

**Universidad Autónoma de Baja California**

**Ingeniería en Computación**



Facultad de Ciencias Químicas e Ingeniería

Microcontroladores

**Práctica 2. Configuración del ambiente de desarrollo**

Alumno: Zavala Román Irvin Eduardo

Docente: Castro Gonzales Ricardo

Grupo: 561

23/02/2022

Periodo 2022-1

## **Objetivo**

El alumno instalará el ambiente de desarrollo y se familiarizará con las herramientas.

## **Material**

- Computadora Personal

## **Teoría**

*IDEs para sistemas embebidos (pros y contras).*

Microchip studio

IDE para desarrollar y depurar microcontroladores AVR y SAM. Incluye varias funcionalidades de Atmel Studio para la compañía Microchip y permite aplicaciones en C/C++ o ensamblador. La desventaja es la limitada cantidad de microcontroladores soportados.

IAR IDE

Entorno de desarrollo para ARM el cual soporta más de 7000 dispositivos de esta arquitectura, la mayoría de 32 bits y unos cuantos de 64 bits. Incluye el linker y el ensamblador para ser el entorno ideal de software embebido. La desventaja aparte del precio es el reporte de crasheos por parte de este IDE.

uVision IDE

Este entorno de desarrollo incluye gestor de proyectos, entorno de tiempo real, depurador de programa y más en un único entorno. Incluye el uVision Debugger para realizar desde la depuración tradicional hasta el control total de los periféricos. La desventaja es el precio que en su versión definitiva por 1 año vale 4500 dólares.

Eclipse Embedded CDT

El IDE en general no está enfocado a sistemas embebidos, pero por medio de plugins en un solo development kit permite crear aplicaciones para ARM y

RISC-V. La principal ventaja es que es gratis y open source, pero puede ser inestable y al no estar enfocado en sistemas embebidos la interfaz puede no ser la mejor.

### CLION JetBrains

Este IDE es originalmente para C/C++, pero se puede hacer para desarrollar software de sistemas embebidos compatibles con GCC o IAR. No es gratis pero JetBrains generalmente da una versión para estudiante, el principal contra es que la interfaz no se adapta para sistemas embebidos.

### NetBeans

Generalmente usado para Java, se pueden desarrollar sistemas embebidos con Oracle Java SE Embedded Support. Aunque es gratis y Java es una opción para sistemas embebidos, generalmente se usa mas C/C++ con ensamblador. Aparte NetBeans suele ser lento en varias partes del desarrollo.

### MPLAB IDE

Creado por la misma Microchip del primer IDE, permite realizar casi el proceso completo de desarrollo para los microcontroladores de esta marca. La principal contra son quejas de los usuarios de los errores de este sistema y el precio de 40 dólares al mes.

### *Técnicas de depuración para sistemas embebidos.*

Existen depuradores a nivel de hardware y software, a nivel de hardware la serie J-Link como la figura 1 permite depurar la mayoría de sistemas embebidos. Existen versiones básicas y complejas variando en velocidades como de RAM o interfaz. Estos depuradores se suelen conectar a placas de desarrollo con “almohadillas de clavos”. A partir de aquí se usan herramientas de software para depurar.



Figura 1. Depurador J-Link Ultra+.

Obtenido de:

<https://www.digikey.com.mx/es/articles/the-professional-guide-to-debugging-tools-and-techniques-for-iot-devices>

Las opciones de depuración con software no son proporcionadas por la marca de J-Link. El primero es J-Scope que es similar a un osciloscopio para mostrar variables en el tiempo. Otra opción es Ozono para analizar rendimiento. Por ultimo SystemView permite ver el comportamiento de la ejecución del sistema RTOS.

