar un mínimo número de señales, *TXD y RXD* así como *GND.* Los microcontroladores utilizan señal TTL por lo que se debe utilizar un conversor de nivel a *RS232*, como el *MAX232*.
En la actualidad los puertos serie en las PC están prácticamente desaparecidos. Como solución de pueden utilizar cables de conversión *SERIE-USB*. La estructura de un dato en comunicación serial se muestra a continuación, Figura 1.

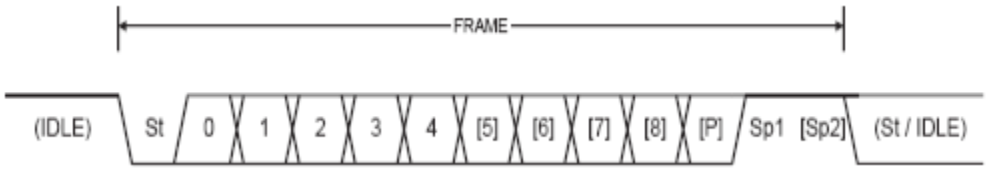


Figura 1. Estructura de un dato en comunicación serial.

Donde:
- St: Bit de Arranque siempre es activo en bajo.
- N: Número de datos de 0 a 8 bits.
- P: Bit de paridad.
- Sp: Bits de parada puede ser 1 o 2. Siempre son activos en altos.

La señal permanece en un nivel lógico alto mientras no realiza ninguna trasferencia de datos. Para empezar a transmitir datos el transmisor coloca la línea en nivel bajo durante el tiempo de

in bit, este se llama bit de arranque, a continuación empieza a transmitir con el mismo intervalo de tiempo los bits de datos, que pueden ser de 7 u 8 bits, comenzando por los bits menos significativos y terminando con los más significativos. Al final de la transmisión se envía un bit de paridad, si estuviera activa esta opción, y por último los bits de parada, que pueden ser 1 o 2, después de esto la línea vuelve a un estado lógico alto, y el transmisor está listo para enviar el siguiente dato.

EQUIPO Y MATERIALES

- Software para programar microcontrolador.
- Software de simulación.
- Microcontrolador.
- Circuito integrado MAX232.
- Cable de conversión SERIE-USB.
- Resistencia variable.
- 2 displays de 7 segmentos.
- Circuito integrado o componentes discretos para desarrollo del puente H.
- Motor DC.
- Motor a pasos.

DESARROLLO DE LA PRÁCTICA

1. Configurar el microcontrolador, Figura 2, considerando:
   a) Configurar pines necesarios del microcontrolador para operar como salidas simples (SO), con el propósito de controlar la activación los displays de 7 segmentos, así como las fases del motor a pasos.
   b) Habilitar el ADC con la mayor resolución posible del microcontrolador.
   c) Habilitar el módulo PWM del microcontrolador.
   d) Habilitar los puertos de comunicación serial del microcontrolador y configurar sus parámetros. Los valores de los parámetros deben ser los mismos tanto la PC como el microcontrolador (velocidad de transmisión (baud rate), bits de datos, bits de parada, paridad) para poder comunicarse.
2. Diseñar un circuito (de acuerdo al diagrama de bloques, Figura 2) para comunicar el microcontrolador con la PC:
   a) Para lograr la comunicación serial entre en microcontrolador y la PC se debe de utilizar un adaptador USB – Serial (en caso de que la PC no cuente con puerto serial).
   b) Conectar la señal de la resistencia variable al ADC.
   c) Conectar las salidas simples de un puerto del microcontrolador a los pines de los displays de 7 segmentos.
   d) Dado que es recomendable proteger el microcontrolador contra sobre corrientes, se debe colocar una etapa de potencia entre este y los motores DC y a pasos.
3. Desarrollar un programa para el microcontrolador el cual consiste en un menú de actividades, que además podrá visualizarse en la PC, este menú debe de contener las siguientes instrucciones: lectura del nivel de voltaje en el ADC, ingreso de un número de dos dígitos como máximo desde la PC para ser mostrado en los displays de 7 segmentos,

controlar el sentido de giro y la velocidad del motor a pasos, controlar el ciclo de trabajo del PWM del microcontrolador, de esta manera controlar la velocidad de un motor DC.

