Práctica 8

- **Manejo de la sección de E/S del microcontrolador ATmega1280/2560**

Objetivo: Mediante esta práctica el alumno analizará la implementación de retardos por software, así como también se familiarizará con la configuración y uso de puertos.

Equipo: - Computadora Personal con AVR Studio y tarjeta T-Juino.
Teoría: 1) Investigación a cerca de ensamblador en línea para GCC.
2) Análisis y cálculo del retardo por SW de la práctica.
3) Teoría sobre puertos de E/S (uC ATmega1280/2560)
4) Técnicas de anti-rebote de botones táctiles.

Descripción:

Implementar un programa en base al Listado 1, el cual revisa el estado del botón, y dependiendo de su duración que esta presionado incrementa o decrementa un contador, que a su vez se muestra en el arreglo de Leds; como se muestra en la Fig. 2.

Listado 1:

```
#include <avr/io.h>

// Macros
#define SetBitPort(port, bit) __asm__ ( ?? )
#define ClrBitPort(port, bit) __asm__ ( ?? )

// Press States
#define NOT_PRESSED 0
#define SHORT_PRESSED 1
#define LONG_PRESSED 2

// Prototypes
Void    delay(uint16_t mseg);
void    InitPorts(void);
uint8_t checkBtn(void);
void    updateLeds(void);

// Global variables
uint8_t  globalCounter;
uint32_t millis;

int main(void){
      InitPorts();

      while(1){
            switch(check_Btn()){
                  case SHORT_PRESSED: globalCounter++;
                                      break;
                  case LONG_PRESSED:  globalCounter--;
                                      break;
            }
            updateLeds();
            delay(1);
            millis++;
      }
}
```

Macros a implementar:

1. `SetBitPort(port, bit)`
   Macro que inserta la instrucción de ensamblador **SBI**, mediante *inline assembly*.

2. `ClrBitPort(port, bit)`
   Macro que inserta la instrucción de ensamblador **CBI**, mediante *inline assembly*.

Funciones a implementar:

3. `void delay(uint16_t mseg);`
   Función que debe de tardarse **n** *ms* en retornar, según se especifique en el parámetro de entrada. Con una exactitud de ±5 *us*.

4. `void InitPorts(void);`
   Inicialización requerida de los puertos utilizados en esta práctica. **PE1** debe dejarse en un nivel alto.

5. `uint8_t check_Btn(void);`
   Retorna el estado del botón, detectando entre `NOT_PRESSED`, `SHORT_PRESSED` y `LONG_PRESSED`. Donde el umbral para una larga duración es cualquiera que sea mayor a 1 *s*. Ignorando el rebote mecánico que se puede apreciar en la Fig 1.

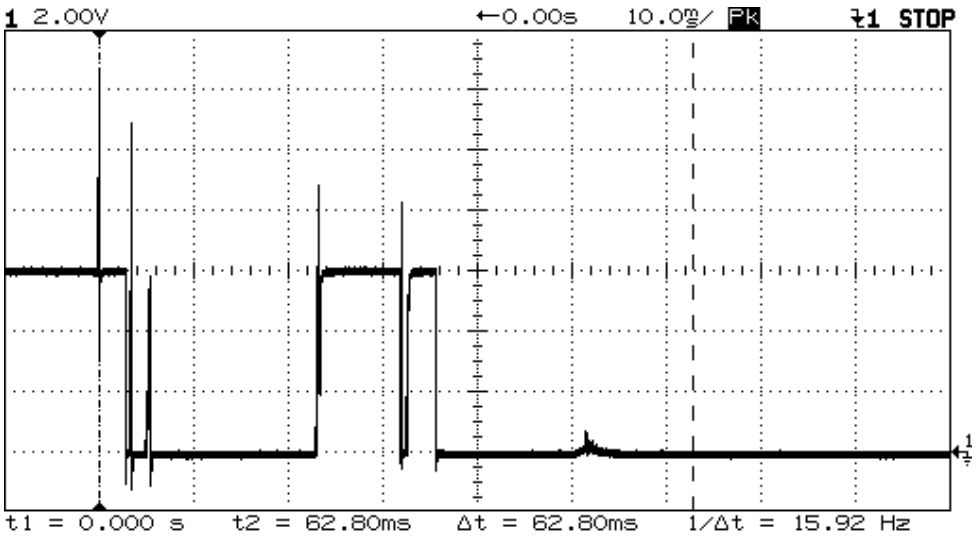

Fig. 1. Ejemplo del rebote mecánico de un botón.

6. `void updateLeds(void);`
   Refleja el valor del contador global en los Leds como se muestra en la Fig. 2. Realizar los ajustes necesarios de tal forma que no se perciba que parpadean.

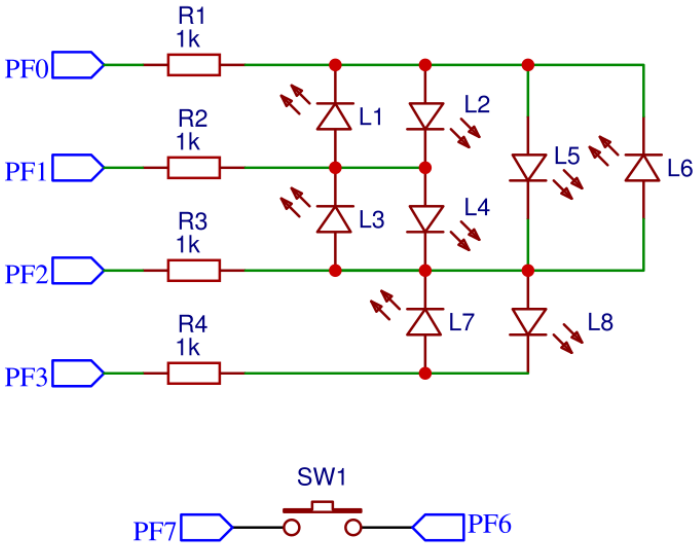

Fig. 2. Esquemático

Comentarios y Conclusiones.

Bibliografía y Referencias.