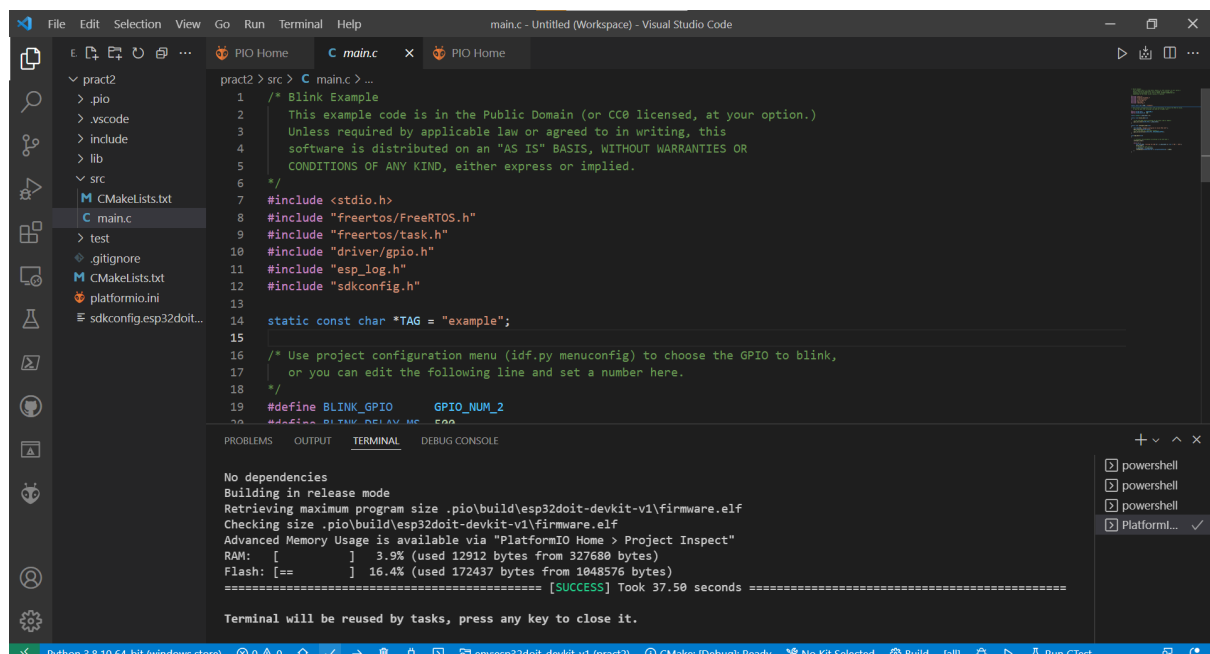
### Instrucciones

- Instalación de VS Code. (<https://code.visualstudio.com/download>)
- Instalación de la extensión de PlatformIO dentro de VS Code.

- Crear un nuevo proyecto dentro de PlatformIO seleccionando alguna tarjeta basada en un uC ESP32 (por ejemplo DOIT ESP32 DEVKIT V1)
- Utilizar el archivo semilla del repositorio y compilar para verificar que las herramientas se han instalado correctamente.
- Cargar el binario generado a la tarjeta de desarrollo, en caso de no contar con el HW, apoyarse con el simulador en línea Wokwi (<https://wokwi.com/projects/new/esp32>)
- Realizar los cambios necesarios para que ahora parpadee en código morse su nombre.
- Anexar captura del analizador lógico como verificación.

## Desarrollo

La ejecución del archivo de ejemplo en VSCode se realizó en la hora de laboratorio, el proceso de compilación se muestra en la figura 1.



```

pract2 > src > C main.c > ...
1  /* Blink Example
2  This example code is in the Public Domain (or CC0 licensed, at your option.)
3  Unless required by applicable law or agreed to in writing, this
4  software is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
5  CONDITIONS OF ANY KIND, either express or implied.
6  */
7  #include <stdio.h>
8  #include "freertos/FreeRTOS.h"
9  #include "freertos/task.h"
10 #include "driver/gpio.h"
11 #include "esp_log.h"
12 #include "sdkconfig.h"
13
14 static const char *TAG = "example";
15
16 /* Use project configuration menu (idf.py menuconfig) to choose the GPIO to blink,
17    or you can edit the following line and set a number here.
18 */
19 #define BLINK_GPIO      GPIO_NUM_2
20 #define BLINK_DELAY_MS  1000

```

```

No dependencies
Building in release mode
Retrieving maximum program size .pio\build\esp32doit-devkit-v1\firmware.elf
Checking size .pio\build\esp32doit-devkit-v1\firmware.elf
Advanced Memory Usage is available via "PlatformIO Home > Project Inspect"
RAM: [    ] 3.9% (used 12912 bytes from 327680 bytes)
Flash: [==] 16.4% (used 172437 bytes from 1048576 bytes)
===== [SUCCESS] Took 37.50 seconds =====
Terminal will be reused by tasks, press any key to close it.

