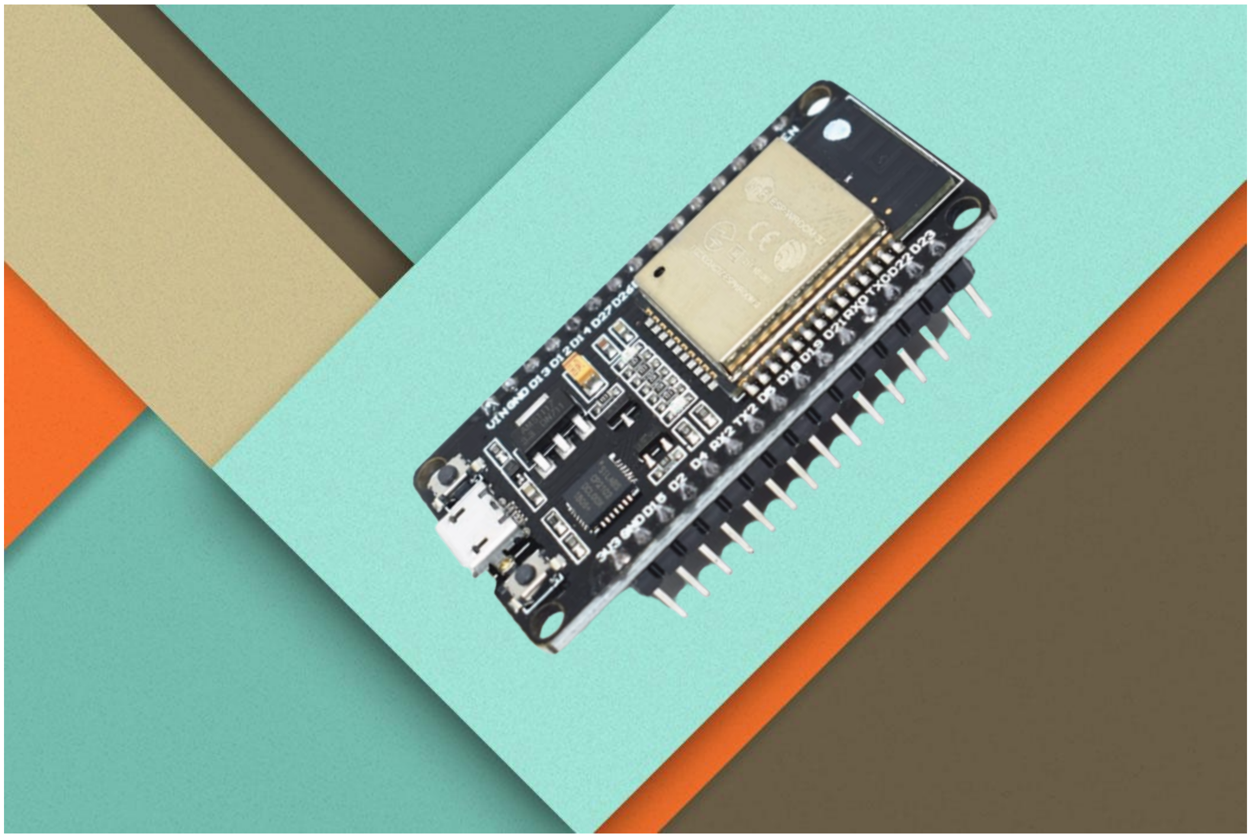


UART DRIVER MANUAL (ESP32)

DECEMBER 2022

Autors:
Axel Gay Díaz
Anahí González Holguin
Carlos Alberto González Vázquez
Carlos Eduardo López Lara

INDEX

1. Generalities

1.1 Developer Recommendations

If you have never used the ESP32 unit, the following link provides detailed information:

ESP32 Wi-Fi & Bluetooth MCU I Espressif Systems

On this page you will find things like:

- Technical information
- Modules information and Development Kits
- Videos with examples

Links will be provided for installing the recommended IDE:

Espressif-IDE Installation Guide (ES):

1) Espressif ESP32 - ESP-IDF - Instalación y Primer Ejemplo - YouTube

Espressif-IDE Installation Guide (EN):

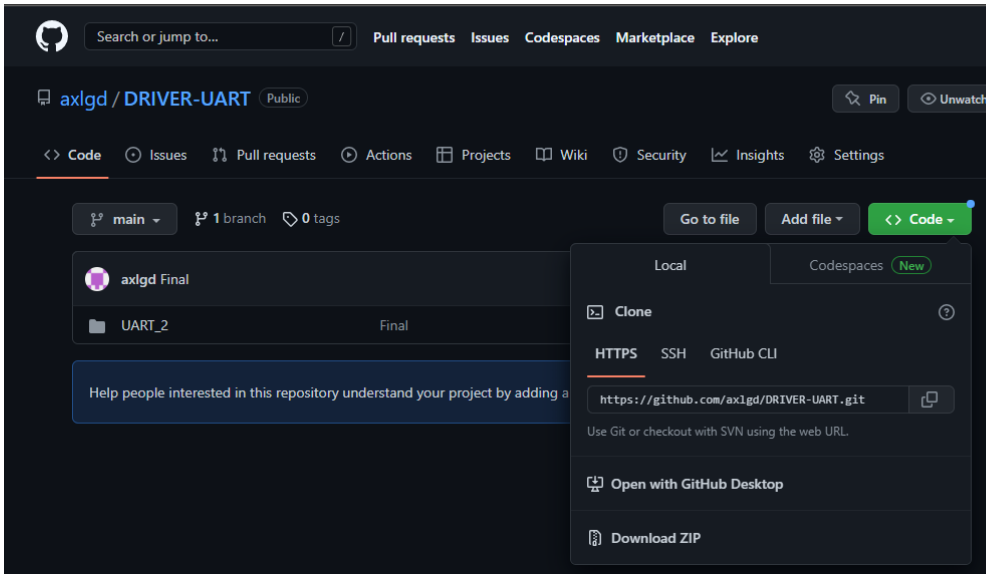
Getting started with ESP-IDF programming using espressif IDE - YouTube

