Guía de Ejercicios Prácticos 2

"Manejo de Timer y RTC"

Objetivos

- Manejo del módulo Timer y RTC del microcontrolador RP2040.
- Control de los periféricos utilizando pooling e interrupciones.
- Manejo de componentes electrónicos básicos como leds, resistencias, displays 7 segmentos, capacitores y transistores.
- Utilización de conocimientos previos.
- Desarrollo de la capacidad de interpretar la documentación relacionada al hardware y correspondiente a las librerías de software utilizadas.

Funciones útiles para está guía

A continuación se listan algunas funciones útiles del Software Development Kit de Raspberry. Está lista no es exhaustiva, por lo tanto se recomienda consultar las restantes. Para saber más sobre el manejo de estas funciones debe consultar la documentación. (Como dato orientativo el nombre de las funciones empieza siempre con el módulo al que pertenece)

Lectura del timer

- static uint32_t time_us_32 (void) [inline], [static]
- uint64_t time_us_64 (void)
- static uint32_t timer_time_us_32 (timer_hw_t * timer) [inline], [static]
- uint64_t timer_time_us_64 (timer_hw_t * timer)

Configuración alarmas e interrupción

- bool hardware_alarm_set_target (uint alarm_num, absolute_time_t t)
- bool hardware_alarm_is_claimed (uint alarm_num)
- void timer_hardware_alarm_force_irq (timer_hw_t * timer, uint alarm_num)
- void timer_hardware_alarm_cancel (timer_hw_t * timer, uint alarm_num)
- static uint hardware_alarm_get_irq_num (timer_hw_t * timer, uint alarm_num) [inline], [static]

Configuración de interrupciones periódicas

- bool alarm_pool_add_repeating_timer_us (alarm_pool_t *pool, int64_t delay_us, repeating_timer_callback_t callback, void *user_data, repeating_timer_t *out)
- static bool alarm_pool_add_repeating_timer_ms (alarm_pool_t *pool, int32_t delay_ms, repeating_timer_callback_t callback, void *user_data, repeating_timer_t *out)
- static bool add_repeating_timer_us (int64_t delay_us, repeating_timer_callback_t callback, void *user_data, repeating_timer_t *out)
- static bool add_repeating_timer_ms (int32_t delay_ms, repeating_timer_callback_t callback, void *user_data, repeating_timer_t *out)
- bool cancel_repeating_timer (repeating_timer_t *timer)

Inicialización, configuración y habilitación del RTC

- void rtc_init (void)
- bool rtc_set_datetime (const datetime_t *t)
- bool rtc_get_datetime (datetime_t *t)
- void rtc_set_alarm (const datetime_t *t, rtc_callback_t user_callback)
- void rtc_enable_alarm (void)

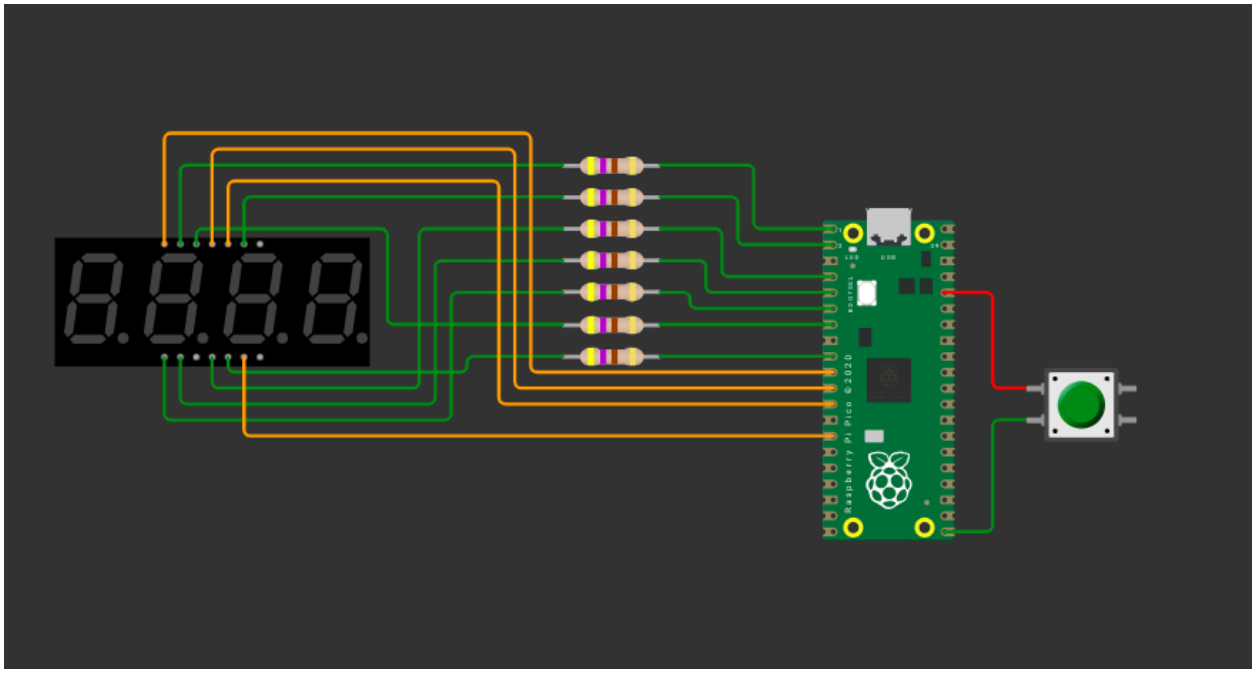
Ejercicio 1 “Blinking”

