Universidad Autónoma de Baja California

Materia:

Microprocesadores y microcontroladores.

Reporte #8

Manejo de la sección de E/S del microcontrolador ATmega1280/2560.

Alumno:

Montoya Valdivia Omar Antonio: 1252892

Profesor:

Jesús García

1) Investigación a cerca de ensamblador en línea y fuera de línea para AVR-GCC

El compilador de C de GNU permite mezclar el lenguaje ensamblador dentro de programas en C. Esta opción se puede utilizar para manualmente optimizar partes críticas del software o para utilizar una instrucción muy específica del procesador, que puede no estar disponible en el lenguaje C.

Con el ensamblador extendido, puede leer y escribir variables C desde el ensamblador y realizar saltos desde el código del ensamblador a las etiquetas C. La sintaxis asm extendida utiliza dos puntos (‘:’) para delimitar los parámetros del operando después de la plantilla del ensamblador:

```
asm ( assembler template
    : output operands                   (optional)
    : input operands                    (optional)
    : clobbered registers list          (optional)
    );
```

Figura 1: Estructura de una línea de ensamblador

Se divide en cuatro partes:
1. Las instrucciones de ensamblador, definidas como una cadena:
2. Las salidas: “=r”
3. Una lista de operandos de entrada separados por comas:
4. Registros afectados

Por ejemplo:

```
int a=10, b;
asm ("movl %1, %%eax;
      movl %%eax, %0;"
     :"=r"(b)        /* output */
     :"r"(a)         /* input */
     :"%eax"         /* clobbered register */
     );
```

Figura 2: Ejemplo de inline asm

Para referirse a los parámetros las salidas empiezan primero con %0, %1…%n dependiendo la cantidad de parámetros de salida, el '=r' (seguido de la variable) de cualquier registro disponible coloca el valor en la variable, luego los de entrada 'r' (seguido de las variables de entrada) con %0, %1…%n pero primero se cuentan los de entrada. Se usan %% para referirse a los registros.

Ensamblador fuera de línea:

Una rutina de lenguaje ensamblador debe declararse como global en el código ensamblador para que sea visible para C compilador. Esto se hace usando la directiva ".global":

```
.global my_assembly_fct
```

Además, un archivo C que intente llamar a la rutina del lenguaje ensamblador necesitará tener un prototipo de función que declare la rutina del lenguaje ensamblador para ser externa:

```
extern unsigned char my_assembly_fct (unsigned char, unsigned int);
```

Registros a tomar en cuenta:

Table 5-1. Summary of the register interfaces between C and assembly.

Register	Description	Assembly code called from C	Assembly code that calls C code
r0	Temporary	Save and restore if using	Save and restore if using
r1	Always zero	Must clear before returning	Must clear before calling
r2-r17	"call-saved"	Save and restore if using	Can freely use
r28			
r29			
r18-r27	"call-used"	Can freely use	Save and restore if using
r0			
r31			

Paso de parámetros

Los argumentos en una lista de argumentos fijos se asignan, de izquierda a derecha, a los registros r25 a r8. Todos los argumentos usan un número par de registros Esto da como resultado argumentos char que consumen dos registros. Argumentos adicionales más allá de eso se pasan a la pila.

Los argumentos en una lista de argumentos variables se insertan en la pila en orden de derecha a izquierda. Los argumentos de Char consumen dos bytes.

Los valores de retorno usan los registros r25 a r18, dependiendo del tamaño del valor de retorno. La relación entre el registrarse y el orden de bytes.