1.2 Download and Driver Implementation

1. Access the following GitHub link:

https://github.com/axlgd/DRIVER-UART.git

2. Click on Code and on Download ZIP inside the repository (UART_2):

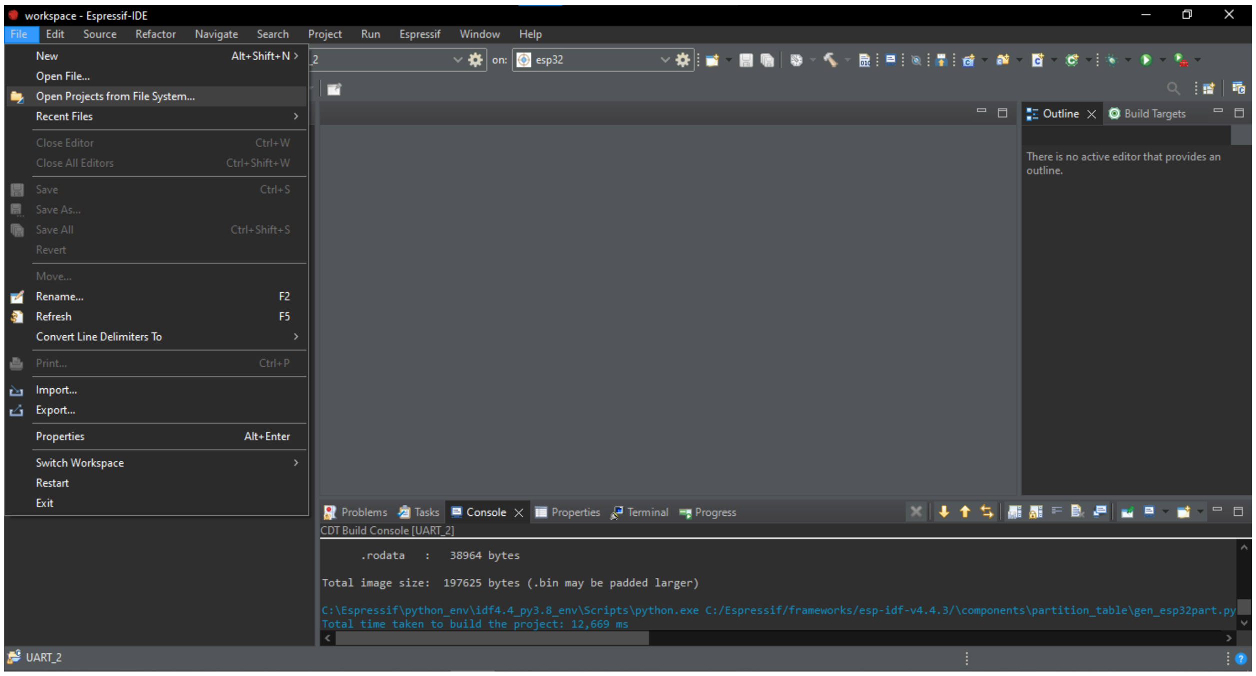


Help people interested in this repository understand your project by adding a

https://github.com/axlgd/DRIVER-UART.git

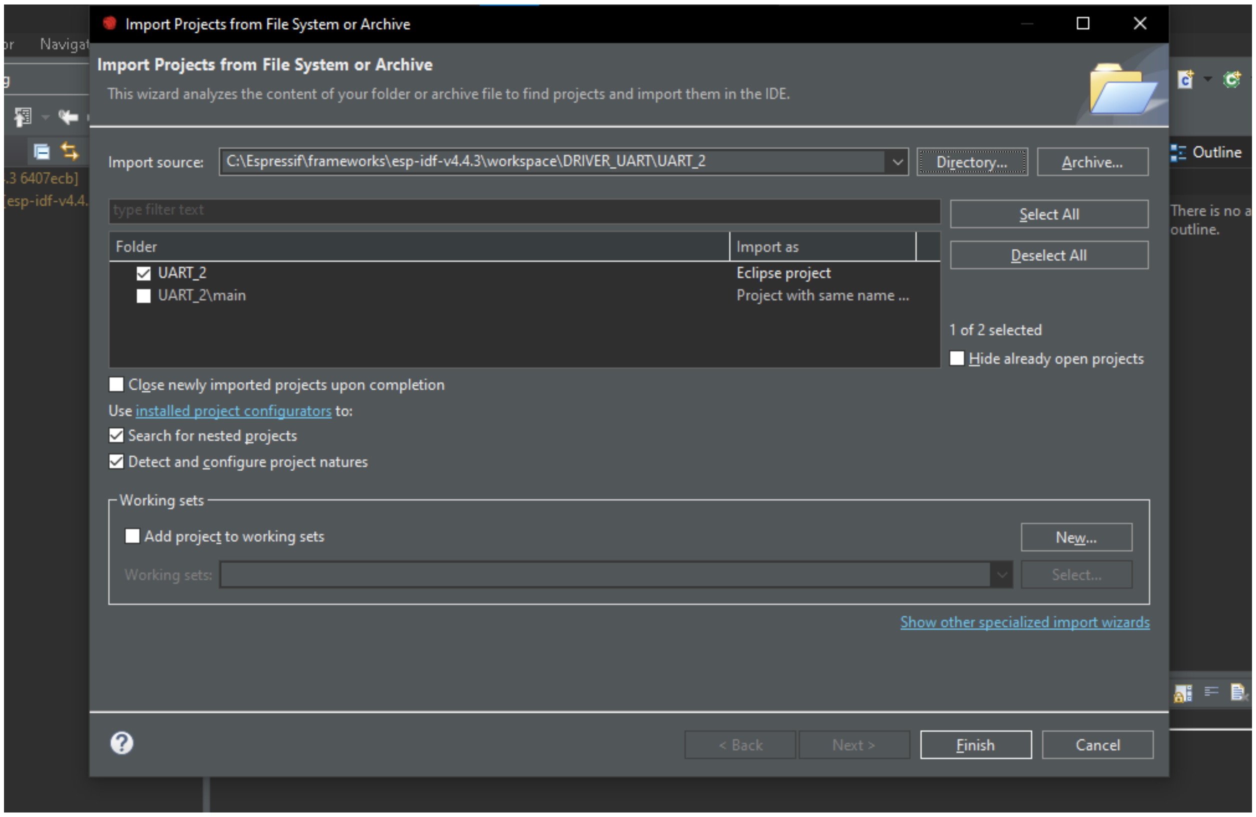
Use Git or checkout with SVN using the web URL.

3. Select the option Open Projects from File System:



```
      .rodata   :   38964 bytes

Total image size:  197625 bytes (.bin may be padded larger)

C:\Espressif\python_env\idf4.4_py3.8_env\Scripts\python.exe C:/Espressif/frameworks/esp-idf-v4.4.3/\components\partition_table\gen_esp32part.py
Total time taken to build the project: 12,669 ms
```

4. Select the path of the project and click on Finish:



Import Projects from File System or Archive

This wizard analyzes the content of your folder or archive file to find projects and import them in the IDE.

Import source: C:\Espressif\frameworks\esp-idf-v4.4.3\workspace\DRIVER_UART\UART_2

type filter text

Folder	Import as
☑ UART_2	Eclipse project
☐ UART_2\main	Project with same name ...

1 of 2 selected

☐ Hide already open projects

☐ Close newly imported projects upon completion

Use installed project configurators to:

