# Práctica 3

### Manejo de la sección de E/S del microcontrolador ESP32

**Objetivo:** El alumno se familiarizará con la configuración y uso de puertos mediante el uso del ESP-IDF SDK.

**Equipo:** - Computadora Personal y tarjeta de desarrollo del ESP32.
**Teoría:** - Técnicas de anti-rebote de botones
- Manejo de Puertos de Entrada y Salida

**Desarrollo:**

- Revisar documentación del SDK sobre la sección de entrada y salida ESP-IDF GPIO.
- Revisar configuración del SDK del parámetro `CONFIG_FREERTOS_HZ`, para esta práctica se necesitara que este configurado a `1000`. Eso logrará que nuestros retardo puedan tener resolución de 1 mili-segundo.
- Implementar el juego del Rebote (algo similar a "Pong" de 1-dimensión) en base al esquemático de la Fig. 1 y el código proporcionado en el repositorio.

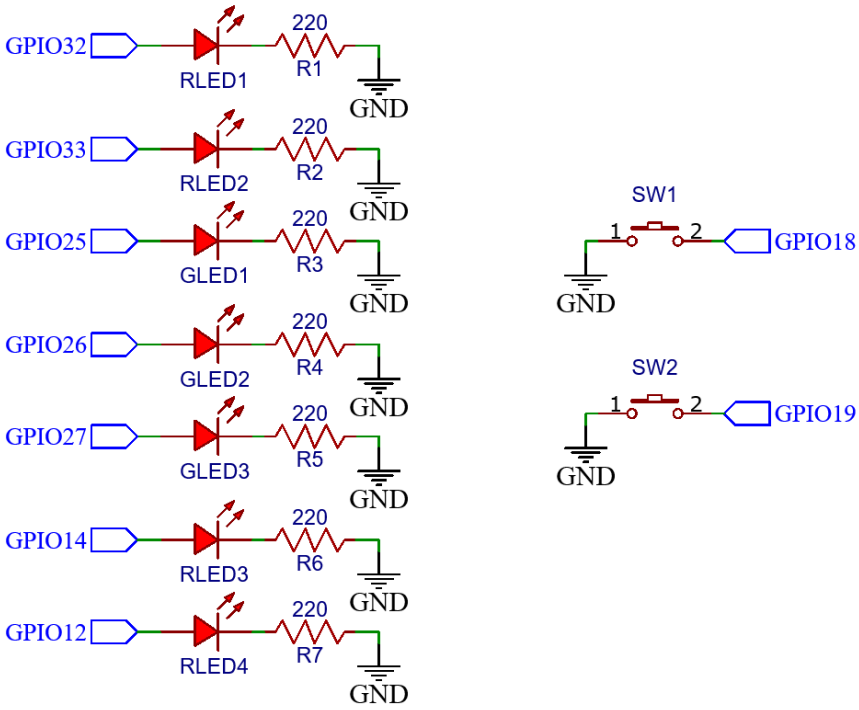


**Fig. 1. Esquemático**

**Funciones a implementar:**

1. *void* ***initIO****(void)*
   Inicialización requerida de los puertos utilizados en esta práctica, según la Fig. 1.

2. *eButtonState_t* ***checkButtons****(eButtonId_t *buttonNumber)*
   Retorna el ID del botón detectando y su possible estado *eBtnUndefined*, *eBtnShortPressed* y *eBtnLongPressed*. Donde el umbral para una larga duración es cualquiera que sea mayor a 1 segundo. Es importante ignorar el rebote mecánico, un ejemplo de este rebote se puede apreciar en la Fig 2.

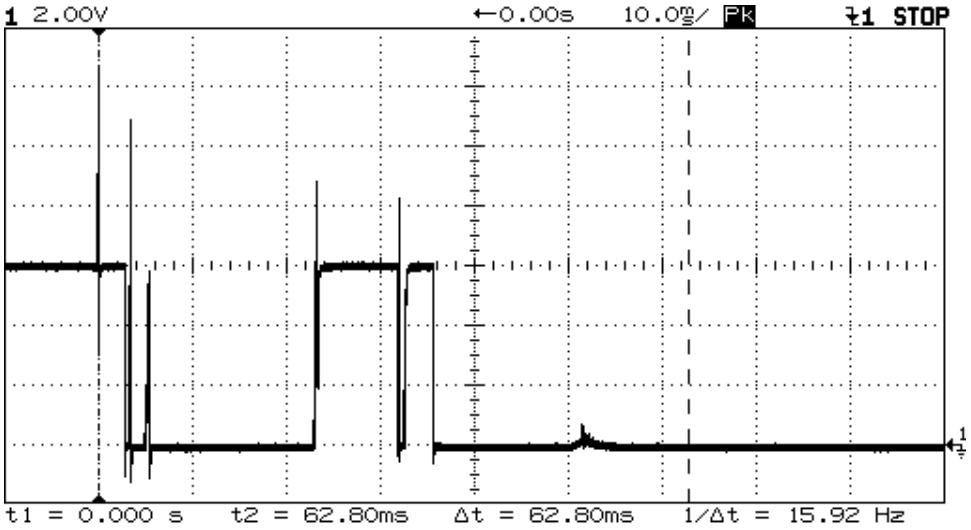


**Fig. 2. Ejemplo del rebote mecánico de un botón.**

3. `bool playSequence(eGameState_t gameState)`
   Muestra el patrón actual que refleja el estado del juego. Estos estados son los siguientes:
   - ***eWaitForStart***:
     Sequencia de *walking-zero* del LED de mas a la izquierda al LED de mas a la derecha, actualizándose cada 500 ms ($s_0$=0b1111, $s_1$=0b0111, …, $s_4$=0b1110, y repetir)
   - ***eOngoingGame:***
     Sequencia de *walking-one* del MSB al LSB con rebote, actualizándose cada 300 ms ($s_0$=0b0001, $s_1$=0b0010, …, $s_3$=0b1000, $s_4$=0b0100, $s_6$=0b0010 y repetir) con decremento de 20 ms en cada rebote.
   - ***eEnd:***
     Muestra el puntaje obtenido por el jugador (con un factor de escala de 0.5) sobre un solo LED, donde el LED de mas a la izquierda representa un puntaje de 2, y el de mas a la derecha de 14. Este LED deberá estar parpadeando con un patrón de 300 ms encendido y 50 ms apagado.

4. `ePlayerInputState_t checkPlayerInput(eButtonState_t buttonState, eButtonId_t buttonId)`

   Revisa el estado actual de los botones y valida que sean presionados en una ventana de tiempo correcta. Esta ventana esta dada de la siguiente manera, cuando la "pelota" vaya en sentido de los LEDs a la izquierda, el usuario debe presionar el botón de la izquierda cuando alguno de los dos LEDs del extremo estén encendidos, y misma racionalización en el otro sentido.

***Nota:***
Es importante tener en cuenta que ***ninguna de las dos funciones anteriores debe bloquear la tarea***, ya que que ambas deben aparentar que están corriendo al mismo tiempo. Para lograr esto, hacer uso de la variable global de mili-segundos.

## Comentarios y Conclusiones.

## Bibliografía y Referencias.