```

**Figura 2.** Compilación del programa Blink Example

En la práctica se pide que el LED del ESP32 parpadee el código morse del nombre del alumno, en este caso se consideró mejor implementar todas las letras en morse y con dependiendo de la cadena mostrar las letras correspondientes. El código base es el mismo aunque se añadieron funciones para tener la base del código morse, el “punto”, la “raya” equivalente a 3 puntos, la “pausa” para espacios entre letras y “espacio” para espacio entre palabras.

```
void raya(){
    s_led_state = true;
    ESP_LOGI(TAG, "Turning the LED %s!", s_led_state ==
true ? "ON" : "OFF");
    blink_led();
    /* Toggle the LED state */
    vTaskDelay((delay_raya* configTICK_RATE_HZ) / 1000);
}
void punto(){
    s_led_state = true;
    ESP_LOGI(TAG, "Turning the LED %s!", s_led_state ==
true ? "ON" : "OFF");
    blink_led();
    /* Toggle the LED state */
    s_led_state = !s_led_state;
    vTaskDelay((delay_punto* configTICK_RATE_HZ) / 1000);
}
void espacio(){
    s_led_state = false;
    ESP_LOGI(TAG, "Turning the LED %s!", s_led_state ==
true ? "ON" : "OFF");
    blink_led();
    /* Toggle the LED state */
    s_led_state = !s_led_state;
    vTaskDelay((300* configTICK_RATE_HZ) / 1000);
}
void pausa(){
    s_led_state = false;
    ESP_LOGI(TAG, "Turning the LED %s!", s_led_state ==
true ? "ON" : "OFF");
    blink_led();
    /* Toggle the LED state */
    s_led_state = !s_led_state;
    vTaskDelay((50* configTICK_RATE_HZ) / 1000);
}
```

En el main solamente se recorre la cadena a convertir con un for y por cada carácter se llaman las funciones correspondientes.