☑ Search for nested projects

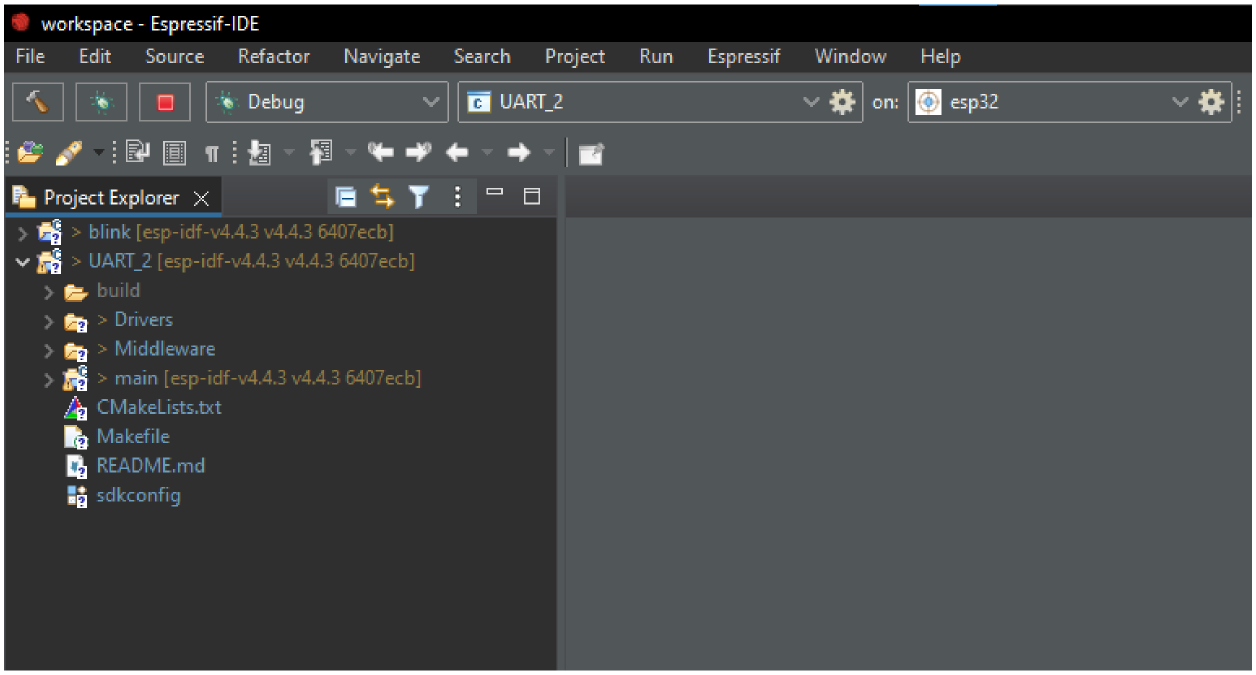
☑ Detect and configure project natures

Working sets

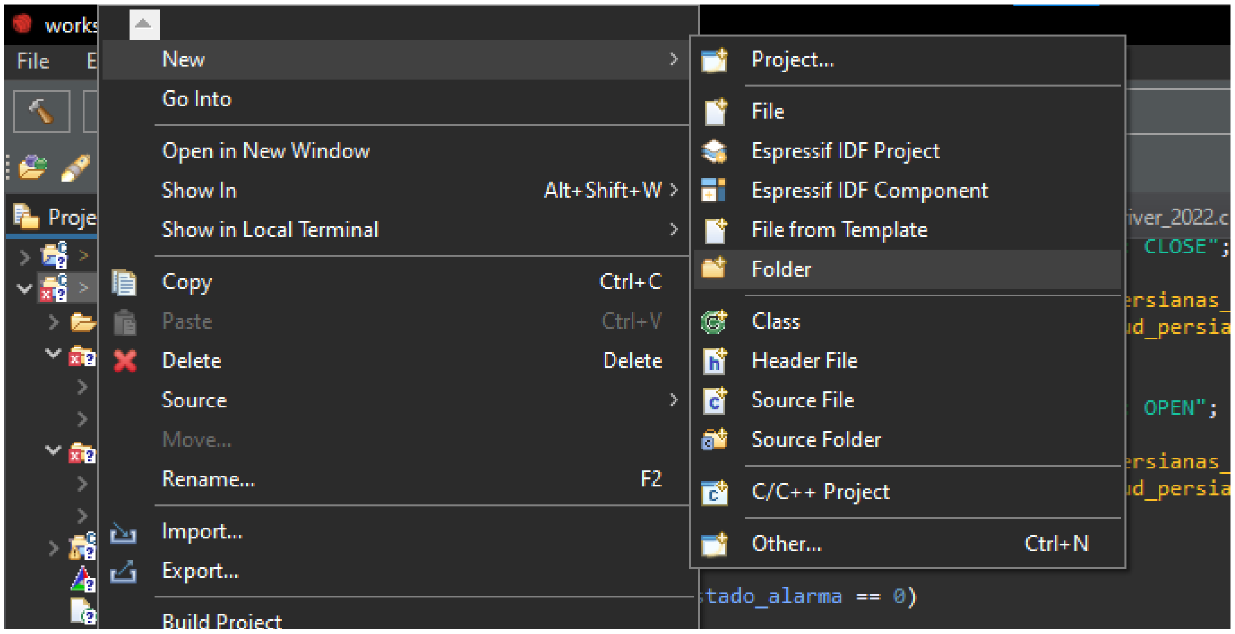
☐ Add project to working sets

Working sets:

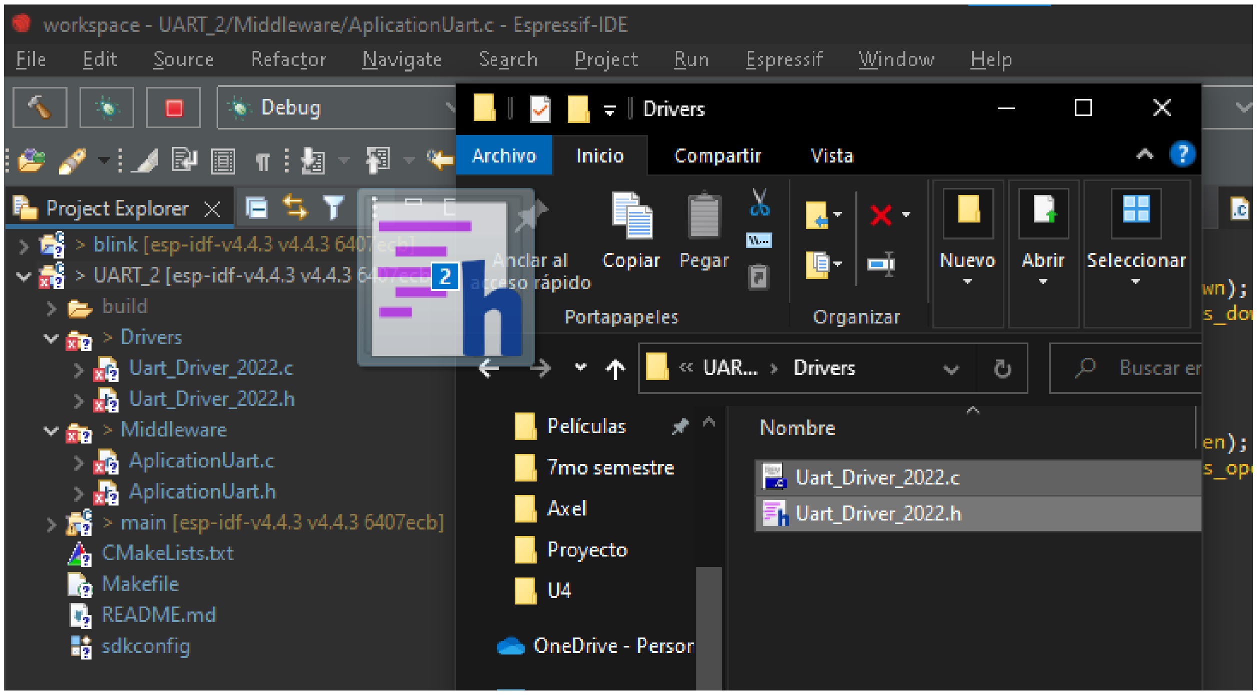
5. Then the project should appear on the left:


workspace - Espressif-IDE
File Edit Source Refactor Navigate Search Project Run Espressif Window Help
Debug
UART_2
on: esp32
Project Explorer
> blink [esp-idf-v4.4.3 v4.4.3 6407ecb]
> UART_2 [esp-idf-v4.4.3 v4.4.3 6407ecb]
build
> Drivers
> Middleware
> main [esp-idf-v4.4.3 v4.4.3 6407ecb]
CMakeLists.txt
Makefile
README.md
sdkconfig

6. In case you need to add the drivers to another project than the example one, you just need to follow the steps below:
   a) Create a new folder inside your project and name it.



b) Drag the Uart_Driver_2022.c and Uart_Driver_2022.h files to the folder you just created.

