Universidad Zaragoza

Trabajo Fin de Grado

Estudio de dispositivo de Edge Computing con acelerador hardware de redes neuronales convolucionales basado en arquitectua RISC-V

Study of an Edge Computing device with a convolutional neural networks hardware accelerator based on the RISC-V architecture

Autor

Enrique Torres Sánchez

Director

Enrique Fermín Torres Moreno

ESCUELA DE INGENIERÍA Y ARQUITECTURA

2020

Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. Enrique Torres Sánchez,

con nº de DNI 78772353K en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado, (Título del Trabajo) Estudio de dispositivo de Edge Computing con acelerador hardware de redes neuronales convolucionales basado en arquitectura RISC-V,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 22 de junio de 2020

Fdo: Enrique

AGRADECIMIENTOS

En primer lugar, agradezco a mi tutor toda su ayuda y cooperación a lo largo del trabajo, así como su apoyo en todas las problemáticas surgidas en el desarrollo de este. Gracias a la Universidad de Zaragoza por haberme facilitado los medios necesarios para la realización de este Trabajo Final de Grado. También agradezco a mis compañeros de clase, y a mi grupo de amigos de la universidad por todo su apoyo, y por todos los buenos momentos a lo largo de estos cuatro años que finalizan con este trabajo. Muchas gracias a mis padres, Enrique y Anabel, por comprenderme y aguantarme incluso en los momentos de estrés a lo largo del grado, y por proporcionarme las herramientas y las condiciones necesarias para lograr terminar esta etapa de mis estudios. Agradezco a mi hermana el ser siempre un apoyo fundamental para mí, y por darme ánimos día tras día. Finalmente, agradezco a Jana su constante apoyo y ayuda, por ser mi motivación para seguir adelante y por hacer lo imposible por ayudarme siempre.

RESUMEN

RISC-V es una arquitectura abierta emergente que está cogiendo fuerza para una gran cantidad de aplicaciones de bajo consumo e IoT. Sin embargo, con la estabilización de las extensiones base de la arquitectura, y el comienzo de comercialización de SoCs basados en RISC-V, como el Kendryte K210 (con un precio de 5 dólares), surge la cuestión de si verdaderamente el estándar abierto acaba siendo una facilidad para los desarrolladores de aplicaciones sobre la plataforma. Se han evaluado los entornos de desarrollo, el toolchain, los procesos de depuración relacionados con la placa de desarrollo Sipeed MAIX Go, así como el SDK standalone y el port de Micro Python para el K210. También se ha estudiado el pipeline de entrenamiento para el acelerador de redes neuronales convolucionales del Kendryte K210, con soporte de Tiny YOLO v2.

Se ha realizado una aplicación prueba de concepto de IoT EDGE de reconocimiento de objetos basada en IA acelerada por hardware, de bajo coste y consumo energético, con la funcionalidad de cámara de seguridad capaz de distinguir en el propio dispositivo, y no en la nube, si lo que aparece en imagen es una persona o una mascota mediante detección de objetos, para reducir el número de interacciones del usuario de la aplicación. Además, se ha probado la versatilidad de la aplicación con el uso de otro modelo preentrenado para la detección de posicionamiento correcto de mascarillas en transeúntes en el contexto de la pandemia COVID-19. Finalmente, se ha analizado el rendimiento del dispositivo SiPEED Maix Go, y el uso de ancho de banda y consumo en comparación con un dispositivo ya en el mercado.

A lo largo del proceso se ha constatado que la documentación para desarrolladores es escasa, los entornos de desarrollo se encuentran en un estado de poca madurez, y los procesos de depuración en ocasiones son inexistentes. Sin embargo, las capacidades de IA, el rendimiento que ofrece, el bajo consumo del dispositivo, y la reducción de uso de ancho de banda pueden resultar en un posible auge del uso de este tipo de dispositivos para IoT, así como el auge del Edge Computing y el AIoT.

Este trabajo fue finalmente sometido al congreso internacional DCIS 2020 como un artículo de seis páginas de extensión con título: **Developing an AI IoT application with open software on a RISC-V SoC** (Anexo A).

ABSTRACT

RISC-V is an emergent open architecture that is gaining strength in low-power IoT applications. The stabilization of the architectural extensions and the start of commercialization of RISC-V based SOCs, like the Kendryte K210 (with a price of just $5), raises the question of whether this open standard will facilitate the development of applications in specific markets or not.

In this document the development environments, the toolchain, the debugging processes related to the Sipeed MAIX Go development board, as well as the standalone SDK and the Micro Python port for the Kendryte K210 are evaluated. The training pipeline for the built-in convolutional neural network accelerator, with support for Tiny YOLO v2, has also been studied.

In order to evaluate all the above aspects in depth, a low-cost, low-power, IoT edge application based on AI has been developed. The application serves the functionality of a surveillance camera that distinguishes wether a recognized movement has been done by a human or by a pet, all in the device the application is running on. In the context of the current COVID-19 pandemic, the application is capable of labeling whether a pedestrian is wearing a face mask or not, doing real-time object recognition at a mean rate of 13 FPS, showing this way the versatility of the application and the device.

Throughout the process, it can be concluded that, despite the potential of the hardware and its excellent performance/cost ratio, the documentation for developers is scarce, the development environments are in low maturity levels, and the debugging processes are sometimes nonexistent. However, the AI capabilities, the performance that is offered by the device, its low consumption, and the possibilites of bandwith reduction, can result in a upswing of these types of devices, as well as of Edge Computing and AIoT.

This work was submitted to the DCIS 2020 international congress as a six pages long paper with the following title: **Developing an AI IoT application with open software on a RISC-V SoC** (Appendix A).

Índice

Capítulo 1

Introducción

1.1. Oportunidades de las tecnologías IA en IoT

Los avances recientes en sistemas empotrados e inteligencia artificial (IA), en combinación con costes de manufactura y precios de cara al consumidor más bajos, han hecho que la arquitectura RISC-V esté en el foco tanto del desarrollo de sistemas empotrados, como en el ámbito de la investigación. Estos hechos en conjunto han conseguido traer la inteligencia artificial y el deep learning a sistemas de bajo consumo, con nuevo silicio que está siendo añadido a los chips, como por ejemplo aceleradores hardware de redes neuronales. Un ejemplo de estos dispositivos es el System on Chip Kendryte K210, que por ejemplo se vende al consumidor por aproximadamente cinco dólares.

El campo de deep learning ha conseguido muchos éxitos en varios dominios. Tanto investigadores como ingenieros han puesto su esfuerzo en aplicar algoritmos de inteligencia artificial a dispositivos empotrados y dispositivos móviles. La visión por computador, así como el reconocimiento de sonido por inteligencia artificial, han ganado popularidad recientemente debido a las nuevas soluciones de bajo coste y consumo. La detección de objetos, la clasificación de imágenes (la detección y el reconocimiento de caras, la obtención del tamaño y coordenadas de un objeto en un imagen, etcétera) ahora pueden ser realizados en tiempo real.

Todos estos conceptos se unen en el término AIoT, o Artificial Intelligence of Things, que trata de traer la inteligencia artifical al EDGE, o allá donde se encuentra la fuente de datos sobre la que se quiere extraer información. A medida que la inteligencia artifical se mueve cada vez más hacia el EDGE y hacia dispositivos como sensores inteligentes o cámaras, en muchas ocasiones esto elimina la necesidad de racks de computación basada en la nube ya que el análisis de los datos se mueve al dispositivo IoT. Esto puede reducir ancho de banda y posible latencia entre dispositivos conectados libremente, o en los que la latencia es crítica y por tanto es ventajoso que el tratamiento de los datos se

realice en el propio dispositivo donde se están recolectando.

La inteligencia artificial a día de hoy está generalizada tanto en las aplicaciones a nivel de particulares, como en las aplicaciones a nivel de empresa. Con el rápido crecimiento del número de dispositivos conectados, la creciente demanda de privacidad o confidencialidad, la necesidad de latencias bajas y las restricciones de ancho de banda, los modelos entrenados de inteligencia artificial que antes corrían en la nube cada vez más necesitan ejecutarse en el EDGE.

Están surgiendo una gran cantidad de placas de desarrollo de bajo coste con capacidad de cómputo suficiente para ejecutar sistemas de detección de objetos en tiempo real como Tiny-YOLO [1], MobileNet-v1 [2] o TensorFlow Lite [3], lo que propicia un auge del desarrollo de AIoT en el futuro cercano, y por tanto un auge del cómputo en el EDGE.

Estas nuevas capacidades de los sistemas empotrados y la aparición del AIoT, junto al cada vez más bajo coste de nuevos transistores gracias a que la Ley de Moore se sigue manteniendo en términos de precio, son las principales motivaciones para el estudio de un dispositivo de bajo coste que contenga estas características mencionadas.

1.2. Objetivos del trabajo

En este Trabajo Final de Grado (TFG), el objetivo principal fijado es el estudio de un dispositivo relacionado con la computación en el EDGE, con capacidades de inteligencia artificial acelerada por hardware. Este objetivo principal se subdivide en varios objetivos, que se proponen a continuación:

- **Objetivo 1:** Analizar el hardware del SoC Kendryte K210 y sus capacidades teóricas, así como el módulo MAIX-I y la placa de desarrollo MAIX Go basada en este SoC.

- **Objetivo 2:** Analizar el toolchain necesario para desarrollar aplicaciones para el SoC Kendryte K210 y el dispositivo derivado MAIX Go, estudiando a su vez el estado de madurez de los entornos de desarrollo disponibles, así como las distintas opciones que existen de programación para el dispositivo.

- **Objetivo 3:** Desarrollar una prueba de concepto de aplicación que haga uso de las capacidades de inteligencia artificial del SoC Kendryte K210.

- **Objetivo 4:** Evaluar el rendimiento del dispositivo, su consumo, sus ventajas con respecto a un dispositivo que se encuentre ya en el mercado, y analizar la capacidad de los modelos de inteligencia artifical entrenados con el pipeline elegido.

Capítulo 2

Descripción del hardware

Para la realización del estudio propuesto, se han valorado distintos dispositivos, como por ejemplo las placas de desarrollo de SiFive, que ofrecen System on Chips basados en la arquitectura RISC-V y son conocidas en el mercado. Sin embargo, la disponibilidad de estas placas de desarrollo, en conjunto con el precio al que podían ser compradas (por ejemplo, la placa de desarrollo más barata que ofrece SiFive, llamada HiFive1 Rev B, se vende por cincuenta y nueve dólares en Estados Unidos) hicieron descartarlas como viables para estudio.

Sin embargo, la empresa Kendryte recientemente ha lanzado al mercado el System on Chip llamado Kendryte K210 basado en la arquitectura RISC-V, el cuál tiene un precio de cara al consumidor de cinco dólares. Además, este SoC tiene hardware dedicado para realizar reconocimiento de objetos en tiempo real en el propio SoC con modelos de inteligencia artificial preentrenados. Por estos motivos, se buscaron placas de desarrollo que utilizaran este SoC, y finalmente se optó por utilizar la placa de desarrollo comercializada por la empresa SiPEED que recibe el nombre de SiPEED Maix Go [4], y que se vende de cara al consumidor a un precio de cuarenta dólares.

La información de este capítulo ha sido recolectada desde multitud de fuentes y mediante la realización de experimentos propios, debido a que la documentación del hardware es escasa o a veces inexistente.

2.1. System-on-Chip (SoC)

Kendryte K210 (**K210**) [5] es un System on chip (SoC) diseñado por la empresa Canaan [6] e implementado sobre el proceso avanzado de 28 nanómetros de TSMC, que ofrece *ultra bajo consumo* e integra visión por computador y reconocimiento de sonido por inteligencia artificial (*machine hearing*) en el propio SoC. Está orientado a los mercados de la inteligencia artificial empotrada y del IoT, pero también puede ser utilizada como una Unidad de Microcontrolador (*Microcontroller Unit* o MCU) de

alto rendimiento. Salió al mercado en Septiembre de 2018 y su actual precio de compra mayorista ronda los cinco dólares.

Los componentes principales del SoC son un procesador de 64 bits con dos cores basado en la arquitectura RISC-V (CPU), un acelerador hardware de redes neuronales convolucionales (KPU), un acelerador de audio (APU), transformación de Fourier rápida en silicio, silicio dedicado para realizar criptografía de SHA256 y 8 MiB de memoria estática de acceso aleatorio (SRAM).

La SRAM está dividida en dos partes diferenciadas: 6 MiB están dedicados a computación de propósito general y los otros 2 MiB están dedicados en exclusiva a la KPU (en caso de que esta esté siendo utilizada). Sin embargo, en caso de que la KPU no esté corriendo, los 2 MiB dedicados pueden ser utilizados como memoria de porpósito general junto a los otros 6 MiB. La figura 2.1 muestra el diagrama de bloques del SoC K210, donde se pueden apreciar todos los componentes descritos.

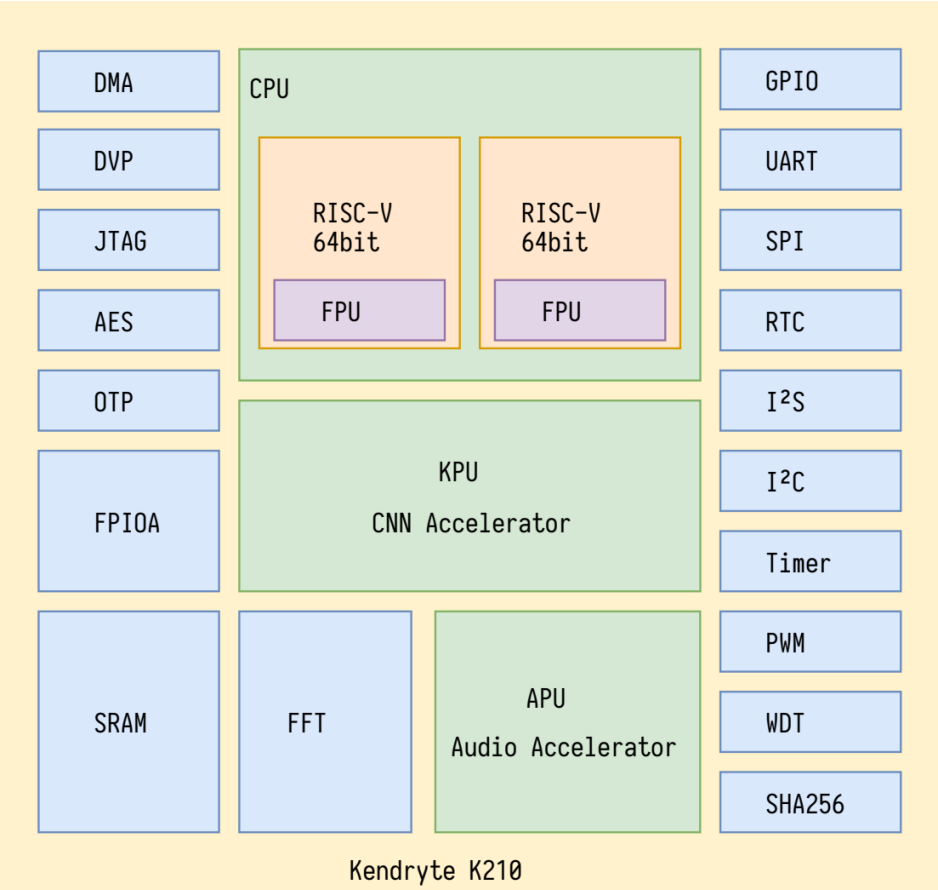


Figura 2.1: Diagrama de bloques del SoC K210

Los cores de la CPU implementan la ISA IMAFDC de la arquitectura RISC-V de 64 bits (**RV64GC**) [7]. Este set de instrucciones es adecuado para tareas generales y provee soporte para diferentes niveles de privilegio para mejorar la seguridad. También provee gestión avanzada de interrupciones que pueden ser enrutadas a cualquiera de

los dos cores ofreciendo así mejor eficiencia en términos de consumo, mejor estabilidad y mejor fiabilidad.

Cada core de la CPU contiene una unidad de coma flotante (FPU) que cumple con el estándar IEE754-2008 y que soporta operaciones de simple y doble precisión como multiplicaciones, divisiones y raíces cuadradas. Cada core tiene además una cache de instrucciones con un tamaño de 32 KiB, así como una cache de datos con el mismo tamaño. La frecuencia de reloj de la CPU puede ser ajustada desde los 400 MHz nominales hasta los 800 MHz. El consumo teórico según las especificaciones del SoC queda por debajo de los 350 mW cuando se ejecutan rutinas de reconocimiento facial, y por debajo de los 35 mW en caso de que ambos cores se programen para que se mantengan en modo WFI (*Wait For Interrupt instruction*).

La **Unidad de Procesamiento del Conocimiento o KPU** (*Knowledge Processing Unit*) es un procesador de redes neuronales de propósito general con convolución integrada (kernels 1x1 y 3x3), normalización en lote, activación (por ejemplo ReLU o sigmoide) y operaciones de agrupación (como función de máximo, o de media). La KPU no tiene un límite directo del número de capas de la red neuronal que se ejecuta dentro de ella. Además, cada capa de la red neuronal convolucional puede ser configurada de manera individual, incluyendo el número de canales de entrada y de salida, así como el ancho de línea y la altura de columna de entrada y de salida.