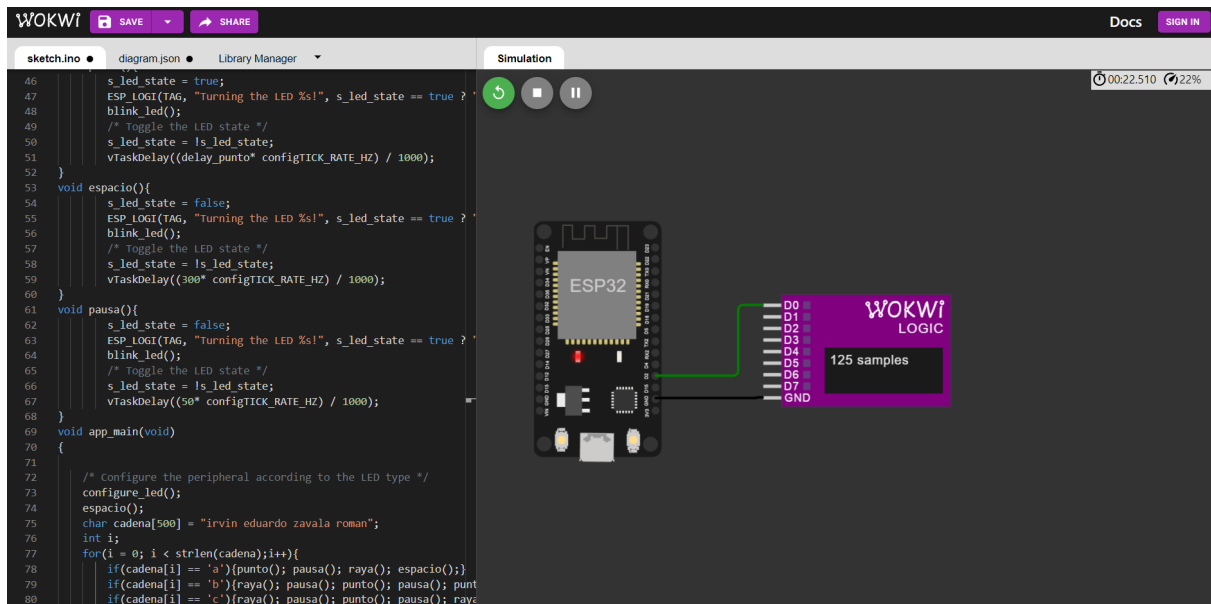
```
configure_led();
espacio();
char cadena[500] = "irvin eduardo zavalá roman";
int i;
```

```

    for(i = 0; i < strlen(cadena);i++){
        if(cadena[i] == 'a'){punto(); pausa(); raya(); espacio();}
        if(cadena[i] == 'b'){raya(); pausa(); punto(); pausa();
punto(); pausa(); punto(); espacio();}
        if(cadena[i] == 'c'){raya(); pausa(); punto(); pausa();
raya(); pausa(); punto(); espacio();}
        if(cadena[i] == 'd'){raya(); pausa(); punto(); pausa();
punto(); espacio();}
        if(cadena[i] == 'e'){punto(); espacio();}
        if(cadena[i] == 'f'){punto(); pausa(); punto(); pausa();
raya(); pausa(); punto(); espacio();}
        if(cadena[i] == 'g'){raya(); pausa(); raya(); pausa();
punto(); espacio();}
        if(cadena[i] == 'h'){punto(); pausa(); punto(); pausa();
punto(); pausa(); punto(); espacio();}
        if(cadena[i] == 'i'){punto(); pausa(); punto();
espacio();}
        if(cadena[i] == 'j'){punto(); pausa(); raya(); pausa();
raya(); pausa(); raya(); espacio();}
        if(cadena[i] == 'k'){raya(); pausa(); punto(); pausa();
raya(); espacio();}
        if(cadena[i] == 'l'){punto(); pausa(); raya(); pausa();
punto(); pausa(); punto(); espacio();}
        if(cadena[i] == 'm'){raya(); pausa(); raya(); espacio();}
        if(cadena[i] == 'n'){raya(); pausa(); punto(); espacio();}
        if(cadena[i] == 'o'){raya(); pausa(); raya(); pausa();
raya(); espacio();}
        if(cadena[i] == 'p'){punto(); pausa(); raya(); pausa();
raya(); pausa(); punto(); espacio();}
        if(cadena[i] == 'q'){raya(); pausa(); raya(); pausa();
punto(); pausa(); raya(); espacio();}
        if(cadena[i] == 'r'){punto(); pausa(); raya(); pausa();
punto(); espacio();}
        if(cadena[i] == 's'){punto(); pausa(); punto(); pausa();
punto(); espacio();}
        if(cadena[i] == 't'){raya(); espacio();}
        if(cadena[i] == 'u'){punto(); pausa(); punto(); pausa();
raya(); espacio();}
        if(cadena[i] == 'v'){punto(); pausa(); punto(); pausa();
punto(); pausa(); raya(); espacio();}
        if(cadena[i] == 'w'){punto(); pausa(); raya(); pausa();
raya(); espacio();}
        if(cadena[i] == 'x'){raya(); pausa(); punto(); pausa();
punto(); pausa(); raya(); espacio();}
        if(cadena[i] == 'y'){raya(); pausa(); punto(); pausa();
raya(); pausa(); raya(); espacio();}
        if(cadena[i] == 'z'){raya(); pausa(); raya(); pausa();
punto(); pausa(); punto(); espacio();}
        if(cadena[i] == ' '){espacio(); espacio();}
    }

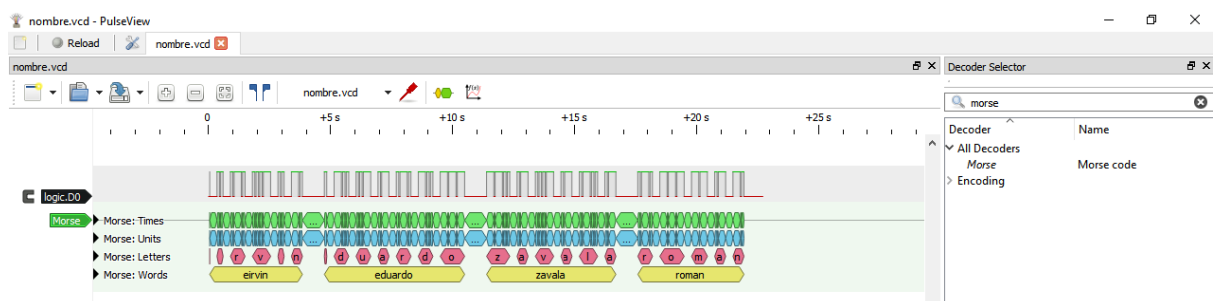
```

