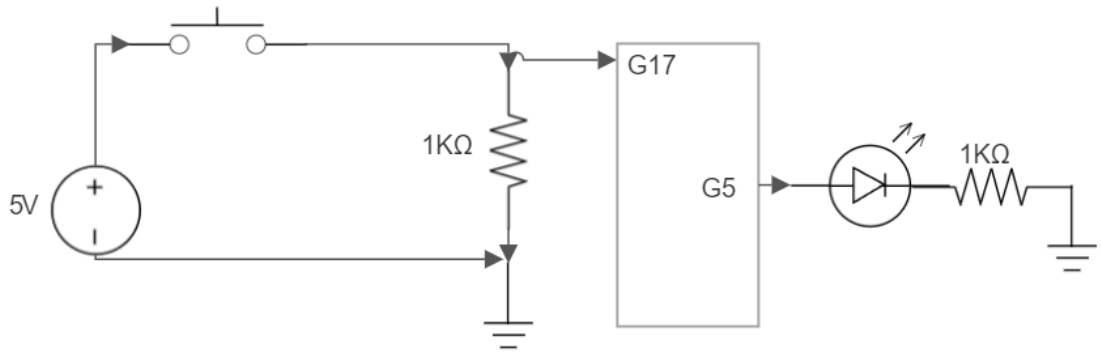
Actividad 2

A partir del proyecto de ejemplo *blink* disponible en el aula virtual, genera un proyecto en el que un led se encienda o apague cada vez que se aprieta un pulsador. Además, se deberá mostrar por pantalla un mensaje de información cada vez que se modifique el estado del led. Para desarrollar la actividad, se propone el siguiente material:

- **1 x diodo led.**
- **2 x resistencia 1KΩ.**
- **1 x pulsador.**

Considera las siguientes conexiones para el montaje del circuito:



Nota: puedes usar el pin V5 de ESP32 como fuente de 5V, así como el pin GND para tierra.

Solución:

De acuerdo con el enunciado, se deberá activar (G5 nivel HIGH) o desactivar (G5 nivel LOW) cada vez que se acciona el pulsador. Ten en cuenta que, mientras el pulsador esté desactivado, la tensión de G17 = 0V (LOW). Cuando se active, la tensión de G17 pasará a 5V (HIGH).

Para resolver el problema anterior, se deben implementar las siguientes funcionalidades:

- Configuración del pin para el led: al igual que en proyecto *Blink,* es necesario configurar un pin GPIO de *salida* para la gestión del led (en este caso, G5).
- Configuración del pin para el pulsador: se debe configurar un pin GPIO de *entrada* que lea el estado del pulsador (G17).
- Encendido/apagado del led: funcionalidad que fije el nivel (LOW o HIGH) de G5.
- Lectura de pulsador y selección gestión del estado del led: se deberá *leer* el estado de G17 (LOW o HIGH). Cada vez que haya un cambio de estado, se modificará el estado de la salida G5.

En primer lugar, vamos a incluir los archivos de cabecera que emplearemos en el código.

```c
#include <stdio.h>

#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/gpio.h"
#include "esp_log.h"
#include "sdkconfig.h"
```

A continuación, se procede a:

- Definir una etiqueta a incluir en los mensajes de información con el texto "Actividad 2".
- Definir un alias para el GPIO del led (LED_GPIO) con valor 5, y para el GPIO del pulsador (BUTTON_GPIO) con valor 17.
- Definir dos variables globales tipo uint8_t para el estado del led (s_led_state) y pulsador (s_button_state), ambas inicializadas a 0. Por un lado, s_button_state pasará a 1 cada vez que se accione el pulsador, y s_led_state estará a 1 cuando el led esté encendido.

```c

static const char *TAG = "Actividad 2";


#define LED_GPIO 5
#define BUTTON_GPIO 17

static uint8_t s_led_state = 0;
static uint8_t s_button_state = 0;

