

03/Jun/2020

TRABAJO PRÁCTICO N° 2

Manejo de Puertos

98934 Vazquez, Rodrigo rvazquez@fi.uba.ar

Docentes:

Campiglio, Guillermo Carlos
Stola, Gerardo Luis
Cofman, Fernando
Salaya Velazquez, Juan Guido

Resumen

En el presente trabajo se desarrollará el diseño de programa de encendido de un LED mediante la pulsación de un botón, sobre el microcontrolador *Atmel 328P* en la plataforma *Arduino*. Luego se armará el circuito de testeo apropiado utilizando un *Shield I/O* con los elementos pertinentes al TP.

Índice

1. Objetivo	2
2. Descripción del proyecto	2
3. Diagrama de conexiones en bloques	2
4. Circuito esquemático	3
5. Listado de componentes y tabla de gastos	3
6. Software	3
7. Resultados	3
7.1. Resistencias de Pull-Up	4
8. Conclusiones	5
9. Anexo - Código	6
10. Anexo - Esquemático Shield I/O	7

1. Objetivo

Manejar los puertos del microcontrolador *Atmel 328P* para capturar los valores que un boton pulsador tiene a la entrada del microcontrolador y asi encender un LED.

2. Descripción del proyecto

Se realizará un programa en lenguaje *Assembly* para la familia de microcontroladores *AVR*. La conexión de los perifericos estará dada por la plataforma utilizada. Siendo la conexión al led en el pin PB3 y el boton pulsador en PD2

