Curso:

Arquitectura y Programación de Sistemas Embebidos

junio de 2024

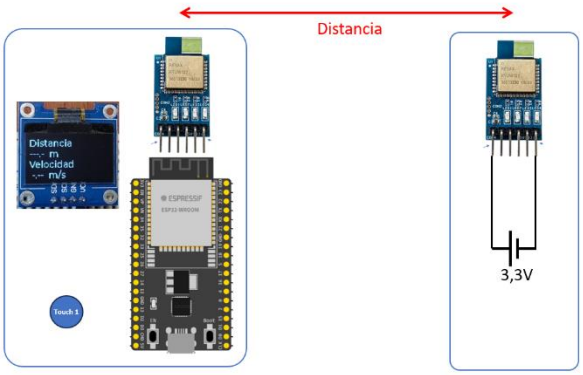
Alumno:

Juan Ignacio Spinetto

Tabla de Contenidos

1. Descripción del Proyecto

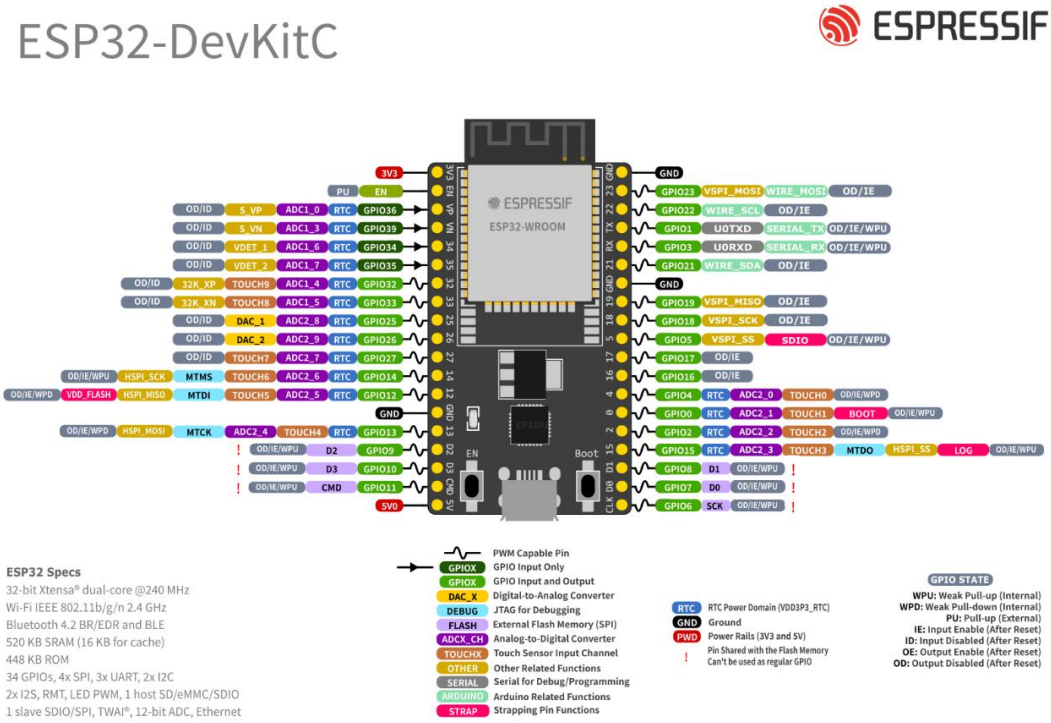
A partir de dos periféricos elegidos, una pantalla oled de 0,96 pulgadas (128x64px) y un transceiver UWB RYUW122 (ANCHOR) capaz de medir distancia a su par equivalente (TAG), se escribirán los drivers para una placa de desarrollo que utiliza un esp32. Para verificar el funcionamiento de estos drivers se implementará un programa básico capaz de obtener la distancia y velocidad relativa entre el par de RYUW122, mostrando estos valores en pantalla oled, con la posibilidad de cambiar las unidades de velocidad presionando un botón táctil capacitivo.