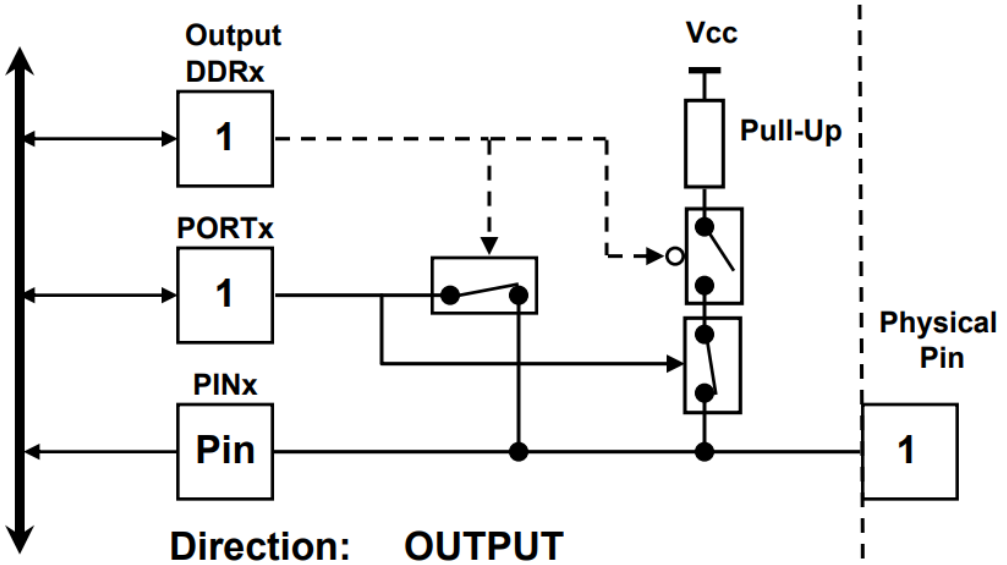
3) Teoría sobre puertos de E/S (uC ATmega1280/2560)

Los puertos de E/S del microcontrolador tienen verdadera funcionalidad de Escribir-Modificar-Leer cuando se utilizan como puertos digitales generales. Lo que significa que la dirección de un solo bit de un puerto puede ser cambiado sin intencionalmente intercambiar la dirección de cualquier otro pin con las instrucciones de SBI y CBI. Lo mismo aplica al modificar el valor de la dirección o la habilitación de resistores pull up internos en caso de que estos sean configurados como entrada.

Bits de Control/Estado por Pin

- DDRx Registro para programar de entrada 0 o salida 1
- PORTx
    - Si está habilitado DDRx como entrada, con 0 se deshabilita el pull-up del puerto, con 1 se habilita el pull-up. Si DDRx esta configurada como salida, se usa PORTx para sacar a puerto.
- PINx
    - SI DDRx está habilitado como entrada, este registro se usa para lectura de datos.

Diagrama de bloques de un pin de E/S

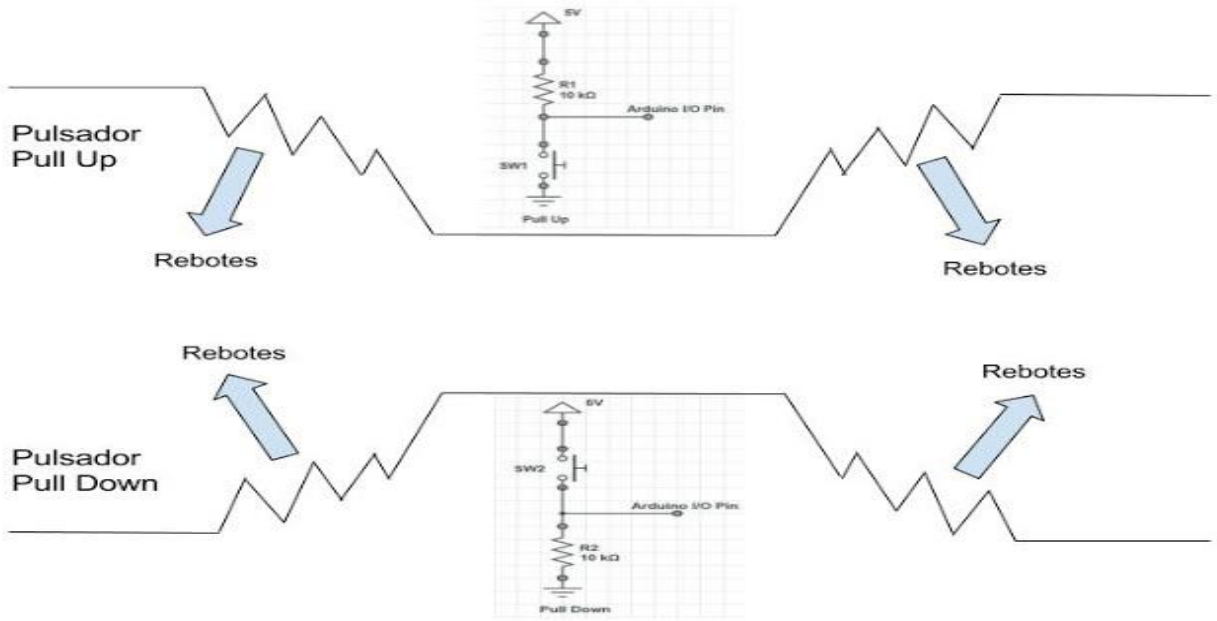


Direction: OUTPUT
Pull-Up: OFF (Tri-State)

Physical Pin x = PORTx
Pin x = Physical Pin x

4) Técnicas de anti-rebote de botones táctiles.