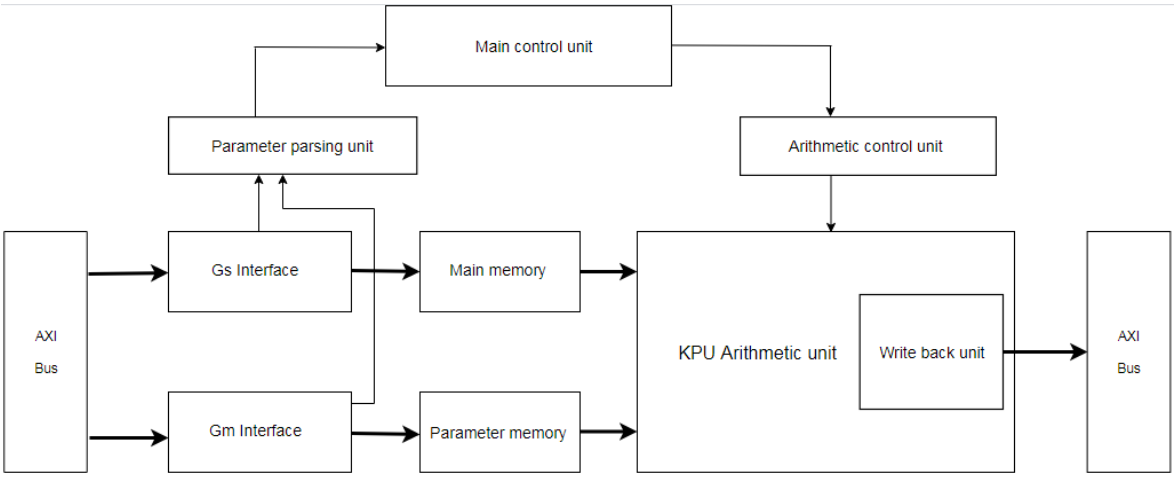


Figura 2.2: Diagrama de flujo de la Knowledge Processing Unit (KPU) [8].

La figura 2.2 muestra el diagrama de flujo que representa el funcionamiento de la KPU del SoC K210. La KPU se comunica con los cores mediante un bus AXI tanto para recibir instrucciones, como para devolver resultados.

La KPU soporta modelos de coma fija. Alcanza un rendimiento teórico de 0.25 TOPS (0.5 TOPS si el SoC se sube de frecuencia a 800 MHz) mientras se ejecutan multiplicaciones de 16 bits en sus 64 unidades KLU (Kendryte Logical Unit), las

cuales disponen de un camino de datos de 576 bits, que utiliza SIMD o Single *Instruction Multiple Data*. Según las especificaciones del dispositivo, se puede alcanzar un rendimiento de reconocimiento en tiempo real de 30 frames por segundo si los parámetros de la red neuronal se mantienen por debajo de los 5.9 MiB de peso. Por otro lado, el espacio disponible en flash es el único limite para aplicaciones que no son en tiempo real.

El módulo **APU de preprocesamiento** es el responsable del preprocesamiento de la direccionalidad del sonido que está analizando (si se utiliza un sistema de micrófonos direccionales con el SoC) y la salida de datos de voz. Con hasta ocho canales de entrada de datos de audio, la APU es capaz de implementar un array de micrófonos, escaneo simultáneo, preprocesamiento, y formación de haz a partir de fuentes de sonido que provienen desde hasta 16 direcciones. La APU utiliza el módulo de transformacion rápida de Fourier y el sistema de DMAC (Direct Memory Access Controller) para almacenar los datos de salida en la memoria del sistema.

El módulo de interfaz de cámara DVP (Digital Video Port) soporta cámaras con un tamaño máximo de frame de 640x480, así como un framerate de 30 frames por segundo, o de 60 frames por segundo en caso de que se esté utilizando QVGA (Quarter Video Graphics Array). El interfaz puede enviar imágenes tanto a la KPU como a una pantalla LCD a través de una interfaz MCU (Microcontroller Unit).

El módulo FPIOA (Field Programmable IO Array) flexible puede mapear hasta 255 funciones a los 48 GPIOs (General Purpose Input/Output) del SoC que pueden provenir desde otros aceleradores y periféricos como UART, WDT, IIC, SPI, I2S, TIMER, RTC, PWM, etcétera.

El SoC K210 tiene aceleradores de los algoritmos AES y SHA256 que proveen al programador o al usuario con funcionalidades básicas de seguridad. La OTP (One-time Programmable Memory Unit) y el acelerador AES permiten el cifrado de firmware, además de la comprobación de integridad para dar soporte a resistencia contra manipulaciónes/alteraciones.

El K210 también permite depuración mediante una interfaz JTAG, así como mediante una UART de alta velocidad. De este modo, se permite establecer breakpoints y la operación del SoC en modo de depuración. También se dispone de registros físicos de monitorización dedicados exclusivamente al conteo de instrucciones y de ciclos de ejecución.

2.2. Módulo MAIX-I

El módulo **Sipeed MAIX-I** [9], también llamado M1w, integra el SoC Kendryte K210 con un circuito de alimentación DC/DC, 8MB/16MB/128MB de memoria flash, y un microcontrolador ESP8285 con Wi-Fi. Todo ello se dispone en un módulo que mide una pulgada por una pulgada de superficie que tiene un formato amigable para incluirlo en placas de circuitos (breadboards), pero que también puede incluirse en circuitos mediante tecnología de montaje superficial (SMT). El módulo está dirigido al mercado de la AIoT (Artificial Intelligence of Things), ofreciendo una solución de computación en el EDGE. Inicialmente, este módulo consiguió la financiación para salir al mercado mediante crowdfunding en Indiegogo, donde la empresa desarrolladora (SiPEED) consiguió un 428 % del objetivo de financiación original.

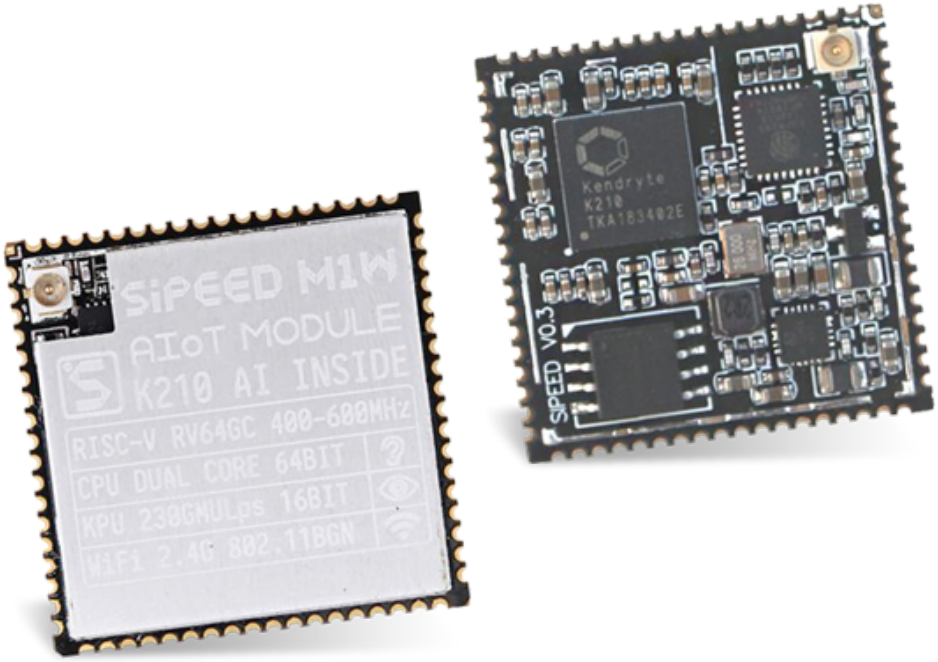


Figura 2.3: Módulo SiPEED MAIX-1 Wi-Fi [9].

La figura 2.3 muestra una imagen del módulo M1. El SoC K210 está ubicado en la parte superior izquierda de la imagen. Al lado de este se encuentran el microcontrolador ESP8285 y el conector de la antena IPX. El chip de memoria flash de 16 MiB se encuentra en la parte inferior izquierda de la imagen

El microcontrolador ESP8285 de la empresa Espressif es un SoC altamente integrado de bajo cosumo, el cual tiene capacidades completas de Wi-Fi en el propio SoC [10]. Integra una versión mejorada del procesador *L106 Diamond series* de 32 bits de Tensilica, SRAM en el propio chip, y hasta 2 MiB de memoria flash. El ESP8285

también integra interruptores de antena, amplificadores de potencia, filtros y módulos de gestión de bajo consumo. En este caso, el ESP8285 actúa como un esclavo (slave) de la MCU del host (K210).

2.3. Placa de desarrollo MAIX Go

SiPEED Maix Go es una placa de desarrollo que tiene una superficie de 88x60 mm. La placa de desarrollo está construida en base al módulo SiPEED Maix M1w. Su pequeño tamaño, el bajo consumo y el bajo coste de cara al consumidor hacen que sea una placa de desarrollo muy atractiva para desarrolladores que quieran iniciarse en el mundo de la inteligencia artificial y de los sistemas empotrados. Además, el conector USB tipo C integrado en la placa de desarrollo puede ser utilizado para alimentar la placa de desarrollo, pero también puede ser utilizado como conexión UART al mismo tiempo. Además, el conector JTAG que ofrece la placa puede ser utilizado para subir código al SoC, y también puede ser utilizado para depurar código, o simplemente como método de comunicación. El soporte de JTAG de la plaza de desarrollo está basado en el microcontrolador STM32F103C8, el cual ofrece depuración y subida de código sin una sonda de depuración externa. La figura 2.4 muestra un diagrama de la placa de desarrollo con los componentes principales que se encuentran on-board.

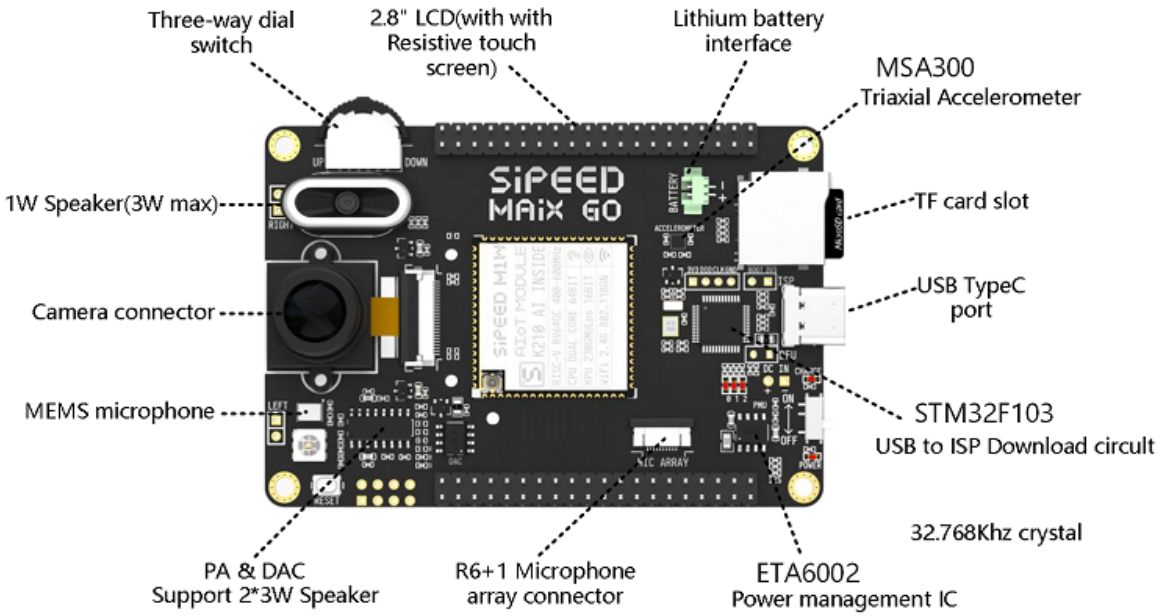


Figura 2.4: Placa de desarrollo Sipeed MAIX Go [4].

La placa Maix Go incluye un acelerómetro digital de tres ejes, un micrófono I2S, un altavoz, un LED RGB, un conector para un array de micrófonos (el cuál puede ser adquirido a parte), una rueda-botón de tres posiciones que también actúa como

potenciómetro, una ranura para tarjetas TF, una batería litio de 500 mAH, un chip de gestión para encaminamiento de corriente (para seleccionar extraer la corriente desde el propio conector USB o desde la batería de litio), un panel LCD de 2.8 pulgadas, una cámara DVP modelo ov2640 con lentes M12, una antena Wi-Fi, una caja de material acrílico sencilla, y finalmente un array de pines/conectores para poder acceder a todos los GPIOs que el SoC K210 tiene disponibles. Todos estos componentes, junto con la placa de desarrollo que ya incluye el SoC K210 cuestan $40. Esto, en comparación con los $59 que cuesta la placa HiFive1 Rev B de cara al consumidor (que solo contiene un SoC RISC-V de 32 bits y pines físicos para conexión rápida) hacen que la placa Maix Go sea un producto interesante para desarrolladores interesados en RISC-V.

Capítulo 3

Software, pipeline de trabajo y entornos de desarrollo

Durante el proceso de desarrollo de una aplicación, uno de los pasos más importantes es la selección de la toolchain que se va a utilizar. Tanto los entornos de desarrollo integrados (IDE) como las herramientas de depuración son elecciones clave cuando se desarrolla software para una plataforma. En el caso de plataformas nuevas, el software o las librerías que están disponibles pueden ser muy limitadas, o simplemente no estar maduras. Pueden tener fallos/bugs debido a su estado temprano de desarrollo, o simplemente pueden tener una documentación insuficiente (o ninguna).

En el caso del SoC Kendryte K210, un desarrollador que opte por realizar software para él, puede elegir entre uno de los pocos IDEs existentes para el lenguaje de programación C, o puede escoger programar con el port de Micro Python provisto por SiPEED (en caso de que el desarrollador esté utilizando una placa de desarrollo que contenga el módulo Maix M1w).

En el caso de la programación en C para el K210, el desarrollador puede elegir principalmente entre una de las siguientes:

- **PlatformIO:** IDE gratuito, de código abierto, y cross-platform. Está basado en Visual Studio Code y está orientado hacia el desarrollo de sistemas empotrados.
- **VisualGDB:** Extensión de Visual Studio propietaria y de pago.
- Utilizar directamente el **SDK Standalone de Kendryte**, sin ninguna capa de abstracción y ningún IDE.

En cualquiera de los tres casos, el previamente mencionado SDK Standalone de Kendryte será utilizado (ya sea directamente, o abstraído de cara al desarrollador por uno de los dos IDEs). El SDK es un set de herramientas y librerías programadas en C sobre las que se puede desarrollar software ejecutable en el SoC K210 sin utilizar un

sistema operativo por debajo. El SDK Standalone es de código abierto y su código está completamente alojado en GitHub [11].

Existe la opción de utilizar FreeRTOS en el Kendryte K210, que ofrece programación de sistemas empotrados de tiempo real. Sin embargo, para el desarrollo de este proyecto, esta opción no ha sido valorada y por tanto tampoco se ha indagado en las posibilidades completas que ofrece.

Como se ha mencionado antes, en el caso de que el desarrollador disponga de una placa de desarrollo que utilice el módulo SiPEED M1w, este puede optar por programar sobre el port de Micro Python realizado por SiPEED. Para programar con este port, las opciones son las siguientes:

- Realizar uso del IDE de código abierto que provee SiPEED, llamado **MaixPy IDE** [12].

- Utilizar el puerto serie para comunicarse mediante UART con la **terminal de Micro Python** que el port de Micro Python ofrece a través de la conexión USB de la placa de desarrollo.

- Cargar scripts de Micro Python **desde una tarjeta MicroSD**, que puede ser insertada en la ranura para tarjetas TF que se ofrece en la placa, tanto desde la terminal antes mencionada, como desde el IDE.

Tanto PlatformIO como VisualGDB han sido probados, y MaixPY IDE ha sido utilizado para el desarrollo de la aplicación prueba de concepto que se expondrá en un apartado posterior. También se ha probado la opción de desarrollar software en C para el SoC K210 sin IDE por si esta opción mejoraba las capacidades de depuración del SoC.

3.1. Desarrollo de bajo nivel y programación en C

El realizar uso de la placa de desarrollo SiPEED Maix Go comenzó inicialmente como un proyecto de investigación, que estaba centrado en probar si la placa de desarrollo podría ser utilizada como una base para que los estudiantes de grado universitario aprendieran a trabajar sobre hardware en bare metal, y más específicamente como toma de contacto con hardware basado en la arquitectura RISC-V.

3.1.1. Desarrollo con PlatformIO y problemáticas

La fase de prueba de la placa de desarrollo comenzó con el análisis del IDE PlatformIO como una opción viable de un IDE fácil de usar, que permitiera a los estudiantes abstraerse del proceso de enlazado y mapeado, y de la compilación. Sin embargo, también se buscaba que al mismo tiempo presentara una forma fácil y universal de realizar depuración de código C sobre la placa, a la vez que permitiera examinar código ensamblador RISC-V y ver el estado arquitectónico del sistema (registros físicos, estado y datos de la memoria RAM, etcétera).