2. Descirpción de Hardware

2.1. Placa de Desarrollo

La placa de Desarrollo utilizada es una ESP32-DevKit V4. Ésta una placa de desarrollo basada en ESP32 de pequeño tamaño producida por Espressif. La mayoría de los pines de E/S están divididos en los cabezales de los pines en ambos lados para facilitar la interfaz. Los desarrolladores pueden conectar periféricos con cables de puente o montar ESP32-DevKitC V4 en una placa.



2.2. Periféricos: OLED

Pantalla OLED monocromo de 0,96" y 128 x 64 pixeles, con conectividad I2C, utiliza el módulo SSD1306.

El SSD1306 se integra con control de contraste, RAM de pantalla y oscilador, lo que reduce el número de componentes externos y el consumo de energía. Tiene control de brillo de 256 pasos. Los datos/comandos se envían desde la MCU general a través de la interfaz I^2C.

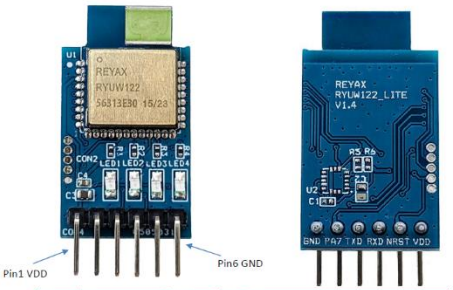
2.3. Periféricos: RYUW122_lite

REYAX RYUW122 está diseñado como un algoritmo inteligente y un módulo UWB (banda ultra ancha) de alta calidad, y permite una medición de distancia segura y precisa. La comunicación con la placa de desarrollo es serial y se controla a través de comandos AT.

En este proyecto se usará la versión REYAX RYUW122_Lite que incluye la placa de evaluación, el propósito es permitirle comprender y probar el módulo REYAX RYUW122 más rápidamente.

Caracteristicas

- Supports IEEE802.15.4 2015 UWB & IEEE802.15.4z (BPRF mode)
- Supports channels 5 & 9 (6489.6MHz & 7987.2 MHz)
- Worldwide UWB Radio Regulatory compliance
- Location to an accuracy of 10 cm
- Control easily by AT commands
- Provides precision location and data transfer simultaneously.
- Designed with integrated antenna.
- Integrated AES 1 28

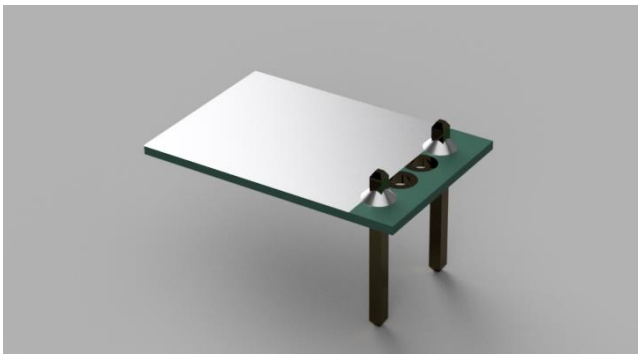


Pin	Name	I/O	Condition
1	VDD	P	Power Supply
2	NRST	I	Low reset trigger input
3	RXD	I	UART Data Input
4	TXD	O	UART Data Output
5	PA7	O	Mode Indicator Hi : Normal mode, Low : Sleep mode.
6	GND	P	Ground

