Fiuba
Facultad de Ingeniería
Universidad de Buenos Aires

(6609) LABORATORIO DE MICROCOMPUTADORAS

Proyecto:
Parpadeo de un LED

Profesor:	**Guillermo Campiglio**
Cuatrimestre / Año:	**1er Cuatrimestre 2020**
Turno de clases prácticas:	**Miércoles**
Jefe de Trabajos Prácticos:	**Pedro Ignacio Martos**
Docente guía:	**---**

Autores			**Seguimiento del proyecto**								
Nombre	**Apellido**	**Padrón**									
Mariano	**Guglieri**	**99573**									

Observaciones:

Fecha de aprobación		

Firma J.T.P.

COLOQUIO	
Nota final	
Firma Profesor	

Contenido:

Objetivos……3
Desarrollo……...3
Diagrama en bloques…… 3
Esquemático y listado de componentes……..4
Diagrama de Flujos……5
Códigos……...6
Resultados……...7
Conclusiones……...7

Objetivo:

El objetivo de este trabajo consiste en realizar dos programas en código assembler que permitan el parpadeo de un LED conectado al pin PB0. Ambos programas serán realizados por métodos diferentes de programación. Uno utilizará todo el puerto B mientras que el otro sólo hará uso de un solo bit del puerto.

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. Una resistencia, un led de color rojo y dos cables. El circuito se explicará con más detalle cuando se analice el esquemático y el diagrama en bloques.

Para lograr el cometido, el código consta en esencia de 3 partes. Primero se declara al puerto B como puerto de salida, ya que se busca brindar una tensión a un dispositivo y no se pretende recibir ningún tipo de lectura. Es decir, se habilita como puerto de escritura.

La segunda parte consta de las instrucciones que determinan el estado del pin PBO. En el caso del código el cual activa un solo pin, se usaron las instrucciones sbi (Set Bit In I/O Register) y cbi (Clear Bit in I/O Register), mientras que para encender todo el puerto se utilizó la instrucción OUT, cargando el registro del puerto entero con el valor correspondiente (0xff nivel alto y 0x00 nivel bajo). Por medio de un salto, estas instrucciones que permiten el encendido y apagado del LED se repiten indefinidamente por medio de un bucle utilizando la instrucción RJMP.

Por último, como el tiempo de conmutación del arduino es de 16MHz, utilizar un bucle no basta ya que el efecto de parpadeo no sería visible. Es por esto que se generan dos bucles secundarios entre cada estado del pin, los cuales tienen como único propósito desperdiciar ciclos de máquina. De esta manera, el proceso de encendido y apagado dura menos que la frecuencia de conmutación, ya que el micro debe procesar dichos ciclos.

Diagrama en bloques:

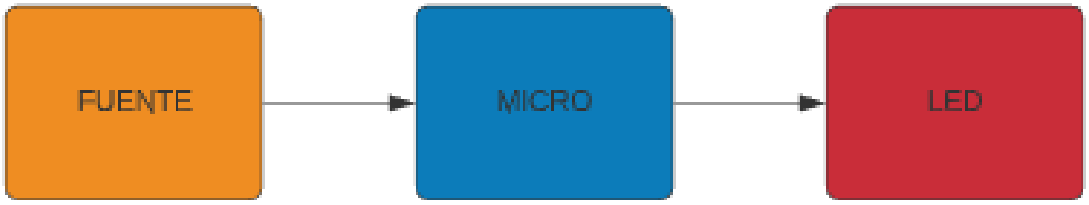


Figura: Diagrama de conexiones en bloques del trabajo

En este esquema se puede ver a grandes rasgos el cometido del proyecto y cuales son los bloques básicos de este. Por un lado, la computadora la cual sirve de fuente y como entorno de desarrollo del código. El micro, el cual permite ejecutar las instrucciones y el LED que va a ser encendido y apagado indefinidamente.