Además, según las especificaciones de la placa de desarrollo, la placa SiPEED Maix Go debería permitir el subir código directamente al SoC y realizar depuración en tiempo real a través del JTAG y UART integrados. Cuando se probó esta capacidad de la placa, se encontró que el depurador universal de PlatformIO era incapaz de realizar depuración a través del puerto USB. Inicialmente se pensó que esto podía deberse a un fallo con el depurador universal de PlaformIO. Para eliminar entropía del error, se optó también por realizar pruebas tanto en el sistema operativo Windows, como en la distribución de Linux Ubuntu, en ambos casos con los drivers necesarios instalados. Sin embargo, el depurador universal de PlaformIO no logró realizar la conexión con el depurador JTAG incluido en la placa en ninguno de los dos sistemas operativos.

A raíz de este problema, se comenzó a realizar una investigación para conocer si el problema era único o si otros desarrolladores también lo habían sufrido. Esta investigación condujo a un post en el blog de GitHub de SiPEED, donde uno de los desarrolladores que trabaja para SiPEED indicaba que la sonda de depuración integrada en la placa SiPEED Maix Go aún no estaba completamente activa ya que su firmware aún estaba en desarrollo [13]. Este problema no estaba mencionado ni en la documentación de la placa de desarrollo, ni en las especificaciones descritas a la hora de comprarla.

Después de encontrar que el problema no tenía solución y dependía del hardware, se intentaron otros métodos para lograr depurar el K210 mediante una sonda de depuación. Se encontró un post de un desarrollador en el blog de SiPEED el cual indicaba que si se cortocircuitaban dos pines de la placa, esta entraba en modo DFU (Device Firmware Update) y por tanto permitía depurar el K210 mediante una sonda de depuración externa [14].

Se cortocircuitaron ambos pines, y se probó la idea realizando una depuración mediante sonda externa a través de la sonda JTAG **Olimex-Tiny-H** [15], con el pinout adecuado configurado. También se tuvo que instalar el driver adecuado para que el Olimex-Tiny-H fuera reconocido por PlatformIO. Con el modo DFU activo y

la sonda Olimex-Tiny-H se logró depurar el primer programa en C codificado para el K210, que era un *LED blink* basado en *sleeps*.

Figura 3.1: Sonda de depuración externa ARM-USB-TINY-H de OLIMEX.

El depurador sin embargo, que es una modificación de **OpenOCD** realizada por Kendryte para soportar el SoC K210, tiene un rendimiento muy bajo cuando se está depurando paso a paso código en C (la ejecución paso a paso es muy lenta). Además, aunque PlatformIO es capaz de leer el estado arquitectónico del K210, no es capaz de realizar desensamblado ni depuración sobre código desensamblado. Aunque el desarrollador sí es capaz de depurar el código en C paso por paso y seguir el camino de ejecución, no se logró realizar el mismo proceso con código desensamblado (instrucciones máquina RISC-V).

Otro fallo importante que se detectó en PlatformIO fue la incapacidad del desarrollador de subir código al SoC a través del JTAG Olimex-Tiny-H. Para mitigar este fallo, la placa de desarrollo debía ser inicializada sin modo DFU primero y se debía subir el código a través de un cable USB tipo C. Tras esto, se debía volver a apagar la placa, y se debía arrancar en modo DFU con el JTAG Olimex-Tiny-H conectado. Si se seguía este proceso, entonces se conseguía depurar el código en C.

Tras la depuración suficientemente adecuada del primer programa, se programó una versión del *LED blink* esta vez basada en interrupciones de timer. Este segundo programa no resultó en un proceso de depuración adecuado, ya que tras la ejecución del entry point al main, y de la ejecución de las funciones de configuración del timer, el depurador universal de PlatformIO quedaba en un estado catatónico y perdía por completo el camino de ejecución (*execution path*). El código por tanto no lograba parar en breakpoints puestos ni en la interrupción de reloj, ni en funciones que se tuvieran que ejecutar más allá del punto descrito con anterioridad.

Si el código se ejecutaba sin depuración, el programa realizaba la funcionalidad esperada de manera correcta. Sin embargo, si el K210 comenzaba la ejecución en modo depuración, la ejecución no funcionaba y el programa fallaba al poco tiempo. Se teorizó

inicialmente que el problema pudiera venir del uso de la sonda externa de Olimex, ya que podría estar mal configurada en algún punto de PlatformIO, o simplemente no estar completamente soportada por este.

Por este motivo y para comprobar esta hipótesis, se decidió adquirir una sonda de depuración **SEGGER'S J-LINK** [16]. No obstante, PlatformIO no logró conectarse mediante JTAG a la placa de desarrollo a través de la sonda J-LINK, incluso tras haber instalado todos los drivers correspondientes y tener la configuración adecuada en el IDE. Se sabe que no es problema de la sonda, ya que esta funcionaba con otros dispositivos. Por si acaso J-LINK simplemente no funcionara correctamente en Ubuntu, se probó también su uso desde Windows, que acabó con el mismo resultado.

Figura 3.2: Sonda de depuración externa J-Link EDU.

3.1.2. Desarrollo con VisualGDB y problemáticas

Para eliminar posibles puntos de fallo y poder descartar si este problema estaba relacionado con el hardware, o era un fallo de PlatformIO, se optó por probar otro IDE. En este caso, se optó por realizar uso de la prueba gratuita de la extensión de pago **VisualGDB** para Visual Studio. Esta extensión se dedica exclusivamente al desarrollo de programas sobre sistemas empotrados, y tiene configuraciones predefinidas para multitud de dispositivos IoT. Además, VisualGDB ofrece técnicas de depuración paso a paso incluso para microcontroladores que no ofrecen modos de depuración (como por ejemplo Arduino, donde se permite la depuración contaminando el código generado por el compilador).

VisualGDB se probó con el código del programa *LED blink* basado en interrupciones de timer que había fallado en PlatformIO. La sonda externa que se utilizó fue Olimex-Tiny-H, debido a que era la única sonda con la que se había logrado la depuración. Se encontró que VisualGDB es capaz de realizar la depuración paso a paso del código C. Además, es capaz de activar breakpoints en la rutina de interrupción de

manera correcta. Asimismo, VisualGDB puede subir código a la RAM del SoC K210 directamente a través de la sonda Olimex-Tiny-H, por lo que se eliminaba la necesidad de realizar el proceso de carga mediante el proceso indicado anteriormente. Se probó también la sonda J-LINK, pero se encontró el mismo resultado que con PlatformIO, y por tanto se ha deducido que J-LINK aún no está adaptado para funcionar con el SoC K210, aunque serían necesarias más pruebas para lograr confirmar esta hipótesis.

Sin embargo, con VisualGDB se pierde la capacidad de visualizar el contenido de los registros físicos del SoC, aunque PlatformIO sí que permitía esta opción. Un análisis más profundo de la configuración que utilizaba VisualGDB del SDK de Kendryte mostró que el IDE utilizaba una versión modificada del código de OpenOCD de Kendryte. Se encontró una página de GitHub [17] que contenía el código de la versión modificada, y en esta misma página los desarrolladores de VisualGDB, Sysprogs, explicaban que la versión original de OpenOCD ofrecida por Kendryte no lograba depurar código que se ejecutara de manera multicore en el SoC K210. Este problema surgía ya que OpenOCD era incapaz de seguir el camino de ejecución de los dos cores y por tanto, en caso de que el código saltara a ejecutar algo en el otro core que no se estaba analizando, (como es el caso de el código de *LED blink* basado en interrupciones ya que la interrupción se trataba en el otro core) OpenOCD fallaba ya que había perdido el hilo de ejecución.

De hecho, el código de OpenOCD dispuesto por Kendryte sí permite depurar los dos cores, pero sólo de manera individual, y el core a depurar debe ser seleccionado al principio de la ejecución. Esto es lo que causa el problema cuando se ejecuta código en el otro core que no ha quedado seleccionado, y por ende, lo que causa que la sesión de depuración falle por completo. Sysprogs arreglaron este fallo mediante una modificación de la lógica de sondeo de OpenOCD para que este pasara a depurar el core sobre el que se estaba ejecutando el código en ese momento. Por otro lado, como Sysprogs indican en esa misma página de GitHub, esto afecta a la lectura de los registros físicos del SoC, que con esta modificación de la lógica de sondeo no pueden ser leídos. Esto último se observó efectivamente cuando se trató de depurar el código sobre VisualGDB.

3.1.3. Comparativa de los entornos

Tanto la placa de desarrollo SiPEED Maix Go como el SoC K210 no tienen una toolchain suficientemente madura como para que un desarrollador pueda programar software en C para los dispositivos basados en este SoC. La toolchain tampoco está suficientemente madura como para servir de herramienta de aprendizaje para estudiantes universitarios en asignaturas como sistemas empotrados o desarrollo bare metal, ya que no permite indagar en el código de bajo nivel, ni permite analizar el estado

arquitectónico del sistema, dos elementos que son imprescindibles cuando se están estudiando dichas asignaturas. Además, la documentación es normalmente inexistente o se encuentra en un estado incompleto.

Tabla 3.1: Comparación-resumen entre PlatformIO y VisualGDB.

	PlatformIO		VisualGDB	
	Integrada	Olimex	Integrada	Olimex
Subida de código	Sí	No	No	Sí
Ejecución paso a paso	No	Sí	No	Sí
Breakpoints multicore	No	No	No	Sí
Desensamblado RISC-V	No	No	No	No
Registros físicos	No	Sí	No	No

La tabla 3.1 muestra un resumen de las capacidades que tienen cada una de las sondas de depuración con las que se ha logrado algún objetivo de los propuestos cuando se analizaba la toolchain de dichos dispositivos, en contraposición a los IDEs que se han podido utilizar para el desarrollo sobre la plataforma.

Si bien VisualGDB carece de la capacidad para analizar el estado arquitectónico del sistema, es la opción más completa en lo que a depuración de software programado en C se refiere. Además, también es la más fiable y la más intuitiva. Sin embargo, hay que tener en cuenta que es una opción propietaria y de pago, lo que puede limitar el acceso a dicha herramienta, y por tanto resulta inoperable dentro del ámbito académico.

3.2. Desarrollo con el port de Micro Python

En caso de que se disponga de una placa de desarrollo de SiPEED que haga uso del módulo Maix M1w, existe la opción de programar sotware para el SoC Kendryte K210 mediante el port de Micro Python realizado por los propios desarrolladores de SiPEED.

Micro Python [18] es una implementación ligera y eficiente del lenguaje de programación Python 3, que incluye un subset de la librería estándar de este y que está optimizada para ejecutarse en microcontroladores y entornos muy limitados. Según sus especificaciones, la implementación base de Micro Python cabe en 16kB de RAM.

En añadido, el port realizado por SiPEED incluye las librerías necesarias para interactuar directamente (pero con la abstracción que un lenguaje de alto nivel ofrece) con el hardware del dispositivo. Esto facilita la programación de software que hace uso de módulos hardware como timers, GPIOs e incluso la KPU del SoC. Si bien el SDK Standalone de Kendryte permite la programación de la KPU para realizar visión por computador, la abstracción que ofrece el port de Micro Python de esta

simplifica el proceso de ejecución de modelos IA preentrenados, y además permite cargarlos directamente desde una micro SD que se encuentre insertada en la ranura para tarjetas TF.

Como se ha indicado en un apartado anterior, hay varias posibilidades en cuanto a pipeline de trabajo se refiere para programar software basado en el port de Micro Python. Todas las opciones requieren que la placa de desarrollo tenga cargado en memoria flash el binario que contiene el intérprete y las librerías del port. Existen varias versiones de este binario, que contienen más o menos funcionalidades y a cambio ocupan más o menos espacio tanto en RAM como en memoria flash.

En el caso de que se quiera utilizar MaixPy IDE para programar software, se debe flashear la versión del binario del port que soporta la conexión del intérprete con el IDE. Si se quiere utilizar la placa de desarrollo con el port de Micro Python para realizar uso de las funcionalidades de visión por computador que esta ofrece, la mejor opción es utilizar el binario del port que ofrece conexión con el IDE (ya que facilita la programación de código y permite el guardado de ficheros en el ordenador sobre el que se esté programando), pero que al mismo tiempo tiene las mínimas librerías extras para poder dejar el máximo espacio en RAM para los pesos del modelo que se vaya a utilizar.

Finalmente, de los tres posibles pipelines de trabajo con Micro Python, el mejor y más completo resulta del uso de MaixPy IDE. Además de las características mencionadas con anterioridad, el IDE permite la conexión directa con la placa de desarrollo, y permite visualizar directamente en el ordenador el denominado *frame buffer*, es decir, el último frame que ha sido capturado por la cámara conectada a la placa. También permite analizar mediante gráficas los niveles de los canales RGB que está viendo la KPU, por lo que el IDE se convierte en una herramienta de análisis importante. Sin embargo, al igual que las otras dos opciones, el IDE no ofrece ejecución paso a paso, ni depuración más allá de *exception handling* y *depuración mediante impresiones en terminal*.

3.3. Puesta a punto pipeline de entrenamiento IA

El SoC Kendryte K210 dispone de una KPU (acelerador de redes neuronales convlucionales) que ejecuta un framework basado en Tiny-YOLO v2. Tiny-YOLO es una variación del backend de detección de objetos YOLO (*You Only Look Once*). YOLO se creó para mejorar la velocidad de detección de los detectores de objetos en dos fases como Fast R-CNN [19]. Aunque las R-CNNs son precisas, son muy lentas incluso cuando corren sobre GPUs.

Por el contrario, los detectores de una fase como YOLO son rápidos, obteniendo normalmente detecciones en tiempo real si se ejecutan sobre GPU. Sin embargo, esto supone que YOLO suele tener menor precisión. Debido a que Tiny-YOLO es una versión limitada de YOLO para poder ejecutarse sobre dispositivos más restringidos, esta versión tiene incluso peor precisión que YOLO. Según los análisis, Tiny-YOLO suele alcanzar un mAP (mean average precision) de 23.7 % de media, aunque este mAP puede ser suficiente para determinados detectores. Según un artículo publicado en 2018 [1], Tiny-YOLO es aproximadamente un 442 % más rápido que YOLO.

Para poder entrenar modelos mediante deep learning para la KPU del K210, existen varias opciones de framework de entrenamiento. Se pueden utilizar TensorFlow Lite [3], Caffe [20], PaddlePaddle [21] y ONNX [22]. El número de frameworks es restringido debido a que la KPU del K210 utiliza un formato binario de pesos llamado KModel. Según Kendryte, KModel es un binario de pesos compilado para obtener una mejor eficiencia mientras estos pesos son utilizados por la KPU. Este formato sólo se puede obtener utilizando la herramienta NNCASE [23] provista por Kendryte, la cual se encarga de compilar mediante inferencia pesos de modelos entrenados con únicamente los frameworks mencionados previamente.

Existe la opción de programar un script de entrenamiento que realice uso de uno de los frameworks mencionados, sin embargo, existen herramientas de entrenamiento que pueden realizar uso de cualquiera de ellos, y a la vez permiten realizar entrenamiento con datasets etiquetados que contienen bounding boxes (cuadrado de coordenadas que indica dónde se encuentra el objeto en la imagen) de los objetos. Debido a que el alcance de este trabajo no incluye la metodología de entrenamiento de un modelo mediante deep learning, se ha optado por utilizar una de estas herramientas para realizar el entrenamiento de los modelos que serán utilizados. La herramienta más completa encontrada se llama aXeleRate [24]. aXeleRate utiliza el formato de anotaciones Pascal VOC, que permite crear etiquetado para distintas clases de objetos en una misma imagen y a la vez asignar *bounding boxes* a dichos objetos.

Para poder ejecutar aXeleRate sólo hay que configurar Anaconda y crear un workspace basado en Python 3.6. Tras esto, se instalan los requisitos (*requirements.txt*) y se configura el dataset y el framework que se va a utilizar en el fichero de configuración de aXeleRate. Una vez se realiza este proceso, se puede optar por configurar CUDA y Tensorflow GPU para poder entrenar con GPU, o simplemente entrenar mediante CPU. Finalmente, se ejecuta el script de entrenamiento y aXeleRate genera un fichero de pesos de TensorFlow Lite, que en definitiva convierte utilizando la herramienta NNCASE al tipo de fichero KModel.

Capítulo 4

Evaluación de las capacidades del dispositivo

4.1. Desarrollo de una aplicación prueba de concepto

Para probar las capacidades AIoT del SoC Kendryte K210 y de la placa de desarrollo SiPEED Maix Go, se ha optado por desarrollar una aplicación prueba de concepto basada en el port de Micro Python realizado por SiPEED.

La aplicación prueba de concepto es una aplicación AIoT que actúa como cámara de videovigilancia. Esta es capaz de reconocer movimiento en su campo de visión, y puede indentificar mediante lógica interna (sin conectarse con un servidor en la nube) si el movimiento ha sido causado por una persona o por un animal de compañía (como por ejemplo un perro o un gato, según el modelo IA que se entrene).