4. Una vez que la simulación sea la correcta, desarrollar el armado físico de circuito.

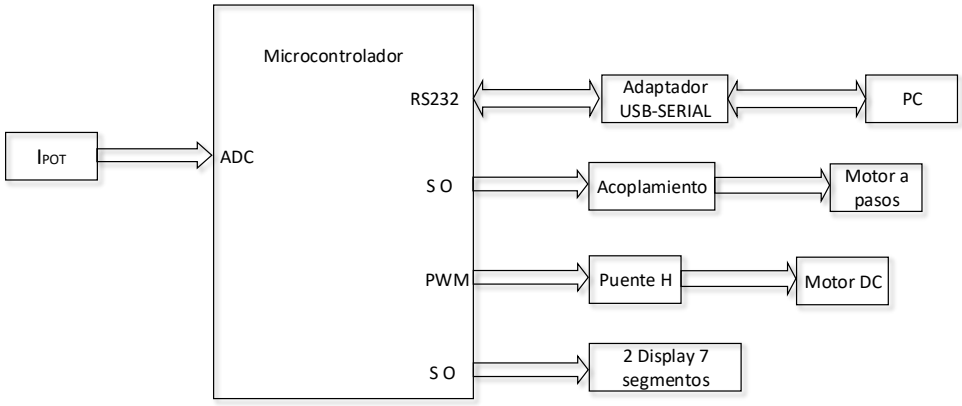
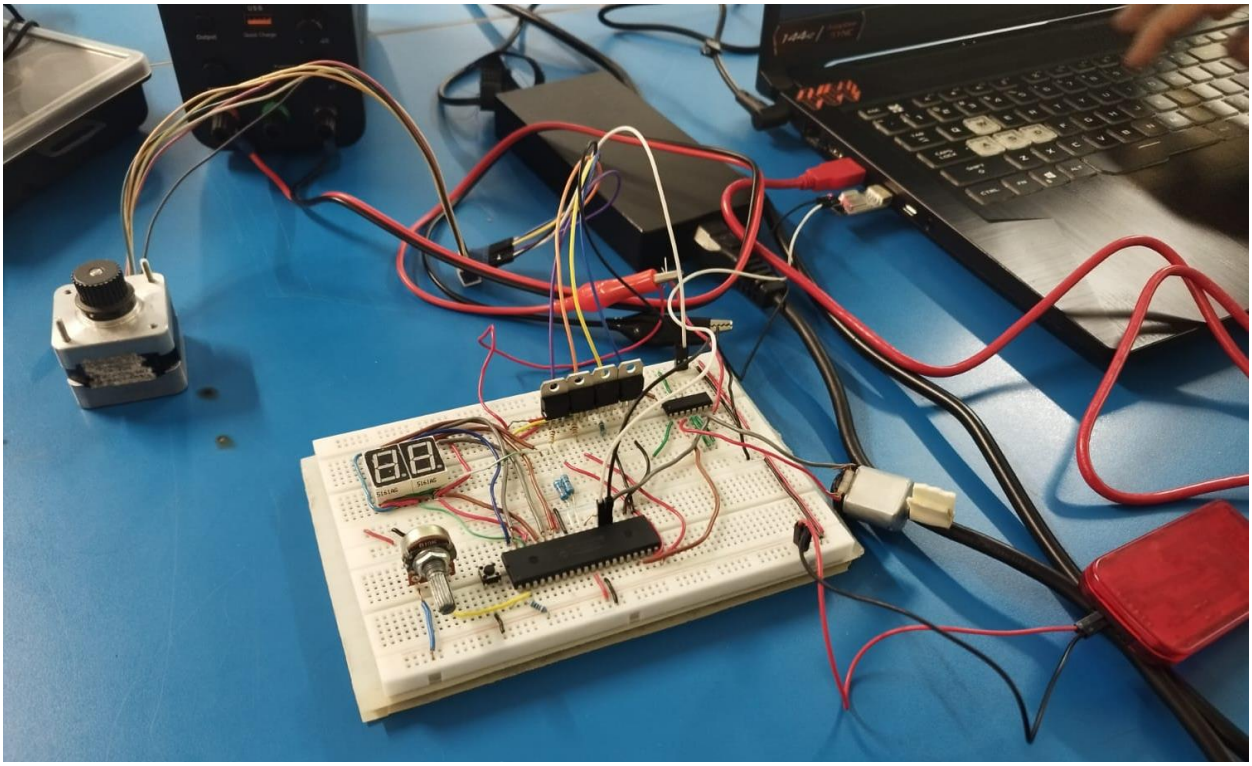
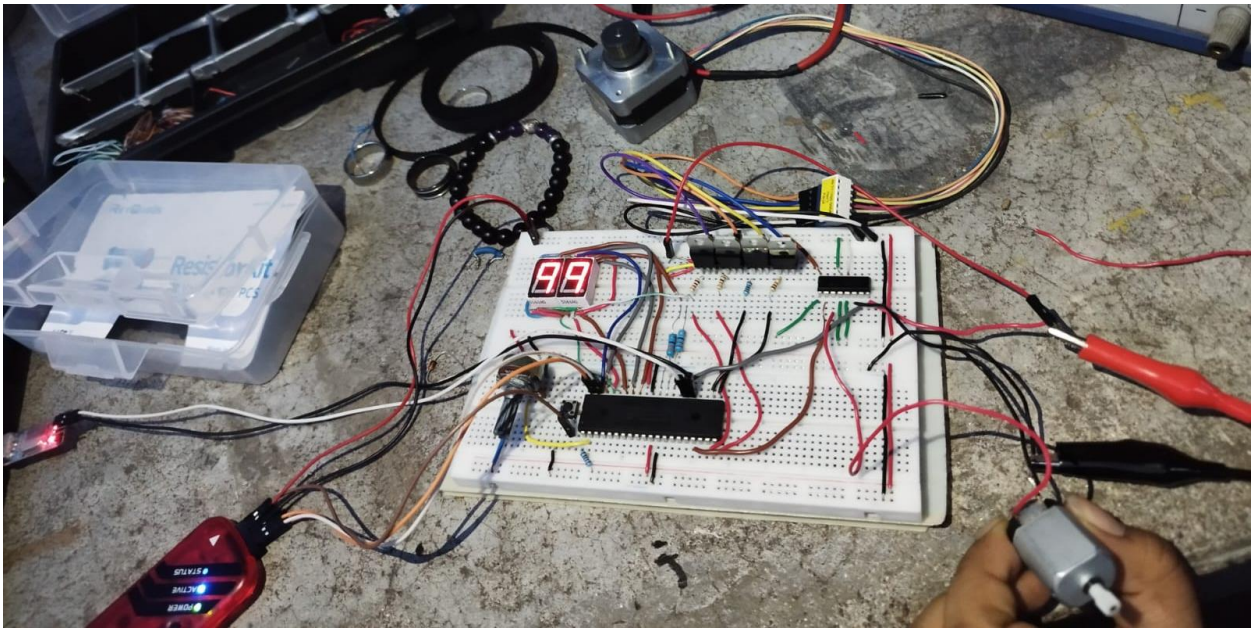