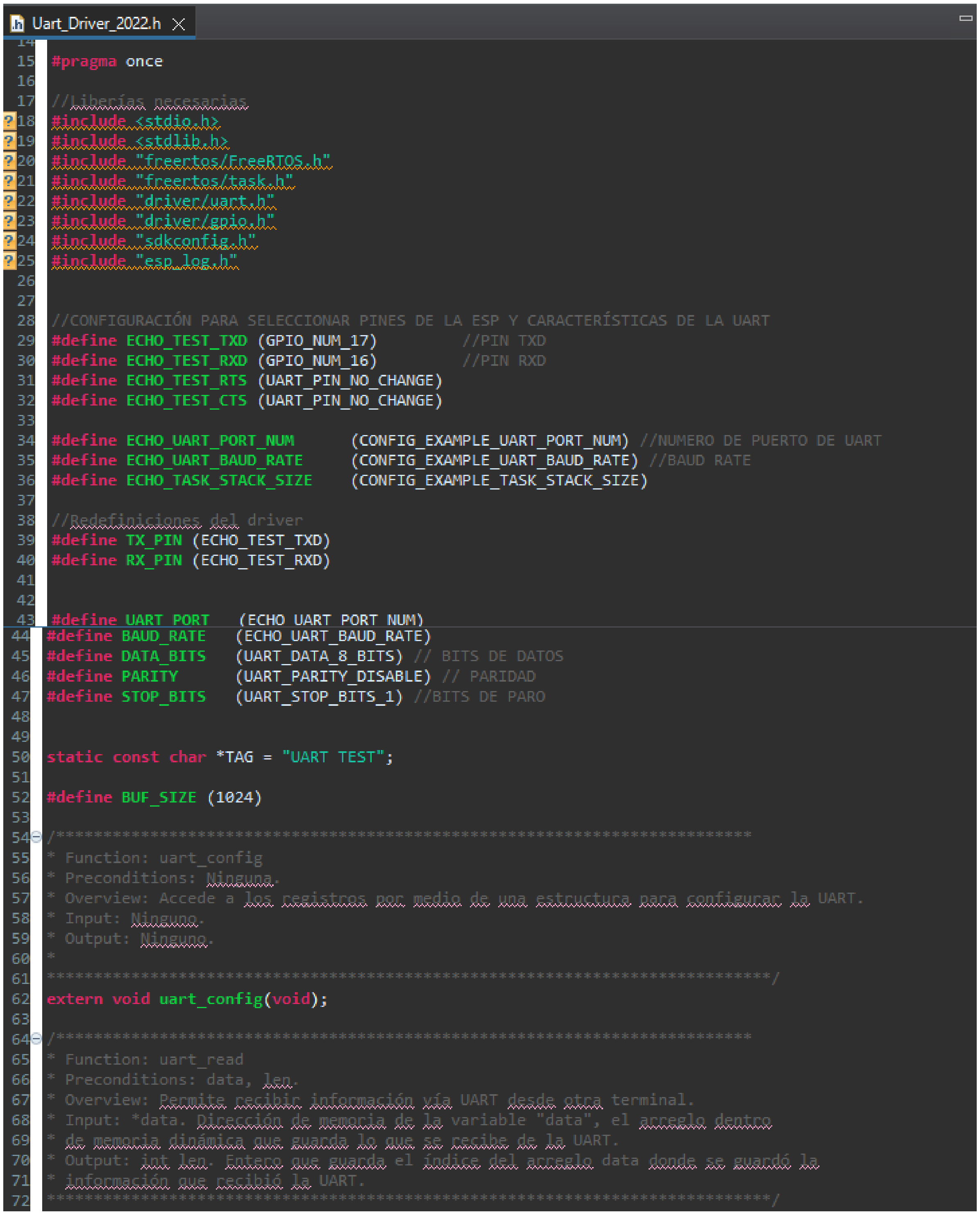


2. General operation

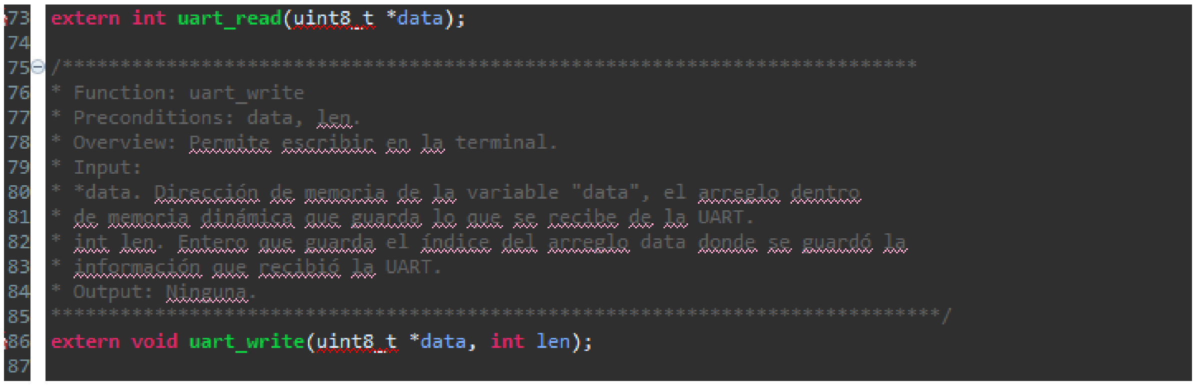
2.1 Header Files(.h)

In these files are all the definitions and redefinitions of functions and constants necessary for the configuration of the UART, pins and an external button.

2.1.1 Uart_Driver_2022.h



```c
#pragma once

//Liberías necesarias
#include <stdio.h>
#include <stdlib.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/uart.h"
#include "driver/gpio.h"
#include "sdkconfig.h"
#include "esp_log.h"


//CONFIGURACIÓN PARA SELECCIONAR PINES DE LA ESP Y CARACTERÍSTICAS DE LA UART
#define ECHO_TEST_TXD (GPIO_NUM_17)         //PIN TXD
#define ECHO_TEST_RXD (GPIO_NUM_16)         //PIN RXD
#define ECHO_TEST_RTS (UART_PIN_NO_CHANGE)
#define ECHO_TEST_CTS (UART_PIN_NO_CHANGE)

#define ECHO_UART_PORT_NUM      (CONFIG_EXAMPLE_UART_PORT_NUM) //NUMERO DE PUERTO DE UART
#define ECHO_UART_BAUD_RATE     (CONFIG_EXAMPLE_UART_BAUD_RATE) //BAUD RATE
#define ECHO_TASK_STACK_SIZE    (CONFIG_EXAMPLE_TASK_STACK_SIZE)

//Redefiniciones del driver
#define TX_PIN (ECHO_TEST_TXD)
#define RX_PIN (ECHO_TEST_RXD)


#define UART_PORT   (ECHO_UART_PORT_NUM)
#define BAUD_RATE   (ECHO_UART_BAUD_RATE)
#define DATA_BITS   (UART_DATA_8_BITS) // BITS DE DATOS
#define PARITY      (UART_PARITY_DISABLE) // PARIDAD
#define STOP_BITS   (UART_STOP_BITS_1) //BITS DE PARO


static const char *TAG = "UART TEST";

#define BUF_SIZE (1024)

/***************************************************************************
* Function: uart_config
* Preconditions: Ninguna.
* Overview: Accede a los registros por medio de una estructura para configurar la UART.
* Input: Ninguno.
* Output: Ninguno.
*
****************************************************************************/
extern void uart_config(void);

/***************************************************************************
* Function: uart_read
* Preconditions: data, len.
* Overview: Permite recibir información vía UART desde otra terminal.
* Input: *data. Dirección de memoria de la variable "data", el arreglo dentro
* de memoria dinámica que guarda lo que se recibe de la UART.
* Output: int len. Entero que guarda el índice del arreglo data donde se guardó la
* información que recibió la UART.
****************************************************************************/
```



```
extern int uart_read(uint8_t *data);

/*****************************************************************************
 * Function: uart_write
 * Preconditions: data, len.
 * Overview: Permite escribir en la terminal.
 * Input:
 * *data. Dirección de memoria de la variable "data", el arreglo dentro
 * de memoria dinámica que guarda lo que se recibe de la UART.
 * int len. Entero que guarda el índice del arreglo data donde se guardó la
 * información que recibió la UART.
 * Output: Ninguna.
 *****************************************************************************/
extern void uart_write(uint8_t *data, int len);
```

