Departamento de Tecnología Electrónica
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Introducción a AVR-STUDIO

Enunciados de Prácticas de Laboratorio
Estructura de Computadores

1. Introducción y objetivos

Los objetivos de la sesión de laboratorio son los siguientes:

- Introducir el entorno de programación y depuración de microcontroladores de ATMEL[1] llamado AVR-STUDIO.
- Realizar la simulaciones de programas escritos en lenguaje ensamblador para el microcontrolador ATMEGA328P.
- Realizar la programación de un microcontrolador mediante AVR-STUDIO mediante la plataforma AVR-DRAGON.
- Realizar la programación, depuración de programas y control del microcontrolador desde AVR-STUDIO.

En esta sesión de laboratorio se utilizará el entorno de desarrollo AVR-STUDIO para programamar el microcontrolador ATMEGA328P que se encuentra en una placa de desarrollo llamada Arduino[2] Duemilanove. La programación se realiza mediante la plataforma de depuración/programación AVR-DRAGON también del fabricante ATMEL.

AVR-STUDIO puede descargarse gratuitamente de desde las páginas del fabricante de ATMEL en http://www.atmel.com. Respecto a los Arduinos, éstos están diseñados para ser programados en un lenguaje de programación propio, transfiriéndose los programas a través de su puerto USB. En esta sesión de laboratorio no se utilizarán estas características, es decir, se programarán directamente en ensamblador. Por ello, se han realizado modificaciones en dichas placas. Aunque no es relevante para esta sesión de laboratorio, se puede consultar toda la información adicional sobre esta placas en http://www.arduino.cc

1 Fabricante de microcontroladores, más información en http://www.atmel.com
2 Plataforma opensource de prototipado electrónico, mas información en http://www.arduino.cc

Durante la sesión de laboratorio se debe disponer de los ficheros indicados en la tabla 1. Algunos de los ficheros deben ser completados en el estudio teórico y otros se completarán durante la sesión de laboratorio.

Nombre del fichero	Contenido	Descripción
contador_0_10.asm	Programa contador de 0 a 10	Debe completarlo el alumno antes de asistir a la sesión de laboratorio.
contador_0_1000.asm	Programa contador de 0 a 1000	Debe completarlo el alumno antes de asistir a la sesión de laboratorio.
conmutadores.asm	Programa de control de conmutadores y leds	Debe completarlo durante la sesión de laboratorio.
contador_bcd.asm	Programa contador de pulsaciones en BCD	Debe completarlo durante la sesión de laboratorio.

Tabla 1. Ficheros necesarios durante la sesión de laboratorio.

Es **obligatorio** traer los programas del estudio teórico preparados para utilizarlos durante el desarrollo de la sesión de laboratorio.

2. Estudio teórico

Se deben realizar dos programas en lenguaje ensamblador que incrementen un valor almacenado desde 0 hasta un valor determinado. Dichos programas cuando terminen la cuenta volverán a empezar de nuevo la cuenta desde 0.

A continuación se detallan los programas:

1. Programa contador 0 a 10: Realizar un programa en ensamblador que cuente de 0 a 10 utilizando un registro del microcontrolador. Cuando termine la cuenta el programa debe invertir el valor del PINC0 y volver a empezar, es decir, volverá a contar de 0 a 10 e invertirá el PINC0. Así indefinidamente.

   Para realizar el programa correctamente se debe configurar el puerto C como salida, para ello se propone comenzar el programa utilizando el siguiente fragmento de código (*fichero contador_0_10.asm*):

```
; Programa contador de 0 a 10
; Cada vez que se pase por 10 se debe invertir el PINC0

.INCLUDE "m328pdef.inc"
.DEF  TMP=R19

            LDI     TMP,$FF
            OUT     DDRC,TMP       ; Configura el puerto C completo como salida
```

Código 1. Fichero contador_0_10.asm, plantilla de código para el programa contador 0 a 10.

2. Programa contador 0 a 1000: Realizar un segundo programa similar al anterior donde ahora la

cuenta debe ser desde 0 a 1000. Tenga en cuenta que los registros del microcontrolador son de 8 bits y solo pueden contar de 0 a 255. El fichero debe llamarse *contador_0_100.asm* y puede utilizar la plantilla de código suministrada.

3. Estudio experimental

El estudio experimental se divide en dos bloques, el primero consiste en utilizar el programa realizado en el estudio teórico para familiarizarse con el entorno de desarrollo AVR-STUDIO. El segundo consistirá en completar fragmentos de código de algunos programas donde se controlará la entrada y salida del microcontrolador.

Antes de comenzar el estudio experimental asegúrese de disponer de todos los ficheros indicados en la tabla 1.

3.1. Introducción a AVR-STUDIO

Se utilizarán los programas realizados en el estudio teórico en el entorno de desarrollo AVR-STUDIO. Los pasos para crear un proyecto nuevo y poder escribir el código del programa se detalla a continuación en esta sección.

Una vez iniciado AVR STUDIO aparece un asistente para creación o apertura de un nuevo proyecto tal y como se muestra en la figura 1. Si no apareciera el asistente hay que acceder al menú Project y seleccionar la opción Project Wizard.

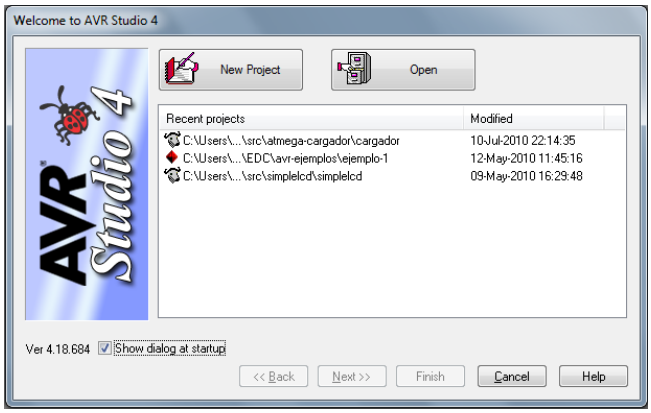

Figura 1. Asistente para creación o apertura de un proyecto.

