Sistemas Embebidos

Introducción a la placa de desarrollo del curso y a VSCode + ESP-IDF plugin

---

Integrantes del grupo:

- Sebastián Berardi
- Gervasio Hernández
- Joaquín Cuitiño

Docentes: Nicolás Calarco y Pablo Alonso

6/05/2025

Objetivo

Interiorizarse con el hardware de la placa de desarrollo del curso (ESP32) y el entorno de desarrollo ESP IDF en VSCode. Comenzar a escribir código en lenguaje de programación C para aplicaciones embebidas. Desarrollar funciones básicas de E/S. Emplear las herramientas de compilación y depuración que ofrece el entorno de desarrollo.

Desarrollo

Introducción al ESP-IDF + VSCode, creación de proyectos y primeros pasos.

1. Creación de un proyecto en VSCode + Espressif IDE (shortcut CTRL + E N)

    a. Abrir el VSCode y crear un nuevo proyecto vacío (View -> Command palette ->ESP-IDF: New Project) y seleccionar sample_project para el ESP32-S2-KALUGA-1. Darle un nombre al proyecto, por ejemplo “Laboratorio” y guardarlo en el directorio que deseen.

    b. Analizar el árbol de directorios creado en el navegador del proyecto (vista “Explorer”, Workspace del VSCode).

    c. Analizar los archivos generados: ¿qué secciones tiene?

    - .devcontainer: Contiene los archivos necesarios para que VS Code, a través de la extensión “Dev Containers”, monte el código, instale dependencias y que todo funcione correctamente sin depender del sistema anfitrión.

    - vscode: Define cómo se compila, depura y ejecuta el código desde VSCode.

    - main: Esta es una carpeta con dos archivos. El main.c que es donde se escribe el código principal de la aplicación. Y un CMakeLists.txt el cual le dice a CMake cómo compilar la parte de la aplicación que vive en esta carpeta main.

    - .gitignore: Es un archivo de texto que le dice a Git qué archivos o carpetas debe ignorar al hacer commits.

    - CMakeLists.txt: Es un archivo que define la estructura del proyecto, nombre, dependencias, etc. Llama a main, components, etc para compilar.

    - README.md: Es un archivo de texto con una breve descripicin de la estructura de sample_project.

    d. Compilar el proyecto tal cual está (click en la barra inferior del VSCode). ¿Qué cosas cambian en el workspace del laboratorio? ¿Qué información nos brinda el compilador?

    Al buildear el proyecto, CMake crea un directorio build donde coloca todos los artefactos de compilación: archivos objeto, librerías estáticas, binarios intermedios y finales y ficheros generados. Ademas crea un archivo sdkconfig el cual es un fichero de texto plano que almacena en formato clave=valor todas las opciones seleccionadas en el menú de configuración de ESP-IDF.

En cuanto a la información brindada, aparece un desglose de tamaños de secciones, que indica cuánto ocupa cada segmento en la imagen y en la RAM del dispositivo.

2. Parámetros de Configuración (shortcut CTRL + E G)

   a. Los dispositivos ESP32 cuentan con múltiples Parámetros de configuración conocido como sdkconfig. Entre estos se pueden habilitar (o no) diversas configuraciones del ESP32, por ejemplo el nivel de log, priorizar la performance o el tamaño final de la compilación, habilitar algunas librerías, entre otras configuraciones.
   Estos Parámetros de configuración se deben configurar en VSCode o mediante el comando idf.py menuconfig en la terminal de ESP-IDF (sobre el proyecto a trabajar).

   b. Los parámetros de configuración y su valor son específicos del proyecto y pueden visualizarse en el documento sdkconfig (si se guardan nuevos parámetros los valores predeterminados están bajo el nombre sdkconfig.old) ubicado en el directorio del proyecto.
   Ver el archivo sdkconfig y analizar los distintos parámetros configurables.

      - Build Type: Define cómo se construye la aplicación y el bootloader
      - Bootloader config: Controla los ajustes del segundo bootloader que se encarga de cargar la aplicación.
      - Security features: Agrupa todo lo relativo a arranque y encriptación segura.
      - Application manager: Define metadatos y cómo se incluyen en la imagen de la app.
      - Comportamiento del Boot ROM: Ajusta la salida de logs y comportamiento del bootloader en ROM.
      - Configuracion del serial flasher: Opciones de esptool.py al flashear y monitorear:
      - Tabla de particiones: Controla cómo se organiza el flash en regiones.

   c. Afortunadamente, Espressif IDF nos permite modificar los parámetros de configuración a través de una interfaz gráfica con el botón (abajo a la izquierda en VSCode)
   Luego de seteados los parámetros, dar click en SAVE para guardar los cambios o DISCARD si no se desea realizar cambios. En caso de hacer click en RESET se restablecen los valores predeterminados en sdkconfig.old.

   d. Investigar qué opciones se pueden modificar en el sdkconfig y ver cómo cambia el archivo al guardar. Comparar diferencias con sdkconfig.old.

      Las opciones modificables son todos aquellos parámetros configurables que se encuentran en el archivo sdkconfig.

      Al guardar las nuevas configuraciones, el archivo que antes tenia las configuraciones pasa a llamarse sdkconfig.old y simplemente se diferencia en el .old y en el valor que tomen las configuraciones.

3. Compilación (shortcut CTRL + E B)
   a. Agregar al archivo main.c la siguiente definición y las siguientes variables:
   ```
   #define ARRAY_SIZE 12

   int exampleData;
   char exampleArray[ARRAY_SIZE];
   ```

b. Compilar el proyecto creado (en caso de errores corregirlos) y ver los archivos generados en la carpeta del proyecto build. Buscar y Analizar particularmente los archivos flasher_args.json y project_description.json. ¿Qué puede comentar acerca de estos archivos?

   El archivo flasher_args.json es un fichero de metadatos que usa la extensión para saber cómo invocar esptool.py al flashear. Es decir, sirve para darle instrucciones precisas al IDE para que flashee correctamente el archivo a la placa, con las configuraciones especificadas.

   El archivo project_description.json sirve como “hoja de ruta” para el IDE, es decir que le indica dónde están las fuentes, el SDK, los artefactos de compilación y cómo lanzar las tareas de build, flash y monitorear.

c. Buscar las variables antes declaradas en el archivo .map dentro de la carpeta build. ¿Qué tamaño ocupan en la memoria y en qué direcciones están alojadas?

   El exampleArray ocupa 0xc, es decir, 12 bytes y sus primeros 10 bytes se alojan en la dirección 0x3ffc1c84 mientras que aparece un *fill* de dos bytes con dirección 0x3ffc1c8e

   El exampleData ocupa 0x4, es decir, 4 bytes y esta alojada en la dirección: 0x3ffc1c90

d. Sustituir ARRAY_SIZE por 10 y ver qué ocurre con exampleArray en la memoria.

   El exampleArray ahora ocupa 0xa, es decir, 10 bytes y se aloja en la dirección: 0x3ffc1c84
   Ocupa menos espacio, por lo que se guarda en una única ubicación de 10 bytes.

Bibliografía

- *Technical Documents. Espressif:* *https://www.espressif.com/en/support/documents/technical-documents*