Mediante el uso de la detección de objetos con Tiny-YOLO v2 que ofrece la KPU, la aplicación analiza las imágenes que se están capturando en tiempo real, y sólo manda una notificación (o graba vídeo y lo manda a través de internet a un servidor) cuando el movimiento ha sido producido por un humano, ahorrando así ancho de banda y espacio en la nube.

Para desarrollar esta prueba de concepto, se ha utilizado la toolchain de Micro Python que realiza uso de MaixPy IDE. Además, se ha utilizado el software aXeleRate para entrenar distintos modelos con distintos mAPs y así poder tener una comparativa general de tiempo de entrenamiento - precisión/recall.

Inicialmente, se entrenaron varios modelos para reconocimiento de humanos con el dataset ofrecido para la competición VOC 2007 [25]. La herramienta aXeleRate permite también inferir el mAP de un modelo tanto en su formato de pesos original, como en KModel. A continuación, en la tabla 4.1 se exponen los resultados de los cuatro modelos entrenados para reconocimiento exclusivo de humanos. Todos los ficheros de pesos de

los modelos entrenados acaban ocupando algo menos de 2 MB.

Tabla 4.1: Resultados de la inferencia de los modelos entrenados (Humanos)

Nº Imágenes	Épocas	Fscore	Precisión	Recall	Tiempo entrenamiento (CPU)
1856	1	0.085	0.3478	0.0484	5 minutos
1942	5	0.206	0.5526	0.1272	14 minutos
1856	20	0.352	0.6451	0.2424	5 h. y 30 min.
1942	200	0.369	0.5471	0.2788	11 h. y 41 min.

Estos modelos se entrenaron para comprobar, previa realización completa de la aplicación, si el reconocimiento de objetos del SoC K210 era suficiente para el objetivo propuesto. Los dos primeros modelos se mostraron insuficientes cuando fueron cargados en la KPU, ya que esta era incapaz de reconocer ningún humano. El tercer modelo sí logro que la KPU reconociera humanos, aunque el resultado era impreciso y en ocasiones intermitente.

Si bien el cuarto modelo tiene una precisión menor que el tercero (pero sí un mayor recall), este sí logró ofrecer un resultado viable en términos de reconocimiento, por lo que se procedió al entrenamiento de modelos que contuvieran también reconocimiento de gatos y perros, además de la clase humanos de la que ya se disponía un dataset.

Para el entrenamiento de los modelos con clases de humano, perro y gato se utilizó el dataset provisto para la competición VOC 2012 [26]. Hasta ahora, los modelos se habían estado entrenando en CPU, pero debido a que los resultados con pocas épocas no eran buenos, y el dataset ahora contenía más imágenes, se optó por configurar TensorFlow GPU con CUDA para obtener un entrenamiento más rápido.

A continuación se presenta una tabla similar a la anterior para los dos modelos entrenados con las nuevas clases. El rendimiento del segundo modelo es efectivamente mayor cuando se ejecuta sobre la KPU del SoC. El primer modelo entrenó altamente el reconocimiento de gatos, pero no logró entrenar correctamente el reconocimiento de perros ni de humanos.

Tabla 4.2: Resultados de la inferencia de los modelos entrenados (Humanos - mascotas)

Nº Imágenes	Épocas	Fscore	Precisión	Recall	Tiempo entrenamiento (GPU)
2274	50	0.4838	0.6716	0.3781	2 h. y 22 min.
2274	250	0.5157	0.6901	0.4117	6 h. y 39 min.

Tras el entrenamiento de los modelos, se procedió al desarrollo de la aplicación prueba de concepto. La aplicación simplemente activa el hardware de cámara y carga los pesos del modelo desde SD en la KPU. Tras esto, comienza a capturar imágenes y estas se analizan en la KPU. La KPU devuelve los objetos reconocidos en una lista con información como su etiqueta o su *bounding box*.

Finalmente, se utiliza esta información para mostrar en el LCD la imagen con cuadrados de colores denotando las *bounding boxes* de los objetos reconocidos. Si el objeto reconocido es un humano, este se denota con el color rojo. En caso contrario, este se denota con el color verde

4.2. Versatilidad de la aplicación y el dispositivo

Debido a que la aplicación desarrollada se basa en un modelo preentrenado para reconocer clases de objetos, y simplemente hace una distinción positiva o negativa de los objetos reconocidos, esta aplicación puede resultar muy versátil dependiendo del modelo que se aplique sobre ella.

En este caso, debido a la situación causada por la pandemia del COVID-19, se ha optado por evaluar la versatilidad de la aplicación y el dispositivo cambiando el modelo preentrenado a uno que sea capaz de reconocer si una persona lleva una mascarilla de forma adecuada o no.

Se publicó un dataset en la página web PyImageSearch [27] que contenía dos clases de imágenes, personas con la mascarilla puesta de forma adecuada, y personas con la mascarilla mal puesta o simplemente no puesta. Este dataset se dice que ha sido creado aritificialmente mediante el posicionamiento de mascarillas sobre las zonas correctas en la cara en imágenes donde no había mascarillas, de forma automática a través de reconocimiento facial y visión por computador. El dataset no está anotado con lo que se ha utilizado el software VOTT de Intel para realizar las anotaciones de forma manual.

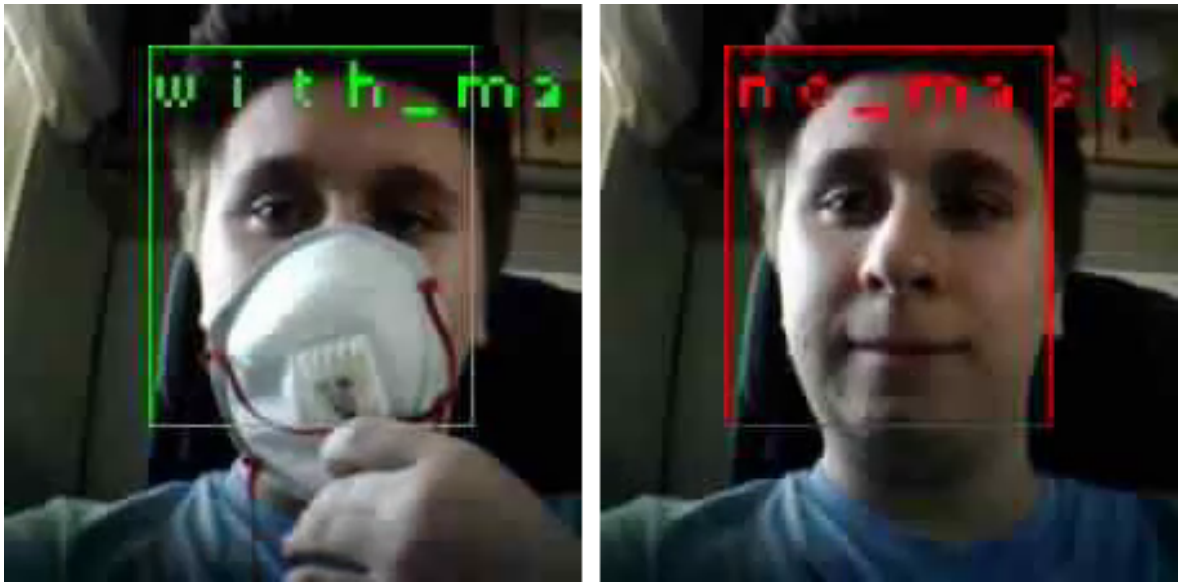


Figura 4.1: Cámara de vigilancia realizando uso del modelo entrenado para reconocimiento de mascarillas.

Sin embargo, aunque se trate de un dataset artificial, se comprueba que funciona de una manera correcta si el modelo IA se entrena lo suficiente. Por ello, se entrenó un modelo con el toolchain de entrenamiento descrito en un apartado anterior y el dataset cedido por PyImageSearch.

Este modelo alcanzó según la inferencia realizada por la herramienta aXeleRate una precisión del 99 %. Este resultado fue después probado en la aplicación desarrollada con anterioridad, en la cual no hubo que modificar ninguna parte del código para que se obtuviera la nueva funcionalidad.

La aplicación se convirtió en una cámara de detección de posicionamiento correcto de mascarilla simplemente cambiando el modelo entrenado. La figura 4.1 muestra dos capturas realizadas directamente desde la placa de desarrollo y la aplicación. La aplicación ofrece una gran versatilidad debido a que sin cambios (simplemente cambiando el modelo IA utilizado) se puede conseguir una funcionalidad completamente distinta.

Por otro lado, el dispositivo SiPEED Maix Go ofrece también una gran versatilidad debido no sólo a sus capacidades de IA, si no a sus capacidades de conectividad gracias al dispositivo ESP8285 que contiene, con lo que se pueden desarrollar dispositivos EDGE con gran facilidad, y destinados a una gran amplitud de campos.

4.3. Evaluación del rendimiento del dispositivo

Para realizar la evaluación del rendimiento del dispositivo, esta se ha centrado en el rendimiento de la parte de inteligencia artificial debido a que se cree la más interesante por ser la que da más capacidades de cómputo en términos de Edge Computing. Se ha configurado un script que utiliza el timer interno del SoC K210 para medir los frames por segundo que el dispositivo es capaz de capturar, analizar y mostrar mediante ejecución en tiempo real.

Se han realizado dos pruebas principales, la primera siendo la ejecución del script simplemente capturando imágenes con la cámara y mostrándolas en el LCD de la placa de desarrollo; y la segunda siendo la ejecución completa de una aplicación de detección de objetos que captura un frame con la cámara, lo analiza utilizando la KPU y un modelo preentrenado (en este caso se ha utilizado como modelo el detector de posicionamiento correcto de mascarilla), y finalmente muestra la imagen capturada con las *bounding boxes* señaladas en pantalla.

Estas mismas pruebas también han sido realizadas sin la necesidad de la aplicación de muestrear por pantalla el frame capturado, para así poder discernir cuánto rendimiento pierde el K210 a la hora de comunicarse con el LCD.

La primera prueba mostró que el SoC K210 era capaz de capturar imágenes y mostrarlas en el LCD con un rendimiento base de 32 frames por segundo de media. Tras esto, se quitó el uso del LCD del script, y este rendimiento subió hasta los 38 frames por segundo de media. Este resultado indica un rendimiento base sin carga mayor en la CPU, y además también indica que el muestreo por pantalla en el LCD tiene un overhead significativo.

La segunda prueba, realizando uso completo de la KPU, mostró un rendimiento medio de 14 frames por segundo cuando se realizaba uso del LCD. Esto implica una reducción del rendimiento a la mitad debido a que el resultado de la KPU es síncrono y por tanto se debe esperar a que esta lo devuelva. Es decir, el tiempo de ejecución sobre un único frame se duplica cuando se trata de detectar objetos utilizando un modelo IA y la KPU. Finalmente, se realizó esta misma prueba pero sin realizar uso del LCD, lo que incrementó el rendimiento medio a 19 frames por segundo (no sólo se elimina el overhead causado por la transmisión al LCD, si no que también se elimina el overhead causado por tener que dibujar las *bounding boxes* en la imagen capturada).

4.4. Análisis del uso de ancho de banda

Una de las ideas principales del Edge Computing es eliminar transferencias innecesarias con un servidor en la nube que gestiona las funcionalidades con más demanda de capacidad de cómputo. Por ejemplo, uno de los principales problemas en el mundo actual donde todo está conectado a Internet, es el ancho de banda disponible. Con el gran volumen de dispositivos, las restricciones físicas de ancho de banda que existen en las conexiones troncales de Internet cada vez se están convirtiendo en una problemática más real.

La idea inicial de la prueba de concepto de aplicación es la de desarrollar una aplicación que realice parte del cómputo que se realizaría en la nube en el propio dispositivo, y por tanto no necesitar enviar datos ni consumir una cantidad amplia de ancho de banda en el proceso. Por ello, se ha evaluado si efectivamente se logra esto realizando el cómputo en el dispositivo.

Se ha escogido una cámara de videovigilancia de la que ya se disponía, el dispositivo llamado YCC365 [28], que graba en calidad 1080p. Esta cámara está conectada a la nube, y es capaz de grabar 24/7 el contenido que visualiza. Además, es controlable mediante control remoto en una aplicación de móvil, y es configurable para sólo grabar en caso de detección de movimiento.

Debido a esta última característica, se ha considerado que se trata de un dispositivo idóneo para comparar el uso de ancho de banda con la aplicación desarrollada.

La comparativa entre el consumo de ancho de banda de la cámara YCC365 y la aplicación prueba de concepto se ha realizado mediante la grabación del vídeo que se obtiene a través de la cámara YCC365. Este vídeo se ha introducido en la micro SD de la placa de desarrollo Maix Go y se ha cargado frame por frame en la KPU. Tras esto, se ha medido la proporción del tiempo del vídeo en la que la KPU reconoce un ser humano y también en la que se reconoce un animal de compañía. Finalmente, en base a los datos de ancho de banda que ofrece la cámara YCC365, y la proporción de tiempo medida con la placa, se ha calculado de manera analítica el ancho de banda consumido por la aplicación prueba de concepto.

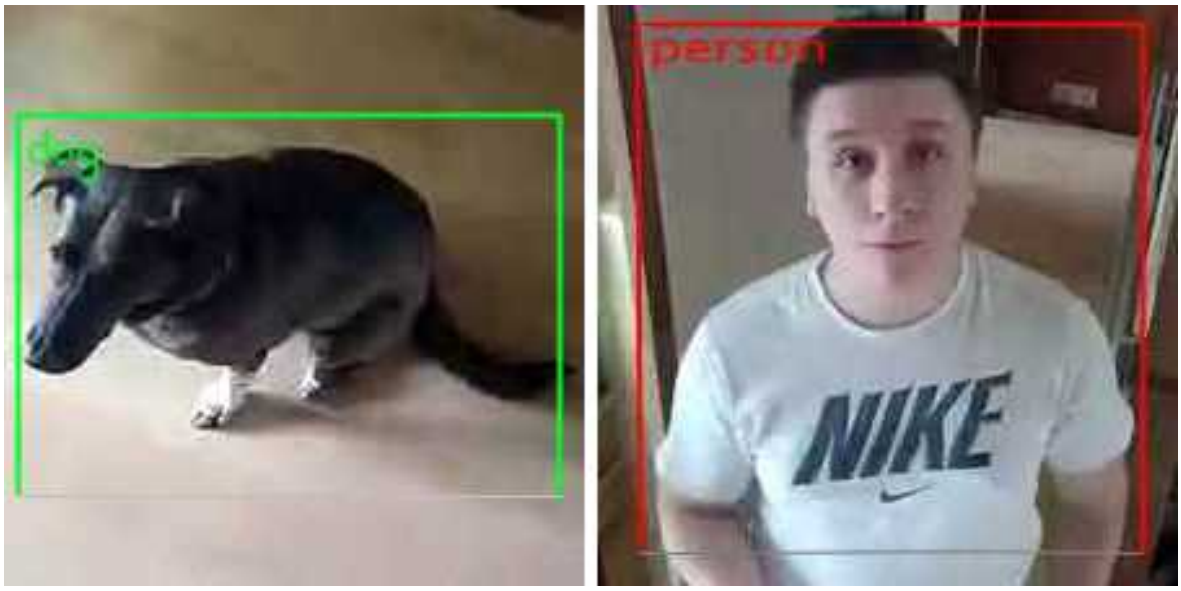


Figura 4.2: Ejemplos de detección de persona y perro en el vídeo grabado.

Según las mediciones realizadas, la cámara YCC365 tiene los siguientes consumos de ancho de banda de media según las siguientes condiciones:

Imagen fija	Imagen fija y sonido	Movimiento	Movimiento y sonido
9.88 kB/s	11.54 kB/s	28.38 kB/s	31.67 kB/s

Sin embargo, como se ha mencionado con anterioridad, la cámara puede configurarse para no grabar si no detecta movimiento, con lo que el consumo de ancho de banda continuo no es un problema ni un posible punto de mejora. Sin embargo, en el caso de el ancho de banda consumido por movimiento, puede reducirse con la aplicación desarrollada.

La figura 4.2 muestra dos frames de ejemplo del vídeo grabado con la cámara YCC365, que ha sido analizado frame a frame por la KPU cargándolo desde la tarjeta micro SD. De los 10 segundos de vídeo grabados, durante 5 de ellos el movimiento de la imagen era causado por un animal de compañía.

Por ello, en el caso de que se hubiera realizado un cribado inteligente en el dispositivo, se habrían dejado de utilizar 32 kB cada segundo de ancho de banda. Considerando que un animal de compañía puede transitar una misma zona habitualmente, a la largo esto supondría una menor transimisón de datos. Concretamente, en esta situación de prueba se habrían dejado de enviar 160 kB de datos a un servidor externo.

Además, la imagen se puede analizar en calidad de fuente. En ocasiones, este tipo de dispositivos comprimen la imagen grabada para poder limitar el ancho de banda utilizado. Por el contrario, con la aplicación desarrollada (y el dispositivo utilizado), la imagen no tendría porqué ser comprimida, y por tanto no se perderían detalles en la imagen previo análisis, algo que sí podría ocurrir si se comprimen.