Se debe seleccionar un nuevo proyecto (botón *New Project*) y aparecerá el siguiente diálogo del asistente (figura 3) donde, habrá que indicar el nombre del proyecto, el directorio y seleccionar la opción *Atmel AVR Assembler.* Antes de pulsar el botón *Next* seleccione adecuadamente la opción *Create initial file*, tiene dos opciones, seleccionar o no seleccionar dicha opción (observe la marca roja en la figura 2):

1. Si no selecciona esta opción, el proyecto se creará sin ningún archivo de texto asociado. Esto le permite posteriormente utilizar un fichero de texto que tenga en si disco con el programa ya escrito. De esta forma evita tener que teclear el programa de nuevo
2. Si lo selecciona, se creará un nuevo fichero vacío en el que deberá teclear el programa. Si ya trae

el programa escrito en otro fichero tendrá que copiar y pegar el código desde el bloc de notas a AVR-STUDIO.

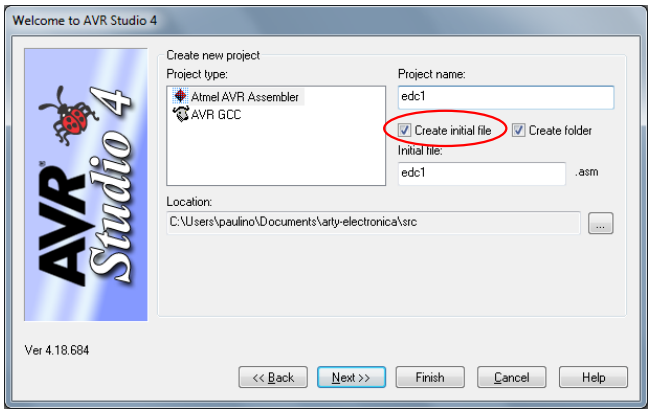

Figura 2. Selección de tipo y nombre de proyecto.

Tras escoger la opción que le interese en cada caso, tras pulsar el botón *Next* aparecerá la última ventana de asistente. Aquí debe seleccionar las opciones *AVR Simulator* y *ATMega328P* tal y como se muestra en la figura 3.

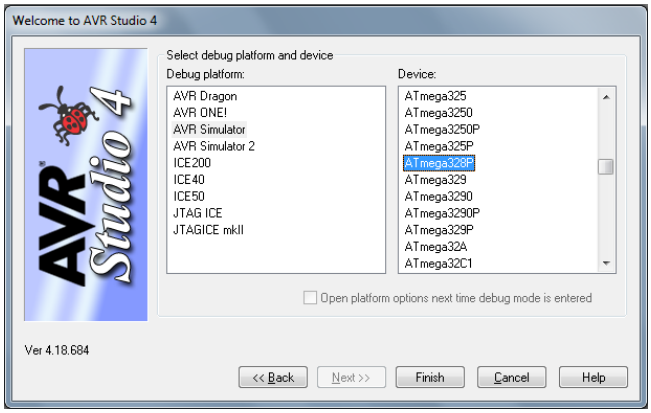

Figura 3. Selección de tipo y nombre de proyecto.

En caso de no haber seleccionado la opción *Create initial file* tendrá un proyecto vacío al que hay que añadir un programa previamente escrito en un fichero. Para realizar esto, hay que utilizar el botón derecho del ratón en la raíz del árbol de proyecto y aparecerá un menú flotante como el mostrado en la figura 4. Con la opción Add files to project podemos seleccionar del disco el fichero con el programa en ensamblador que se desee.

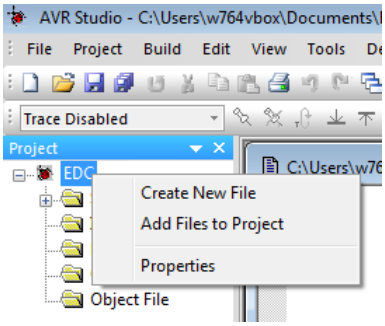

Figura 4. Añadir ficheros al proyecto.

Tras estos pasos aparece en el árbol de proyecto el nombre del fichero ensamblador sobre el que trabajar. Pulsando con el ratón dos veces sobre este nombre del fichero aparece una ventana en la que puede editar el código ensamblador del programa.

Una vez se ha terminado de escribir el programa hay que realizar el ensamblado del código. Este paso se realiza en menú Build, opción Build. También puede utilizar el icono  de la barra de herramientas. Si el código es correcto debe aparecer en la parte inferior información referente al programa ensamblado:

```
C:\Users\paulino\Documents\edc1\edc1.asm(4): Including file 'C:\Program Files (x86)\Atmel\AVR
Tools\AvrAssembler2\Appnotes\m168DEF.INC'

C:\Users\paulino\Documents\edc1\edc1.asm(23): No EEPROM data, deleting
C:\Users\paulino\Documents\edc1\edc1.eep

ATmega168 memory use summary [bytes]:
Segment   Begin    End      Code   Data   Used    Size   Use%
----------------------------------------------------------------
[.cseg] 0x000000 0x000022     34      0     34   16384   0.2%
[.dseg] 0x000100 0x000100      0      0      0    1024   0.0%
[.eseg] 0x000000 0x000000      0      0      0     512   0.0%

Assembly complete, 0 errors. 0 warnings
```

Código 2. Salida de la construcción del programa contador.