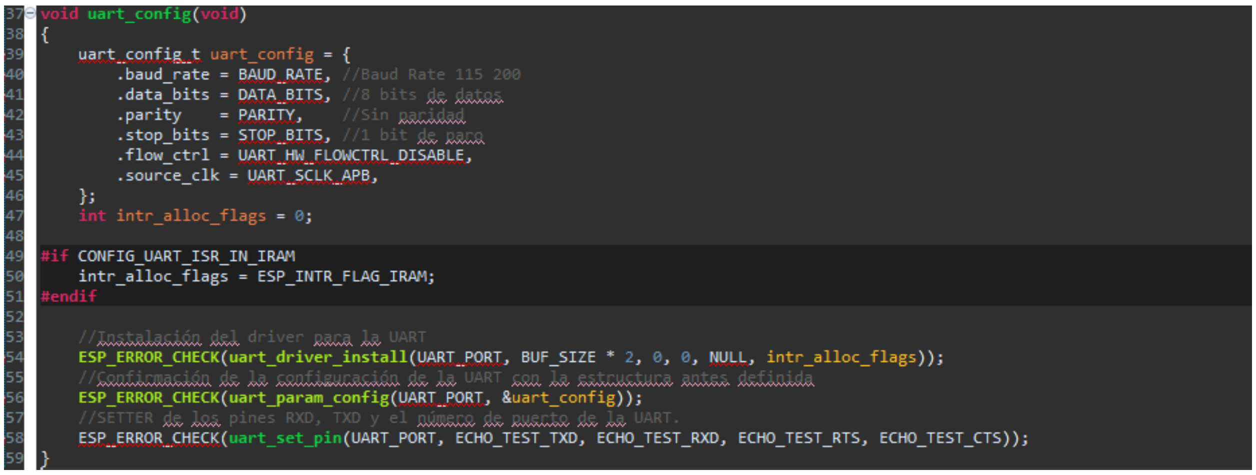
2.2 Functions (.c)

In these files you will find the implementation of all the functions used in the Driver.

2.2.1 Uart_Driver_2022.c

uart_config()

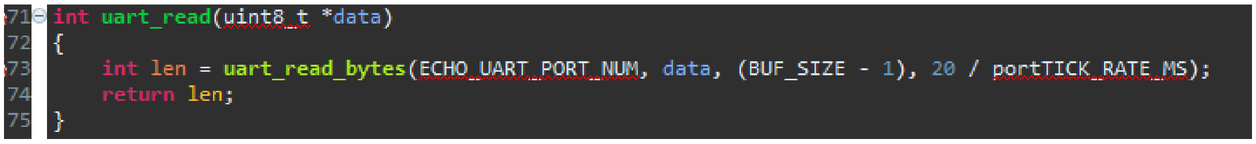
This function is used to configure the UART accessing memory through a structure.



```c
void uart_config(void)
{
    uart_config_t uart_config = {
        .baud_rate = BAUD_RATE, //Baud Rate 115 200
        .data_bits = DATA_BITS, //8 bits de datos
        .parity    = PARITY,    //Sin paridad
        .stop_bits = STOP_BITS, //1 bit de paro
        .flow_ctrl = UART_HW_FLOWCTRL_DISABLE,
        .source_clk = UART_SCLK_APB,
    };
    int intr_alloc_flags = 0;

#if CONFIG_UART_ISR_IN_IRAM
    intr_alloc_flags = ESP_INTR_FLAG_IRAM;
#endif

    //Instalación del driver para la UART
    ESP_ERROR_CHECK(uart_driver_install(UART_PORT, BUF_SIZE * 2, 0, 0, NULL, intr_alloc_flags));
    //Confirmación de la configuración de la UART con la estructura antes definida
    ESP_ERROR_CHECK(uart_param_config(UART_PORT, &uart_config));
    //SETTER de los pines RXD, TXD y el número de puerto de la UART.
    ESP_ERROR_CHECK(uart_set_pin(UART_PORT, ECHO_TEST_TXD, ECHO_TEST_RXD, ECHO_TEST_RTS, ECHO_TEST_CTS));
}
```

uart_read()

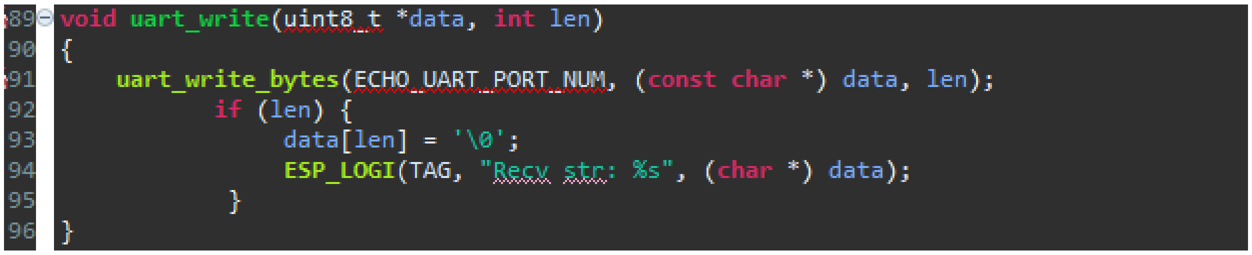
This function is the one that allows receiving information from another terminal through UART.



```c
int uart_read(uint8_t *data)
{
    int len = uart_read_bytes(ECHO_UART_PORT_NUM, data, (BUF_SIZE - 1), 20 / portTICK_RATE_MS);
    return len;
}
```

uart_write()