4.5. Análisis del consumo del dispositivo

Debido a que los dispositivos cada vez tienen más potencia de cálculo por el mismo o similar coste, esto puede suponer que el consumo de estos dispositivos aumente. Además, en el caso del SoC K210 o de la placa de desarrollo Maix Go, este consumo podría verse aumentado aún más debido al uso del acelerador hardware de redes neuronales, o al uso de una arquitectura que aún está en pleno desarrollo.

Por este motivo, se ha decidido realizar un análisis del consumo energético que realiza la placa de desarrollo Maix Go en distintas circunstancias, y además realizar una comparativa directa entre esta y la cámara YCC365 utilizada para el análisis del uso de ancho de banda. Se optado por realizar esta comparativa debido a que la cámara YCC365 cumple la misma funcionalidad que la aplicación prueba de concepto, pero sin reconocimiento de objetos ni cálculo alguno en el propio dispositivo.

Figura 4.3: Multímetro Joulescope de Jetperch.

Se ha utilizado el dispositivo Joulescope [29] de la empresa Jetperch para realizar las mediciones de consumo y de corriente tanto de la placa de desarrollo como de la cámara YCC365. Se ha escogido este dispositivo debido a su rango dinámico de medición, que permite medir desde los nano amperios hasta los pocos amperios, y también porque

tiene librerías en Python. Concretamente, se ha desarrollado un script de Python que lee tanto el consumo en vatios como la corriente en amperios directamente del dispositivo.

De esta forma, se han realizado mediciones de consumo cuando la KPU está realizando cálculos y cuando no lo está en el caso de la placa de desarrollo; y se han realizado mediciones de consumo cuando la cámara YCC365 está en modo de bajo consumo (*standby*) y cuando está realizando una grabación activa.

La figura 4.4 muestra el consumo de la placa de desarrollo SiPEED Maix Go mientras se ejecuta sólo una rutina que captura una imagen a través de la cámara y cuenta los frames por segundo que es capaz de alcanzar; y también muestra el consumo de la placa mientras se ejecuta una rutina de captura de imagen y reconocimiento de objetos a través de un modelo cargado en la KPU. Además, para esta medición se ha desactivado el LCD y se ha desconectado ya que podría tener un consumo residual y se quiere eliminar al máximo el posible ruido en las medidas tomadas.

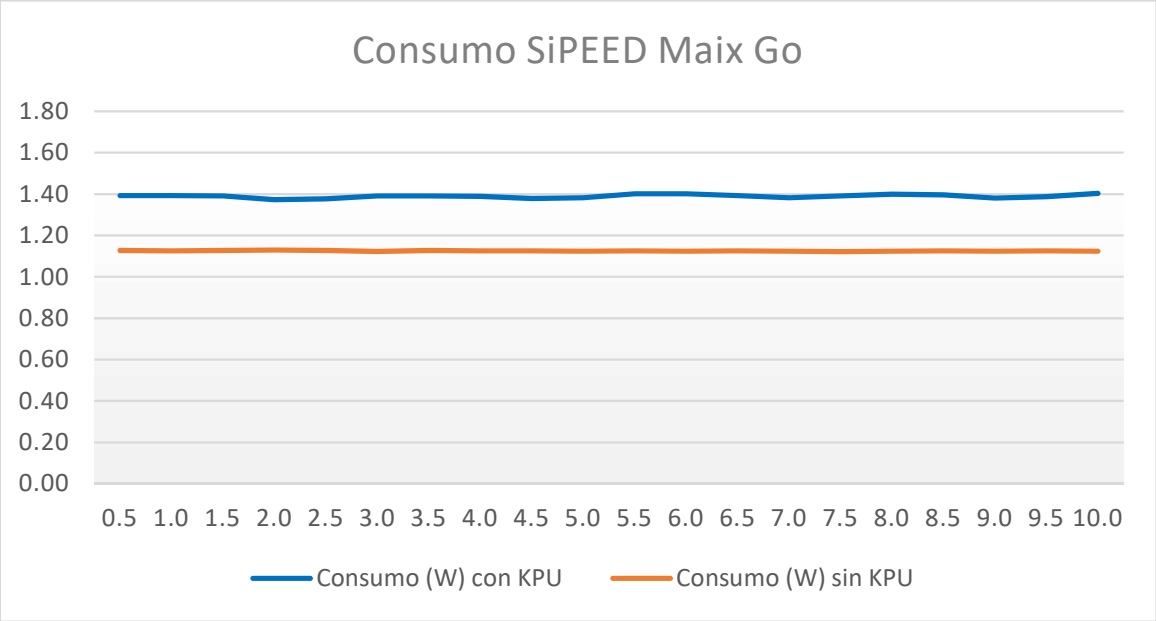


Figura 4.4: Consumo de Maix Go cuando se está haciendo trabajo en el procesador sólo, y consumo mientras se realiza trabajo en el procesador y en la KPU.

Si se comparan las dos mediciones a lo largo del tiempo, se puede apreciar que la KPU sólo añade 0.2 W de media al consumo total de la placa de desarrollo, o lo que equivale a un 16 % del consumo total del hardware. Es decir, a pesar de estar realizando cálculos que en computadores convencionales y de uso general realizarían uso completo de componentes con consumos cercanos a los 100 W o 200 W, el consumo del acelerador hardware, así como el consumo del hardware al completo, es mínimo.

Por ejemplo, si se tiene en cuenta una tarjeta gráfica de gama media como la NVIDIA GTX 1060 [30], se puede ver que consume 197 W bajo carga [31]. Tiny-YOLO

v2 corre cerca de los 240 frames por segundo en esta tarjeta gráfica. Si se comapra el ratio consumo - frames de esta con respecto al SoC K210 (18 frames por segundo a 1.4 W), se puede apreciar que este último ofrece un consumo por frame mejor (0.82 W por frame de la tarjeta gráfica contra 0.08 W por frame del SoC K210).

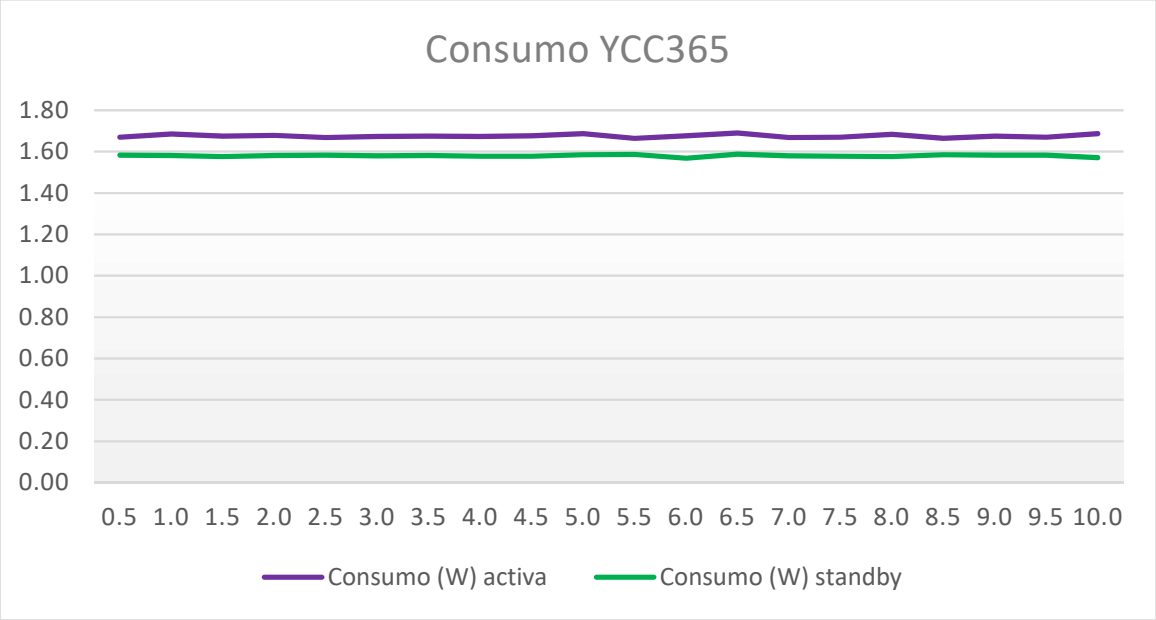


Figura 4.5: Consumos de la cámara YCC365 activa y en espera.

La figura 4.5 muestra el consumo de la cámara YCC365 cuando se encuentra activa realizando una grabación, y también cuando se encuentra en de bajo consumo (donde técnicamente está apagada).

El consumo en standby también es un punto fuerte de la placa de desarrollo Maix Go. El SoC Kendryte K210 soporta el modo WFI, que permite reducir el consumo del SoC al mínimo y esperar a que ocurra una interrupción externa. Este modo permite que el dispositio consuma cerca de 1 mW en standby, y que sea despertado por una interrupción externa, como por ejemplo un detector de movimiento externo.

Sin embargo, en el caso de la cámara YCC365, esta no puede entrar en un modo de bajo consumo. Como se puede apreciar, la cámara YCC365 reduce en tan sólo 0.1 W de media el consumo con respecto a su estado activo, mientras que el consumo de la placa de desarrollo Maix Go es despreciable.

Como se puede apreciar, el consumo de la cámara no disminuye en ningún momento de los 1.66 W, un consumo que es superior al de la placa de desarrollo Maix Go al completo, incluso cuando se está usando su capacidad de cómputo al máximo.

Además, la cámara YCC365 no tiene procesamiento de imagen ni tiene lógica interna más allá del reconocimiento de movimiento a través de un sensor de movimiento, y del envío de imagen a un servidor en la nube.

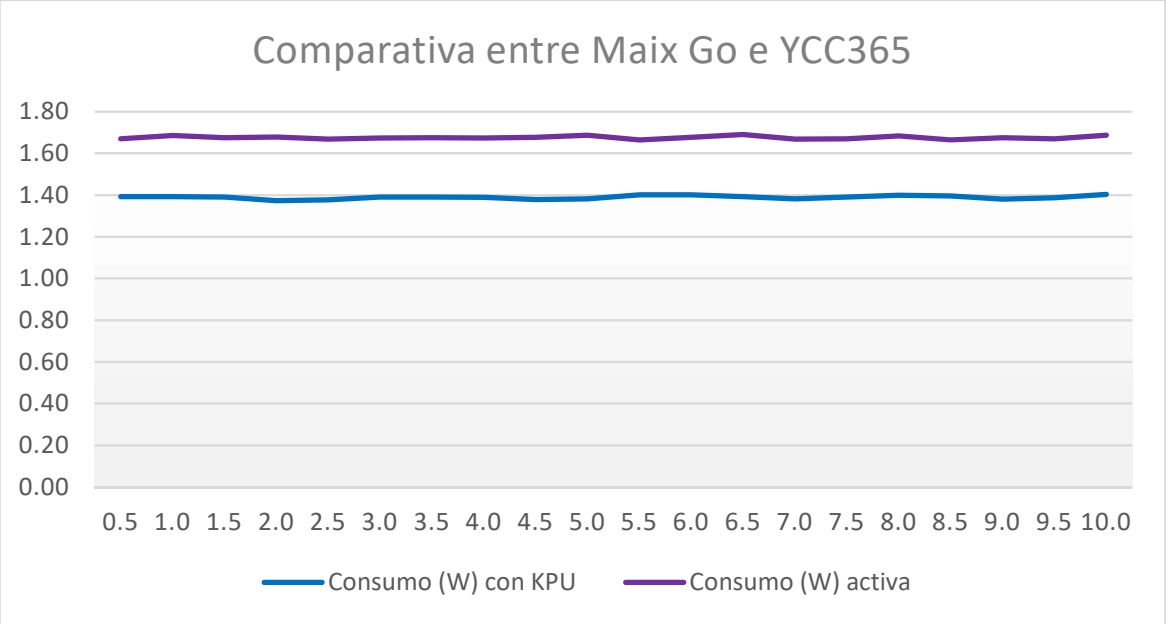


Figura 4.6: Comparativa entre cámara YCC365 en estado activo y Maix Go con KPU activa.

La figura 4.6 muestra una comparativa de los consumos de la placa de desarrollo SiPEED Maix Go mientras ejecuta reconocimiento de objetos y por tanto hace uso de la KPU, y la cámara YCC365 mientras se encuentra en un estado activo donde está realizando una grabación y a la vez retransmitiendo dicha grabación a la nube.

Incluso cuando se realiza una utilización completa de la capacidad de cómputo de la placa de desarrollo Maix Go, el consumo nunca supera al de la cámara YCC365, mostrando así el avance que puede suponer la salida al mercado de dispositivos cada vez de más bajo consumo, pero con más capacidad de cómputo.

Capítulo 5

Conclusiones

Este trabajo describe todos los elementos estudiados del dispositivo SiPEED Maix Go y su relación con la tecnología emergente del Edge Computing. Además, el estudio hace incapié en las capacidades AIoT del dispositivo, así como la arquitectura RISC-V, sobre la que está basado el SoC K210.

A lo largo del proceso de investigación, se ha encontrado que la madurez de una nueva plataforma, así como su nivel de adopción por parte del consumidor, pueden suponer grandes barreras para los nuevos desarrolladores que deseen iniciarse en ella.

En relación con el hardware que se ha utilizado para el desarrollo del proyecto, se ha encontrado que la documentación es escasa y que no hay un único lugar que la recoja toda. Sin embargo, el rendimiento que puede ofrecer, los precios bajos tanto del SoC K210 como de la placa Maix Go, así como la versatilidad que ofrecen sus componentes, podrían permitir en el futuro la generalización de hardware dedicado a Edge Computing.

En el caso de la placa de desarrollo SiPEED Maix Go, se ha encontrado que los entornos de desarrollo para la porgramación en C no se encuentran en un estado de madurez adecuado, y que la documentación, tanto de las capacidades del dispositivo como de las librerías de las que se hace uso, es insuficiente.

Además, las posibilidades de depuración, un elemento imprescindible cuando se desarrolla para cualquier plataforma, son insuficientes y en ocasiones inexistentes. Sólo existe una opción privativa y de pago que ofrece un entorno y una funcionalidad suficiente para realizar desarrollo simple para la plataforma en lenguaje C.

Sin embargo, las capacidades de inteligencia artificial del SoC K210, en conjunto con el port de Micro Python y el entorno Maix Py IDE desarrollados por SiPEED, crean un nuevo caso de uso para la plataforma. Tanto el IDE como el port de Micro Python son sencillos de utilizar para cualquier desarrollador que haya realizado uso de Python.

En conjunto, permiten crear aplicaciones versátiles como la cámara de

videovigilancia con reconocimiento de objetos, que si bien pueden verse restringidas por el tamaño de la memoria del dispositivo, ofrecen resultados suficientes para una aplicación en el EDGE. Además, estas aplicaciones resultan fáciles de desarrollar y se pueden lograr pruebas de concepto de una manera muy eficaz.

Asímismo, el entrenamiento de modelos IA y la conversión al formato KModel no son procesos complicados para un desarrollador que haya entrenado cualquier tipo de modelo con Keras y TensorFlow previamente. El rendimiento del dispositivo es además interesante si se considera que se trata de una nueva plataforma y de silicio reciente, llegando a alcanzar los 18 frames por segundo en detección de objetos.

Todo esto bajo un consumo similar o mejor al de dispositivos que se encuentran ya en el mercado, y que no realizan ningún tipo de procesamiento en el propio dispositivo, como la cámara YCC365 contra la que se han realizado las comparaciones de consumo y ancho de banda.

Dispositivos como este ofrecen además la posibilidad de liberar carga en los servidores cloud, así como uso de ancho de banda y transmisión de datos. Asimismo, debido a que los datos pueden ser procesados en el propio dispositivo, no son necesarios procesos de compresión, y por tanto estos pueden ser analizados con más detalle.

En definitiva, el Edge Computing es una tecnología que puede cambiar por completo el paradigma de la computación. Además, la arquitectura RISC-V también está emergiendo en los dispositivos de bajo consumo, permitiendo crear nuevos dispositivos cada vez más eficientes y más abiertos.

Este trabajo me ha enseñado que aunque una nueva plataforma pueda resultar muy interesante y tener un elevado potencial, a veces el hecho de ser nueva propicia una experiencia inadecuada para un desarrollador que se quiera iniciar en ella. Por otro lado, a veces puede llegar a ser imposible encontrar información acerca de algo que necesitas para continuar con el desarrollo, o incluso te puedes llegar a encontrar con problemas que nadie sabe resolver. Sin embargo, el poder recabar toda esta información permite ayudar a los desarrolladores futuros de la plataforma.

Este trabajo, que ha supuesto 527 horas de desarrollo (las cuales quedan desglosadas y explicadas por tareas en el Anexo B de este documento), fue finalmente sometido al congreso internacional DCIS 2020 como un artículo de seis páginas de extensión (el cual se encuentra en el Anexo A de este documento) con título: **Developing an AI IoT application with open software on a RISC-V SoC**.

Capítulo 6

Bibliografía

[1] R. Huang, J. Pedoeem, and C. Chen, "Yolo-lite: A real-time object detection algorithm optimized for non-gpu computers," *arXiv preprint 1811.05588*, 2018.