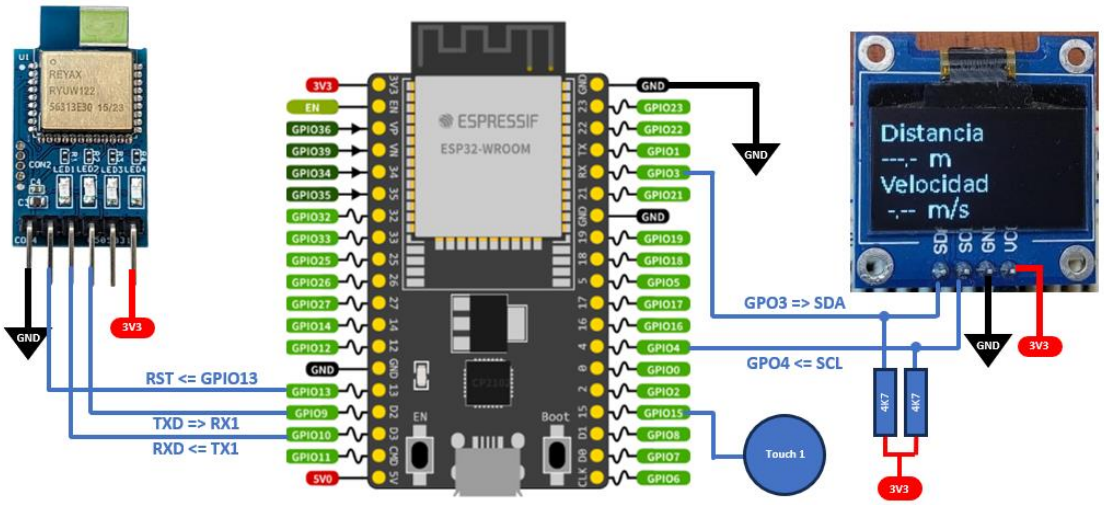
2.4. Periféricos: Botón tactil

El ESP32 cuenta con puertos que pueden ser usados como contactos capacitivos. Se implementa uno de estos puertos a través de conexión a una pequeña superficie metálica usada como sensor táctil. En esta implementación se conecta al TOUCH02 / GPIO15 de la placa de desarrollo.

Se implementará un driver específico para hardware, para identificar cuando el botón es presionado (o tocado) independizando al programa principal de los detalles específicos de los mandos táctiles.



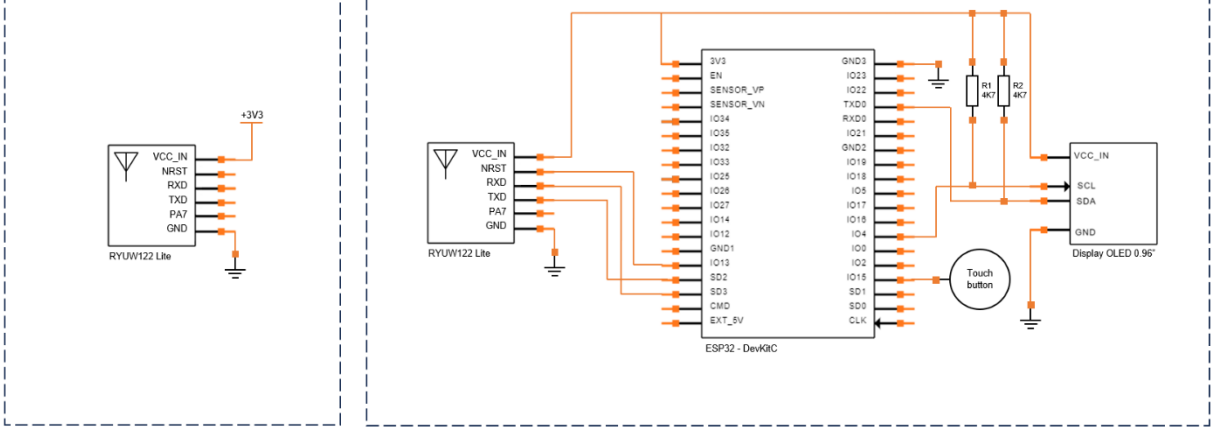
2.5. Conexiones



2.6. Esquemático

TAG
(No requiere un sistema programable)