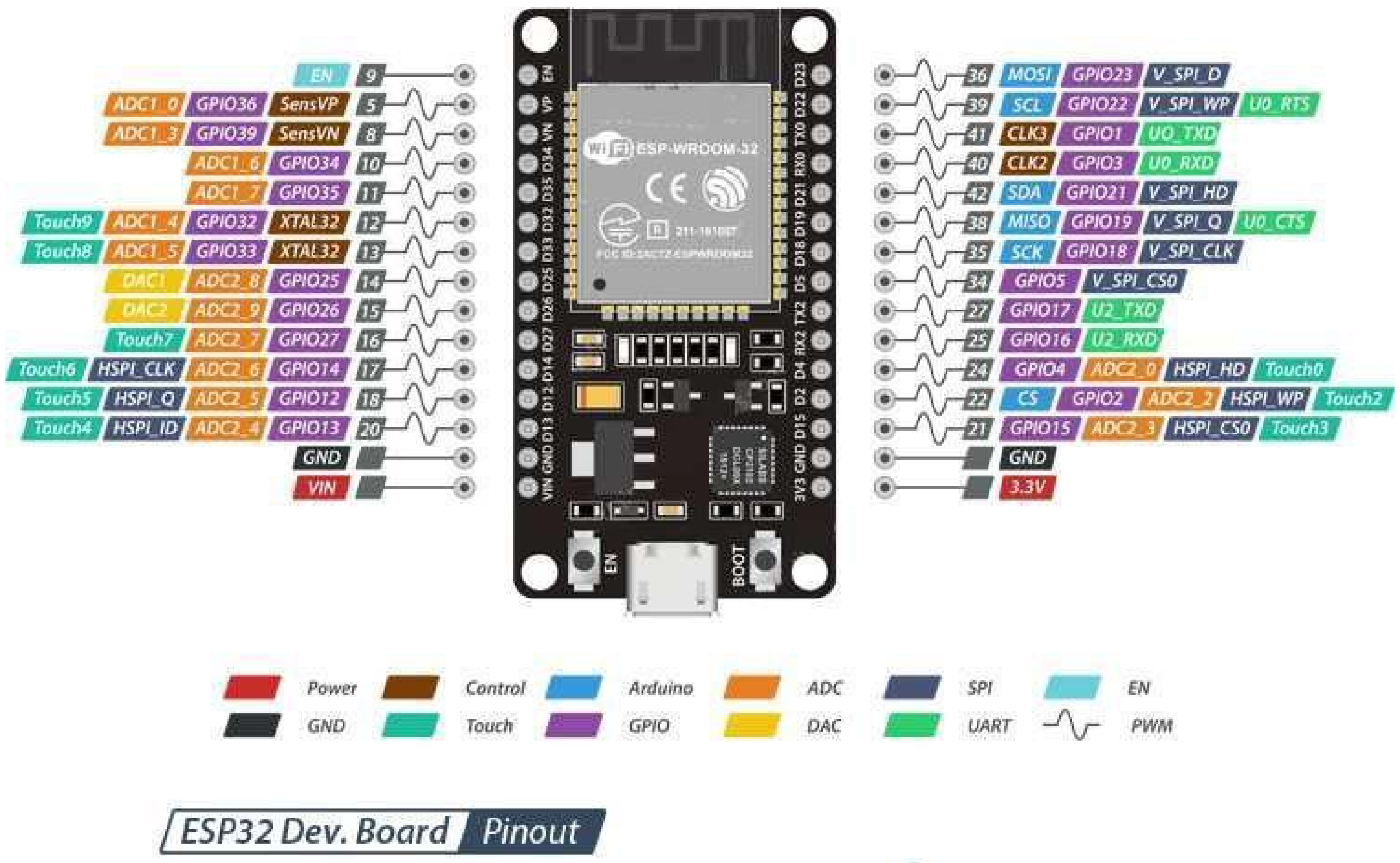
This function allows us to write to the terminal.



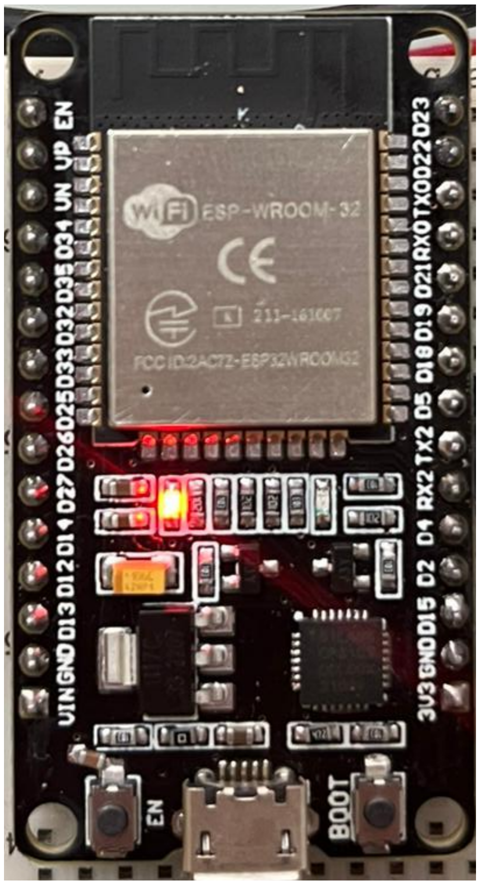
```
void uart_write(uint8_t *data, int len)
{
    uart_write_bytes(ECHO_UART_PORT_NUM, (const char *) data, len);
           if (len) {
                data[len] = '\0';
                ESP_LOGI(TAG, "Recv str: %s", (char *) data);
            }
}
```

3. DevKitC V1 Setup

3.1 Pinout


EN 9
ADC1_0 GPIO36 SensVP 5
ADC1_3 GPIO39 SensVN 8
ADC1_6 GPIO34 10
ADC1_7 GPIO35 11
Touch9 ADC1_4 GPIO32 XTAL32 12
Touch8 ADC1_5 GPIO33 XTAL32 13
DAC1 ADC2_8 GPIO25 14
DAC2 ADC2_9 GPIO26 15
Touch7 ADC2_7 GPIO27 16
Touch6 HSPI_CLK ADC2_6 GPIO14 17
Touch5 HSPI_Q ADC2_5 GPIO12 18
Touch4 HSPI_ID ADC2_4 GPIO13 20
GND
VIN
EN VP VN D34 D35 D32 D33 D25 D26 D27 D14 D12 D13 GND VIN
ESP-WROOM-32
D23 D22 TX0 RX0 D21 D19 D18 D5 TX2 RX2 D4 D2 D15 GND 3V3
EN
BOOT
36 MOSI GPIO23 V_SPI_D
39 SCL GPIO22 V_SPI_WP U0_RTS
41 CLK3 GPIO1 U0_TXD
40 CLK2 GPIO3 U0_RXD
42 SDA GPIO21 V_SPI_HD
38 MISO GPIO19 V_SPI_Q U0_CTS
35 SCK GPIO18 V_SPI_CLK
34 GPIO5 V_SPI_CS0
27 GPIO17 U2_TXD
25 GPIO16 U2_RXD
24 GPIO4 ADC2_0 HSPI_HD Touch0
22 CS GPIO2 ADC2_2 HSPI_WP Touch2
21 GPIO15 ADC2_3 HSPI_CS0 Touch3
GND
3.3V
Power
GND
Control
Touch
Arduino
GPIO
ADC
DAC
SPI
UART
EN
PWM
ESP32 Dev. Board Pinout

3.2 Physical features


ESP-WROOM-32
VIN GND D13 D12 D14 D27 D26 D25 D33 D32 D35 D34 VN VP EN
3V3 GND D15 D2 D4 RX2 TX2 D5 D18 D19 D21 RX0 TX0 D22 D23
EN
BOOT

