Universidad de Buenos Aires

Facultad de Ingeniería

Trabajo Práctico N° 1 – Manejo de Puertos

ALUMNO

Mundani Vegega, Ezequiel **emundani@fi.uba.ar** **102312**

ASIGNATURA

86.07 - Laboratorio de Microprocesadores

4 de noviembre de 2022

Introducción y objetivos

En este trabajo se busca programar un controlador ATMega328P en Assembler de manera que haga parpadear un LED, para lo cual se utilizó un Arduino Nano con su microcontrolador.

Desarrollo y análisis

El software fue desarrollado con Microchip Studio, para luego ser ensamblado y subido al microprocesador con AVRDude. Se optó por dejar prendido y apagado el LED durante un segundo, lo que equivale a 8.000.000 ciclos de reloj del dispositivo (siendo que su frecuencia es 8MHz).

Para programar los 8.000.000 ciclos de reloj, se utilizó la subrutina `delay8Mcicles`. Cada vez que se llama a esta rutina pasan exactamente 8.000.000 ciclos de reloj hasta luego del retorno.
Para calcular la cantidad de ciclos que pasan primero considerar las llamadas a `call` (4 ciclos), a `ret` (4 ciclos), los 3 `ldi` (3 ciclos en total) y `nop` (1 ciclo), dando un total de 12 ciclos.
El primer loop en el que se decrementa `r20` hasta 0 por primera vez:

$$124 \cdot 3 + 2 = 374 \text{ ciclos}$$

El segundo loop que decrementa r19 hasta 0 por primera vez (sin considerar lo anterior):

$$149 \cdot (255 \cdot 3 + 2 + 3) + 2 = 114.732 \text{ ciclos}$$

Y por último el loop que contiene a los otros dos (sin considerar los ciclos que se tardó en llevar a r19 y r20 a valer 0):

$$40 \cdot (255 \cdot (255 \cdot 3 + 2 + 3) + 255 \cdot 3 + 2 + 2 + 3) + 2 = 7.884.882 \text{ ciclos}$$

Sumando todo queda:

$$12 + 374 + 114.732 + 7.884.882 = 8.000.000$$

```
delay8Mcicles:
    ldi r18, 41 ;41
    ldi r19, 150 ;150
    ldi r20, 125 ;5
    L1:
        dec r20
        brne L1
        dec r19
        brne L1
        dec r18
        brne L1
    nop
    ret
```

Subrutina para esperar 8.000.000 ciclos.

Al momento de calcular los ciclos se ve que se multiplica el valor del registro menos uno, esto se debe a que en la última iteración la cantidad de ciclos de `brne` es 2 en vez de 3, por lo que se resta 1 al valor del registro y luego se suma 2. En el primer loop esto es fácil de ver, en los siguientes sigue ocurriendo pero es más difícil de ver.

Una vez que el registro llegó a cero, como los loops están anidados cuando se lo vuelva a llamar empezará desde 0, lo que equivale a 256 iteraciones, pero en vez de aparecer el 256 aparece el 255 por la misma razón que antes. En los loops 2 y 3 se puede ver que aparece un +3, este se debe al `inc` y al `brne` luego de iterar por el primer loop.

Se obtuvo la fórmula genérica de cuántos ciclos tardará la función en base a los valores de `r18`, `r19` y `r20`:

$$12 + (\texttt{r20} - 1) \cdot 3 + 2 + (\texttt{r19} - 1) \cdot (255 \cdot 8) + 2 + (\texttt{r18} - 1) \cdot (255 \cdot (255 \cdot 3 + 5) + 255 \cdot 3 + 7) + 2$$

Teniendo en cuenta esta fórmula se puede ver que cambiando los valores de los registros con esta subrutina como mínimo se podrán esperar 18 ciclos y como máximo 50.463.248 ciclos.

El programa (subido a GitHub) primero configura el pin 2 del puerto B como salida y luego lo que se hace es setear y resetear ese pin esperando ocho millones de ciclos de por medio. Este comportamiento se refleja en el siguiente diagrama:

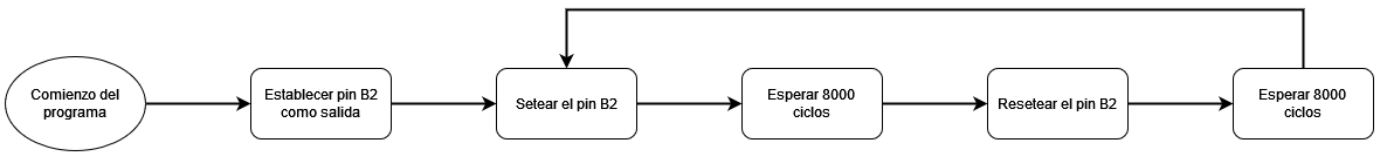

Figura 1: Diagrama del flujo del programa.

El diagrama en bloques del circuito planteado sería el siguiente:

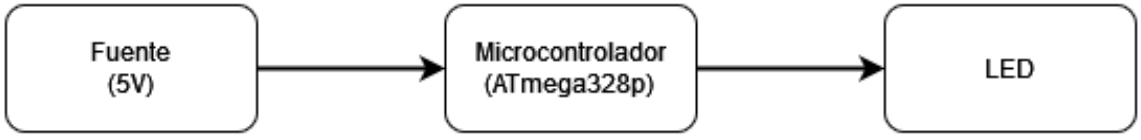

Figura 2: Diagrama en bloques del circuito.

Y para el correcto funcionamiento, los dispositivos se deben conectar de la siguiente manera:

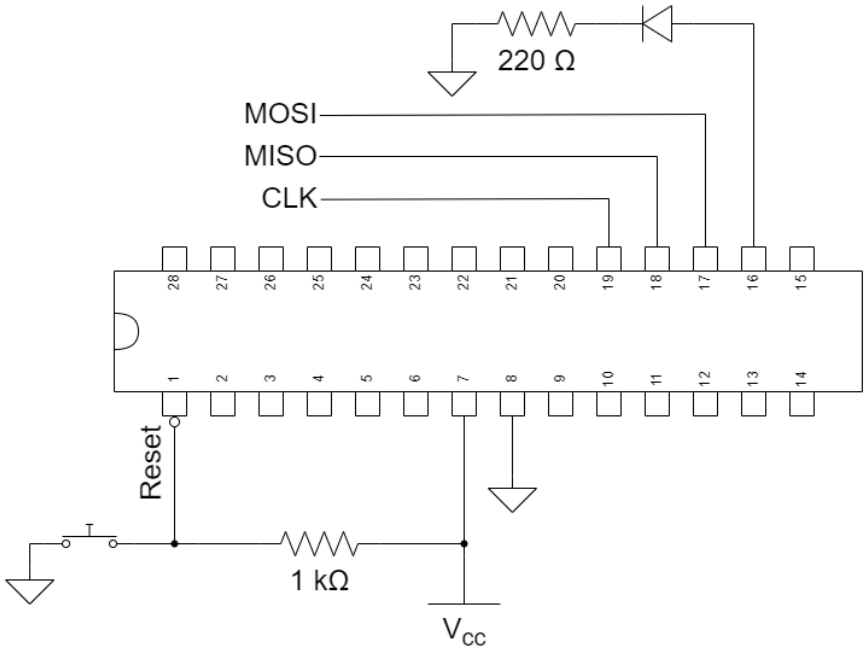

Figura 3: Diagrama esquemático del hardware.