Figura 2. Diagrama de bloques para el circuito *Comunicación RS232.*

Resultados de la práctica





Circuito armando

Código explicado

```
#include <18f4550.h>   // Libreria del Microcontrolador
#device adc = 10 // Resolucion del ADC en bits
#fuses INTRC, NOWDT, NOPROTECT, NOLVP, CPUDIV1, PLL1 // Fusibles
(Configuraciones del microcontrolador)
#use delay(clock = 8M)
#use rs232(rcv = pin_c7, xmit = pin_c6, baud = 9600, bits = 8, parity = n)
// rs232(rcp = (Pin receptor), xmit = (pin transmisor), baud = (Velocidad
de transferencia), bits = 8, parity = n)

// CONSTANTES
#define SALIR '*'

#define D1 PIN_D7
#define D2 PIN_D6

const int16 DutyMAX = 500; // Maximo valor Duty Cicle
const int16 Minms = 10, Maxms = 300; // Maximo y minimo valor para ms de
motor a pasos

// Variables GLOBALES
int16 duty_actual = DutyMAX;
int16 MS_actual = 10; // Variable de milisegundos en motor a pasos
int Num_Catodo[] = {0x3F, 0x06, 0x5B, 0x4F, 0x66,
0x6D,0x7D,0x07,0x7F,0x67}; // Numeros para display Catodo
char Numeros[] = {'0','0'}; // Numeros Impresos en los display
int contnum = 0; // Contador para Ingreso de numeros

// ESTRUCTURAS
struct menu
```

```
{
   char Opcion; // Opcion actual de menu
   int Impbool; // Booleano de impresion de menu
};

struct menu Principal = {'0',1};
struct menu MotorDC = {'0',1};
struct menu MotorPasos = {'0',1};
struct menu Displays = {'0',1};
struct menu DutySelect ={'0',1};
struct menu VelocidadMP ={'0',1};

struct Booleanos
{
   int MotorDC;
   int MotorPasos;
   int SentidoMotorPasos; // 0 izquierda 1 derecha
   int Displays;
};

struct Booleanos Bool = {0,0,0,0};

// INTERRUPCION
#int_rda
void RecibirDatos()
{
   switch(Principal.Opcion) // MENU PRINCIPAL
   {
      case '0': // No se ha recibido opcion
         Principal.Opcion = getc();
      break;
      case '2': // Motor de corriente directa
      {
         switch(MotorDC.Opcion)
         {
            case '0':
               MotorDC.Opcion = getc();
            break;

            case '3':
               DutySelect.Opcion = getc();
            break;
         }
      }
      break;
      case '3': // Motor a pasos
      {
         switch(MotorPasos.Opcion)
         {
            case '0':
               MotorPasos.Opcion = getc();
            break;
            case '3':
               VelocidadMP.Opcion = getc();
```

```
            break;
         }
      }
      break;
      case '4': // Displays
      {
         switch(Displays.Opcion)
         {
            case '0':
               Displays.Opcion = getc();
            break;
            case '2':
               numeros[contnum] = getc();
               printf("%c", numeros[contnum++]);
            break;
         }
      }
      break;
   }
}

// FUNCIONES
void IMP_Menus();

void ROTACION_MDC();
void ROTACION_MP();
void IMP_Display();

void main()
{
   // HABILITAMOS INTERRUPCIONES
   enable_interrupts(GLOBAL);
   enable_interrupts(int_rda);

   // CONFIFGURACION DE ADC
   setup_adc(adc_clock_div_2); // Sincronizamos las frecuencias
   setup_adc_ports(AN0);
   delay_us(10);
   set_tris_a(0b00000001); // 1 entrada 0 salida

   // CONFIGURACION DE PWM
   setup_timer_2(T2_DIV_BY_16, 124, 1); // Primer parametro modificamos el
preescalador
   setup_ccp1(CCP_PWM | CCP_PWM_HALF_BRIDGE, 1); // Activa el PMW en P1A y
P1B. (Activar PWM, Modo de puente H, Desfase (En funcion de la cuenta del
timer ))

   while(true)
   {
      // IMPRESION DE MENUS
      IMP_Menus();

      // EJECUCION DE INSTRUCCIONES
      ROTACION_MDC();
```

```
      ROTACION_MP();