En caso de producirse errores, en la ventana inferior aparecerá el número de línea del programa donde está el error.

3.1.1. Ejecución en el simulador del programa

AVR-STUDIO incluye un simulador con el cual se puede visualizar el estado del microcontrolador durante la ejecución de un programa. Entre las diversas opciones que ofrece el simulador nos centraremos en la posibilidad de ejecutar instrucción a instrucción un programa y la posibilidad de ejecutar un programa hasta que llegue a una instrucción determinada.

Para comenzar la simulación del programa hay que acceder al menú Debug y utilizar la opción Start Debugging. Tras esto aparecen diferentes ventanas (ver figura 5) que componen el simulador:

- Ventana del Procesador: Situada en la parte izquierda, muestra el estado interno de procesador (Frecuencia, contador de ciclos del reloj) y el contenido de los registros: PC, SP, X, Y, Z, SREG y los 32 registros internos.
- Ventana de dispositivos de E/S: Situada en la parte superior derecha, muestra en forma de árbol todos los dispositivos que tiene el microcontrolador seleccionado. En esta primera práctica de debe seleccionar el puerto C, de igual modo que se ha seleccionado la en la figura 5. De esta forma se visualizan los tres registros que forman el puerto.
- Ventana de visualización de Memoria: Situada en la parte inferior derecha, permite ver en tiempo real el contenido de la memoria del microcontrolador. Se puede seleccionar entre memoria de programa, memoria SRAM y EEPROM. Principalmente interesará ver el contenido de la memoria

SRAM a partir de la dirección $100. Se debe recordar que hasta la dirección $99 están mapeados los periféricos, por lo que no se debe usar como espacio de almacenamiento para los programas.

Para comprobar el funcionamiento del programa se debe realizar la ejecución paso a paso observando como cambian los valores de los registros y el PINC0. Hay que desplegar los registros en la ventana del procesador y el puerto C en la ventana de E/S para visualizar los registros del puerto.

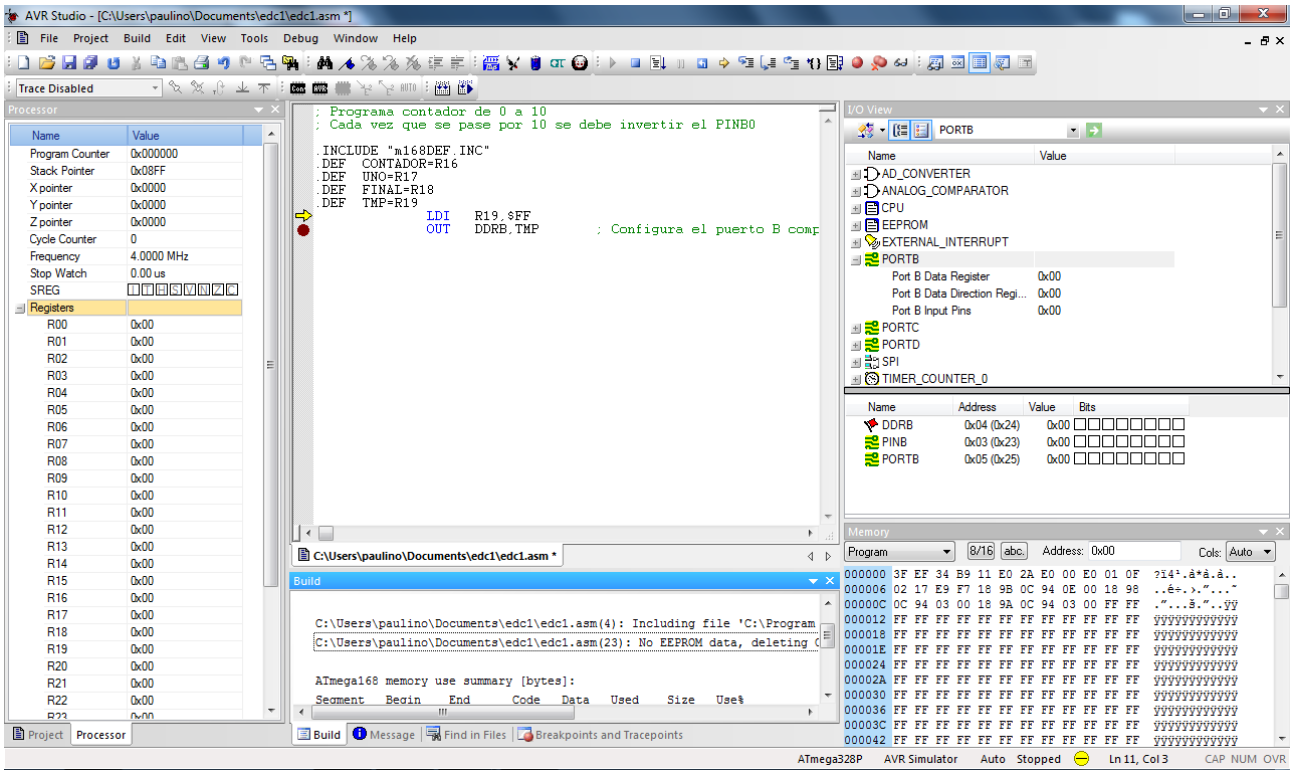

Figura 5. Visión global del modo de depuración de AVR-STUDIO.