Esquemático y listado de componentes:

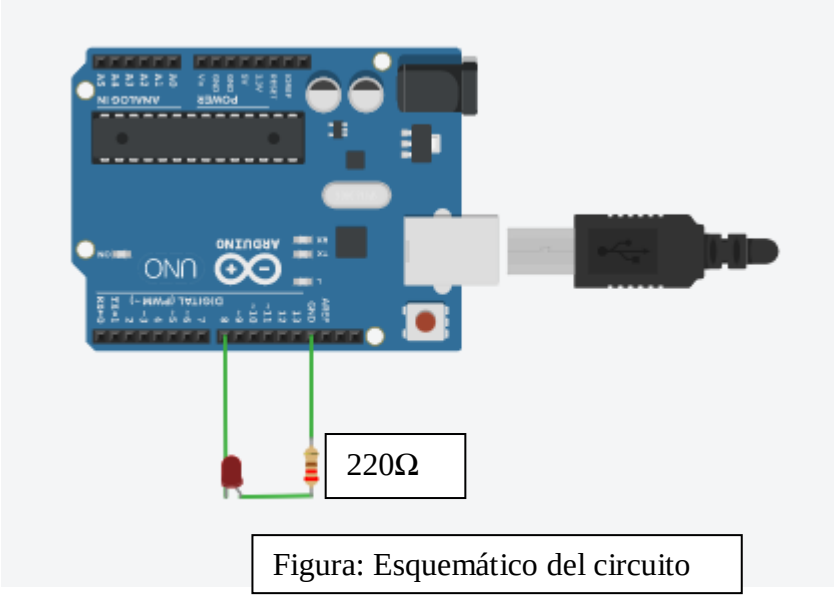
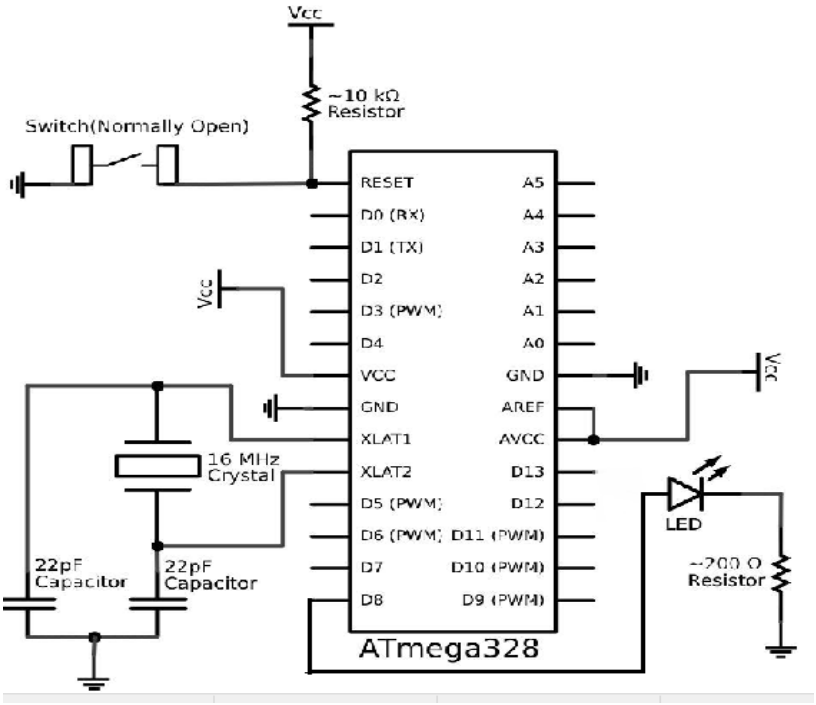


Figura: Esquemático del circuito

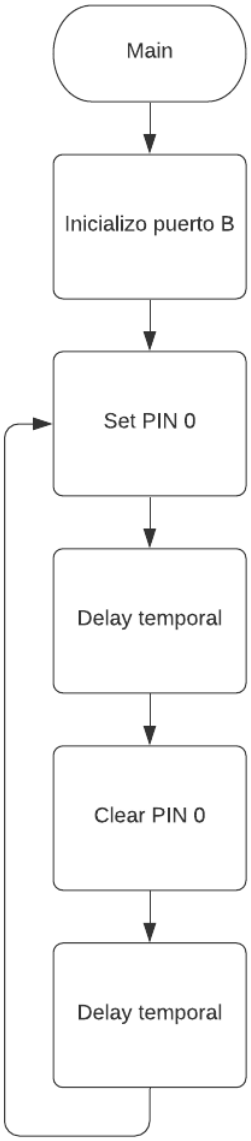


En el esquemático se observa la disposición en serie del led con la resistencia. Esto es así debido a que el led no soportaría una caída de tensión de 5V y se quemaría. El Arduino proporciona la alimentación necesaria al microcontrolador. El pin 8 equivale precisamente al pin 0 del puerto de salida B (PB0). Se utilizó una Protoboard para conectar los componentes. El segundo esquemático muestra la conexión del cristal y la resistencia de reset.