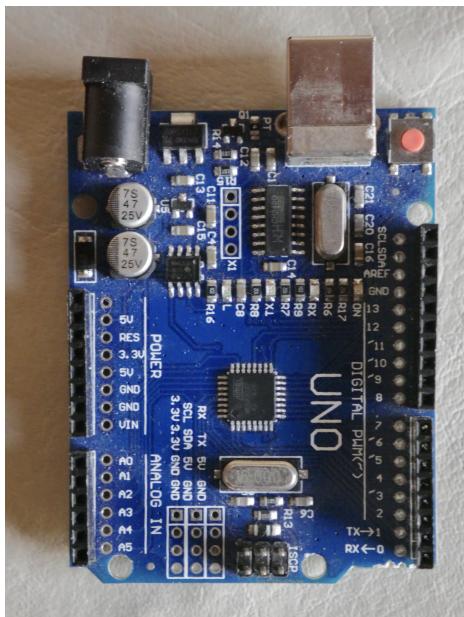


Figura 1: Placa Arduino UNO utilizada



Figura 2: Placa *Shield I/O* utilizada para la conexión LED

3. Diagrama de conexiones en bloques

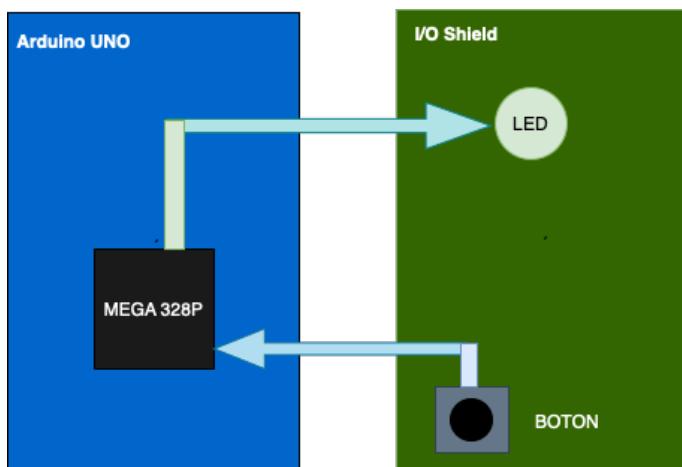


Figura 3: Diagrama en bloques del hardware

4. Circuito esquemático

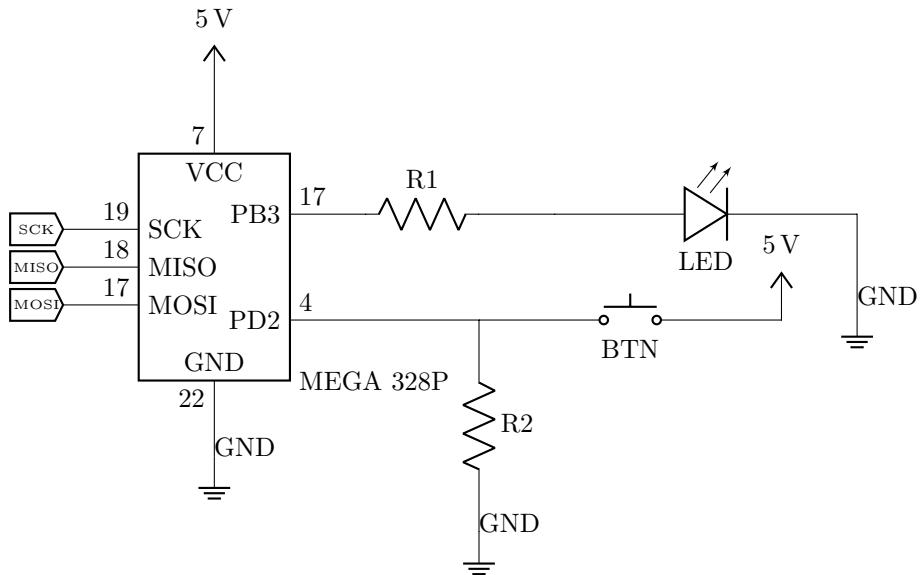


Figura 4: Circuito para encendido de LED con pulsador

5. Listado de componentes y tabla de gastos

Componente	Precio
Arduino UNO	\$750
I/O Shield	\$950

Tabla 1: Tabla de componentes

6. Software

El programa (anexado al final del documento) comienza definiendo el principio de la memoria de programa via la directiva `.org` e inicializando el stack pointer. Luego con la configuracion del PIN 3 del puerto B como salida y el PIN 2 del puerto D como entrada, esto se realiza cargando un 1 en el cuarto bit del registro `DDRB` y un 0 en el tercer bit del registro `DDRD`.

El software implementado tambien toma codificaciones hechas en ejercicios previos como la subrutina `DELAY`, esta actua como espera de tiempo 1 ms y la utilidad es desestimar rebotes en el pulsador.

7. Resultados

Para la escritura del programa se mantuvo la utilizacion del editor de texto *Atom*, pero ahora se incluyo la confección de un archivo *Makefile* para poder ser reutilizado en futuros trabajos.

Como se utilizaron varias subrutinas y saltos, se prestó especial atencion a la memoria de stack, teniendo cautela de no hacer llamados sin utilizar los retornos. Para lidiar con el rebote propio de un botón pulsador, se implemento una subrutina de *delay* muy corta pero con la suficiente duracion para poder evitar visualizar el rebote en la salida.

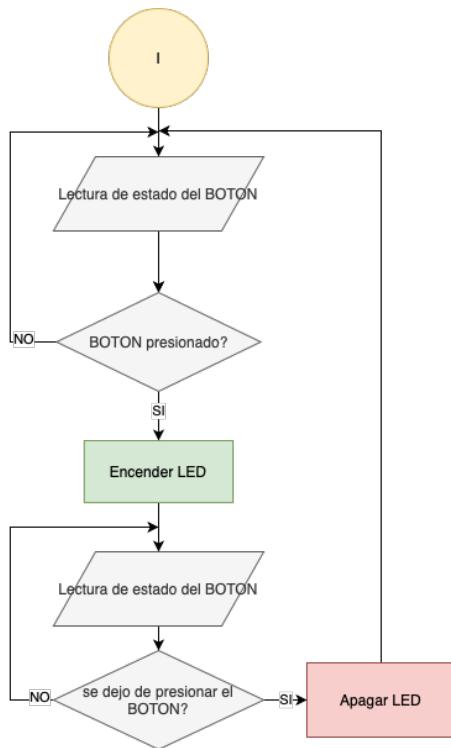


Figura 5: Diagrama de flujo del programa para encender un LED con botón pulsador

7.1. Resistencias de Pull-Up

El microcontrolador posee la función de activar resistencias de pull-up internas a través de poner unos en los pines donde se quiera activar la resistencia. Para ello hice una subrutina llamada `activate_portd_pullup` donde se activan las resistencias en todo el puerto D. Por último, como al invertirse los valores lógicos recibidos al apretar y soltar el botón, será necesario cambiar las instrucciones `sibs` por `sibc` y viceversa. Asimismo, activarlas trae una ventaja tanto en facilidad de armado del circuito como en eventual ahorro en costos, ya que utilizando esta función no es más necesaria la resistencia R2 que se utilizó para forzar un estado lógico cuando el botón se encuentre abierto, ya que el microcontrolador reconocerá como 1 a una entrada de alta impedancia. Entonces se debe modificar el programa para que al ingresar un 1 el led se apague y viceversa