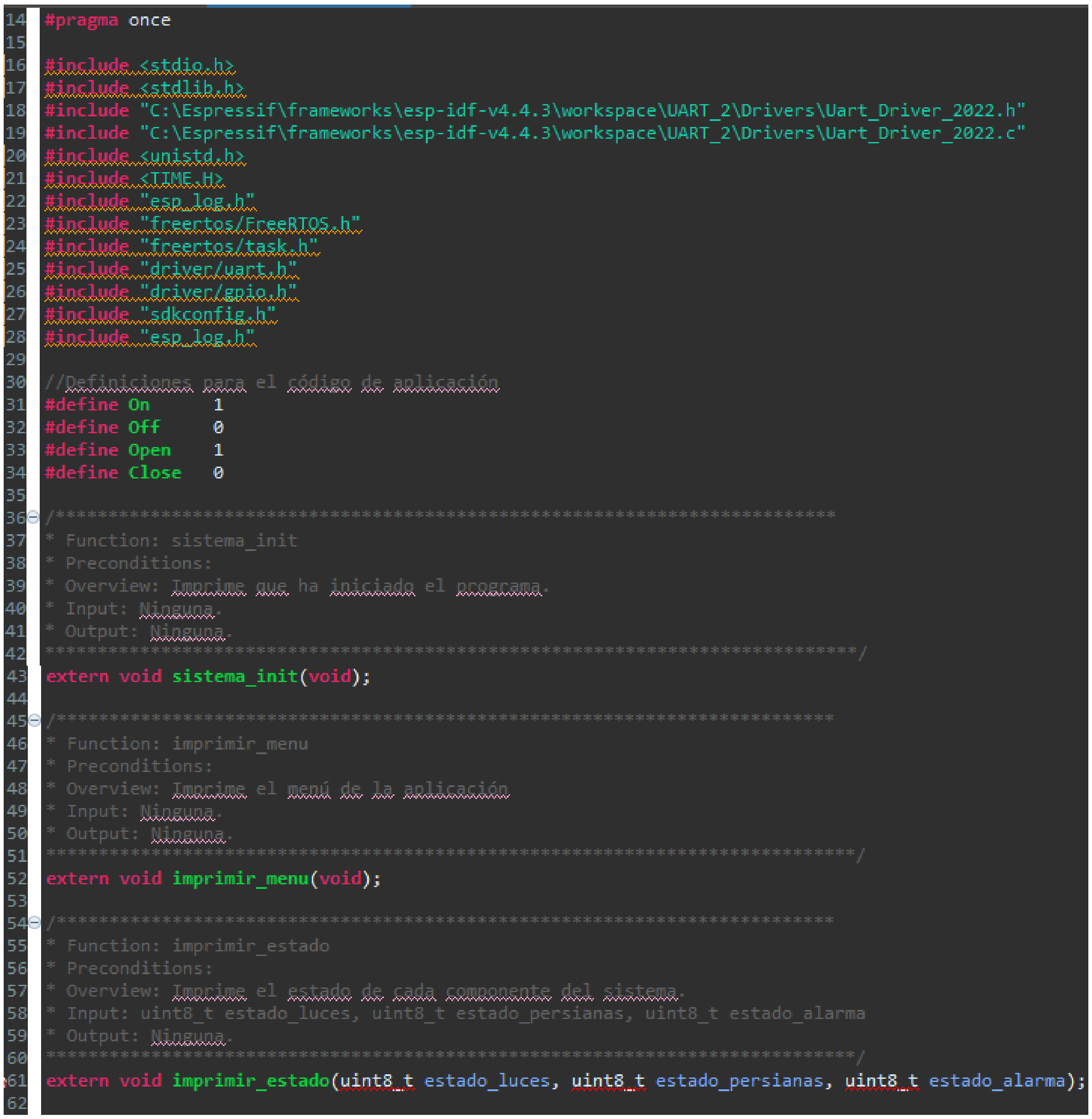
Some of its main physical characteristics are:

- SoC Tensilica Xtensa 32-bit LX6
- 2 buttons, one to activate Bootloader mode, and another to Reset
- 30 pins. We find pins for GPIO, UART, GND, 5V or 3.3V supply, ADC, etc.
- WiFi & Bluetooth modules
- Micro USB port
- LED indicator

4. Examples

4.1 AplicationUart.h

In this header file we find the definitions of functions and constants to be used in the application example.



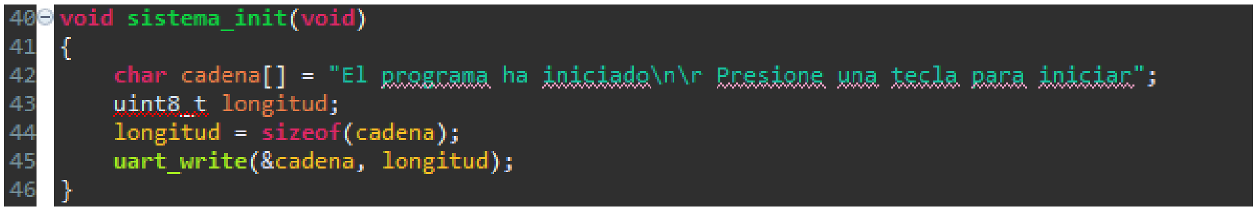
```c
#pragma once

#include <stdio.h>
#include <stdlib.h>
#include "C:\Espressif\frameworks\esp-idf-v4.4.3\workspace\UART_2\Drivers\Uart_Driver_2022.h"
#include "C:\Espressif\frameworks\esp-idf-v4.4.3\workspace\UART_2\Drivers\Uart_Driver_2022.c"
#include <unistd.h>
#include <TIME.H>
#include "esp_log.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/uart.h"
#include "driver/gpio.h"
#include "sdkconfig.h"
#include "esp_log.h"

//Definiciones para el código de aplicación
#define On      1
#define Off     0
#define Open    1
#define Close   0

/************************************************************************
 * Function: sistema_init
 * Preconditions:
 * Overview: Imprime que ha iniciado el programa.
 * Input: Ninguna.
 * Output: Ninguna.
 ************************************************************************/
extern void sistema_init(void);

/************************************************************************
 * Function: imprimir_menu
 * Preconditions:
 * Overview: Imprime el menú de la aplicación
 * Input: Ninguna.
 * Output: Ninguna.
 ************************************************************************/
extern void imprimir_menu(void);

/************************************************************************
 * Function: imprimir_estado
 * Preconditions:
 * Overview: Imprime el estado de cada componente del sistema.
 * Input: uint8_t estado_luces, uint8_t estado_persianas, uint8_t estado_alarma
 * Output: Ninguna.
 ************************************************************************/
extern void imprimir_estado(uint8_t estado_luces, uint8_t estado_persianas, uint8_t estado_alarma);
```

4.2 AplicationUart.c

This file contains the implementation of each of the functions to be used in the application example.

sistema_init()

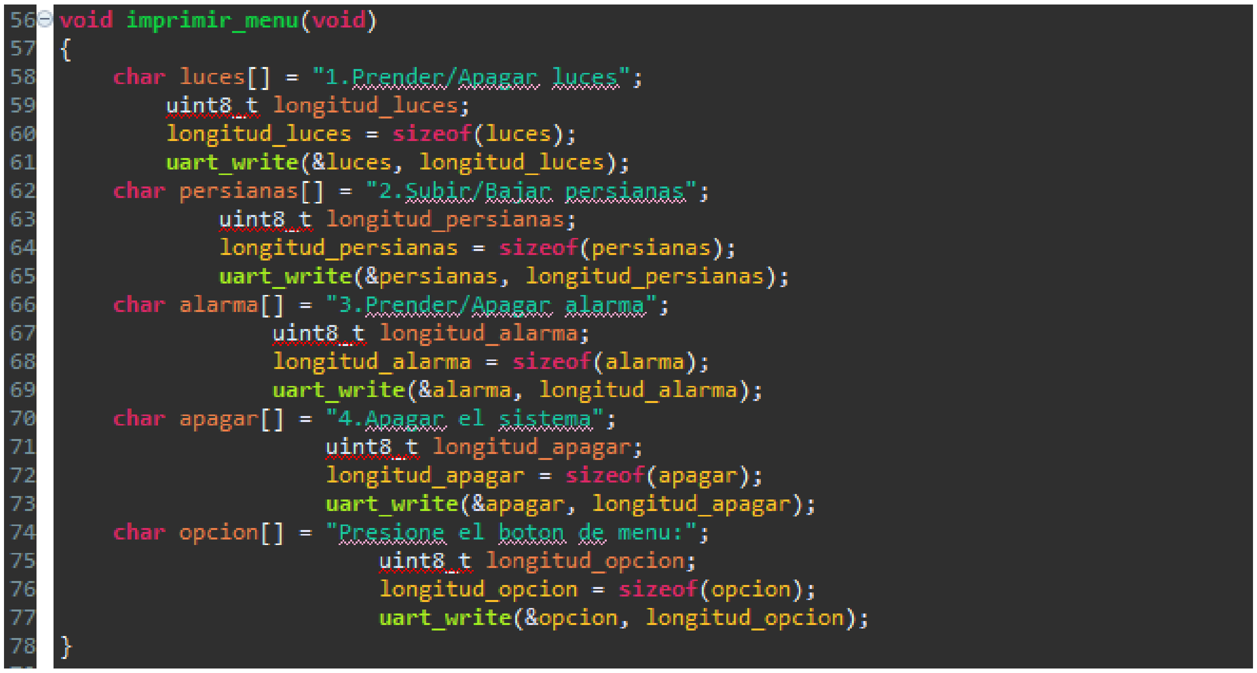
This function tells us that the program has started, by printing in the terminal and telling us to press a key to start.



```c
void sistema_init(void)
{
    char cadena[] = "El programa ha iniciado\n\r Presione una tecla para iniciar";
    uint8_t longitud;
    longitud = sizeof(cadena);
    uart_write(&cadena, longitud);
}
```

imprimir_menu()