-Resistencia de 220Ω
-LED de color rojo
-Protoboard
-2 cables para protoboard

Los componentes se compraron hace varios años, por lo que se habrá gastado un total de 300 pesos. Hoy en día eso equivaldría a 1400 pesos. Contando el Arduino.

Diagrama de Flujos:



La lógica de ambos programas es similar, la única diferencia es que, para usar todo el puerto, las instrucciones de set bit y clear bit se reemplazan por OUT y se debe cargar un registro con el valor de salida para encender el led. En el diagrama no se encuentra, pero cuando los registros no son 0, se vuelven a decrementar.

Códigos:

Utilizando el pin PBO:

```
.include "m328pdef.inc" ;carpeta que incluye nombres de puertos y constantes para
el atmega328p

.cseg ;indica que todo lo que viene a continuación tiene que estar en la memoria
del programa. Es código ejecutable

.org 0x0000          ;que empiece en la dirección 0
              jmp           main ;salto a la etiqueta de cógido donde se encuentra
main

.org INT_VECTORS_SIZE ;salta a donde hay espacio libre en donde no hay espacio
reservado
main:

; Configuro puerto B
              ldi           r20,0xff      ; cargo r20 con el valor 0xff

              out           DDRB,r20      ;(PORTB como salida)

prendo:       sbi           PORTB,0       ; encendido del led

demora1:
              ldi           r20, 32              ; cargo r20 con el valor 32
loop3:        ldi           r21, 255             ; cargo r21 con el valor 255
loop2:        ldi           r22, 255             ; cargo r22 con el valor 255

loop1:        dec           r22                  ; decremento r22
              brne          loop1         ; veo el flag de Z para ver si llego a 0
              dec           r21                  ; decremento r21
              brne          loop2         ; veo el flag de Z para ver si llego a 0
              dec           r20           ; decremento r20
              brne          loop3         ; veo el flag de Z para ver si llego a 0

              cbi           PORTB,0              ; apagado del led

demora2:
              ldi           r20, 32
loop6:        ldi           r21, 255
loop5:        ldi           r22, 255

loop4:        dec           r22
              brne          loop4
              dec           r21
              brne          loop5
              dec           r20
              brne          loop6

              RJMP          prendo               ; reinicio el ciclo
```

Utilizando todo el puerto:

```
.include "m328pdef.inc"

.cseg
.org 0x0000
              jmp           main

.org INT_VECTORS_SIZE
main:

; Led en PB5
; Configuro puerto B
              ldi           r20,0xff      ;(PORTB como salida)
              ldi           r23, 0xff    ;cargo r23 con el valor 0xff
              out           DDRB,r20

; rutina de encendido y apagado

prendo:              out PORTB,r23                ; encendido del led

demora1:
              ldi           r20, 32
loop3:        ldi           r21, 255
loop2:        ldi           r22, 255

loop1:        dec           r22
              brne          loop1
              dec           r21
              brne          loop2
              dec           r20
              brne          loop3

              out           PORTB,r20            ; apagado del led

demora2:
              ldi           r20, 32
loop6:        ldi           r21, 255
loop5:        ldi           r22, 255

loop4:        dec           r22
              brne          loop4
              dec           r21
              brne          loop5
              dec           r20
              brne          loop6

              RJMP   prendo        ; reinicio el ciclo
```

Resultado:

Se pudo controlar el encendido y apagado del led. Además, manipulando la cantidad de repeticiones del ciclo se puede determinar la frecuencia de estas repeticiones.

Conclusiones:

Comparando ambos programas, se puede observar que, para utilizar todo el puerto B, se necesita ocupar un registro con la información necesaria para encender el led. Esta acción provoca un ciclo más en contraposición con utilizar el pin 0.

Si bien se sabe que no es óptimo generar retardos de esta manera, se buscó desarrollar un retardo alternativo al dado en la clase, a modo de práctica. El código realizado funciona a partir de decrementos de los registros en vez de incrementos. Esto permite obviar la instrucción cpi, ya que el condicional se fijará en el flag de zero para tomar la decisión en vez del flag de carry. Aún así, este tipo de retardo utiliza varias líneas de código, usos de registros e instrucciones, con lo cual no es aconsejable.