      IMP_Display();
   }
}

void IMP_Menus() // IMPRESION DE MENUS
{
   static int booldigitos = 1;
   int16 data;

   if(Principal.Impbool) // Impresion de menu 1
   {
      printf("\r\n\r\nMENU PRINCIPAL\r\n\r\n1. Imprimir valor
Potenciometro\r\n\r\n2. Motor de corriente directa\r\n\r\n3. Motor a
pasos\r\n\r\n4. Displays\r\n ");
      printf("\r\nIngrese presione el numero de la opcion que desea\r\n");

      Principal.Impbool = 0; // Ya se imprimio el menu
   }

   switch(Principal.Opcion)
   {
      case '1': // Imprimir valor Potenciometro
      {
         set_adc_channel(0);
         delay_us(10);

         data = read_adc();
         printf("\r\n\r\nEL VALOR DEL ADC ES: %li\r\n\r\nQUE EQUIVALE A %f
Volts\r\n", data , (float) data/1023.0*5.0);

         // Volver a recibir el menu despues de hacer la accion
         Principal.Opcion = '0';
         Principal.Impbool = 1;
      }
      break;
      case '2': // Motor de corriente directa
      {
         if(MotorDC.Impbool)
         {
            printf("\r\n\r\nMOTOR DE CORRIENTE DIRECTA\r\n\r\n1.
Encender/Apagar motor\r\n\r\n2. Cambiar sentido de giro\r\n\r\n3.
Modificar valor Duty Cicle\r\n");
            printf("\r\nIngrese presione el numero de la opcion que desea,
para salir presione *\r\n");

            MotorDC.Impbool = 0; // Ya se imprimio el menu
         }

         switch(MotorDC.Opcion)
         {
            case '1': // Encender/Apagar motor
            {
               Bool.MotorDC = !Bool.MotorDC;
```

```
               printf("\r\nProceso Exitoso\r\n");

               // REALIZAR ACCION CORRESPONDIENTE Y REINICIAR MENU ACTUAL
PARA VOLVER A ESPERAR RESPUESTA
               MotorDC.Opcion = '0';
               MotorDC.Impbool = 1;
            }
            break;
            case '2': // Cambiar sentido de giro
            {
               duty_actual = dutyMAX - duty_actual;

               printf("\r\nProceso Exitoso\r\n");

               // REALIZAR ACCION CORRESPONDIENTE Y REINICIAR MENU ACTUAL
PARA VOLVER A ESPERAR RESPUESTA
               MotorDC.Opcion = '0';
               MotorDC.Impbool = 1;
            }
            break;
            case '3': // Modificar valor Duty Cicle
            {
               // VER SI PUEDO PONER OTRO MENU Y PONERLE OPCIONES PARA EL
% DEL DUTY CICLE
               if(DutySelect.Impbool)
               {
                  //printf("\r\n\r\n  seleccione el valor del duty
cicle\r\n\r\n1. 0 \r\n\r\n2. 25 \r\n\r\n3. 75  \r\n\r\n4. 100 ");
                  printf("\r\n\r\nVALOR DEL DUTY CICLE \r\n\r\n1. 0 %%
\r\n\r\n2. 30 %% \r\n\r\n3. 70 %%  \r\n\r\n4. 100 %%\r\n");
                  printf("\r\nPresione el numero de la opcion que desea,
para salir presione *\r\n");

                  DutySelect.Impbool = 0; // Ya se imprimio el menu
                }

               //printf("\r\nProceso Exitoso\r\n");
              switch(DutySelect.Opcion)
              {
                  case '1':
                  {
                     duty_actual = 0;
                     printf("\r\nProceso Exitoso\r\n");

                     // REALIZAR ACCION CORRESPONDIENTE Y REINICIAR MENU
ACTUAL PARA VOLVER A ESPERAR RESPUESTA
                     DutySelect.Opcion = '0';
                     DutySelect.Impbool = 1;
                  }
                  break;
                  case '2':
                  {
                     duty_actual = dutyMAX*0.3;
```

```
                    printf("\r\nProceso Exitoso\r\n");

                    // REALIZAR ACCION CORRESPONDIENTE Y REINICIAR MENU
