Facultad de Ingeniería
Universidad de Buenos Aires

Laboratorio de Microprocesadores - 86.07

Trabajo Práctico Obligatorio Nº1

Profesor:			Ing. Guillermo Campiglio									
Cuatrimestre/Año:			1º/2020									
Turno de las clases prácticas			Miércoles									
Jefe de trabajos prácticos:			Ing. Pedro Ignacio Martos									
Docente guía:			Ing. Fabricio Baglivo, Ing. Fernando Pucci									
Autores			Seguimiento del proyecto									
Mariano Federico	Guglieri	99573										

Observaciones:

.....
.....
.....
.....
.....
.....
.....
.....
.....

Fecha de aprobación				Firma J.T.P

Coloquio	
Nota final	
Firma profesor	

Índice

1. Objetivos

El objetivo de este trabajo práctico es hacer un programa que prenda un LED cuando se presiona el pulsador utilizando la resistencia de pull up y que lo apague cuando se deje de presionar.

2. Desarrollo

Para la realización del trabajo se utilizó la plataforma de desarrollo de Atmel, Atmel Studio versión 7.0, en donde se implementó el Software. Para la parte física se requirió de un Arduino UNO, el cual sirvió como programador para el integrado ATMEGA328P, además de suplirlo con la energía necesaria para su funcionamiento. Dos resistencias, un led de color rojo, un pulsador y tres cables.

El código del programa consta de dos versiones, la primera versión surge sin mucho análisis y consiste en concentrarse en poder observar el funcionamiento del circuito, declarando las macros necesarias, los puertos de entrada y los ciclos correspondientes a la lógica del programa. La segunda versión consta de un mejoramiento de recursos respecto al anterior código. Esto se podrá analizar observando ambos diagramas. Además, la versión definitiva usa la resistencia de pull up, lo cual la configuración del circuito debe modificarse. Con esta modificación nos ahorramos de uns resistencia.

3. Diagrama en bloques

El diagrama de conexiones en bloque del trabajo se presenta a continuación:

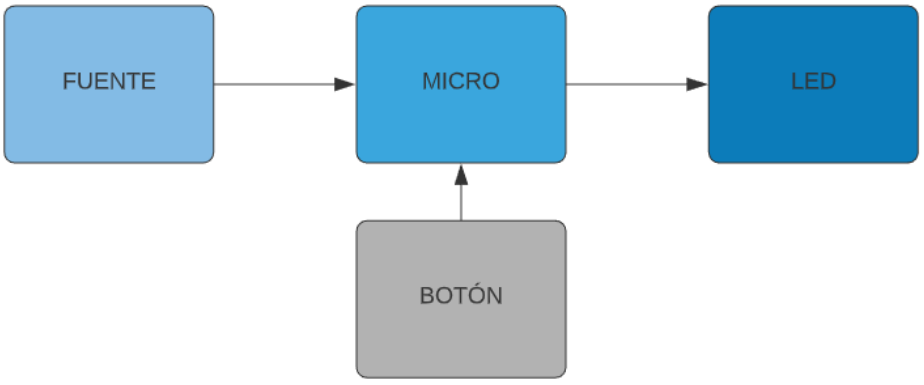

Figura 1: Diagrama de conexiones en bloque

Se observa que el funcionamiento surge primero de la pc donde se encuentra el código que será descargado en el micro y a su vez sirve como una fuente. El micro recibe información del estado del botón. En base a como se encuentre este, el micro encenderá o no al led. Esto es lo que se ve en el diagrama.

4. Esquemático y listado de componentes

Para el primer circuito que no utiliza la resistencia de pull up, se utilizaron:

- 1 resistencia de 220Ω
- 1 resistencia de 10kΩ
- 1 led de color rojo
- 1 pulsador
- 3 cables para protoboard
- 1 placa arduino con atmega328p