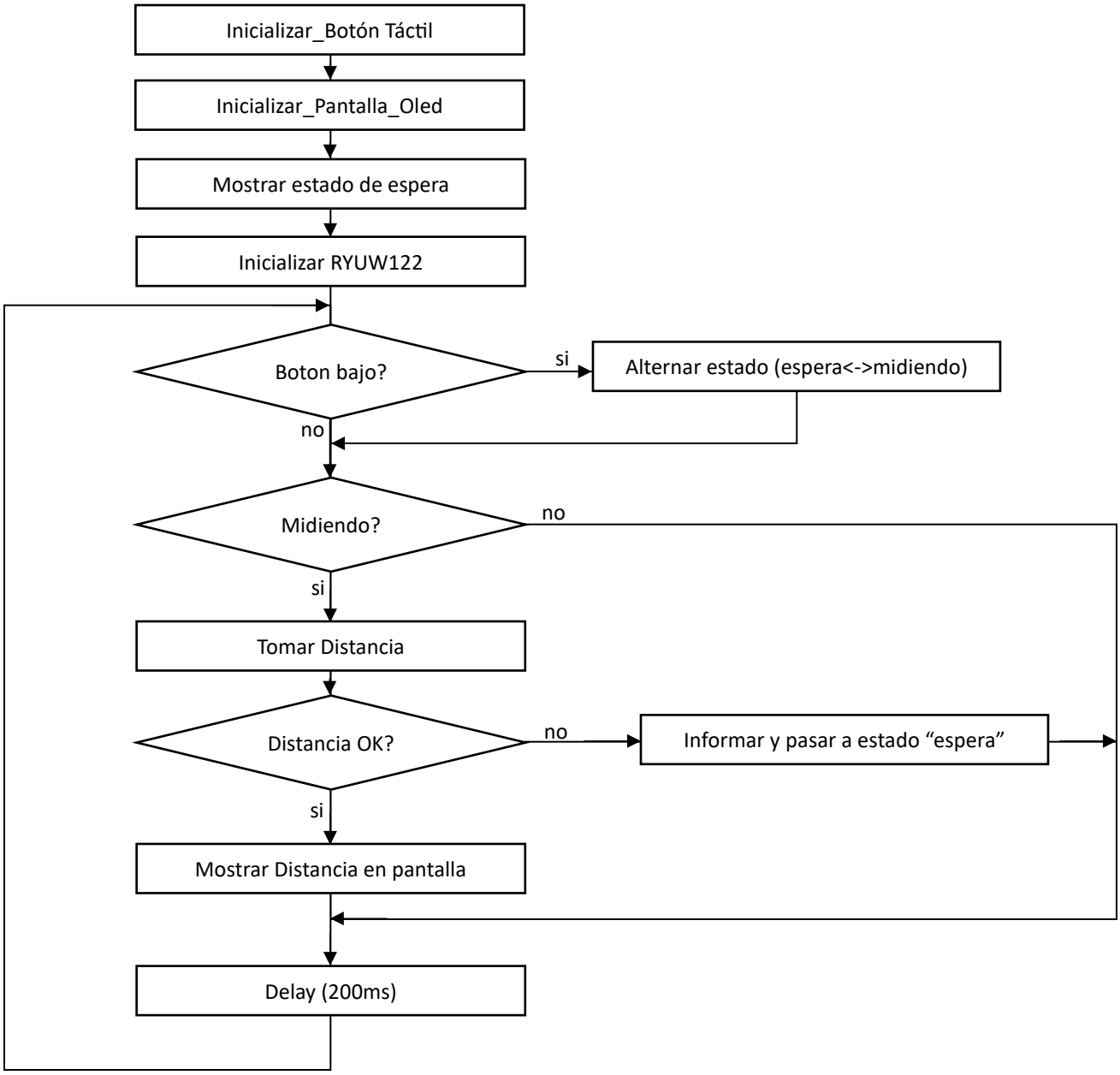
Sistema Principal
(Aloja el firmware desarrollado)



* El TAG para la aplicación de medición de distancia funciona solo como “eco”, por lo que no necesita estar conectado a un sistema programable. El TAG debe configurarse una única vez en modo TAG, nombre de red, etc.