El simulador permite la ejecución instrucción a instrucción del programa. El indicador situado en la parte izquierda indica la siguiente instrucción que se ejecutará. En el menú Debug (figura 6) se pueden encontrar diversas acciones útiles durante la simulación. En la ejecución paso a paso las funciones más utilizadas son:

- Step Over: (icono) Ejecuta instrucciones hasta la siguiente línea, en caso de ser una llamada a una subrutina, la ejecuta completamente para avanzar a la siguiente línea de código.
- Step Into: (icono) Ejecuta una instrucción, en caso se existir una llamada a subrutina, realiza la llamada y se sitúa en la primera instrucción de la subrutina.
- Step Out: (icono) Ejecuta instrucciones hasta encontrar una instrucción de retorno de subrutina.
- Reset: (icono) Reinicia la simulación y sitúa la ejecución en la primera instrucción del programa.
- Toogle Breakpoint: (icono) Establece un punto de ruptura de ejecución. Cuando se ejecute el comando *Run* (icono), el programa se ejecutará hasta encontrar algún punto de ruptura.

- Run to Cursor: (icono) Ejecuta instrucciones hasta la instrucción en la que está el cursor.

A continuación inicie la simulación con Start Debuggin y ejecute paso a paso el programa del estudio teórico. Puede utilizar la tecla F10 para no tener que utilizar los menús. Compruebe que su programa opera correctamente desplegando el puerto C en la ventana E/S y realice las siguientes tareas:

1. Ensamble y simule el primer programa del estudio teórico (*contador_0_10.asm*) programa y obtenga el número de ciclos que tarda el su programa en conmutar de valor el PINC0.
2. Calcule la frecuencia del señal cuadrada que se generará sabiendo que el procesador funciona a una frecuencia de 1Mhz. A partir de la frecuencia y el número de ciclos que tarda su programa en conmutar el PINC0 puede realizar este cálculo.
3. Utilice la opción AutoStep (icono) y compruebe la ejecución animada que realiza el simulador del programa.

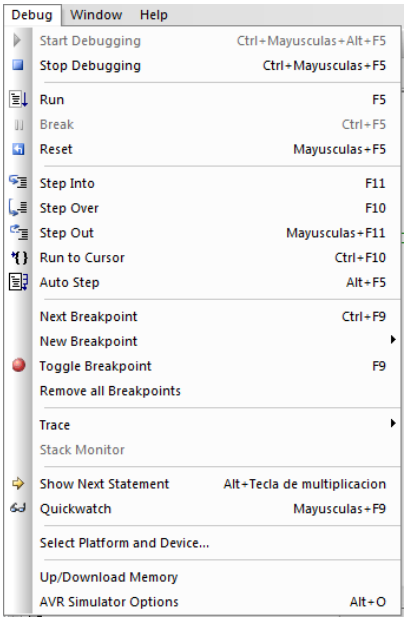

Figura 6. Menú de depuración.

3.1.2. Programación del microcontrolador

El siguiente paso consiste en la programación con el programador AVR-DRAGON (figura 7a) de un microcontrolador ATMEGA328P en una placa Arduino (figura 7b). El entorno de pruebas utilizado en esta sesión de laboratorio está formada por tres componentes: programador AVR-DRAGON, placa de prototipo Arduino Duemilanove y placa de expansión para Arduino con componentes E/S.

La placa de expansión mostrada en la figura 8 está conectada a la placa Arduino, quedando ocultos todos los componentes del Arduino. En la placa de expansión están disponibles todos los puertos del microcontrolador en los laterales de la placa, además, estos puertos también están conectados a diversos componentes como son, leds, displays, conmutadores, etc. Estos componentes se utilizarán posteriormente para realizar programas que controlen la entrada/salida.

En primer lugar se deben conectar ambas placas a los conectores USB. No es necesaria ninguna alimentación adicional ya utilizan los 5V suministrados por el Bus USB. Tras la conexión USB puede aparecer en el ordenador algún cuadro de diálogo indicando que se ha encontrado nuevo hardware. Si esto ocurriera, debe instalar los controladores, no cancele la instalación o tendrá problemas de programación del microcontrolador.