ACTUAL PARA VOLVER A ESPERAR RESPUESTA
                    DutySelect.Opcion = '0';
                    DutySelect.Impbool = 1;
                 }
                 break;
                 case '3':
                 {
                    duty_actual = dutyMAX*0.7;
                    printf("\r\nProceso Exitoso\r\n");

                    // REALIZAR ACCION CORRESPONDIENTE Y REINICIAR MENU
ACTUAL PARA VOLVER A ESPERAR RESPUESTA
                    DutySelect.Opcion = '0';
                    DutySelect.Impbool = 1;
                 }
                 break;
                 case '4':
                 {
                    duty_actual = dutyMAX;
                    printf("\r\nProceso Exitoso\r\n");

                    // REALIZAR ACCION CORRESPONDIENTE Y REINICIAR MENU
ACTUAL PARA VOLVER A ESPERAR RESPUESTA
                    DutySelect.Opcion = '0';
                    DutySelect.Impbool = 1;
                 }
                 break;
                 case SALIR:
                    // REINICIAMOS ESTE MENU Y EL ANTERIOR
                    MotorDC.Opcion = DutySelect.Opcion = '0';
                    MotorDC.Impbool = DutySelect.Impbool = 1;
                 break;
                 default:
                    DutySelect.Opcion = '0';
                 break;
              }
           }
           break;
           case SALIR:
           {
              // REINICIAMOS ESTE MENU Y EL ANTERIOR
              Principal.Opcion = MotorDC.Opcion = '0';
              Principal.Impbool = MotorDC.Impbool = 1;
           }
           break;
           default: // Opcion no valida
              MotorDC.Opcion = '0';
           break;
        }
     }
     break;
```

```
      case '3': // Motor a pasos
      {
         if(MotorPasos.Impbool)
         {
            printf("\r\n\r\nMOTOR A PASOS\r\n\r\n1. Encender/Apagar
motor\r\n\r\n2. Cambiar sentido de giro\r\n\r\n3. Modificar
velocidad\r\n");
            printf("\r\nIngrese presione el numero de la opcion que desea,
para salir presione *\r\n");

            MotorPasos.Impbool = 0; // Ya se imprimio el menu
         }

         switch(MotorPasos.Opcion)
         {
            case '1': // Encender/Apagar motor
            {
               Bool.MotorPasos = !Bool.MotorPasos;
               printf("\r\nProceso Exitoso\r\n");

               // REALIZAR ACCION CORRESPONDIENTE Y REINICIAR MENU ACTUAL
PARA VOLVER A ESPERAR RESPUESTA
               MotorPasos.Opcion = '0';
               MotorPasos.Impbool = 1;
            }
            break;
            case '2': // Cambiar sentido de giro
            {
               Bool.SentidoMotorPasos = !Bool.SentidoMotorPasos;
               printf("\r\nProceso Exitoso\r\n");

               // REALIZAR ACCION CORRESPONDIENTE Y REINICIAR MENU ACTUAL
PARA VOLVER A ESPERAR RESPUESTA
               MotorPasos.Opcion = '0';
               MotorPasos.Impbool = 1;
            }
            break;
            case '3': // Modificar velocidad
            {
               if(VelocidadMP.Impbool)
               {
                  //printf("\r\n\r\n  seleccione el valor del duty
cicle\r\n\r\n1. 0 \r\n\r\n2. 25 \r\n\r\n3. 75  \r\n\r\n4. 100 ");
                  printf("\r\n\r\nVELOCIDAD DEL MOTOR A PASOS \r\n\r\n1.
Muy lento \r\n\r\n2. Lento \r\n\r\n3. Rapido  \r\n\r\n4. En friega\r\n");
                  printf("\r\nPresione el numero de la opcion que desea,
para salir presione *\r\n");

                  VelocidadMP.Impbool = 0; // Ya se imprimio el menu
               }

               switch(VelocidadMP.Opcion)