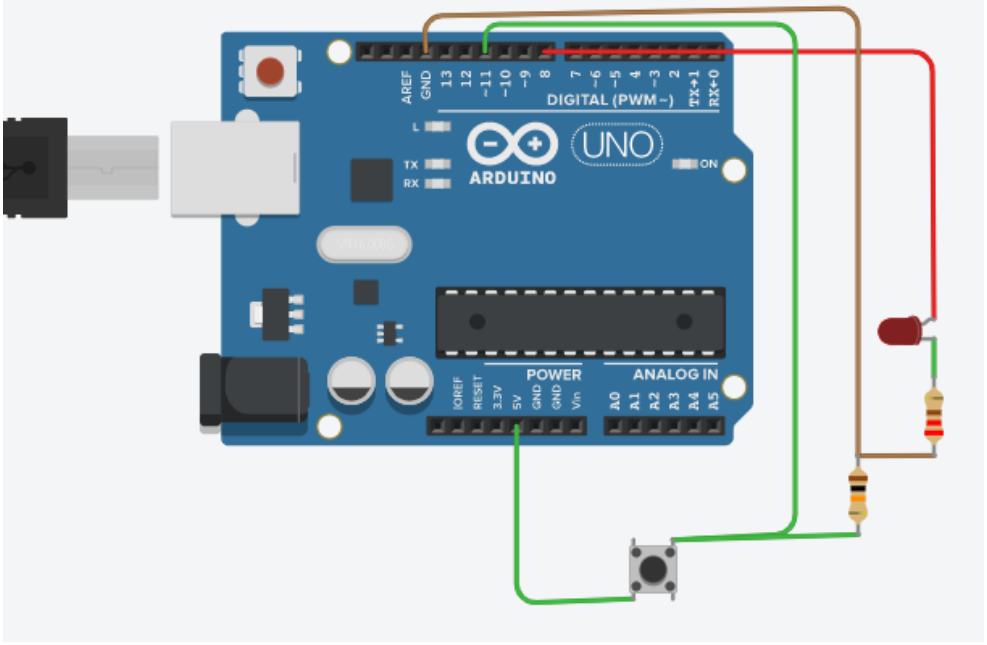

Figura 2: Esquemático sin la resistencia de pull up

En este primer esquema se utiliza una resistencia externa al microcontrolador. Este circuito tiene una lógica de pull down es decir, cuando el botón no se presiona, el pin se conecta a tierra.

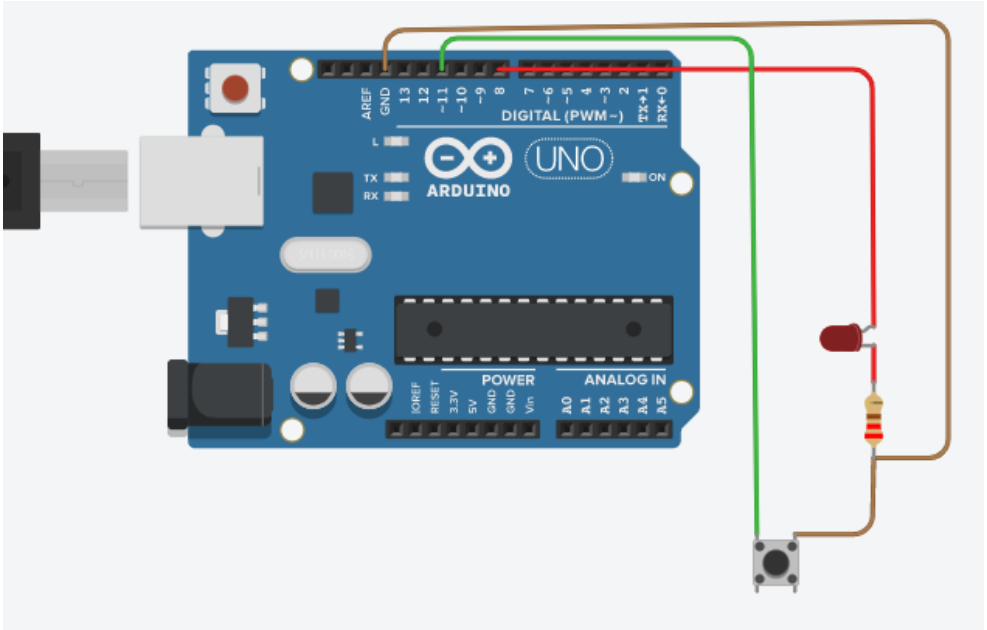

Figura 3: Esquemático con la resistencia de pull up

Utilizando la resistencia de pull up se ve una notoria simplificación en el circuito. Claro esta que ahora la lógica del programa se encuentra invertida. A estados altos del pin de entrada, el led se encontrará apagado.

Como se ve, para este circuito no fue necesario la resistencia de $10k\Omega$.

En total se habrá gastado unos 50 pesos para el circuito. Sin contar la plataforma arduino.

La plataforma arduino le agrega al microcontrolador un cristal de 16Mhz y una resistencia de $10k\Omega$, conectados de la siguiente manera.