Implemente un proyecto donde se haga parpadear un led (puede ser el propio de la Raspberry Pico) a una frecuencia de 1[Hz] utilizando el Timer.

Ejercicio 2 “Cronómetro”

En este proyecto el alumno deberá construir un cronómetro con inicio, detención y restauración (vuelta a cero).
El circuito sugerido coincide con el del ejercicio “Contador Extendido” de la Guia 1.

Circuito Propuesto



No olvide colocar los transistores en corte-saturación que activan los distintos dígitos del display.
También se sugiere la colocación de un capacitor de 100[nF] en paralelo al pulsador (este cumple la función de antirebote)

Consideraciones de software

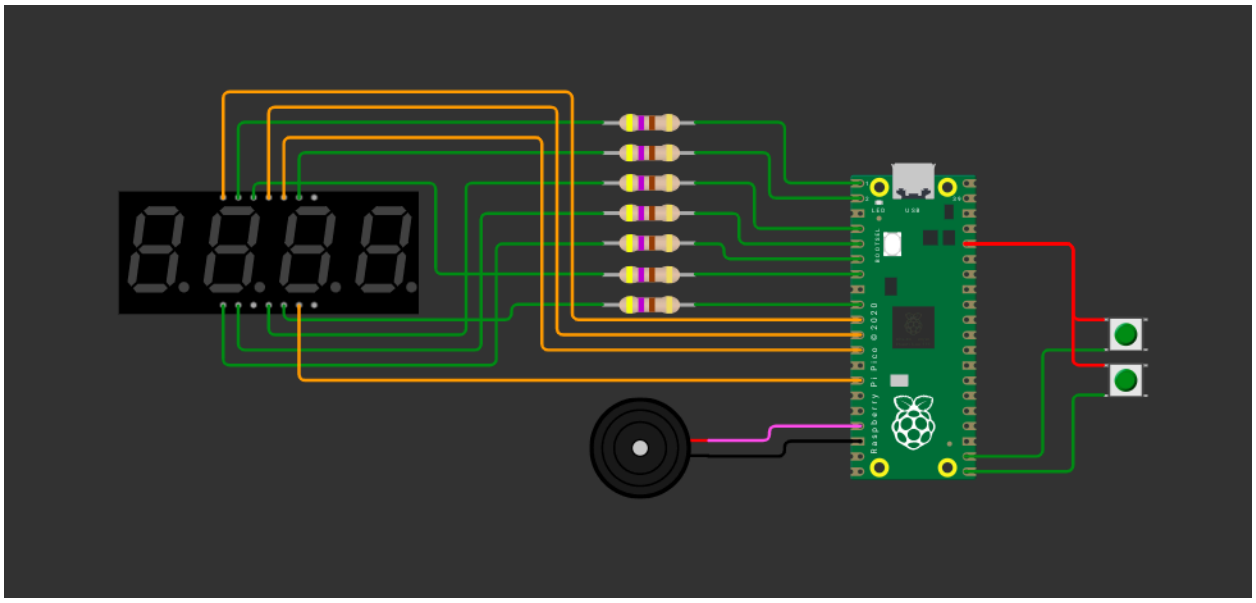
Para este problema se espera la utilización del módulo Timer, este se puede utilizar por diferencia en la cuenta o por interrupciones periódicas.

Recordar configurar la correspondiente resistencia de pull-down interna del pin asociado al pulsador.

Ejercicio 3 “Reloj despertador”

En este ejercicio se espera que el alumno construya un reloj despertador, el objetivo es que configure correctamente el RTC del microcontrolador y habilite una interrupción de alarma que haga sonar un buzzer (o simplemente encienda un led). La hora debe mostrarse en el display de 4x7segmentos en formato Hora-Minuto (HHMM).
La hora en que inicia el reloj y la alarma pueden ser iniciadas por default (desde programación), o si el alumno lo prefiere puede agregar el hardware que considere necesario para setear la hora (y/o la hora del alarma).

Circuito Propuesto



No olvide colocar los transistores en corte-saturación que activan los dígitos del display. También se sugiere la colocación de un capacitor de 100[nF] en paralelo al pulsador (este cumple la función de antirebote). También puede ser necesario el uso de un transistor para activar el buzzer.