Entornos de desarrollo para la tarjeta ESP32

Con una gran cantidad de características, periféricos y la disponibilidad de conexión para Wi-Fi han hecho que la tarjeta de desarrollo ESP32 sea la más popular en aplicaciones IoT y también como un microcontrolador eficiente y buena opción. Otra de las características importantes es su versatilidad a la hora de programarla, como hace ya casi 10 años desde que salió su primer modelo y con su extendida fama han hecho que organizaciones y usuarios busquen implementarla en sus proyectos, a razón de esto cada quien ha buscado una forma cómoda para programar, implementando los entornos más fáciles o con mejores prestaciones y en los lenguajes de programación más cómodos para el desarrollador y aquí radica la versatilidad de la tarjeta, mientras algunos microcontroladores están obligados a programarse en un solo entorno desarrollado por el mismo comerciante, los módulos ESP tienen esa facilidad para implementarse en diferentes entornos. Entre los lenguajes más usados se encuentran los preferidos para la programación embebida que son: C, C++, el lenguaje interpretado Python y en raros casos Rust, a continuación detallaremos los entornos disponibles y las ventajas de usar cada uno.

Arduino IDE

Una característica de las tarjetas ESP es que poseen un microprocesador basado en una arquitectura propia llamada Xtensa, el IDE de Arduino está pensado para trabajar con arquitecturas AVR o ARM(Por ejemplo). Una de las ventajas con este entorno es que existe una gran comunidad que desarrollan librerías y API's para trabajar más cómodamente con las tarjetas Arduino, entre todas estas extensiones existe la posibilidad de compilar y cargar en las tarjetas ESP el código usado en los módulos de Arduino. Por lo tanto al usar el IDE de Arduino podremos escribir código en lenguaje C++ utilizando las abstracciones que ofrece, facilitando la escritura del código.

Ventajas:

- Alto nivel de abstracción.
- Facilidad en la escritura de código.
- Gran cantidad de cabeceras preestablecidas listas para integrarse.
- Comunidad amplia en constante desarrollo.
- Gran cantidad de documentación y guías para indagar.

Desventajas:

- Consumo de recursos no deseados: Si bien las cabeceras preestablecidas son muy útiles también acarrean el inconveniente de integrar módulos o partes de código que puedan ser innecesarias para nuestro proyecto, consumiendo recursos de forma innecesaria.
- Carente de flexibilidad en la configuración de los módulos: Al ser preestablecidos, es difícil configurar y adaptar a los requerimientos del proyecto de modo que no se le puede sacar el máximo rendimiento a las tarjetas.

MicroPython

Se trata de una implementación del lenguaje Python para microcontroladores, la gran ventaja con este, es su baja curva de aprendizaje, un alto nivel de abstracción y una comunidad que brinda soporte constantemente. Estamos hablando del lenguaje más optimo para acercarse a las tarjetas ESP, ideal para principiantes. Existen diferentes entornos con los que se puede trabajar en MicroPython, el más popularizado es Thonny, aunque como tal no es necesario utilizar un IDE para trabajar con este, ya que las herramientas necesarias pueden ser accedidas desde consola sin demasiada dificultad a diferencia de otros entornos. En este punto es importante remarcar el hecho de que Python es un lenguaje interpretado y que a diferencia de los lenguajes compilados este necesita de un interprete para poder ejecutar el código, por lo que es necesario cargarlo en las tarjetas ESP antes de poder cargar código de MicroPython.

Ventajas:

- Baja curva de aprendizaje y facilidades de implementación.
- Gran cantidad de soporte, documentación y recursos.
- Facilidad en la escritura de código.

- Gran cantidad de módulos para trabajar.
- Capacidad para integrar módulos escritos en C, mejorando asi la velocidad de ejecución

Desventajas:

- La necesidad de cargar el interprete en la memoria reduce su capacidad lo cual debe ser considerado si se busca implementar códigos muy grandes.
- Al ser un lenguaje interpretado no es recomendado en aplicaciones que requieran ejecutarse en tiempo real.

Espressif IoT Development Framework(ESP-IDF)

EL framework desarrollado por la misma Espressif con el que originalmente se deben programar las tarjetas ESP, esta entrega las herramientas, API's y librerías necesarias para poder trabajar y desarrollar proyectos. La misma desarrolladora otorga un IDE del mismo nombre para poder usarlo, a la fecha de redacción de este documento(22 de marzo de 2023) el IDE ESP-IDF es relativamente nuevo y anteriormente se implementaba junto a los IDE's Eclipse, Platformio y Visual Studio Code(Sí, VSCode es un IDE). Este framework utiliza l lenguaje C, de nivel intermedio y estructurado es ideal para implementarse en sistemas embebidos, entregando la versatilidad de trabajar a nivel de bits y con suficiente abstracción para facilitar la codificación, por lo que se trata del lenguaje ideal para poder sacar el máximo provecho a cada periférico gracias a las API´s proporcionadas. Al trabajar de esta manera requiere tiempo ya que la curva de aprendizaje es alta, sumado a su muy baja popularidad hace difícil encontrar documentación clara sobre la cual guiarse.

Ventajas:

- Acceso a todas las funcionalidades completas de la tarjeta.
- Al emplear un lenguaje de nivel medio es ideal para usarse tanto con la posibilidad de trabajar a bajo y alto nivel.
- Al usar el framework es posible configurar cualquier aspecto de la tarjeta con total libertad.
- Gran cantidad de librerías las cuales permiten trabajar con cada periférico lo que permite emplear un alto nivel de abstracción facilitando la programación.
- Opción para trabajar con programación modular, pudiendo manipular la estructura del proyecto y realizar una depuración de forma más cómoda.
- Amplia documentación con ejemplos para cada funcionalidad de la tarjeta.
- Soporte para integración de módulos escritos en C++.

Desventajas:

- Si bien existe gran cantidad de documentación esta no es clara por lo que encontrar guías o ayuda en la programación se vuelve una tarea complicada.
- A pesar de mantener soporte y actualizaciones constantes, el framework aún posee errores y algunas carencias que lo vuelven complejo de usar, en especial para desarrolladores poco experimentados.
- Alta curva de aprendizaje.

Cada IDE posee sus ventajas y desventajas lo que los vuelve ideales en determinadas situaciones y se debe dejar en consideración de las necesidades del proyecto sobre cual entorno es el más eficiente.