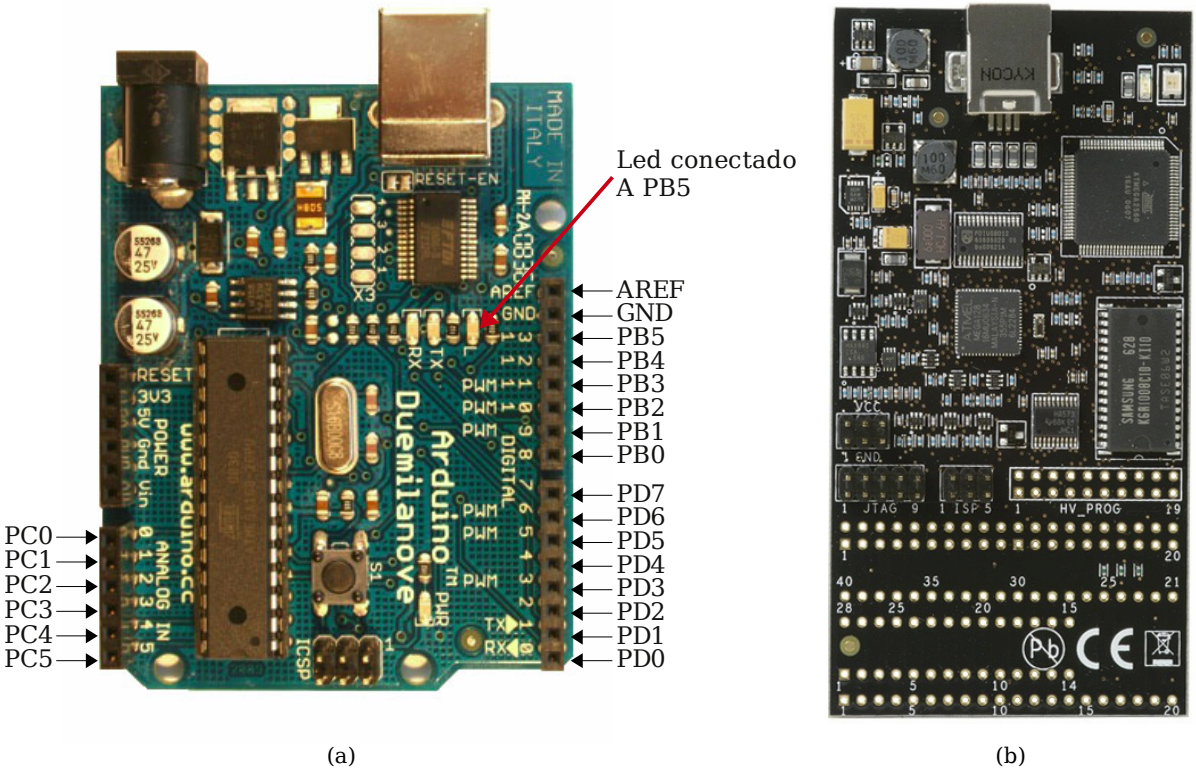

Figura 7. a) Placa de desarrollo Arduino b) Programador/Depurador AVR-DRAGON.

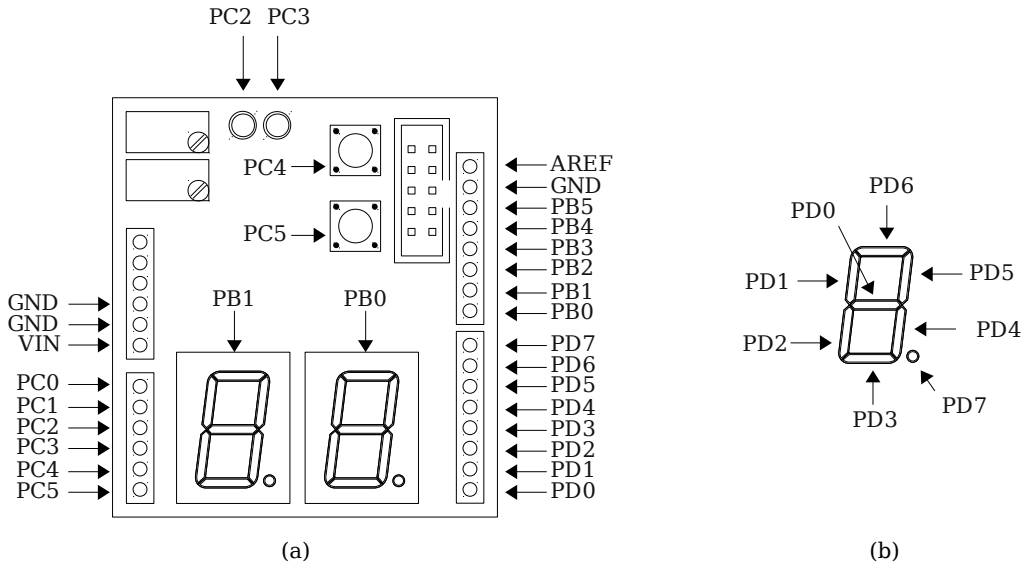

Figura 8. a) Placa de expansión E/S para Arduino. b) Detalle de conexión de los segmentos a los puertos.

La placa AVR-DRAGON dispone de dos leds, inicialmente se iluminan uno en verde y otro en rojo. El led de color rojo cambiará de color indicando el estado de la comunicación con AVR-STUDIO. La tabla 2 muestra el significado de los diferentes colores de dicho led, debemos observarlo durante los siguientes pasos para detectar posibles problemas en la programación del microcontrolador.

Color	Descripción
Rojo	En reposo, no hay conexión con AVR Studio
Apagado	En reposo, conectado a AVR Studio
Verde	Transfiriendo datos
Amarillo	Inicialización o actualización del firmware

Tabla 2. Indicaciones del led de AVR-DRAGON.

Antes de realizar la programación se debe verificar la correcta configuración de AVR-STUDIO realizando una prueba de conexión con el microcontrolador. Accediendo al menú Tools hay que usar el submenú Program AVR y, opción Connect. Aparecerá el diálogo mostrado en la figura 9. Alternativamente, dicho diálogo se puede obtener de manera directa utilizando el botón de la barra de herramientas.

En este diálogo hay que establecer la configuración indicada en la figura 9: plataforma AVR-DRAGON y puerto USB. Tras pulsar el botón *Connect*, si la conexión es correcta, debe aparecer automáticamente el diálogo mostrado en la figura 10 y el led rojo de AVR-DRAGON se apagará.

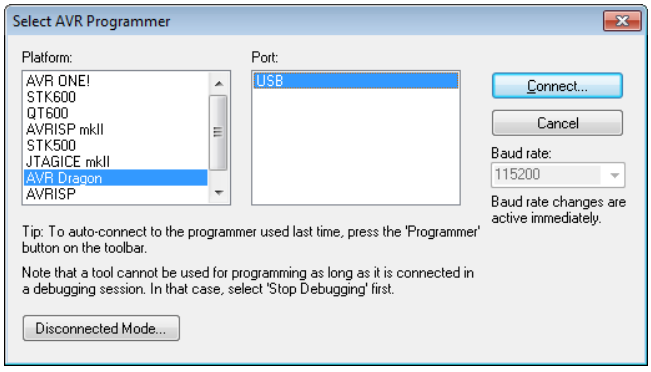

Figura 9. Selección del programador y el puerto

En caso de no aparecer automáticamente el dialogo de la figura 10 se puede utilizar el botón de la barra de herramientas o, la opción de menú Tools submenú Program AVR. Tras esto finalmente aparecerá la ventana mostrada en la figura 10.