[2] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[3] Tensorflow lite landing page https://www.tensorflow.org/lite. [Online]. Available: https://www.tensorflow.org/lite

[4] Sipeed maixgo datasheet v1.1. [Online]. Available: https://dl.sipeed.com/MAIX/HDK/Sipeed-Maix-GO/Specifications/Sipeed%20MaixGo%20Datasheet%20V1.1.pdf

[5] (2020) Kendryte k210 web page. [Online]. Available: https://canaan.io/product/kendryteai

[6] (2020) Canaan: Landing page. [Online]. Available: https://canaan.io/

[7] A. Waterman, Y. Lee, D. Patterson, and K. Asanovic, "The risc-v instruction set manual, volume i: user-level isa, version 2.0, eecs department," *University of California, Berkeley, Tech. Rep. UCB/EECS-2014-54*, 2014.

[8] K210 datasheet. [Online]. Available: https://s3.cn-north-1.amazonaws.com.cn/dl.kendryte.com/documents/kendryte_datasheet_20181011163248_en.pdf

[9] Sipeed home page, https://www.sipeed.com. [Online]. Available: https://www.sipeed.com

[10] Esp8285 datasheet. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/0a-esp8285_datasheet_en.pdf

[11] Github page for the standalone sdk for kendryte k210. [Online]. Available: https://github.com/kendryte/kendryte-standalone-sdk

[12] (2020) Maixpy - micropython to k210. [Online]. Available: https://maixpy.sipeed.com/

[13] Github issue: Debugging maixgo on platformio using kendryte standalone sdk. [Online]. Available: https://github.com/sipeed/platform-kendryte210/issues/10#issuecomment-510744986

[14] Sipeed - blog: Platformio ide's debugging guide. [Online]. Available: https://blog.sipeed.com/p/727.html

[15] (2020) Olimex arm-usb-tiny-h. [Online]. Available: https://www.olimex.com/Products/ARM/JTAG/ARM-USB-TINY-H/

[16] (2020) J-link debug probes. [Online]. Available: https://www.segger.com/products/debug-probes/j-link/

[17] Github page for sysprogs' modified openocd. [Online]. Available: https://github.com/sysprogs/openocd-kendryte

[18] (2020) Micro python web page. [Online]. Available: https://micropython.org/

[19] R. Girshick and M. Research, "Fast r-cnn," *arXiv preprint arXiv:1504.08083*, 2015.

[20] Caffe deep learning framework landing page. [Online]. Available: https://caffe.berkeleyvision.org/

[21] Paddlepaddle deep learning framework landing page. [Online]. Available: https://www.paddlepaddle.org.cn/

[22] Onnx open machine learning model format landing page. [Online]. Available: https://onnx.ai/

[23] Github page for nncase: Open deep learning compiler stack for kendryte k210. [Online]. Available: https://github.com/kendryte/nncase

[24] Github page for axelerate: Keras-based framework for ai on the edge. [Online]. Available: https://github.com/AIWintermuteAI/aXeleRate

[25] (2007) Visual object classes challenge 2007. [Online]. Available: http://host.robots.ox.ac.uk/pascal/VOC/voc2007/

[26] Visual object classes challenge 2012. [Online]. Available: http://host.robots.ox.ac.uk/pascal/VOC/voc2012/

[27] Covid-19: Face mask detector with opencv, keras/tensorflow, and deep learning. [Online]. Available: https://www.pyimagesearch.com/2020/05/04/covid-19-face-mask-detector-with-opencv-keras-tensorflow-and-deep-learning/

[28] (2007) Amazon.es: Cámara de vigilancia 1080p. [Online]. Available: https://www.amazon.es/Vigilancia-Inclinaci%C3%B3n-Detecci%C3%B3n-Movimiento-Seguridad/dp/B07MS85YLT

[29] (2020) Joulescope: Precision dc energy analyzer. [Online]. Available: https://www.joulescope.com/products/joulescope-precision-dc-energy-analyzer

[30] (2020) Nvidia geforce gtx 1060 specifications. [Online]. Available: https://www.geforce.co.uk/hardware/desktop-gpus/geforce-gtx-1060/specifications

[31] (2020) Nvidia geforce gtx 1060 review. [Online]. Available: https://www.techspot.com/review/1209-nvidia-geforce-gtx-1060/page8.html#:~:text=Power%20Consumption%20%26%20Temperatures,-The%20following%20power&text=The%20GTX%201060%20allowed%20for,improvement%20over%20the%20GTX%20980.

[32] Kendryte home page, https://kendryte.com/. [Online]. Available: https://kendryte.com/

[33] European comission, coordinated plan on the development and use of artificial intelligence made in europe – 2018. [Online]. Available: https://ec.europa.eu/digital-single-market/en/news/coordinated-plan-artificial-intelligence

[34] Tensorflow lite. [Online]. Available: https://www.tensorflow.org/lite/

[35] (2020) Platformio: A new generation ecosystem for embedded development. [Online]. Available: https://platformio.org/

[36] (2020) Visualgdb - serious cross-platform support for visual studio. [Online]. Available: https://visualgdb.com/

Lista de Figuras

Anexos

Anexos A

Artículo sometido al congreso internacional DCIS 2020

Developing an AI IoT application with open software on a RISC-V SoC

1st Enrique Torres-Sánchez 2nd Jesús Alastruey-Benedé 3rd Enrique Torres-Moreno

***Abstract*—RISC-V is an emergent architecture that is gaining strength in low-power IoT applications. The stabilization of the architectural extensions and the start of commercialization of RISC-V based SOCs, like the Kendryte K210, raises the question of whether this open standard will facilitate the development of applications in specific markets or not.**

In this paper we evaluate the development environments, the toolchain, the debugging processes related to the Sipeed MAIX Go development board, as well as the standalone SDK and the Micropython port for the Kendryte K210. The training pipeline for the built-in convolutional neural network accelerator, with support for Tiny YOLO v2, has also been studied. In order to evaluate all the above aspects in depth, a low-cost, low-power, IoT edge application based on AI has been developed. In the context of the current COVID-19 pandemic, the application is capable of labeling whether a pedestrian is wearing a face mask or not, doing real-time object recognition at a mean rate of 13 FPS. Throughout the process, we can conclude that, despite the potential of the hardware and its excellent performance/cost ratio, the documentation for developers is scarce, the development environments are in low maturity levels, and the debugging processes are sometimes nonexistent.

***Index Terms*—RISC-V, IoT, AI, AIoT, Kendrite K210, Sipeed MAIX, CNN Hardware Accelerators**

I. Introduction

Recent advances in embedded systems and artificial intelligence (AI), in combination with lower manufacturing costs and end-consumer prices, have led to the **RISC-V** architecture being in the spot for embedded system development, research, and education. This has also brought AI and deep learning to low-power systems, with new silicon being added to chips, such as neural networks hardware accelerators.

Deep learning has achieved many successes in various domains. Recently, researchers and engineers make efforts to apply AI algorithms to mobile or **embedded devices**. Machine vision and hearing has gained popularity due to new low cost and low power solutions. Object detection, image classification (face detection and recognition, obtaining size and coordinates, etc.) can be done in real time.

The **AIoT** term is where AI and IoT meet, bringing intelligence to the edge. As AI moves closer to the edge and into devices, such as smart sensors and cameras. In many cases it eliminates the need for racks of cloud-based computing and instead moves the analysis to the IoT device itself, reducing bandwidth and removing any delay in loosely connected devices or when latency is critical.

AI is pervasive today, from consumer to enterprise applications. With the rapid growth of connected devices, combined with a demand for privacy/confidentiality, low latency and bandwidth constraints, AI models trained in the cloud increasingly need to be run at the edge.

The European Commission published in 2018 its Artificial Intelligence Plan under title "Coordinated Plan on the Development and Use of Artificial Intelligence Made in Europe – 2018" [1]. It emphasizes the importance of **university education** in different aspects of AI and highlights the importance of moving concepts from the theoretical framework to the real world of embedded systems as a priority in the context of the transformation of the industry. These systems must have enough computing power to read their sensors and extract information by analyzing their environment and taking actions, or through connectivity, share it with other intelligent devices.

Low-cost development boards with enough computing power to run real-time object detection systems such as Tiny-YOLO [2], MobileNet-v1 [3], and TensorFlow Lite [4] are emerging. The usefulness of these systems in education of both undergraduate courses in Computer Architecture and advanced courses in Embedded Systems or Machine Learning depends greatly on the maturity of the programming tools and available documentation.

This paper is organized as follows: Section 2 introduces the Sipeed MAIX Go development board and the RISC-V SoC used. Section 3 describes the software development environments available for the platform focusing particularly on debugging tools. Section 4 then combines the results from the previous section to implement a practical AI embedded application able to recognize and label whether a pedestrian is wearing a face mask or not. Finally, Section 5 concludes.

II. Hardware

The Sipeed MAIX-Go development board [6] based on the MAIX M1w module with the Kendryte SoC K210 [7] was selected for this study due to its low cost and availability. In this section we will describe its hardware components.

A. System-on-Chip (SoC)

The **Kendryte K210** (**K210**) is a system-on-chip (SoC) implemented in TSMC ultra-low-power 28-nm advanced process that integrates machine vision and machine hearing [5]. It targets the embedded AI and IoT markets, but it can be also used as a high-performance MCU. Released in Sep. 2018, its current batch price is around $6.

Its main components are a dual-core **RISC-V 64-bit processor** (CPU), a convolutional neural network (**CNN**) hardware accelerator (KPU), an audio accelerator (APU), hardcore FFT,

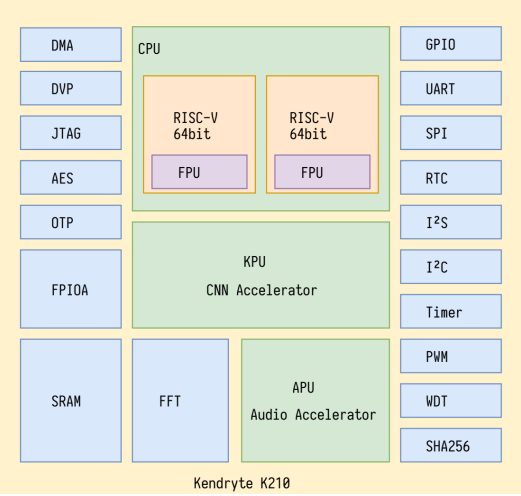


Fig. 1. K210 SoC block diagram [5].

SHA256, and a 8 MiB static random-access memory (SRAM). The SRAM is split into two parts: 6 MiB are devoted for general-purpose computation and 2 MiB for the KPU. The AI SRAM can be used as main memory if the KPU is not using it. Figure 1 shows the block diagram of the K210 SoC.

The CPU cores implement the RISC-V 64-bit IMAFDC ISA (**RV64GC**) [8]. This instruction set is suitable for general tasks and provides support for different privilege levels to improve safety and advanced interrupt management routeable to any of the two cores for better power efficiency, stability and reliability.

Each CPU core contains an IEE754-2008 compliant floating-point unit (FPU) that supports single- and double-precision multiply, divide, and square-root operations. Each core also has a 32 KiB instruction cache and a 32 KiB data cache. Frequency can be adjusted from the nominal 400 MHz up to 800 MHz. Power consumption is kept below 350 mW when running face detection routine and 35 mW with both cores in WFI mode (*Wait For Interrupt instruction*).

The **Knowledge Processing Unit** (KPU) is a general-purpose neural network processor with built-in convolution (1x1 and 3x3 kernels), batch normalization, activation (e.g. ReLU, sigmoid), and pooling operations (e.g. max, average). There is no direct limit on the number of network layers. Each CNN layer can be configured separately, including the number of input and output channels, and the input and output line width and column height. It supports a fixed-point model. It reaches a peak performance of 0.25 TOPS (0.5 TOPS overclocked) executing 16-bit multiplications from its 64 KLU (576 bit SIMD datapath). Real time at $\geq$ 30 fps can be achieved if the size of neural network parameters is kept below 5.9 MiB. Available flash capacity is the limit in non real-time applications. The KPU flow diagram is shown in Figure 2.

The **APU pre-processing** module is responsible for the pre-processing of voice direction and voice data output. With up to 8 channels of audio input data, it is able to implement a

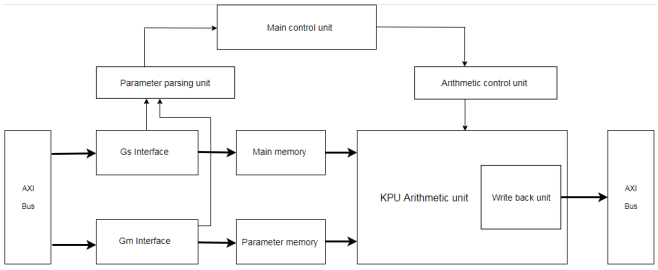


Fig. 2. Knowledge Processing Unit (KPU) flow [5].

mic array, simultaneous scanning, pre-processing, and beam-forming for sound sources in up to 16 directions. It uses the built-in FFT and the system DMAC to store output data in system memory.

The DVP camera interface module supports cameras with a maximum frame size 640x480 at 30 fps or 60 fps at QVGA. It can output images to both KPU and through the MCU interface to an LCD display. The flexible FPIOA (Field Programmable IO Array) can map 255 functions to all 48 GPIOs on the chip from many other accelerators and peripherals: UART, WDT, IIC, SPI, I2S, TIMER, RTC, PWM, etc.

The K210 embeds AES and SHA256 algorithm accelerators to provide users with basic security features. One-time programmable memory unit (OTP) and the AES accelerator allows firmware encryption and integrity check for tamper resistance support.

Debugging is supported by a JTAG interface and a high-speed UART. It allows hardware breakpoints and Debug Mode operation and has monitoring registers limited to count instructions and execution cycles.

B. MAIX-I module

Sipeed MAIX-I [6], or called M1, integrates the K210, DC/DC power supply circuit, 8MB/16MB/128MB Flash (M1w add ESP8285 Wi-Fi chip) into a 1x1 inch breadboard-friendly and also SMT-able module. Its target is the Artificial Intelligence of Things (AIoT) edge computing solution. Crowdfunded in Indiegogo, Sipeed successfully reached a 428% of the initial goal.

Figure 3 shows a picture of the M1 module. The K210 is located in the top-left, next to it is the ESP8285 and the IPX antenna connector. The 16 MiB flash chip can be seen in the bottom left.

Espressif's **ESP8285** is a highly integrated low-power SoC with the complete and self-contained Wi-Fi networking capabilities, in this case working as a slave to the host MCU [9].

C. MAIX Go Development Board

Sipeed MAIX Go (Figure 4) is a 88x60 mm development board built around the M1w module. Its small physical size, low-power footprint and low cost make it really appealing for developing and learning IA embedded systems. The on-board USB type C connector can be used for powering as well as UART. The **JTAG** connector may be employed for uploading

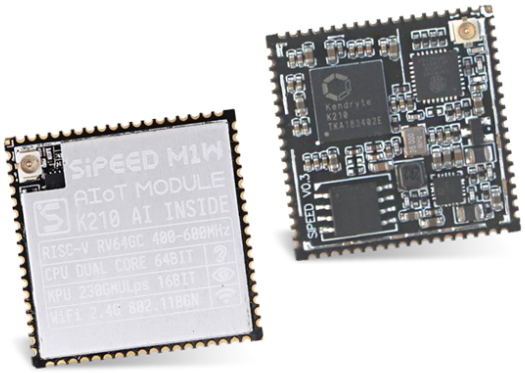


Fig. 3. SiPEED MAIX-1 Wi-Fi module [6].

code, debugging or communication. JTAG support is based on a STM32F103C8 so there is no need for an external Jlink.

The development board also comes with a 3-axis digital accelerometer, a I2S Mic, a speaker, a RGB LED, a Mic array connector, a three-way thumbwheel, a TF card Slot, a lithium battery manager chip with power path management function, and all the K210 GPIO pins available. The full suite, with a street price of $40, comes with a 500 mAH lithium-ion battery, a 2.8 inch LCD, a ov2640 with M12 lens DVP camera, a Wi-Fi antenna, and a simple acrylic case.

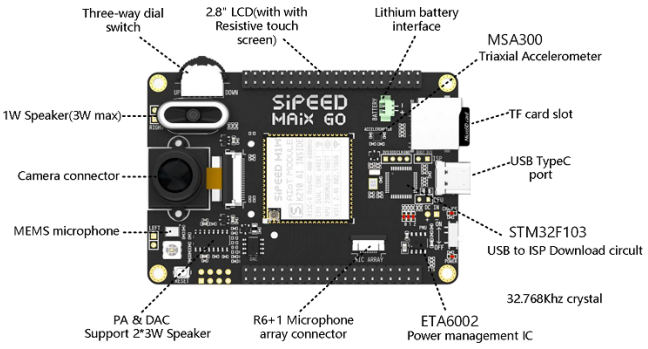


Fig. 4. Sipeed MAIX Go development board [10].

III. Software

During the development process of an application, one of the most important steps is selecting the toolchain. Both the integrated **development environment** (IDE) and the **debugging** tools are key choices when developing code on a platform. For newer platforms, the software and the libraries may be very limited, and/or not mature. Thus, it might have bugs due to its early stage of development, or it might have poor documentation.

In the case of the RISC-V based K210, a developer can either choose between a) one of the few existent IDEs for the C programming language or b) the **MicroPython** port provided by Sipeed in case of using a development board based on a MAIX module.