3. Descripción de Software - Main

Main



4. Driver OLED ssd1306

Para esta aplicación se implementará un driver básico que controla las funciones de comunicación i2c, permita Inicializar, Configurar y escribir texto de tamaño único a partir de una fila y columna.

```c
/**
 * @file      driver_oled_ssd1306.h
 * @brief     driver ssd1306 basic header file
 * @version   1.0.0
 * @author    Juan I Spinetto
 * @date      14/07/2024
 *
 */


/*---------------
--   include    -
----------------
*/
#include "esp_err.h"
#include <stdbool.h>

/**
 * @brief     basic init de la pantalla oled
 * @return    ESP_OK
 *            ESP_FAIL
 * @note      inicializa el oled e ic2
 */
esp_err_t  oled_ssd1306_init(void);

/**
 * @brief  borra toda la información de la pantalla
 * @return none
 * @note   none
 */
void oled_ssd1306_clear_display(void);

/**
 * @brief  Escribe un caracter en una posición especifica
 * @param[in] c es el caracter en ascii
 * @param[in] x es la coordenada x en pixels de 0 a 127
 * @param[in] y es la coordenada y en filas de 0 a 7
 * @return    none
 * @note      none
 */
void oled_ssd1306_draw_char(char c, int x, int y);

/**
 * @brief  escribe un string a partir de una posición especifica
 * @param[in] str es el string a escribir
 * @param[in] x es la coordenada x en pixels de 0 a 127
 * @param[in] y es la coordenada y en filas de 0 a 7
 * @return    none
 * @note      none
 */
void oled_ssd1306_draw_string(const char *str, int x, int y);

/**
 * @brief  basic example clear
 * @param[in] x es la coordenada x en pixels de 0 a 127
 * @param[in] y es la coordenada 7 en pixels de 0 a 63
 * @return none
 * @note   none
 */
void oled_ssd1306_draw_pixel(int x, int y);
```