```

Definimos la función de configuración del led, *configure_led()*:

```c
// Function to configure led
static void configure_led(void)
{
    ESP_LOGI(TAG, "Configuring LED!");
    gpio_reset_pin(LED_GPIO);
    /* Set the GPIO as a push/pull output */
    gpio_set_direction(LED_GPIO, GPIO_MODE_OUTPUT);
}
```

Definimos la función de configuración del pulsador, *configure_button()*. Observa que la principal diferencia con respecto a led reside en la dirección, ahora de entrada (GPIO_MODE_INPUT).

```c
// Function to configure button
static void configure_button(void)
{
    ESP_LOGI(TAG, "Configuring BUTTON!");
    gpio_reset_pin(BUTTON_GPIO);
    /* Set the GPIO as a push/pull output */
    gpio_set_direction(BUTTON_GPIO, GPIO_MODE_INPUT);
}
```

Definimos una función *blink_led()* encargada de gestionar el estado del led en función de *s_led_state:*

```c
static void blink_led(void)
{
    /* Set the GPIO level according to the state (LOW or HIGH)*/
    gpio_set_level(LED_GPIO, s_led_state);
    ESP_LOGI(TAG, "Led is %s", s_led_state == true ? "ON" : "OFF");
}
```

Finalmente, creamos la tarea principal *app_main()*. El objetivo de esta función será:

1. Llamar a la función *configure_led()*.
2. Llamar a la función *configure_button()*.
3. Comprobar de forma continua el estado del pulsador:

   Si el estado está a 1 (se acciona el pulsador)

   - Se muestra un mensaje informativo sobre la activación del pulsador.
   - Si s_button_state era 0, implica que se acaba de activar el pulsador, por lo que se debe modificar el estado del led, s_led_state. Además, se actualiza el s_button_state a 1.

   Si el estado está a 0 (pulsador no accionado), el código no debe hacer nada.

Para conseguir el efecto de comprobación continua del estado del pulsador, se programa un bucle infinito *while.* Para evitar problemas de ejecución, se añade un delay muy bajo al final de cada iteración. En este caso, se ha propuesto 100ms.

```c
void app_main(void)
{
    /* Configure the peripheral according to the LED type */
    configure_led();
    configure_button();

    while (1) {

        if (gpio_get_level(BUTTON_GPIO)==1){
            printf("LEVEL TO ONE %i",gpio_get_level(BUTTON_GPIO));
            if (s_button_state == 0)
            {
                s_led_state = !s_led_state;
                blink_led();
                s_button_state = 1;

            }


        }else {
            s_button_state = 0;
        }


        vTaskDelay(100 / portTICK_PERIOD_MS);
    }
}
```

El resultado tras el *build*, *flash* y *monitor* es el siguiente:

```
W (292) spi_flash: Detected size(4096k) larger than the size in the binary image header(2048k). Using the size i
n the binary image header.
I (306) main_task: Started on CPU0
I (316) main_task: Calling app_main()
I (316) Actividad 2: Configuring LED!
I (316) gpio: GPIO[5]| InputEn: 0| OutputEn: 0| OpenDrain: 0| Pullup: 1| Pulldown: 0| Intr:0
I (326) Actividad 2: Configuring BUTTON!
I (326) gpio: GPIO[17]| InputEn: 0| OutputEn: 0| OpenDrain: 0| Pullup: 1| Pulldown: 0| Intr:0
LEVEL TO ONE 1I (1636) Actividad 2: Led is ON
LEVEL TO ONE 1I (2836) Actividad 2: Led is OFF
LEVEL TO ONE 1I (5136) Actividad 2: Led is ON
LEVEL TO ONE 1I (7236) Actividad 2: Led is OFF
```

Se observa que cada vez que se acciona el pulsador aparece el mensaje “LEVEL TO ONE”. A continuación, se muestra el nuevo estado del led (ON/OFF).

Nota: existen distintas alternativas para resolver este ejercicio. Una variante podrías ser el uso de interrupciones con FreeRTOS asociada al pulsador.