               {
                  case '1':
```

```
                {
                   Ms_actual = Maxms;
                   printf("\r\nProceso Exitoso\r\n");

                   // REALIZAR ACCION CORRESPONDIENTE Y REINICIAR MENU
ACTUAL PARA VOLVER A ESPERAR RESPUESTA
                   VelocidadMP.Opcion = '0';
                   VelocidadMP.Impbool = 1;
                }
                break;
                case '2':
                {
                   Ms_actual = Maxms - (Maxms - Minms) * 0.45;
                   printf("\r\nProceso Exitoso\r\n");

                   // REALIZAR ACCION CORRESPONDIENTE Y REINICIAR MENU
ACTUAL PARA VOLVER A ESPERAR RESPUESTA
                   VelocidadMP.Opcion = '0';
                   VelocidadMP.Impbool = 1;
                }
                break;
                case '3':
                {
                   Ms_actual = Maxms - (Maxms - Minms) * 0.75;
                   printf("\r\nProceso Exitoso\r\n");

                   // REALIZAR ACCION CORRESPONDIENTE Y REINICIAR MENU
ACTUAL PARA VOLVER A ESPERAR RESPUESTA
                   VelocidadMP.Opcion = '0';
                   VelocidadMP.Impbool = 1;
                }
                break;
                case '4':
                {
                   Ms_actual = Maxms - (Maxms - Minms);
                   printf("\r\nProceso Exitoso\r\n");

                   // REALIZAR ACCION CORRESPONDIENTE Y REINICIAR MENU
ACTUAL PARA VOLVER A ESPERAR RESPUESTA
                   VelocidadMP.Opcion = '0';
                   VelocidadMP.Impbool = 1;
                }
                break;
                case SALIR:
                   // REINICIAMOS ESTE MENU Y EL ANTERIOR
                   MotorPasos.Opcion = VelocidadMP.Opcion = '0';
                   MotorPasos.Impbool = VelocidadMP.Impbool = 1;
                break;
                default:
                   VelocidadMP.Opcion = '0';
                break;
             }

          }
```

```
            break;
            case SALIR:
            {
               // REINICIAMOS ESTE MENU Y EL ANTERIOR
               Principal.Opcion = MotorPasos.Opcion = '0';
               Principal.Impbool = MotorPasos.Impbool = 1;
            }
            break;
            default: // Opcion no valida
               MotorPasos.Opcion = '0';
            break;
         }
      }
      break;
      case '4': // Displays
      {
         if(Displays.Impbool)
         {
            printf("\r\n\r\nCONTROL DE DISPLAYS\r\n\r\n1. Encender/Apagar
display\r\n\r\n2. Ingresar digitos\r\n");
            printf("\r\nIngrese presione el numero de la opcion que desea,
para salir presione *\r\n");

            Displays.Impbool = 0; // Ya se imprimio el menu
         }

         switch(Displays.Opcion)
         {
            case '1': // Encender/Apagar display
            {
               Bool.Displays = !Bool.Displays; // Activamos los displays

               printf("\r\nProceso Exitoso\r\n");

               // REALIZAR ACCION CORRESPONDIENTE Y REINICIAR MENU ACTUAL
PARA VOLVER A ESPERAR RESPUESTA
               Displays.Opcion = '0';
               Displays.Impbool = 1;
            }
            break;
            case '2': // Ingresar digitos
            {
               if(booldigitos)
               {
                  printf("\r\n\r\nIngrese los 2 numeros para el display:
");
                  booldigitos = 0;
               }

               // Detectar si el numero es valido
               if((numeros[0] < '0') || (numeros[0] > '9') || (numeros[1]