5. Driver RYUW122

Se implementará el driver completo controla la comunicación serie con el periférico, permite Inicializar, Configurar, Comunicación con otro dispositivo, obtención de distancia entre el ANCHOR y el TAG.

```c
/**
 * @file driver_RYUW122.h
 *
 */

#ifndef DRIVER_RYUW122_H
#define DRIVER_RYUW122_H

/*********************
 *      INCLUDES
 *********************/
#include "freertos/FreeRTOS.h"

/*********************
 *      DEFINES
 *********************/
#define My_Network_ID "NTJIS001";
#define My_Address_ID "ADJIS001";

/**********************
 *      TYPEDEFS
 **********************/
typedef enum {
    RYUW122_UART_CONFIG_0,         // Set RYUW122 uses UART_0
    RYUW122_UART_CONFIG_1,         // Set RYUW122 uses UART_1
    RYUW122_UART_CONFIG_2,         // Set RYUW122 uses UART_2
} ryuw122_UART_conf_t;

typedef enum {
    RYUW122_MODO_TAG,          // Set/get RYUW122 TAG mode
    RYUW122_MODO_ANCHOR,       // Set/get RYUW122 ANCHOR mode
    RYUW122_MODO_SLEEP        // Set/get RYUW122 SLEEP mode
} ryuw122_modo_t;

typedef enum {
    RYUW122_IPR_9600 ,       // Set/get RYUW122 baud rate to 9600
    RYUW122_IPR_57600,       // Set/get RYUW122 baud rate to 57600
    RYUW122_IPR_115200,      // Set/get RYUW122 baud rate to 115200
    RYUW122_IPR_DEFAULT      // Set deafaul value to 115200
} ryuw122_ipr_t;

typedef enum {
    RYUW122_CHANNEL_6489,      // Set/get RYUW122 Channel to 6489.6MHz
    RYUW122_CHANNEL_7987,      // Set/get RYUW122 Channel to 7987.2 MHz
    RYUW122_CHANNEL_DEFAULT    // Set Default Channel to 6489.6MHz
} ryuw122_channel_t;

typedef enum {
    RYUW122_BANDWIDTH_850Kb,      // Set/get RYUW122 Bandwidth to 850 Kbps
    RYUW122_BANDWIDTH_68Mb,       // Set/get RYUW122 Bandwidth to 6.8 Mbps
    RYUW122_BANDWIDTH_DEFAULT     // Set Default Bandwidth to 850 Kbps
} ryuw122_bandwidth_t;

typedef struct {
    ryuw122_modo_t rModo;             /*Modo*/
    ryuw122_ipr_t rIpr;               /*Baud Rate*/
    ryuw122_channel_t rChannel;       /*Channel*/
    ryuw122_bandwidth_t rBandwidth;   /*Bandwidth*/
    char rNetworkId[20];              /*Network ID*/
    char rAddressId[20];              /*Address ID*/
} ryuw122_config_t;

typedef struct {
    ryuw122_UART_conf_t rUartConf;      /*UART connection*/
    int resetGPIO;                    /*Reset GPIO*/
} ryuw122_hardward_config_t;

/*======================
 * FUNTIONS
 *======================*/

/**
-------------------------------------------------------------------------------------------+
-  RYUW122_Init_Default
-  @brief  Inicializa Transsiver RYUW122
-
-  @note
-  Hardward - Por defecto establece la la UART_1 y el GPIO 13 para reset
-    Para usar otra configuración debe llamarse previamente a "Set_Hardward_Config"
-  @param  none
-  @return
-     - ESP_OK    Success
-     - ESP_FAIL Parameter error
-------------------------------------------------------------------------------------------+
```

```
*/
esp_err_t RYUW122_Init_Default (void);

/**
-------------------------------------------------------------------------------------------+
-  RYUW122_Init
-  @brief  Inicializa Transsiver RYUW122
-
-  @note
-  Hardward - Por defecto establece la la UART_1 y el GPIO 13 para reset
-    Para usar otra configuración debe llamarse previamente a "Set_Hardward_Config"
-  @param  rConfigSetting Incluye la configuracion completa del transiver
-                         debe definise antes de llamar a la función Init
-  @return
-     - ESP_OK   Success
-     - ESP_FAIL Parameter error
-------------------------------------------------------------------------------------------+
*/
esp_err_t RYUW122_Init (ryuw122_config_t rConfigSetting);

/**
-------------------------------------------------------------------------------------------+
-  Start_Continue_Measurement
-  @brief  Inicial el uso de la interrupciónX. Cada X ms obtendrá un nuevo valor de distacia
-          y calculará la velocidad en función de la distancia anterior y el intervalo.
-          Los valores de Distancia y Velocidad se actualizarán en las posiciones de memoria
-          enviadas como parametros
-
-  @note   Esta función utiliza la interrupción X
-          Los valores de distancia son filtrados o promediados para disminuir el ruido
-          generado por la variacion de la medicion.

-  @param  interval El intervalo de activación de la interrupcion en ms
-  @param  dist  Direccion de memoria donde se actulizarán el valor de distancia
-  @param  Vel  Direccion de memoria donde se actulizarán el valor de Velocidad
-  @param  newData  Direccion de memoria a la bandera de nuevo dato (0 - No hay nuevo dato; 1 hay nuevo Dato)
-  @return
-     - ESP_OK   Success
-     - ESP_FAIL Parameter error
-------------------------------------------------------------------------------------------+
*/
esp_err_t Start_Continue_Measurement (int interval, int* dist, int* Vel, int* newData);

/**
-------------------------------------------------------------------------------------------+
-  Stop_Continue_Measurement
-  @brief  Deshabilita la interrupción iniciada en "Start_Continue_Measurement"
-  @note   Los valores de Distancia y velocidad, ya no se actualizarán
-  @return
-     - ESP_OK   Success
-     - ESP_FAIL Parameter error
-------------------------------------------------------------------------------------------+
*/
esp_err_t Stop_Continue_Measurement (void);

/**
-------------------------------------------------------------------------------------------+
-  Tomar_Distancia_Actual
-  @brief  Realiza una medición puntual de distancia
-  @note   Para que esta medición arroje un valor valido, el ANCHOR y TAG deben estar
-          encencidos, y con capacidad de comunicarse, con la misma configuracion de
-          BR, Channel, Bandwidth, Network ID y Address ID.
          Un valor de retorno 0 corresponde a un error
-  @return  Devuelve el valor de distacia expresado en cm
-           Si el Valor es 0, corresponde a un error.
-------------------------------------------------------------------------------------------+
*/
int Tomar_Distancia_Actual (void);

/**
-------------------------------------------------------------------------------------------+
-  Get_Config_Setting
-  @brief  Lee los valores de configuración del RYUW122
-  @note   Para leer estos valores el transiver debe inicializarse, si no es así ésta
-          función lo inicializará.
-  @return
-     - ESP_OK   Success
-     - ESP_FAIL Parameter error
-------------------------------------------------------------------------------------------+
*/
esp_err_t Get_Config_Setting(ryuw122_config_t *rConfigSetting);

/**
-------------------------------------------------------------------------------------------+
-  Set_Config_Setting
-  @brief  Reescribe los valores de configuración del RYUW122
-  @note   Para escribir estos valores el transiver debe inicializarse, si no es así ésta
-          función lo inicializará.
-  @return
-     - ESP_OK   Success
-     - ESP_FAIL Parameter error
-------------------------------------------------------------------------------------------+
```

```
*/
esp_err_t Set_Config_Setting (ryuw122_config_t rConfigSetting);

/**
-------------------------------------------------------------------------------------------+
-  Set_Hardward_Config
-  @brief  Define los valores de configuración de hardware
-  @return
-     - ESP_OK   Success
-     - ESP_FAIL Parameter error
-  @note Se usa en el caso que no sequieran usar los valores por defecto.
         UART_1 y el GPIO 13 para reset
-------------------------------------------------------------------------------------------+
*/
esp_err_t Set_Hardward_Config (void);

/**
-------------------------------------------------------------------------------------------+
-  @brief  Se usa para tester la comunicación con el periferico
-  @return
-     - ESP_OK   Success
-     - ESP_FAIL Parameter error
-------------------------------------------------------------------------------------------+
*/
esp_err_t Test_Communication (void);

#endif /*DRIVER_RYUW122_H*/
```