De las múltiples pestañas que contiene sólo utilizaremos la primera y segunda: *Main* y *Program*. En primer lugar se realizará una prueba de comunicación siguiendo estos pasos:

1. Seleccionar la pestaña *Main*.
2. Seleccionar el microcontrolador correcto del cuadro desplegable indicado con *Device and Signature Bytes*. En estas placas disponemos del microcontrolador ATMega328P.

3. Pulsar el botón *Read Signature*. El programa debe responder con el texto *Signature matches selected device*. Si respondiera con un error, se debe volver a desplegar el cuadro selector de microcontrolador, seleccionar el correcto, y volver a realizar el test de comunicación.

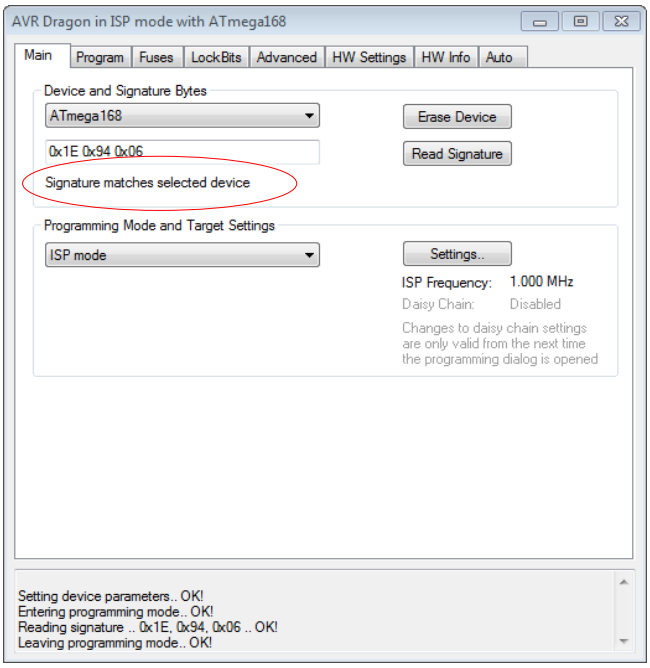

Figura 10. Pestaña principal de la ventana de programación del microcontrolador.

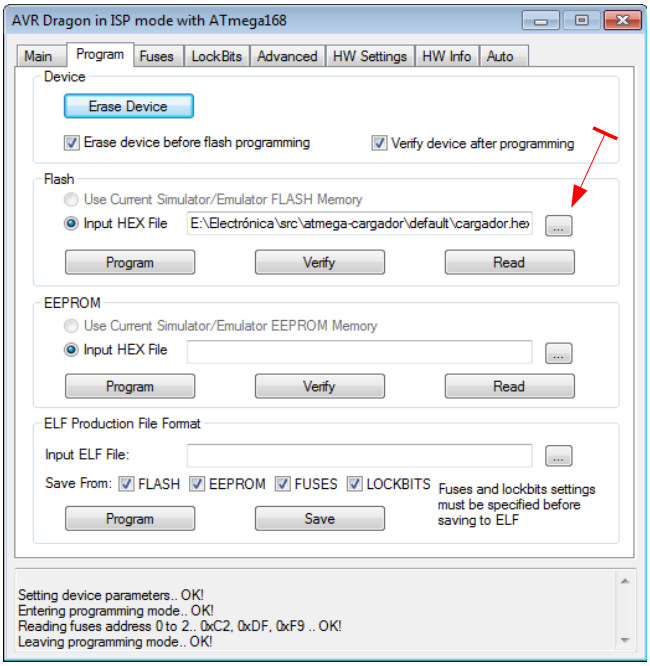

Figura 11. Pestaña de programación de la ventana de programación del microcontrolador.

El siguiente paso consiste en realizar la programación del microcontrolador con el código que se ha ensamblado, para ello, seleccione la pestaña *Program*. Si el ensamblado se realizó con éxito habrá generado un fichero con extensión *.hex* dentro del directorio del proyecto y con el nombre del proyecto. En la figura 11 se muestra el diálogo de programación donde hay que seleccionar el fichero *.hex*. Este diálogo tiene tres cuadros donde se puede seleccionar un fichero: *Flash, EEPROM* y *ELF*. Hay que utilizar la sección sección *Flash* y el botón de selección de fichero (indicado con la flecha roja en la

figura 11). Tras esto basta con pulsar el botón *Program* para realizar la programación del microcontrolador.

Una vez realizada la programación hay que comprobar si el programa se está ejecutando correctamente. Para ello, debe conectar un canal del osciloscopio en el PINC0 y comprobar si se visualiza una señal cuadrada entre 0 y 5 voltios (no olvide conectar la tierra del osciloscopio). Utilice el esquema de la figura 8 para realizar las conexiones y realice las siguientes tareas:

1. Programe en el microcontrolador con el primer programa (*contador 0_10.asm*) y compruebe con el osciloscopio la frecuencia de la señal cuadrada generada en el PINC0.
2. Utilizando el segundo programa del estudio teórico (*contador_0_1000.asm*), compruebe con el osciloscopio la frecuencia de la señal cuadrada generada en el PINC0.
3. De forma experimental y ayudado por el resultado anterior, modifique el valor de final de cuenta del programa hasta conseguir que la frecuencia de dicha señal sea aproximadamente 5HZ. Programe el microcontrolador según modifique el programa hasta conseguir visualizar la frecuencia de 5Hz en el osciloscopio.
4. Modifique el programa de forma que el PIN que conmute sea el PINC2, de esta forma debe observar el LED amarillo parpadear 5 veces por segundo.

3.2. Realización de diversos programas de control E/S

En esta sección se van realizar una serie de programas para controlar los componentes de entrada salida de la placa de desarrollo: conmutadores, leds y displays 7 segmentos.

Debe utilizar las plantillas de código preparadas para cada programa y escribir el fragmento de código indicado en cada uno de ellos.