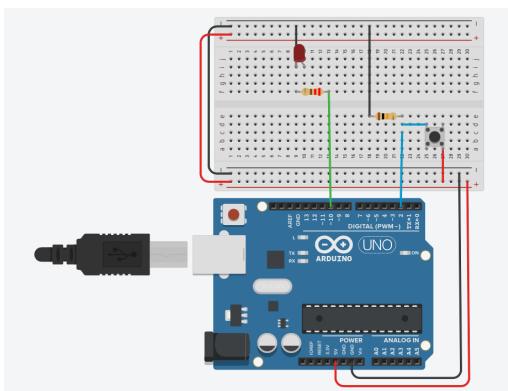


Figura 6: Circuito con necesario sin activar resistencias de pull-up internas

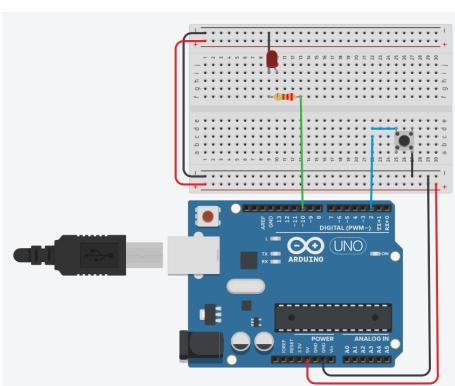


Figura 7: Circuito usado al activar resistencias de pull-up internas

8. Conclusiones

Las principales conclusiones de este trabajo recaen en el manejo apropiado de los puertos de entrada y salida del microcontrolador Atmel Mega 328P via el lenguaje de *assembly*, para ello siendo necesario un profundo conocimiento de la memoria del microcontrolador y de las instrucciones la administran.

Bibliografía

MAZIDI, Muhammad Ali; NAIMI, Sarmad; NAIMI, Sepehr (2011). "The AVR microcontroller and embedded systems. Embedded system using Assembly and C". Pearson Education Inc. New Jersey.

9. Anexo - Código

```

.include "m328Pdef.inc"
.org $00
#####
; Seteo de hardware
.equ LED_PIN = 3 ; conecto led en PB3
.equ BTN_PIN = 2 ; conecto boton en PD2
sbi DDRB, LED_PIN ; declaro a PB3 como salida
; call activate_portd_pullup ; activa resistencias de pullup en PORTD
cbi DDRD, BTN_PIN ; declaro a PD2 como entrada
#####
; Inizializador de stack
stack_ini:
ldi r16, HIGH(RAMEND)
out SPH, r16
ldi r16, LOW(RAMEND)
out SPL, r16 ; inicializa el stack al fondo de la memoria
clr r16
#####
; rutinas de chequeo del boton
check_press:
    sbis PIND, BTN_PIN
    rjmp check_press
    call delay
    call turnon_led
    rjmp check_release
check_release:
    sbic PIND, BTN_PIN
    rjmp check_release
    call delay
    call turnoff_led
    rjmp check_press
#####
; rutinas de encendido y apagado del LED
turnon_led:
    sbi PORTB, LED_PIN
    ret
turnoff_led:
    cbi PORTB, LED_PIN
    ret
#####
; DELAY de 1ms
delay:
    push r17
    push r18
    ldi r18, 209
L2:
    ldi r17, 255
L1:
    dec r17
    brne L1
    dec r18
    brne L2
    pop r18
    pop r17

```

```

ret
#####
; Activa resistencias de pullup en PORTD
activate_portd_pullup:
push r16
ldi r16, $FF
out DDRD, r16 ; declara PORTD como salida
out PORTD, r16 ; activa resistencias de pullup en PORTD
pop r16
ret

```

10. Anexo - Esquemático Shield I/O

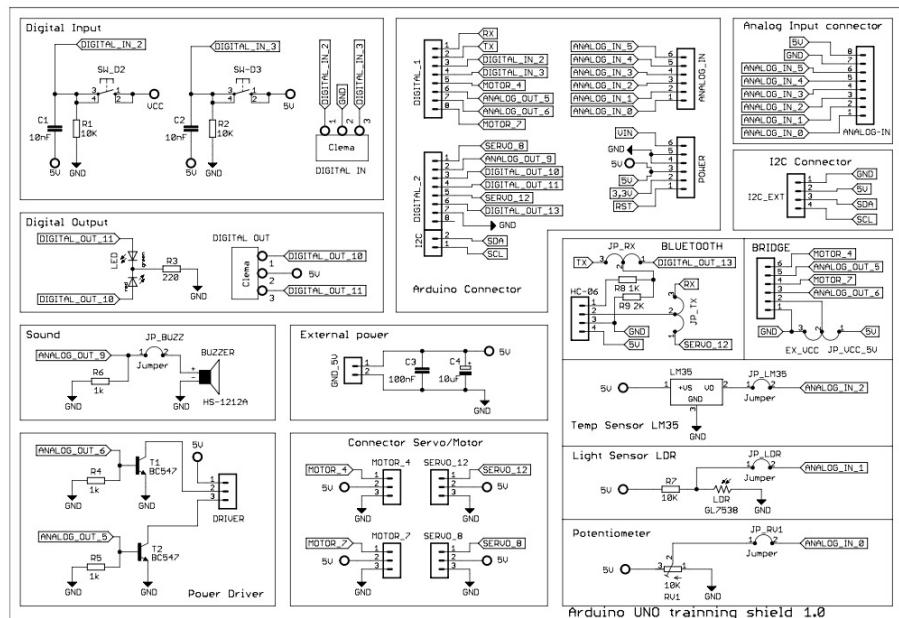


Figura 8: Circuito esquemático Shield I/O