6. Driver botón táctil

Se implementará el driver que maneja la comunicación con el control de los puertos TOUCH devolviendo solo la información se el botón está presionado (siendo tocado) o no. Se agregan una función por si fue tocado por más de 300ms y otra si fue tocado y liberado.

```c
/**
 *
 * @file      driver_touch.h
 * @brief     driver for a touch button
 * @version   1.0.0
 * @author    Juan I Spinetto
 * @date      20/07/2024
 *
 *
 */


/*---------------
--   include    -
----------------
*/
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_log.h"
#include "driver/touch_pad.h"


/**
 * @brief     init button
 * @return    ESP_OK Success
 *            ESP_ERR_NO_MEM Touch pad init error
 *            ESP_ERR_NOT_SUPPORTED Touch pad is providing current to external XTAL
 * @note      Include init and config
 */
esp_err_t init_touch_button(void);

/**
 * @brief     indicates if the button is touched
 * @return    TRUE   the button is touched (the value of touch_pad_read is bellow the threshold)
 *            FALSE  the button is not touched (the value of touch_pad_read is above the threshold)
 * @note     The threshold is a defined constant, and its value is obtained empirically
 */
bool touch_button_down (void);

/**
 * @brief     indicates if the button is touched for at least 300ms
 * @return    TRUE   the button is touched for 300ms or more (the value of touch_pad_read is bellow the threshold in two reads separated by
300ms)
 *            FALSE  the button is not touched for 300ms or more (the value of touch_pad_read is above the threshold)
 * @note     The threshold is a defined constant, and its value is obtained empirically
 */
bool touch_button_pressed_300 (void);

/**
 * @brief     indicates if the button is touched for more tha 50ms and then released
 * @return    TRUE   the button is touched for 50ms or more and then released
 *            FALSE  the button is not touched for 200ms or more (the value of touch_pad_read is above the threshold)
 * @note     if the button is touched, the function will return control to the program once the button is released
 */
bool touch_button_press_release (void);
```