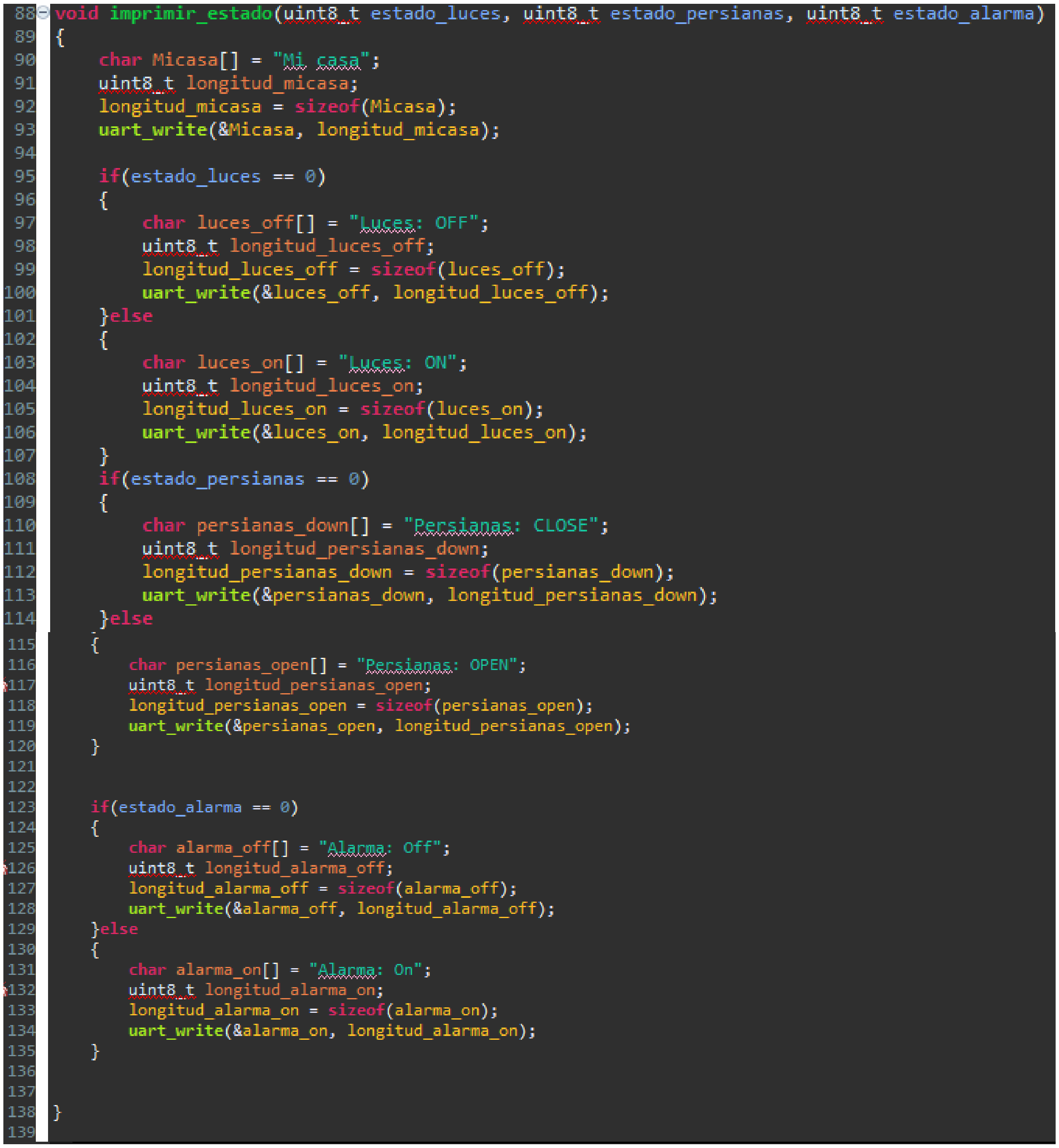
This function prints the application menu, displaying all the available options, which are to control the lights, blinds, alarm or turn off the system.



```
void imprimir_menu(void)
{
    char luces[] = "1.Prender/Apagar luces";
        uint8_t longitud_luces;
        longitud_luces = sizeof(luces);
        uart_write(&luces, longitud_luces);
    char persianas[] = "2.Subir/Bajar persianas";
            uint8_t longitud_persianas;
            longitud_persianas = sizeof(persianas);
            uart_write(&persianas, longitud_persianas);
    char alarma[] = "3.Prender/Apagar alarma";
                uint8_t longitud_alarma;
                longitud_alarma = sizeof(alarma);
                uart_write(&alarma, longitud_alarma);
    char apagar[] = "4.Apagar el sistema";
                    uint8_t longitud_apagar;
                    longitud_apagar = sizeof(apagar);
                    uart_write(&apagar, longitud_apagar);
    char opcion[] = "Presione el boton de menu:";
                        uint8_t longitud_opcion;
                        longitud_opcion = sizeof(opcion);
                        uart_write(&opcion, longitud_opcion);
}
```

imprimir_estado()

This function prints the status of each system component.



```c
void imprimir_estado(uint8_t estado_luces, uint8_t estado_persianas, uint8_t estado_alarma)
{
    char Micasa[] = "Mi casa";
    uint8_t longitud_micasa;
    longitud_micasa = sizeof(Micasa);
    uart_write(&Micasa, longitud_micasa);

    if(estado_luces == 0)
    {
        char luces_off[] = "Luces: OFF";
        uint8_t longitud_luces_off;
        longitud_luces_off = sizeof(luces_off);
        uart_write(&luces_off, longitud_luces_off);
    }else
    {
        char luces_on[] = "Luces: ON";
        uint8_t longitud_luces_on;
        longitud_luces_on = sizeof(luces_on);
        uart_write(&luces_on, longitud_luces_on);
    }
    if(estado_persianas == 0)
    {
        char persianas_down[] = "Persianas: CLOSE";
        uint8_t longitud_persianas_down;
        longitud_persianas_down = sizeof(persianas_down);
        uart_write(&persianas_down, longitud_persianas_down);
    }else
    {
        char persianas_open[] = "Persianas: OPEN";
        uint8_t longitud_persianas_open;
        longitud_persianas_open = sizeof(persianas_open);
        uart_write(&persianas_open, longitud_persianas_open);
    }


    if(estado_alarma == 0)
    {
        char alarma_off[] = "Alarma: Off";
        uint8_t longitud_alarma_off;
        longitud_alarma_off = sizeof(alarma_off);
        uart_write(&alarma_off, longitud_alarma_off);
    }else
    {
        char alarma_on[] = "Alarma: On";
        uint8_t longitud_alarma_on;
        longitud_alarma_on = sizeof(alarma_on);
        uart_write(&alarma_on, longitud_alarma_on);
    }


}
```

4.3 Links

Links:

-Driver UART ZIP file:

https://github.com/axlgd/DRIVER-UART.git

-Video tutorials (Playlist):

https://www.youtube.com/playlist?list=PLUqGq5mzysq71o7x0OJAr6qn0i-qRyWpq