Los pulsadores son dispositivos que tienen un defecto, el cual se llama rebote. Cuando se presiona o se suelta el pulsador, se produce una fluctuación entre sus contactos internos, por lo tanto cuando se va a pasar de un 1 (HIGH) a un 0 (LOW) o viceversa, esas fluctuaciones son también leídas por el Arduino y se produce un comportamiendo inesperado en el funcionamiento de nuestros proyectos, por ejemplo, el usuario puede presionar una sola vez el pulsador pero por culpa del rebote el Arduino podría interpretarlo como si se hubiese presionado varias veces.

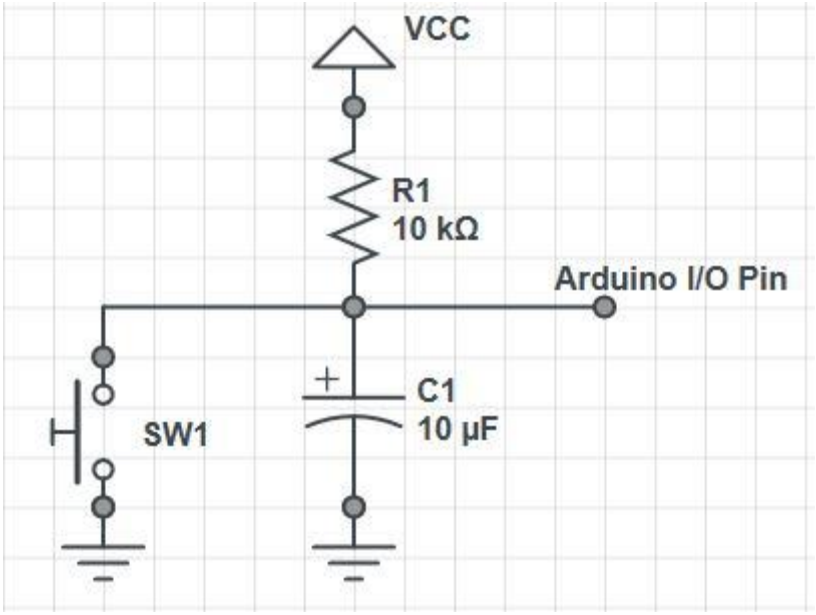


El antirrebote viene a solucionar este problema. Puede realizarse por software y por hardware. En todos los proyectos de este blog se realizarán ambos tipos de antirrebote al mismo tiempo, para tener seguridad de que ningún pulsador va a producir un comportamiento indeseado.

