Práctica 5

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 y fuera de línea para AVR-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 dado en ***Prac5_IO.c***, el cual es un simple juego basado en el tiempo de reacción del jugador. Tiene como entrada un botón y dependiendo de su duración que está presionado determina la acción en el juego. A su vez se muestra en el arreglo de Leds un patrón que refleja el estado actual del juego; los cuales se deberán conectar como se muestra en la Fig. 2. Este juego pasa de estar en estado ***eWaitForStart*** para iniciar, cuando el jugador presiona el botón este estado cambia al ***eStartCount*** donde se muestra una secuencia aleatoria cada 100 ms durante 5 segundos, si en ese tiempo no se presionó el botón se pasa al estado ***eEndCount*** y si el jugador reacciona a tiempo después de que esta secuencia se mostró este gana o pierde y de ahí se pasa a ***eYouWin*** o ***eYouLoose,*** sin importar a cuál estado se cambio, el juego se queda ahí hasta que el jugador presione el botón de nuevo y se pase de nuevo a ***eWaitForStart*** **.**

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. `uint8_t myRand(uint8_t seed)`
   Función que deberá estar definida en ensamblador en un archivo .S, y tendrá que ser llamada desde C.

4. `void delay(uint8_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,* deberá estar definida en ensamblador en un archivo .S.

5. `void delay1us(uint8_t useg);`

Función que debe de tardarse **n** *us* en retornar, según se especifique en el parámetro de entrada. Con una exactitud del 100%, deberá estar definida en ensamblador en un archivo .S.

6. `void InitPorts(void);`
   Inicialización requerida de los puertos utilizados en esta práctica, pueden basarse en la Fig. 2, revisen los gameState para revisar la inicialización correcta.

7. `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.

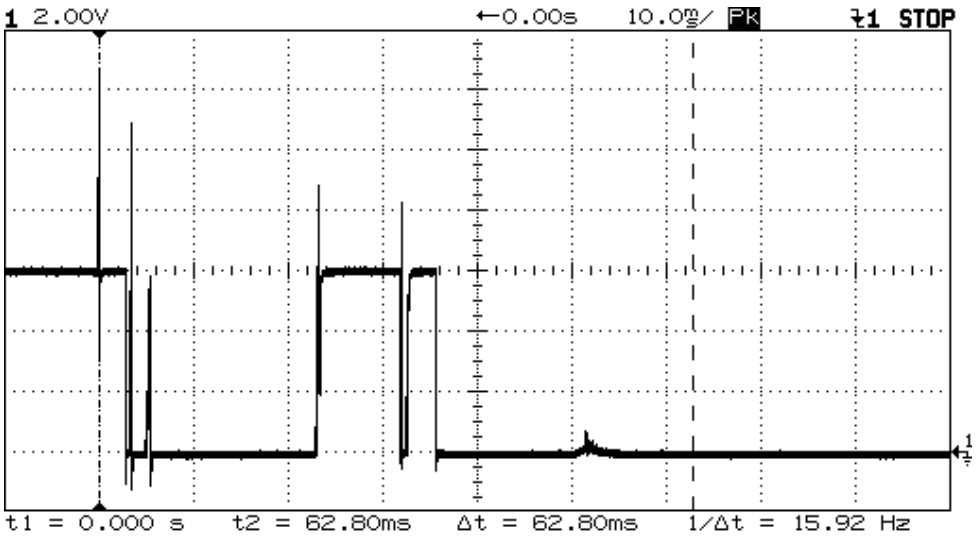

Fig. 1. Ejemplo del rebote mecánico de un botón.

8. `void updateLeds(uint8_t gameState)`
   Muestra el patrón actual que refleja el estado del juego. Estos estados son los siguientes:
   - ***eWaitForStart***:
     Sequencia de *walking-cero* del MSB al LSB, actualizándose cada 100 ms (s_0=0b11111111, s_1=0b01111111, …, s_8=0b11111110, y repetir), el 0 en el patrón indica el led a encender.
   - ***eStartCount:***
     Secuencia aleatoria mostrada en los leds que cambia cada 100 ms.
   - ***eEndCount:***
     Todos los LEDs apagados.
   - ***eYouLoose:***
     Secuencia donde se encienden en secuencia (s_0=0b11111111, s_1=0b01111110, s_2=0b10111101, s_3=0b11011011, s_4=0b11100111, y regresar de s4 a s0 luego repetir la secuencia) actualizándose cada 100 ms.
   - ***eYouWin:***

Secuencia que alterna entre pares y nones los LEDs, estos se encienden y apagan simultáneamente alternando entre pares y nones cada 250 ms.

Realizar los ajustes necesarios de tal forma que no se perciba que parpadean.

La Fig. 2 es solo una referencia, el juego requiere otra matriz de leds, deben apoyarse de esa imagen y utilizar los mismos puertos para los leds y para el botón.

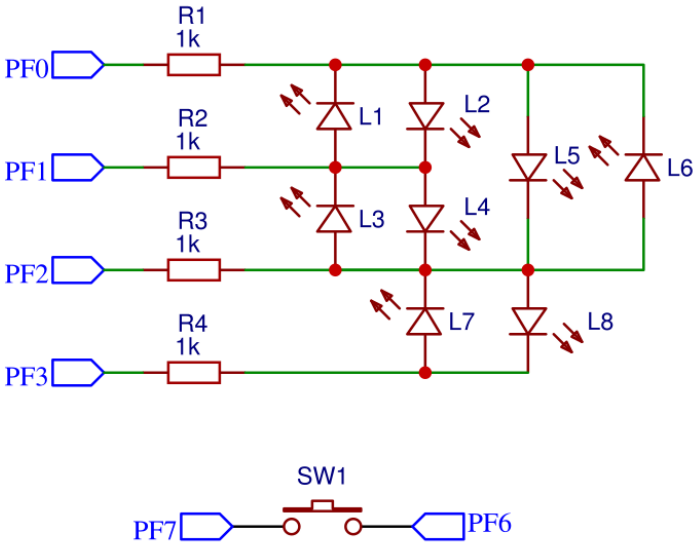

Fig. 2. Esquemático

Comentarios y Conclusiones.

Bibliografía y Referencias.