The options for developing in C for the K210 are:

- **PlatformIO** [11]: free, open source, cross-platform IDE based on Visual Studio Code that is catered towards embedded development.
- **VisualGDB** [12]: proprietary add-on for Visual Studio.
- Plain Kendryte Standalone SDK [13].

In all three cases, the aforementioned Kendryte Standalone SDK is used, which is a set of C tools and libraries for developing software to run on a K210 without operating system. The standalone SDK is open source and its code is completely hosted on GitHub [13].

It is also possible to use **FreeRTOS** with a different toolchain but we have not tested this option.

In case of having a Sipeed MAIX development board and choosing the **MicroPython** port, the options are the following:

- Using the open source IDE provided by Sipeed, called **MaixPy IDE** [14].
- Using the serial port **MicroPython terminal** that the MaixPy MicroPython port provides over the development board's USB connection.
- Loading MicroPython code files **from the SD** card that can be inserted into Sipeed development boards, from either the IDE or the MicroPython terminal.

Both PlatformIO and VisualGDB have been tested, and MaixPy IDE has been used to develop the object recognition software that will be presented in a following section. Not using the IDE is also a path that was taken in order to check the possibility of improving the debugging process.

A. Low-level and C development in PlatformIO

Using the Sipeed MAIX Go development board initially started as a research project which was focused on testing whether the development board could be used as a base for undergraduate students to learn working on bare metal hardware, specifically on RISC-V based hardware. The testing phase started by analyzing **PlatformIO** as a viable option for an easy to use IDE, which let the students both abstract themselves from the linking, compiling and mapping process, and at the same time presented an easy to use debugging interface where students could look into assembly code, as well as the architectural state of the system.

The Sipeed MAIX Go on board JTAG and UART should allow uploading and debugging through the USB port. When this was tested, the PlatformIO debugger did not work. Initially, this was thought to be an error with PlatformIO universal debugger. In order to eliminate some possible variables, PlatformIO was tested both in Windows and in Ubuntu operating systems, with the proper drivers installed. In both cases, the on-board JTAG debugger did not connect with PlatformIO universal debugger. After some research, one blog post comment made by a Sipeed developer was found mentioning that the on-board debugging probe was not fully active as its firmware development was not finished yet [15]. This

issue was mentioned neither in the documentation nor in the specifications sheet.

After this problem was encountered, other methods were pursued in order to debug the K210 via a debugging probe. Some research led to a developer blog post indicating that shorting two pins on the board allowed it to switch to DFU mode, which permitted external debugging probes [16]. This was tested to be true by using an **Olimex-Tiny-H** external debugging JTAG with the proper pinout set up on the board, as well as the USB driver installed in Ubuntu. With this external probe, the board in DFU mode, and the correct board specification in PlatformIO, the initial test code, which was a sleep-based LED blink, could be debugged.

The debugger, which is based on **OpenOCD** but modified to support the K210, was found to have poor performance. In addition to this, PlatformIO was able to read the processor physical registers but not to dissasembly code in real time. Even though the developer could go through the C code step by step, following the execution path, the same could not be performed at the assembly level (RISC-V machine instructions). Another main issue of the Olimex-Tiny-H was the inability to upload code through the USB port. The development board had first to be started without DFU mode to upload the code through USB using either Kendryte provided tools for code flashing, or PlatformIO's uploader, which ends up using the same tool but abstracts the user from using the command line.

After successfully debugging the initial test code, a new version of the LED blink code based on timer interrupts was tested. This second test code did not result in a successful debugging procedure. The code executed correctly and provided the expected functionality, but when the debugging probe was started, the code did not stop at any break point, and the program running on the K210 crashed. The problem was initially theorized to be linked to the Olimex-Tiny-H JTAG, and a **SEGGER'S J-LINK** was acquired to test this hypothesis. Using the J-LINK, PlatformIO was unable to connect to the SoC. This was tested in Ubuntu and in Windows, with the appropriate drivers installed, but it did not function in any of the two platforms.

B. Low-level and C development in VisualGDB

In order to eliminate possible failure points, another IDE was tested. In this case, the free trial for **VisualGDB** was used to test both its debugging capabilities via the external debugging probes, as well as the possibilities that the IDE offered for disassembly and memory look ups. The VisualGDB add-on for Visual Studio was found to be successful at debugging with the Olimex-Tiny-H, but not with the J-LINK. Furthermore, VisualGDB was able to inject code to RAM memory directly using the JTAG interface, so no further procedures like the ones used for PlatformIO were needed to upload the code and debug it. On the other hand, VisualGDB was not able to retrieve the physical registers from the cores, so during the debugging process the developer is not able to analyze the architectural state of the machine.

TABLE I
COMPARISON OF PLATFORMIO AND VISUALGDB FEATURES.

	PlatformIO		VisualGDB	
	On Board	Olimex	On Board	Olimex
Upload	Yes	No	No	Yes
Step by Step	No	Yes	No	Yes
Multicore Brk	No	No	No	Yes
RISC-V ASM	No	No	No	No
Register Values	No	Yes	No	No

In addition to the results mentioned above, VisualGDB was able to correctly debug the timer interrupt based LED blink. Further analysis showed that VisualGDB had a modified version of the OpenOCD. This version of the software was able to set breakpoints on both cores of the K210. The explanation for the fix is given by Sysprogs, the developers of VisualGDB, in a GitHub page [17] where the modified version of Kendryte OpenOCD is uploaded. The original software requires the developer to select which core will be probed at the start of the session. This causes an issue when a breakpoint is triggered on the SoC other core, and the debugging session will crash instead of the breakpoint being acknowledged by OpenOCD. Sysprogs fixed this by modifying the polling logic to check the status of both cores and automatically switching to debugging the core that triggered the last breakpoint. On the other hand, this causes OpenOCD to fail when polling for the contents of the physical registers, which is the result that was observed when debugging the code with VisualGDB.

C. Observations

Both the Sipeed MAIX Go development board and the K210 have been seen as not mature enough in order for them to be viable as educational devices in computer architecture courses. The toolchain has some bugs which sometimes completely disrupt the development process. The development environments are not mature enough for a teacher to base any type of educational assignment on the platform, and they lack features that are deemed mandatory in most of the fields in which the development board could be used as an educational tool. E.g., VisualGDB can debug any software programmed in C for the K210, even if the code uses both cores when running, but it can not analyze the architectural state of the cores, whereas PlatformIO is able to manage the latter (Table I). Furthermore, the most viable option for development and debugging on the platform in C language is proprietary and requires funding, which would make it inaccessible to some students.

IV. APPLICATION DEVELOPMENT UNDER MAIXPY

Following the testing phase as a viable educational tool, the AIoT capabilities of the platform were tested. Specifically, the object detection capabilities of the K210 integrated hardware accelerator.

Even though the Kendryte standalone SDK libraries support C programming and using the KPU for object detection or classification, the abstraction that the Sipeed MicroPython port

provides to the SoC machine vision components gives the developer an appealing alternative that makes both prototyping and development more efficient. This comes at a cost, as the required MicroPython interpreter and the MicroPython libraries reduce the amount of RAM available for model parameter loading. In the case of the developed application, MicroPython port **MaixPy** was chosen as the programming language because of the advantages mentioned above.

A. Development Process

The proposed application is an AIoT domestic surveillance camera which is capable of recognizing movement in a house and autonomously identify whether it was caused by a human or by a house pet, like for example a dog or a cat, using object detection with the YOLO v2 object detection backend. Then, the camera would only send real-time video when it has recognized a human, instead of reacting to any type of movement, and thus saving bandwidth and storage. In order to achieve this, several steps were taken.

Firstly, the toolchain was set up. When using MaixPy, different versions of the port can be chosen, with different components being available or not through importing libraries. In order to minimize the RAM memory that the MicroPython interpreter uses, the 0.5.0 minimum version of MaixPy has been used for the application, with IDE support. The other options come with LVGL support, which consumes more RAM, or without IDE support. The IDE support was chosen as a feature because it provides an easy to use environment to write code, link to the MicroPython interpreter through the serial port, and directly allows code execution on the development board. Furthermore, if the the camera sensor is used, its images will be saved to a frame buffer on the RAM, which is then directly shown on the IDE. This frame buffer can be modified during run time, allowing the developer to draw bounding boxes around detected objects, as well as text, and it will be displayed in the IDE window. The IDE is cross platform, and it was tested for both Windows and Linux.

Secondly, the K210 was discovered to use a specific binary file format for neural network weights. This file format could be obtained by converting any TFLite [4], Caffe [18], PaddlePaddle [19] or ONNX [20] based network using the tool NNCASE, which is open source [21] and is given by Kendryte. The binary file obtained is referred to as KModel or K210Model, and it is a compiled weights format optimized to run on the K210 KPU. A training pipeline based on Tensorflow and Keras can be developed, which in the end outputs a TFLite file. This TFLite file can then be converted to a KModel file using NNCASE, which in turn can be loaded into RAM and executed by the KPU. For the sake of the development of this application, a working training pipeline was found on GitHub, called aXeleRate [22]. This software provides a training pipeline that can directly export to KModel format. It uses the Pascal VOC annotation format to tag classes in an image, and assign a bounding box to the object contained in the image.

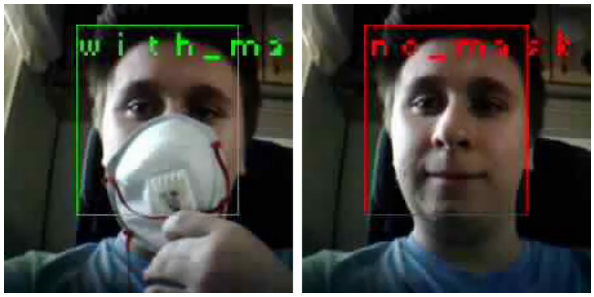


Fig. 5. Mask detection application sample images. Left side shows a correct mask placement detection. Right side shows the opposite.

The first network models were compiled using a human data set to test the real-time object recognition capabilities of the K210 KPU. Initial tests proved that the object recognition was accurate enough for the intended application, and the performance, with a mean of 13 frames per second, was enough for a surveillance camera. This performance was achieved by using the camera sensor to capture 224 x 224 pixels photos, which would then be ran on the KPU with the pre-trained model. Once the KPU returns the recognized bounding boxes, this are drawn on the frame buffer, followed by a string that specifies what object class is recognized.

After having a complete toolchain, with a working training pipeline, a human, dog and cat classes data set was gathered from the VOC 2012 competition [23]. The data set consisted of 2274 images, evenly distributed between all the different included classes. It was used to train different models with different numbers of epochs. Specifically, two models were trained: one with 50 epochs and one with 250 epochs. The precision of the models could be inferred using aXeleRate's inferring script, and it was found to be 67% and 70% for the first and second model, respectively. A testing script was programmed to check the accuracy of the model once it was compiled into the KModel file format. The script while running the second model on the KPU was able to recognize all the objects in the given validation images, which was the same subset of images used to infer the precision of the model previous to the conversion.

Finally, in the context of the COVID-19 pandemic, a data set was published by PyImageSearch [24]. It contained tagged pictures of people wearing a mask, some correctly and some incorrectly. This, in connection with the research that was on going about the viability of the Sipeed MAIX Go as an educational tool, motivated the development of a surveillance camera application to check whether a person was wearing a mask correctly or not (Figure 5).

The data set was not provided with Pascal VOC annotation files, so Intel's VOTT software was used to tag and assign bounding boxes to the given data set manually, and to export the data set with these annotations. A model was trained using the same training pipeline as the one used in the human-pet model. When the precision was inferred prior to the conversion to KModel format, the model was found to have a precision of

99%. This model was then loaded by the previous surveillance camera application. The camera application was then able to recognize when a person was correctly wearing a mask or not, with a high degree of precision if the camera was taking images in good lightning conditions.

B. Observations

The surveillance camera application proved to be a viable concept, and also proved that MaixPy is suitable for embedded AI teaching. It was also found to have a generic purpose, as depending on the model and the classes given to it, it could also function as a mask detector. On the other hand, the hardware that comes with the Sipeed MAIX Go development board was found to be insufficient for a real world use, as the camera sensor is only suitable on good-lightning conditions. The K210 is also very limited in terms of RAM memory, as the pre-trained models that can be run on the KPU are small and thus take some precision compromises.

V. Conclusions

This paper describes the initial process that a developer needs to follow to develop software for a SoC implementation of the RISC-V architecture, focusing on the Kendryte K210. Its toolchain, its ease of use from a developer's perspective and its real-time AI capabilities have also been the main concepts that have been put to test.

Throughout the initial research process, where the focus of the study was on using the platform as an educational tool, some issues were found. Even though the performance of the SoC is high, taking into account its low power consumption and its low cost, the development toolchain for the platform is not ready for these purposes. Although available IDEs do provide code auto completion when programming in C, the debugging capabilities are not complete. Both PlatformIO and VisualGDB fail to provide low-level debugging.

The two IDEs use a version of OpenOCD that is optimized by Kendryte for the K210 SoC, which has bugs and is missing features like run-time disassembly which are necessary for embedded systems labs. Furthermore, VisualGDB is proprietary and needs a paid license, which makes the IDE not viable for widespread student use. The documentation for the platform is also scarce, and the libraries given by Kendryte are obscure and documentation could be improved. All in all, the C language toolchain is not yet fully ready, as can be expected from a new platform without widespread use.

On the other hand, the AI capabilities of the SoC, in conjunction with Sipeed's port of MicroPython for its development boards, brings a new use to the platform. The MicroPython port and the MaixPy IDE are easy to use for a developer. Training a model and converting the trained weights to the KModel file type is not a complicated task for a developer that has trained any type of AI model with Keras or TensorFlow, and the performance of the hardware accelerator is more than acceptable, averaging at 13 fps when doing real-time object recognition and running the developed surveillance camera application.

With all the downsides that the C language toolchain has, the low cost of Sipeed development board, the K210 low power consumption even under heavy load and the AI capabilities of the SoC, make the platform an interesting opportunity for IoT to become more intelligent and efficient in the near future.

The surveillance camera application will be open-sourced on GitHub to help future developers.

Red-RISCV por promover el "Open Hardware"? The RISC-V ISA - European Processor Initiative

References

[1] European comission, coordinated plan on the development and use of artificial intelligence made in europe – 2018. [Online]. Available: https://ec.europa.eu/digital-single-market/en/news/coordinated-plan-artificial-intelligence

[2] R. Huang, J. Pedoeem, and C. Chen, "Yolo-lite: A real-time object detection algorithm optimized for non-gpu computers," in *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 2503–2510.

[3] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[4] Tensorflow lite. [Online]. Available: https://www.tensorflow.org/lite/

[5] K210 datasheet. [Online]. Available: https://s3.cn-north-1.amazonaws.com.cn/dl.kendryte.com/documents/kendryte_datasheet_20181011163248_en.pdf

[6] Sipeed home page, https://www.sipeed.com. [Online]. Available: https://www.sipeed.com

[7] Kendryte home page, https://kendryte.com/. [Online]. Available: https://kendryte.com/

[8] A. Waterman, Y. Lee, D. Patterson, and K. Asanovic, "The risc-v instruction set manual, volume i: user-level isa, version 2.0, eecs department," *University of California, Berkeley, Tech. Rep. UCB/EECS-2014-54*, 2014.