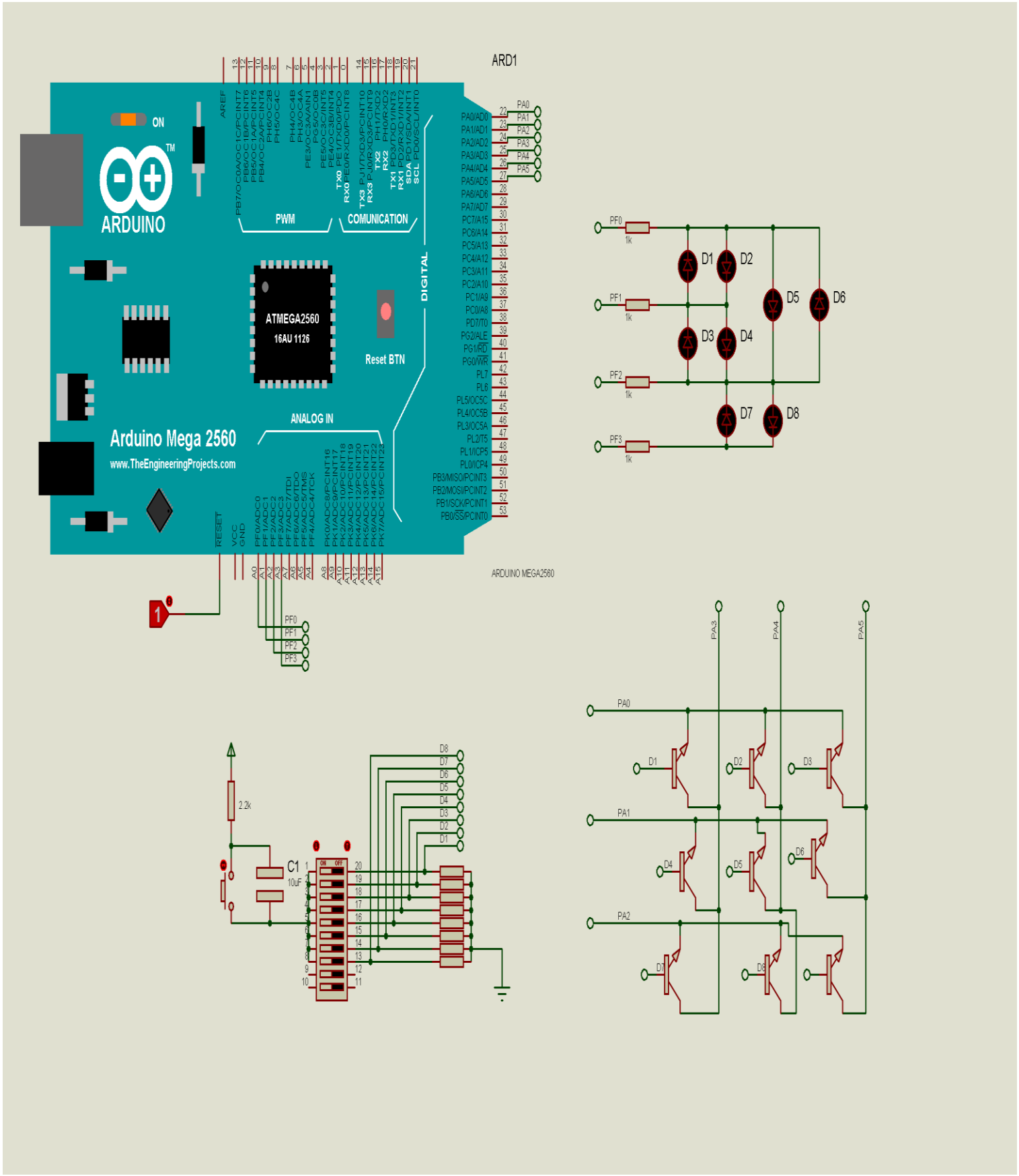
Por software, se coloca un condicional en donde una variable cambia su valor cuando el pulsador se presiona, es decir, cuando se encuentra en LOW. Luego de que la variable tenga su nuevo valor y cuando se suelte el pulsador, se realiza entonces la acción deseada en el proyecto.

```
if (digitalRead(pulsador) == LOW)     //Pregunta si el pulsador está presionado
    {
      presionado = 1;     //La variable cambia de valor
    }
    if (digitalRead(pulsador) == HIGH && presionado == 1)
    {
      //Realiza la acción deseada
      presionado = 0;    //La variable vuelve a su valor original
   }
```

Es necesario poner un condensador de 10uF en el pulsador. Esto es para realizar un antirrebote por hardware, el cual es muy efectivo. El circuito quedaría así para la configuración en Pull Up:



Circuito realizado:



Al realizar la práctica comprendí el funcionamiento de los registros de control del atmega 2560 usando el registro DDRx para configurar el puerto como entrada o como salida, el puerto PORTx para deshabilitar o habilitar el pull-up de los puertos y el registro PINx para leer los datos de un determinado puerto.

También aprendí técnicas de anti rebote, tanto por software como por hardware, al realizar la práctica opté por usar ambas técnicas y así el valor leído fuera más estable y preciso.

Aprendí el uso de ensamblador inline, la sintaxis esta algo rara en términos generales ya que se reciben varios parámetros para el asm. Inclusive aprendí a llamar funciones hechas en ensamblador desde C, comprendí el funcionamiento de cómo se pasan los registros y como regresarlos.