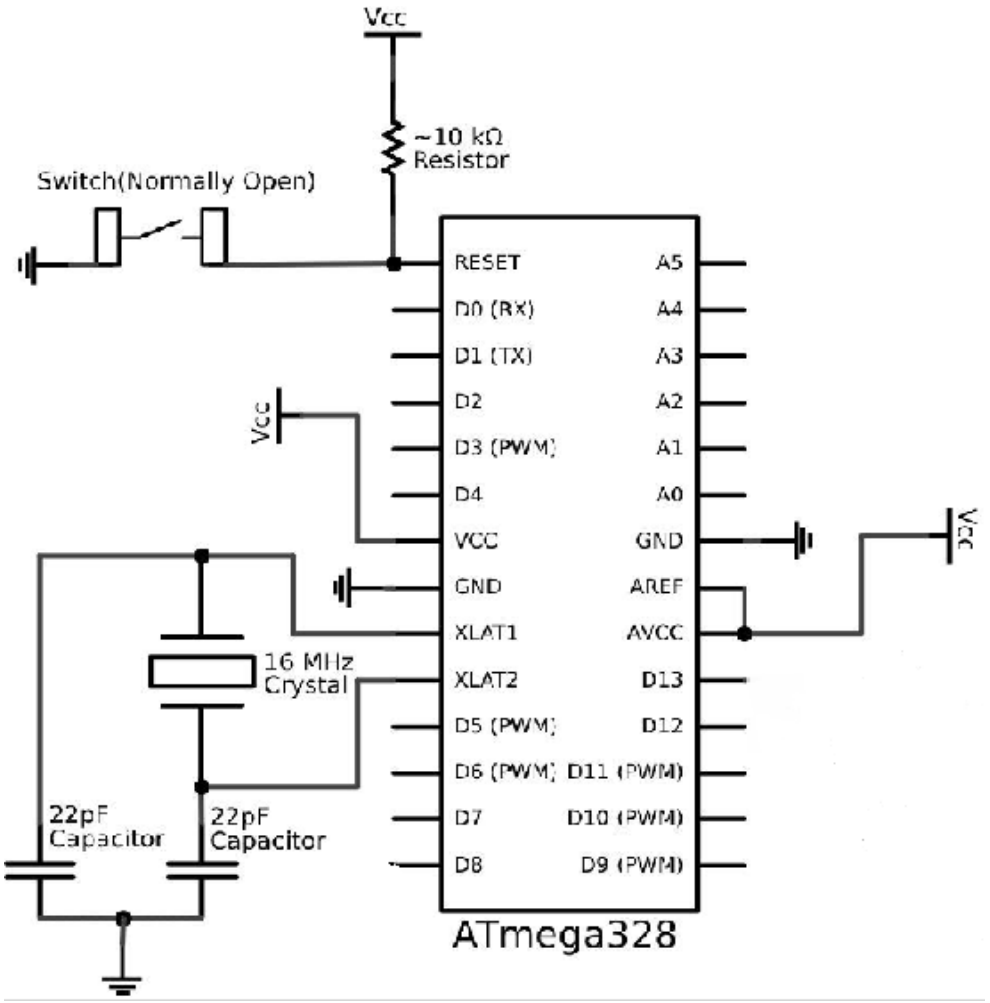

Figura 4: Esquemático con la resistencia de pull up

5. Diagrama de flujos

Los diagramas de flujo que se presentarán a continuación son sólo para los códigos en los cuales se utilizó la resistencia de pull up. Para los que no se utilizo dicha resistencia la lógica sigue siendo parecida, sólo que los procesos de decisión se invierten. Esto quiere decir que, si un condicional verificá que el puerto de entrada se encuentre en estado alto (SBIS), cuando no se tiene la resistencia de pull up este condicional tendrá que verificar si el puerto de entrada se encuentra en estado bajo (SBIC).

El primer diagrama es del primer intento o primera versión del código.

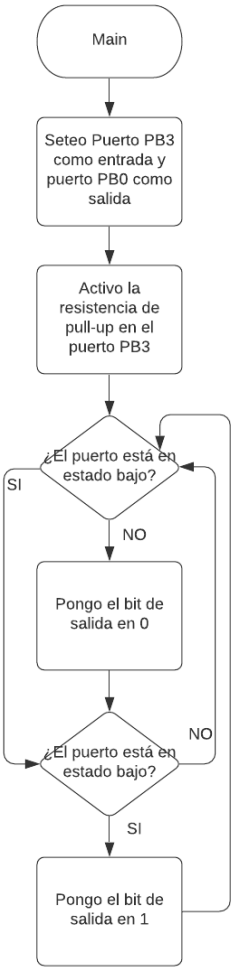

Figura 5: Diagrama de flujo de la primera versión

En este diagrama se observa que cuando el pin de entrada se encuentra en estado alto, es decir que el pulsador no está siendo presionado, el programa recorre un loop del cual sólo puede salir cuando el pin de entrada se encuentre en estado bajo. Si bien esto funciona, el loop setea el bit de salida en cero innecesariamente dentro del loop. Planteando un diagrama con una lógica mas simple se llema a la segunda versión del código,