Con esto se compila el código para buscar errores y se implementa en el simulador wokwi.



**Figura 3.** Simulación del código

La simulación generó un archivo con extensión vcd el cual se importará a Pulse View para ver si la conversión a código morse fue exitosa.



**Figura 4.** Comprobación del archivo generado por el simulador wokwi

Se puede observar que el nombre es el correcto por lo que se comprueba el funcionamiento del programa, este programa puede mostrar cualquier cadena



propuesta pero se puede hacer aun mas especifico para que solo muestre el nombre.

## **Conclusiones**

Esta primera práctica introduce la forma en que se programa un ESP32, aunque no se entendieron ciertos componentes del código se puede observar que es un tipo de C con las modificaciones necesarias para desarrollar software de microcontroladores. Además las herramientas del simulador ESP32 y el PulseView no se habían visto antes, por lo que son nuevas herramientas para mejorar las capacidades al programar este tipo de dispositivos.

## **Bibliografía**

Beningo, J. (2021, Marzo 23). Las herramientas de depuración reducen el tiempo de desarrollo del IoT | DigiKey. Digikey. Consultado el 21 de febrero de 2022, desde <https://www.digikey.com.mx/es/articles/the-professional-guide-to-debugging-tools-and-techniques-for-iot-devices>

CLion: IDE multiplataforma para C y C++ de JetBrains. (n.d.). JetBrains. Consultado el 21 de febrero de 2022, desde <https://www.jetbrains.com/es-es/clion/>

Eclipse Embedded CDT (C/C++ Development Tools). (n.d.). Projects. Consultado el 21 de febrero de 2022, desde <https://projects.eclipse.org/projects/iot.embed-cdt>

IAR Embedded Workbench for Arm. (n.d.). IAR Systems. Consultado el 21 de febrero de 2022, desde <https://www.iar.com/products/architectures/arm/iar-embedded-workbench-for-arm/>

MPLAB® X IDE. (n.d.). Microchip Technology. Consultado el 21 de febrero de 2022, desde <https://www.microchip.com/en-us/tools-resources/develop/mplab-x-ide>

Oracle. (n.d.). B Developing Embedded Applications in NetBeans IDE (Release 8).

Consultado el 21 de febrero de 2022, desde

<https://docs.oracle.com/javase/8/embedded/develop-apps-platforms/develop-apps-in-netbeans.htm>

TikTak Draw. (2019, Febrero 5). CURIOSIDADES SOBRE EL CÓDIGO MORSE |

Draw My Life. YouTube. Consultado el 21 de febrero de 2022, desde

<https://www.youtube.com/watch?v=b55qZVwbomU>

µVision IDE. (n.d.). Keil. Consultado el 21 de febrero de 2022, desde

<https://www2.keil.com/mdk5/uvision/>