3.2.1. Programa para controlar los conmutadores

Se pretende realizar un programa que permita manejar los puertos de entrada salida. En concreto, se trata de activar los leds cuando se pulsa un conmutador.

En el esquema de la figura 8 aparecen dos leds y dos conmutadores que debe operar de la siguiente forma: cuando se pulse el conmutador conectado a PC5 debe encenderse el led conectado a PC2 y mantenerse encendido hasta que se vuelva a pulsar el conmutador. Además, al pulsar el conmutador conectado a PC4 se encenderá el led conectado a PC3 y permanecerá encendido hasta que se pulse nuevamente el conmutador. La tabla 3 muestra los puertos y los bits asociados a los componentes así como la configuración necesaria para que operen correctamente.

Puerto	**Bit**	**Componente**	**Configuración**	**Funcionamiento**
PORTC	2	Led	Como salida DDRC2=1	PINC2=0 ⇒ apagado PINC2=1 ⇒ encendido
PORTC	3	Led	Como salida DDRC3=1	PINC3=0 ⇒ apagado PINC3=1 ⇒ encendido

Puerto	Bit	Componente	Configuración	Funcionamiento
PORTC	4	Conmutador	Como entrada DDRC4=0	PINC4=0 ⇒ pulsado PINC4=1 ⇒ no pulsado
PORTC	5	Conmutador	Como entrada DDRC5=0	PINC5=0 ⇒ pulsado PINC5=1 ⇒ no pulsado

Tabla 3. Configuración de los puertos e/s de los leds y conmutadores

Utilizando el la plantilla de código 3 (fichero *conmutadores.asm*) debe realizar la siguientes tareas:

1. Cree un nuevo proyecto utilizando el código suministrado en el fichero *conmutadores.asm* y complete el programa.
2. Utilice el simulador para comprobar que funciona correctamente. Debe conmutar manualmente los pines PC4 y PC5 desde el simulador. Esto se consigue desplegando el puerto C en árbol de dispositivos que muestra el AVR-STUDIO en la parte derecha durante la simulación y pulsando el botón del ratón sobre el cuadro que representa el bit correspondiente. Cuando el cuadro está relleno de color negro significara que el bit está a 1, si está en blanco es 0.
3. Una vez comprobado en el simulador el correcto funcionamiento, repita los pasos realizados en la sección 3.1.2. para programar el microcontrolador con este nuevo programa. Compruebe que funciona correctamente pulsando los conmutadores.

```
     .include "m328pdef.inc"
     .def temp = r16       /* Define un registro para uso temporal se ha utilizado el r16
                              para poder emplear los modos con direccionamiento indirecto. */

     ldi     temp,$c
     out     ddrc,temp     // Configura portc[3:2] como salidas y el resto como entradas

     ldi     temp,$30
     out     portc,temp    // Activa las resistencias de pull-up del portc[1:0]

             // Debe completar un bucle que lea continuamente PC5 y PC4.
             // Cuando cambie el pin PC5 a 1 debe invertir el valor del pin PC4
             // Cuando cambie el pin PC4 a 1 debe invertir el valor del pin PC3
             // El bucle debe ser infinito
bucle:       // Escriba el código a partir de aquí

    ...
```

Código 3. Fichero conmutadores.asm, plantilla de código para el controlador de pulsadores y leds.

3.2.2. Programa contador de pulsaciones

El nuevo programa a completar debe contar el número de pulsaciones de un conmutador y mostrarlo en el display 7 segmentos. Habrá que completar tres fragmentos de código; el primero es la inizialización correcta de los puertos, el segundo es una subrutina que crea una tabla en memoria con el código 7 segmentos y, el tercero es el programa principal.

La tabla 4 muestra la información de los componentes de entrada/salida que se usarán. Se incluyen los puertos, los bits asociados a los componentes así como la configuración necesaria para que operen correctamente.

Puerto	Bit	Componente	Configuración	Funcionamiento
PORTD	0-7	Segmentos de los displays	Como salida DDRD=0xFF;	PORTDX=0 ⇒ apagado PORTDX=1 ⇒ encendido
PORTB	0	Display 0	Como salida DDRB0=1	PORTB0=0 ⇒ encendido PORTB0=1 ⇒ apagado
PORTB	1	Display 1	Como salida DDRB1=1	PORTB1=0 ⇒ encendido PORTB1=1 ⇒ apagado

Tabla 4. Configuración de los puertos e/s de los displays 7 segmentos.

Utilizando el fichero *contador_bcd.asm* mostrado en el listado de código 4 debe realizar las siguientes tareas:

1. Completar la subrutina de inicialización de puertos llamada *inicializa_puertos.* Puede utilizar como ejemplo de inicialización la utilizada en el programa de la sección anterior (listado de código 3). Debe inicializar los puertos con la siguiente configuración:

   1.1. En el puerto C los pines 3 y 2 deben ser salidas, el resto deben ser entradas.

   1.2. El puerto D está conectado a los segmentos del display, deben ser todos salida.

   1.3. El puerto B controla en encendido o apagado completo de cada uno de los dos displays. Debe configurarlo como salida, así, poniendo un 0 en PORTB0 se activará el display 0 y poniendo un 0 en PORTB1 se activará el display 1.

2. Completar la subrutina que crea una tabla para el convertidor de 7 segmentos llamada *inicializa_tabla7seg.* Esta tabla contiene los códigos 7 segmentos de los dígitos 0 – 9. Al escribir un elemento de esta tabla en el puerto D aparecerá un número BDC en los displays. Como ejemplo se muestran 2 números, donde se puede observar la correspondencia de los bits a uno con la activación de los segmentos mostrados en la figura 8. Complete los números que faltan, del 2 al 9.

