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

Enrique Torres-Sánchez
Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza
Zaragoza, Spain
https://orcid.org/0000-0002-3512-2396

Jesús Alastruey-Benedé, Enrique Torres-Moreno
Dept. Informática e Ingeniería de Sistemas - I3A
Universidad de Zaragoza
Zaragoza, Spain
{jalastru,ktm}@unizar.es

***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, two low-cost, low-power, IoT edge applications based on AI have been developed. The first one 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. In the context of the current COVID-19 pandemic, the second 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. As a result, real time object detection can be performed in a low cost and low power system.

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, it may eliminate the need for racks of cloud-based computing and instead move 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 on the maturity of the programming tools and 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 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.

978-1-7281-9132-4/20/$31.00 ©2020 IEEE

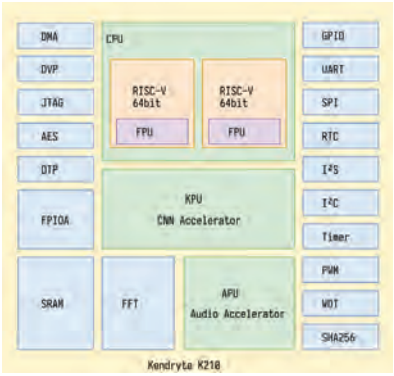

Fig. 1. K210 SoC block diagram [5].

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 also be 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, 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 routines 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.

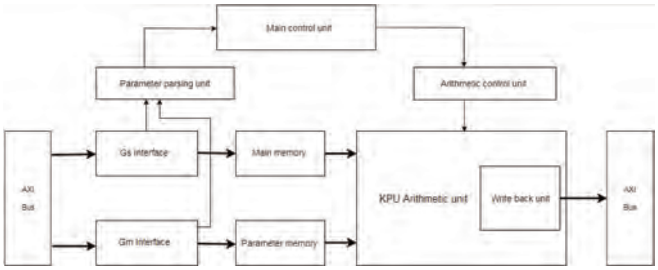

Fig. 2. Knowledge Processing Unit (KPU) flow [5].

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 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 of 640x480 at 30 fps or 60 fps at QVGA. It can output images to both KPU and a LCD display through the MCU interface. 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.

B. MAIX-I module

Sipeed MAIX-I [6], also called M1, integrates the K210, DC/DC power supply circuit, 8MiB/16MiB/128MiB Flash (M1w add ESP8285 Wi-Fi chip) into a 1x1 inch breadboard-friendly and SMT-able module. Its target is the Artificial Intelligence of Things (AIoT) as an 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,

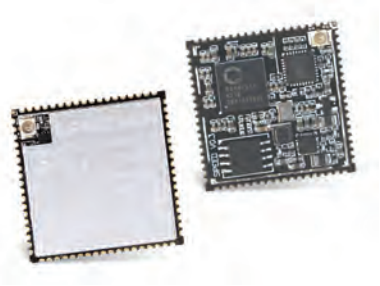

Fig. 3. SiPEED MAIX-1 Wi-Fi module [6].

low-power footprint and low cost make it really appealing for developing and learning AI 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 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, an I2S Mic, a speaker, an 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 consumer price of $40, comes with a 500 mAH lithium-ion battery, a 2.8-inch LCD, a ov2640 with M12 lens DVP camera, an Wi-Fi antenna, and a simple acrylic case.

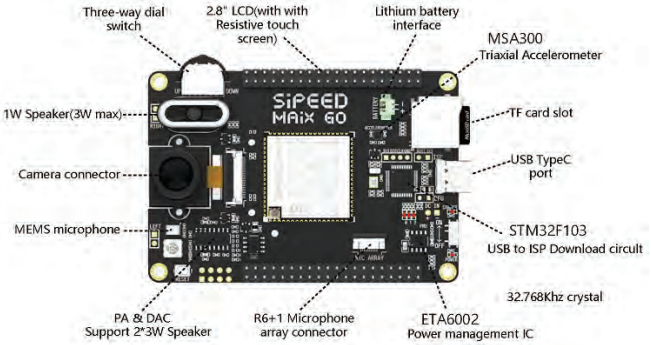

Fig. 4. Sipeed MAIX Go development board [10].

III. Software

During the development process of an application, one important step 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 dissasemble code in real time. Even though C step by step execution is possible, 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.

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 registers, which is the result that was observed when debugging the code with VisualGDB.

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

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 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 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 NNCASE [21], an open source tool provided 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 in the image.

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 bounding boxes it has recognized, they are drawn on the frame buffer, followed by a string that specifies what object class is recognized.

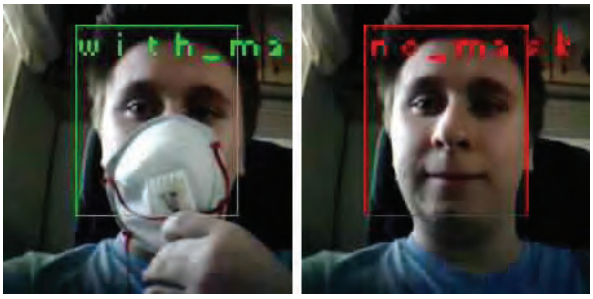

Fig. 5. Mask detection application sample images. Left side shows a correct mask placement detection. Right side shows the opposite.

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, the platform is 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.

VI. Acknowledgments

This work was supported by MINECO/AEI/ERDF (EU) (grants TIN2016-76635-C2-1-R and PID2019-105660RB-C21), Aragón Government (T58_20R research group), and ERDF 2014-2020 "Construyendo Europa desde Aragón".

Thanks to Red-RISCV and the RISC-V ISA - European Processor Initiative for promoting open hardware.

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