< '0') || (numeros[1] > '9') || (numeros[1] == 13))
               {
                  if(numeros[1] == 13)
```

```
                  {
                     numeros[1] = numeros[0];
                     numeros[0] = '0';
                  }
                  else
                  {
                     contnum = 0;
                     booldigitos = 1;
                     numeros[0] = numeros[1] = '0';

                     printf("\r\n\r\nIngrese numeros validos");
                  }
               }

               // REALIZAR ACCION CORRESPONDIENTE Y REINICIAR MENU ACTUAL
PARA VOLVER A ESPERAR RESPUESTA
               if(contnum > 1)
               {
                  printf("\r\n");
                  contnum = 0;
                  booldigitos = 1;

                  Displays.Opcion = '0';
                  Displays.Impbool = 1;
               }
            }
            break;
            case SALIR:
            {
               Principal.Opcion = Displays.Opcion = '0'; // Ya no tenemos
opciones en los menus
               Principal.Impbool = Displays.Impbool = 1; // Activamos la
impresion de los menus
            }
            break;
            default: // Opcion no valida
               Displays.Opcion = '0';
            break;
         }
      }
      break;
      default: // Opcion no valida
         Principal.Opcion = '0';
      break;
   }
}

void ROTACION_MDC() // ROTACION MOTOR CORRIENTE DIRECTA
{
   if(Bool.MotorDC)
      set_pwm1_duty(duty_actual);
   else
      set_pwm1_duty(dutyMAX / 2);
}
```

```
void ROTACION_MP() // ROTACION MOTOR A PASOS
{
   static int16 Posicion = PIN_D0 - 1;

   if(Bool.MotorPasos)
   {
      if(Bool.SentidoMotorPasos) // Derecha
      {
         Posicion++;
         if(Posicion > PIN_D3)
            Posicion = PIN_D0;
      }
      else // Izquierda
      {
         Posicion--;
         if(Posicion < PIN_D0)
            Posicion = PIN_D3;
      }

      output_high(Posicion);

      if(Bool.Displays) // Si estan activados los display
         for(int i = 0; i < (Ms_actual/10);i++)
            IMP_Display();
      else
         delay_ms(Ms_actual);

      output_low(Posicion);
   }
}

void IMP_Display() // IMPRESION EN DISPLAY
{
   if(Bool.Displays)
   {
      // Encendemos primer display
      output_high(D2);
      output_low(D1);

      output_b(Num_Catodo[Numeros[0] - 48]);

      delay_ms(5);

      // Encendemos segundo display
      output_high(D1);
      output_low(D2);

      output_b(Num_Catodo[Numeros[1] - 48]);

      delay_ms(5);
   }
   else
   {
```

```
      output_high(D1);
      output_high(D2);
   }
}
```

Conclusiones de la práctica

Joel Zúñiga:

Logramos mediante un USB to TTL comunicarnos con el pic desde una computadora, aprendí sobre el protocolo de comunicación RS232, y como utiliza algo parecido al PWM para enviar información.

Daniela Manríquez:

En esta práctica reforcé los conocimientos adquiridos en la práctica pasada de cómo utilizar el puente H. Así como también aprendí el concepto y como es que funciona la comunicación serial mediante un módulo TTL para controlar dispositivos. En la cuestión de la programación aprendí como es que funcionan las interrupciones en el código, las cuales usamos para recolectar datos en tiempo real sin necesidad de detener el código y por ende todo lo demás seguía funcionando.

Bibliografía

García E. (2008). Compilador C CCS y simulador PROTEUS para microcontroladores PIC: Alfa Omega.

Valez F. (2007). Microcontroladores: Fundamentos y aplicaciones con PIC. España: ALFAOMEGA.