[9] Esp8285 datasheet. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/0a-esp8285_datasheet_en.pdf

[10] Sipeed maixgo datasheet v1.1. [Online]. Available: https://dl.sipeed.com/MAIX/HDK/Sipeed-Maix-GO/Specifications/Sipeed%20MaixGo%20Datasheet%20V1.1.pdf

[11] (2020) Platformio: A new generation ecosystem for embedded development. [Online]. Available: https://platformio.org/

[12] (2020) Visualgdb - serious cross-platform support for visual studio. [Online]. Available: https://visualgdb.com/

[13] Github page for the standalone sdk for kendryte k210. [Online]. Available: https://github.com/kendryte/kendryte-standalone-sdk

[14] (2020) Maixpy - micropython to k210. [Online]. Available: https://maixpy.sipeed.com/

[15] Github issue: Debugging maixgo on platformio using kendryte standalone sdk. [Online]. Available: https://github.com/sipeed/platform-kendryte210/issues/10#issuecomment-510744986

[16] Sipeed - blog: Platformio ide's debugging guide. [Online]. Available: https://blog.sipeed.com/p/727.html

[17] Github page for sysprogs' modified openocd. [Online]. Available: https://github.com/sysprogs/openocd-kendryte

[18] Caffe deep learning framework landing page. [Online]. Available: https://caffe.berkeleyvision.org/

[19] Paddlepaddle deep learning framework landing page. [Online]. Available: https://www.paddlepaddle.org.cn/

[20] Onnx open machine learning model format landing page. [Online]. Available: https://onnx.ai/

[21] Github page for nncase: Open deep learning compiler stack for kendryte k210. [Online]. Available: https://github.com/kendryte/nncase

[22] Github page for axelerate: Keras-based framework for ai on the edge. [Online]. Available: https://github.com/AIWintermuteAI/aXeleRate

[23] Visual object classes challenge 2012. [Online]. Available: http://host.robots.ox.ac.uk/pascal/VOC/voc2012/

[24] Covid-19: Face mask detector with opencv, keras/tensorflow, and deep learning. [Online]. Available: https://www.pyimagesearch.com/2020/05/04/covid-19-face-mask-detector-with-opencv-keras-tensorflow-and-deep-learning/

Anexos B

Dedicación de tiempo: Diagrama de Gantt

Diagrama Gantt

Project Start: 20/01/2020

Enrique Torres

TASK	START	END	DAYS
Fase de investigación			
Estudio de la arquitectura RISC-V	20/01/2020	30/01/2020	11
Análisis de oferta de mercado placas RISC-V	31/01/2020	10/02/2020	11
Selección del dispositivo a estudiar	11/02/2020	14/02/2020	4
Estudio de entornos de desarrollo y viabilidad educativa			
Investigación y documentación del hardware	14/02/2020	20/02/2020	7
Estudio herramientas de desarrollo de Kendryte	21/02/2020	04/03/2020	13
Preparación del pipeline de desarrollo para C	05/03/2020	07/03/2020	3
Evaluación IDE PlatformIO con distintos JTAGs	08/03/2020	24/03/2020	17
Evaluación IDE VisualGDB con distintos JTAGs	22/03/2020	30/03/2020	9
Desarrollo scripts de prueba en C	08/03/2020	01/04/2020	25
Micro Python, pipeline IA y aplicación prueba de concepto			
Puesta a punto del pipeline de entrenamiento	02/04/2020	17/04/2020	16
Preparación pipeline de Micro Python	17/04/2020	20/04/2020	4
Búsqueda de datasets	21/04/2020	29/04/2020	9
Entrenamiento de modelos	30/04/2020	06/05/2020	7
Desarrollo prueba de concepto y prueba de modelos	07/05/2020	11/05/2020	5
Dataset mascarillas y modelo COVID-19	09/05/2020	13/05/2020	5
Escritura artículo, evaluación app y dispositivo, escritura de la memoria			
Redacción artículo para congreso DCIS 2020	14/05/2020	22/05/2020	9
Evaluación versatilidad aplicación y dispositivo	23/05/2020	25/05/2020	3
Aprendizaje Joulescope y mediciones de consumo	26/05/2020	02/06/2020	8
Mediciones ancho de banda y comparativa	03/06/2020	05/06/2020	3
Redacción memoria	03/06/2020	22/06/2020	20

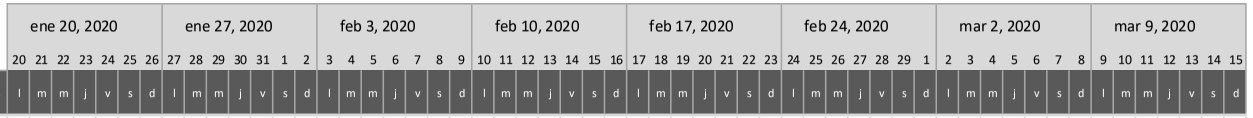


Diagrama Gantt

Project Start: 20/01/2020

Enrique Torres

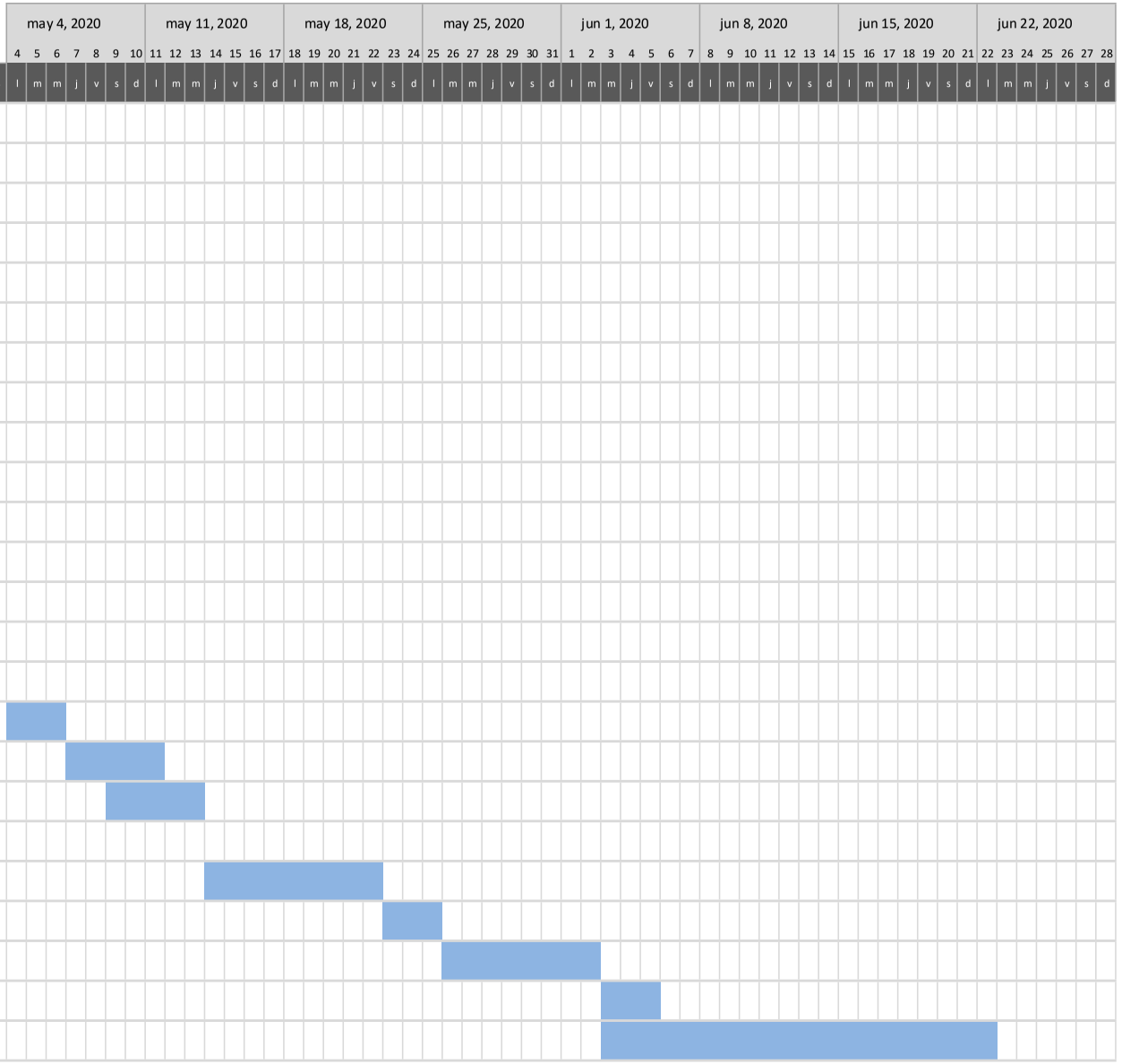
TASK	START	END	DAYS
Fase de investigación			
Estudio de la arquitectura RISC-V	20/01/2020	30/01/2020	11
Análisis de oferta de mercado placas RISC-V	31/01/2020	10/02/2020	11
Selección del dispositivo a estudiar	11/02/2020	14/02/2020	4
Estudio de entornos de desarrollo y viabilidad educativa			
Investigación y documentación del hardware	14/02/2020	20/02/2020	7
Estudio herramientas de desarrollo de Kendryte	21/02/2020	04/03/2020	13
Preparación del pipeline de desarrollo para C	05/03/2020	07/03/2020	3
Evaluación IDE PlatformIO con distintos JTAGs	08/03/2020	24/03/2020	17
Evaluación IDE VisualGDB con distintos JTAGs	22/03/2020	30/03/2020	9
Desarrollo scripts de prueba en C	08/03/2020	01/04/2020	25
Micro Python, pipeline IA y aplicación prueba de concepto			
Puesta a punto del pipeline de entrenamiento	02/04/2020	17/04/2020	16
Preparación pipeline de Micro Python	17/04/2020	20/04/2020	4
Búsqueda de datasets	21/04/2020	29/04/2020	9
Entrenamiento de modelos	30/04/2020	06/05/2020	7
Desarrollo prueba de concepto y prueba de modelos	07/05/2020	11/05/2020	5
Dataset mascarillas y modelo COVID-19	09/05/2020	13/05/2020	5
Escritura artículo, evaluación app y dispositivo, escritura de la memoria			
Redacción artículo para congreso DCIS 2020	14/05/2020	22/05/2020	9
Evaluación versatilidad aplicación y dispositivo	23/05/2020	25/05/2020	3
Aprendizaje Joulescope y mediciones de consumo	26/05/2020	02/06/2020	8
Mediciones ancho de banda y comparativa	03/06/2020	05/06/2020	3
Redacción memoria	03/06/2020	22/06/2020	20

Diagrama Gantt

Project Start: 20/01/2020

Enrique Torres

TASK	START	END	DAYS
Fase de investigación			
Estudio de la arquitectura RISC-V	20/01/2020	30/01/2020	11
Análisis de oferta de mercado placas RISC-V	31/01/2020	10/02/2020	11
Selección del dispositivo a estudiar	11/02/2020	14/02/2020	4
Estudio de entornos de desarrollo y viabilidad educativa			
Investigación y documentación del hardware	14/02/2020	20/02/2020	7
Estudio herramientas de desarrollo de Kendryte	21/02/2020	04/03/2020	13
Preparación del pipeline de desarrollo para C	05/03/2020	07/03/2020	3
Evaluación IDE PlatformIO con distintos JTAGs	08/03/2020	24/03/2020	17
Evaluación IDE VisualGDB con distintos JTAGs	22/03/2020	30/03/2020	9
Desarrollo scripts de prueba en C	08/03/2020	01/04/2020	25
Micro Python, pipeline IA y aplicación prueba de concepto			
Puesta a punto del pipeline de entrenamiento	02/04/2020	17/04/2020	16
Preparación pipeline de Micro Python	17/04/2020	20/04/2020	4
Búsqueda de datasets	21/04/2020	29/04/2020	9
Entrenamiento de modelos	30/04/2020	06/05/2020	7
Desarrollo prueba de concepto y prueba de modelos	07/05/2020	11/05/2020	5
Dataset mascarillas y modelo COVID-19	09/05/2020	13/05/2020	5
Escritura artículo, evaluación app y dispositivo, escritura de la memoria			
Redacción artículo para congreso DCIS 2020	14/05/2020	22/05/2020	9
Evaluación versatilidad aplicación y dispositivo	23/05/2020	25/05/2020	3
Aprendizaje Joulescope y mediciones de consumo	26/05/2020	02/06/2020	8
Mediciones ancho de banda y comparativa	03/06/2020	05/06/2020	3
Redacción memoria	03/06/2020	22/06/2020	20



El trabajo comienza desde mi interés personal por la arquitectura RISC-V y en conocimiento de mi futuro proyecto a realizar en la asignatura de Laboratorio de Empotrados.

Durante el período que transcurre desde la finalización del primer semestre hasta el comienzo de la asignatura de Laboratorio de Empotrados, se dedicaron entre tres horas y tres horas y media diarias al estudio y comprensión de la arquitectura RISC-V por completo. Esto incluye la lectura de la ISA RISC-V, así como de las extensiones y su funcionamiento. Además, también se leyeron multitud de artículos de investigación para lograr obtener una mejor comprensión del tema sobre el que se iba a trabajar. Esta primera tarea conllevó 34 horas y media de trabajo.

Otro punto sobre el que quería centrar el trabajo final de grado era el Edge Computing, debido a su capacidad para cambiar el paradigma de la informática. Por ello, se comenzó a realizar un estudio de mercado de las placas de desarrollo disponibles, que funcionaran sobre la arquitectura RISC-V, pero que a la vez tuvieran suficiente capacidad de cómputo para realizar trabajo en el propio dispositivo, y que además tuvieran cierto grado de conectividad para poder pertenecer al ámbito de IoT. Esta segunda tarea supuso una carga de 29 horas de trabajo.

Finalmente, en esta primera fase de investigación, y con la idea de comenzar trabajando sobre un dispositivo físico al comienzo de la asignatura de Laboratorio de Empotrados, se eligió entre la segmentación de mercado realizada el dispositivo SiPEED Maix Go debido a su uso del SoC Kendryte K210 y sus capacidades IA. Este proceso sin embargo, tuvo aproximadamente 8 horas de trabajo, debido a que se necesitaba realizar la adquisición del dispositivo, y por tanto la elección debía ser definitiva y consistente.

La segunda fase, el estudio de entornos de desarrollo y viabilidad educativa de la plataforma, comienza en la asignatura de Laboratorio de Empotrados, pero se profundiza y finaliza a lo largo del desarrollo del Trabajo Final de Grado. Esta fase comienza con una investigación más en profundidad acerca de las capacidades del hardware elegido, así como con un proceso de documentación de este debido a que era inexistente. Especialmente por este último motivo, esta tarea supuso 29 horas de trabajo.

La tarea relacionada con el estudio de las herramientas de desarrollo de Kendryte supuso una gran cantidad de trabajo debido a la escasa documentación que existía al respecto. La mayor parte del tiempo dedicado a esta tarea consistió en la recolección y centralización de la documentación acerca del pipeline de trabajo. Por estos motivos, se realizaron en definitiva 45 horas de trabajo durante esta tarea. Debido a problemáticas de compatibilidad y drivers, la preparación del pipeline de desarrollo para C supuso 8

horas de trabajo.

Finalmente, la segunda fase del trabajo culminó con las pruebas de los distintos entornos de desarrollo, y los intentos de depuración de distintos scripts desarrolados en el lenguaje de programación C, con mejores y peores resultados. En especial, los procesos de depuración y los intentos realizados para lograr arreglar y solucionar estos problemas, consumieron la mayor parte del tiempo de estas tres tareas, que en total supusieron 83 horas de trabajo.

La tercera fase del trabajo se centra en el port de Micro Python existente para la placa de desarrollo SiPEED Maix Go, en el estudio del pipeline de entrenamiento para los modelos de inteligencia artificial que esta admite, y en el desarrollo de una aplicación prueba de concepto que pone en conjunto los dos primeros elementos. La primera tarea de esta fase resultó ser la más costosa, debido a la poca información acerca de la metodología para el entrenamiento de modelos para el SoC Kendryte K210, así como la incompatibilidad de pipelines existentes con las distintas versiones de los sistemas operativos disponibles. Esta tarea supuso también el intercambio de *issues* en GitHub para lograr finalmente el correcto funcionamiento de la herramienta aXeleRate. Por todo ello, la primera tarea supuso 43 horas.

La segunda tarea consistió en entender correctamente el funcionamiento de los distintos binarios disponibles de Micro Python que se ofrecían en la página web de SiPEED. Además, supuso la preparación del entorno de desarrollo Maix Py IDE, así como de los drivers necesarios para la conexión directa entre la placa de desarrollo y el IDE. En total, se han computado 14 horas en esta tarea.

La tarea relacionada con la búsqueda de un dataset adecuado para la aplicación prueba de concepto que se deseaba desarrollar supuso 31 horas de trabajo totales debido a la restricción que imponía aXeleRate para la nomenclatura de las anotaciones de objetos en cada imagen.

En la tarea de entrenamiento de modelos se computan las horas relacionadas con la resolución de problemas relacionados con el entrenamiento. Debido a la reducida memoria RAM de la placa de desarrollo, se encontraron diversos problemas de espacio relacionados con los ficheros de pesos de los modelos. Además, debido a la poca documentación que existe al rededor de la plataforma, no se encontró hasta después de varias horas de solución de errores, que la versión del tipo de fichero KModel había cambiado, y que por tanto no se podía utilizar la última versión del port de Micro Python ya que esta no la soportaba. Por todos estos motivos, en total esta tarea supuso 35 horas de trabajo.

Finalmente, la tercera fase del proyecto concluyó con el desarrollo de la aplicación prueba de concepto, la prueba de varios modelos y la medición de precisiones/recall

mediante inferencia, así como la adaptación del dataset público de mascarillas en el contexto de la pandemia del COVID-19. Estas últimas dos tareas costaron 24 horas de trabajo.

La cuarta y última fase del trabajo comienza con la oportunidad de redactar un artículo de investigación para el congreso DCIS 2020 ya que la uno de los apartados de este tenía que ver con la arquitectura RISC-V. Sin embargo, el período de desarrollo de este artículo fue muy reducido debido a que la fecha de entrega era el 22 de Mayo. Por ello, el artículo se escribió en algo más de una semana, con una media de 8 horas diarias de trabajo. En total, la escritura del artículo supuso 73 horas de trabajo. Todo lo desarrollado en el artículo era el trabajo realizado hasta ese momento, con lo que supuso parte de la escritura de la memoria final.

Las tareas de evaluación de la versatilidad, las mediciones de consumo y las mediciones de ancho de banda supusieron un total de 40 horas de trabajo a lo largo de 14 días.

Todo este proyecto culmina con la escritura de la memoria, tarea para la cual se han computado 64 horas de trabajo. Sin embargo, aunque se encuentren bajo la tarea de desarrollo del artículo de investigación, las 73 horas que este conllevaron podrían incluirse también bajo el desarrollo de la memoria debido a que lo desarrollado en el artículo es en gran medida lo desarrollado en la memoria y en el trabajo completo.