3. El bucle principal del programa comienza a partir de la etiqueta *bucle*. Aquí debe escribir el programa que cuente las pulsaciones detectada en un conmutador. El programa se puede realizar siguiente estos pasos:

   3.1. Escribir un bucle que espere hasta detectar que el conmutador se ha pulsado. Un valor 1 en el pin correspondiente al conmutador indica que se ha pulsado.

   3.2. Tras detectar la pulsación hay que incrementar el contador en 1.

   3.3. Comprobar si el contador ha llegado a 10 para ponerlo de nuevo a cero.

   3.4. Esperar en un bucle hasta que se suelte en botón, fíjese que este fragmento de código ya está hecho y corresponde a la etiqueta *espera.*

4. Construya el programa y programe el microcontrolador para comprobar si funciona. Si no opera correctamente puede utilizar el simulador para detectar los errores. Tenga en cuenta que a veces existen problemas de rebotes en los conmutadores, esto significa que, al pulsar una vez el conmutador se detectan varias pulsaciones y el valor mostrado en el display se incrementa en más de una unidad.

```
     .include "m328pdef.inc"
     .def temp = r16      /* Define un registro para uso temporal se ha utilizado el r16 para
                             poder emplear los modos con direccionamiento indirecto.*/
     .def contador = r17 // Cuenta el número de pulsaciones
     .def cero= r18

     .dseg
     .org $100

TABLA7SEG: .byte 10 // Se reservan 10 bytes para una tabla de valores del convertidor bin7seg
     .cseg
     .org $0

     rcall inicializa_puertos      // Rutina que inicializa los puertos
     rcall inicializa_tabla7seg    // Rutina que inicializa la tabla del convertidor

bucle:                             // Programa principal
     rcall  display                // Representamos el valor de contador en el display

     /* Aquí debe escribir el programa que haga lo siguiente:
        1. Esperar hasta que se pulse un pulsador
        2. Si se pulsa el pulsador incrementar el contador
        3. Si el contador llega a 10 hay que ponerlo a cero */

    ...

espera:             // Este fragmento de código esperamos a que se suelte sw1
     sbis   pinc,4  // si no se pone se incrementaría muchas veces el contador
     rjmp   espera  // en el momento que se pulse
     rjmp   bucle

inicializa_puertos:
     // Aquí debe configurar portc[3:2] como salidas y el resto como entradas

     ...

     // Aquí debe configurar puerto D y el puerto B completo como salida


     out    portc,temp  // Se Activan las resistencias de pull-up del portc[1:0] y apaga leds
     ldi    temp,$ff
     ret

/* La siguiente rutina inicializa la tabla de 7 segmentos. Esta rutina sería innecesaria si se
hubiera utilizado la memoria de programa para almacenarla */

inicializa_tabla7seg:
     ldi    zh,high(TABLA7SEG)  // Utilizamos Z para apuntar a la tabla
     ldi    zl,low(TABLA7SEG)   // low() high() son directivas que devuelven el byte bajo o
                                // el byte alto de la dirección que se le pasa como
                                // argumento respectivamente
     ldi    temp,0b01111110                    //Código 7 segmentos del 0
     st     z+,temp
     ldi    temp,0b00110000                    //Código 7 segmentos del 1
     st     z+,temp

      /* Aquí debe completar los códigos de los dígitos que faltan: del 2 al 9 */

     ...

     ret

/* La siguiente rutina permite representar un número en el display de 7 segmentos. Utiliza
para ello el registro Z, que inicialmente apunta a la tabla de 7 segmentos. A este registro se
le suma Cont que es una variable entre 0 y 9 y, después, mediate acceso indirecto se carga el
código 7 segmentos correspondiente en el puerto.*/

display:
     ldi    zh,high(TABLA7SEG)
     ldi    zl,low(TABLA7SEG)
     add    zl,contador    // El registro Z es de 16 bits, mientras que contador es de 8
     adc    zh,cero        // No olvidar sumar el acarreo que se genera del byte bajo a ZH
```

```
    ld      temp,z
    out     portd,temp
    sbi     portb,1         // Apaga el display 1
    cbi     portb,0         // Activa el display 0
    ret
```

Código 4. Fichero contador_bcd.asm, plantilla de código para el contador BCD.

3.2.3. Opcional: Mejoras en el programa contador de pulsaciones

Opcionalmente se proponen hacer algunas mejoras en el programa contador, estas son: utilizar los dos conmutadores para incremente/decrementar y realizar un control de rebote del conmutador

Se propone realizar las siguientes tareas donde debe partir del programa realizado en la sección anterior:

1. Modifique el bucle principal del programa para que se comprueben las pulsaciones de los dos conmutadores. Si se pulsa el primero, el contador debe incrementarse, si se pulsa el segundo, el contador debe decrementarse. No olvide comprobar antes el decremento si es cero para establecerlo a 9, si no el decremento fallará.
2. Para realizar el control de rebotes se propone paralizar la ejecución del programa entre 5-10mseg. Para ello realice las siguientes modificaciones en el programa:

   2.1. Cree una subrutina llamada *no_rebote* que consista en un bucle que espere un tiempo determinado. Sabiendo que el procesador funciona a 1Mhz utilice un contador que mantenga un bucle contando durante unos 10ms aproximadamente. Esta subrutina es similar al programa realizado en el estudio teórico.

   2.2. Modifique el programa principal incluyendo una llamada a esta nueva subrutina cuando detecte que el conmutador se ha pulsado. Además, debe llamar de nuevo a esta subrutina cuando detecte que el conmutador se ha soltado, ya que los rebotes pueden ocurrir tanto en al pulsar como al soltar el conmutador.

   2.3. Pruebe su nuevo programa en la placa de desarrollo.