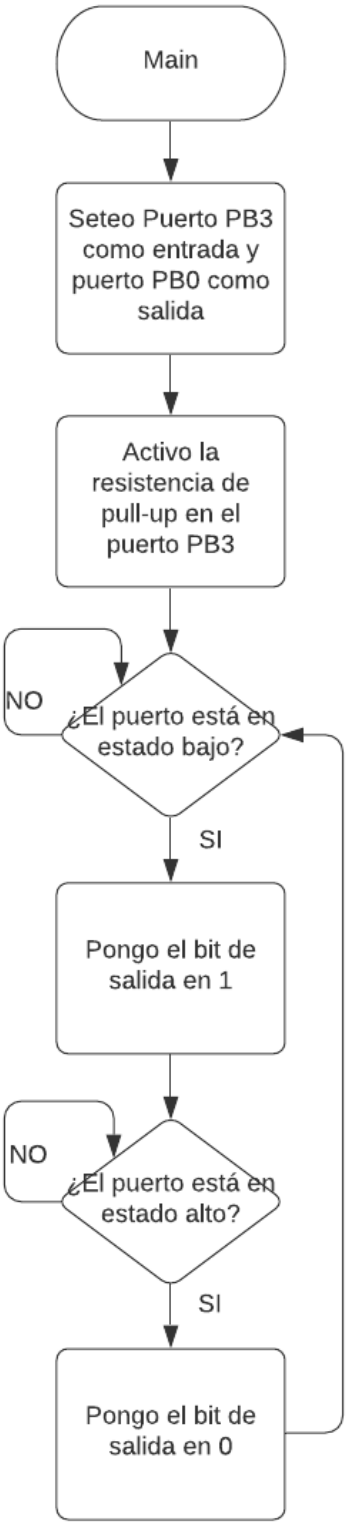

Figura 6: Diagrama de flujo de la segunda versión

En este diagrama se ve una lógica mas compacta en donde la recursividad se da sin necesidad de implementar funciones en el medio, gastando recursos. claro está que para lograr esto se tuvo que recurrir a utilizar también la directiva SBIS

6. Códigos

```
.include "m328pdef.inc"

.cseg
```

```
;declaro macros
.EQU    entrada =       PINB
.EQU    salida  =       PORTB
.EQU    pull_up =       PORTB

                jmp             main

.org INT_VECTORS_SIZE

main:
;declaro los puertos de entrada y salida
                cbi             DDRB,           3
                sbi             pull_up,        3       ;activo la resistencia de pull up
                sbi             DDRB,           0

ACA:    SBIC    PINB, 3
                CBI             PORTB, 0
                SBIC    PINB, 3
                RJMP    ACA
                SBI             PORTB, 0
                RJMP    ACA
```

Segundo código que se implemento:

```
.include "m328pdef.inc"

.cseg

;declaro macros
.EQU    entrada =       PINB
.EQU    salida  =       PORTB
.EQU    pull_up =       PORTB

                jmp             main

.org INT_VECTORS_SIZE

main:
;declaro los puertos de entrada y salida
                cbi             DDRB,           3
                sbi             pull_up,        3       ;activo la resistencia de pull up
                sbi             DDRB,           0


no_pulsado:             sbic entrada,   3
                                jmp no_pulsado
                                sbi salida,             0

pulsado:                sbis entrada,   3
                                jmp pulsado
                                cbi salida,             0

                                jmp no_pulsado
```

El primer código es la primera versión del programa y la otra es la segunda. Se ve que el código de la segunda también es más fácil de interpretar, nombrando a los ciclos por el estado en que se encuentra el pulsador.

7. Resultados

Como resultado se pudo controlar el encendido y apagado del led mediante el pulsador sin mayor inconveniente.

8. Conclusiones

Me pareció útil contrastar los dos códigos realizados para ver en qué cosas uno se puede fijar para mejorar el funcionamiento del programa. Las dos cosas a destacar fueron en minimizar los ciclos utilizados y hacer el código más legible.

Por otro lado, se concluye que es útil tener conocimiento y saber manipular la resistencia de pull up del circuito. La implementación de está ahorro una resistencia del circuito y también un cable. Es por eso que es algo necesario poder saber utilizarla.

En este trabajo se aplico el uso de macros, cosa que en el anterior no. Al tener tanto pines de entrada como salida era necesario una clara distinción para no confundirse.