Inference speed comparison using convolutions in neural networks on various SoC hardware platforms using MicroPython

Kristian Dokic, PhD[a], Hrvoje Mikolcevic[b] and Bojan Radisic[a]

[a] *Polytechnic in Pozega, Vukovarska 17, Pozega, Croatia*
[b] *Tehnical school, Ratarnicka 1, Pozega, Croatia*

Abstract

In recent years we have witnessed the rapid development of machine learning algorithms, and the same can be said for IoT. Developments in both fields have also influenced the growth of machine learning algorithms in IoT devices. The authors of a series of papers cite several reasons to argue this trend. This paper explores the possibility of using the Python programming language in different versions to create, train, and implement convolutional neural networks on two SoCs based on different architectures (ARM and RISC-V). The influence of the number of filters in the convolutional layer on the inference speed is also investigated. The number of filters has a different effect on inference speed depending on the existence of additional components that accelerate individual operations of convolutional neural networks (convolution, batch normalization, activation, and pooling operations).

Keywords 1

MicroPython, SoC, CNN, Convolution Neural Network, RISC-V

1. Introduction

In the last few years, the terms *Internet of Things* and *Machine learning* have been increasingly mentioned in the field of information and communication technologies. Definitions of both terms are given below. Internet of things is “a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction” [1]. Machine learning “is the study of computer algorithms that improve automatically through experience” [2].

Powerful computers are most often required to implement machine learning, and IoT devices are often low in processing power. Several authors have stated that there are few advantages of transferring part of machine learning data processing from large and cloud computers to IoT devices. Zhang et al. state that these are energy savings, network traffic reduction, privacy problems and delays in data transmission [3]. Sakr et al. also cites similar benefits of moving computation toward the edge, such as lower energy consumption, lower response latency, higher security, lower bandwidth occupancy and expected privacy [4].

Since Python is a programming language most often used for machine learning and very rarely for the development of IoT devices, in this paper, the aim was to compare two different SoCs' performance using Python and MicroPython in the development of machine learning models and IoT devices. The C/C++ programming language is most often used to develop IoT devices, but if a version of Python were used for this task, all phases of a machine learning model development for IoT devices would be significantly accelerated and simplified.

Two development boards that hit the market a year ago based on different architectures were used in this paper, and their inference

Proccedings of RTA-CSIT 2021, May 2021, Tirana, Albania
EMAIL: kdjokic@vup.hr (Kristian Dokic); mikhadrugi@gmail.com (Hrvoje Mikolcevic); bradisic@vup.hr (Bojan Radisic)

performance was tested with a different number of filters in the first layer of the convolutional neural network used to detect handwritten digits. Easy use of convolutional neural networks on microcontrollers has been possible in the last year or two, so there is not much information about microcontrollers' performance on these tasks. A convolution neural network is a deep neural network class usually used for image processing, classification, and other similar tasks.

2. Literature review

The first part of this section provides an overview of papers in which the authors used MicroPython to solve various problems. The second part provides an analysis of the used hardware platforms.

2.1. MicroPython

MicroPython is a programming language partially compatible with Python 3. It is optimized to run on microcontrollers, and it includes various modules for low-level hardware access. It was released in 2014. and it is ported on various platforms (ARM Cortex-M, RISC-V, ESP32, ESP8266, PIC, STM32) [5] [6]. It is also available on BBC micro:bit development board [7] [8]. Finally, a new Raspberry product Raspberry Pi Pico can be used with MicroPython [9]. It is evident that MicroPython has become quite popular in a short time, and it has been ported to various platforms without the significant influence of large companies.

MicroPython is used in various IoT projects. Regnath et al. used MicroPython on ESP32 platform to test their approach to verify data integrity and consensus on a blockchain used on IoT platforms [10]. Cardenas et al. used MicroPython in Low-Cost and Low-Power Messaging System. Their solution was based on ESP32 and LoRa Wireless Technology [11]. Tzounis et al., in their review of the IoT platform in agriculture, also mention MicroPython as a programming language for the LoPy development platform based on ESP32 SoC [12]. LoPy development board and MicroPython have been used by Sayed et al. in their multi-sensing platform for the ORCA Hub integrated with Robot Operating System [13]. Kodali et al. used MicroPython to develop a low-cost ambient monitor device based on ESP8266 SoC. They used a DHT22 temperature and humidity sensor and OLED 128x64 display [14].

Khamphroo et al. introduced a mobile educational robot based on MicroPython. The development's primary goal is to create a robot that will be easy to use for beginners. STM32L432KC ARM Cortex-M4 168 MHz was used in the presented project [15] [16]. Tariq et al. used MicroPython for noise filtering on an STM32F10RB platform to implement an early warning seismic event detection algorithm [17]. Zhang et al. used MicroPython in their quadruped robot project that deals with the twisting trunk's effect on the tumble stability from an energy viewpoint. Their development board was based on STM32F405 [18]. Wie et al. used MicroPython in their wearable bio-signal processing system. Devices were based on the STM32F722 microcontroller [19]. Ibba et al. developed an impedance analyzer for fruit quality monitoring based on the STM32L486 microcontroller. They also used MicroPython for microcontroller programming [20]. Crepaldi et al. developed the body channel communication system used for landmark identification. They used MicroPython and STM32L486 microcontroller [21].

Bahmanian et al. used MicroPython in their wide-band frequency synthesizer that can lock on any of the optical reference harmonics between 2 GHz and 20 GHz. They used the FE310 microcontroller based on RISC-V to control the DA converters [22].

At the end of the MicroPython part, we should mention the Arduino IDE, which is undoubtedly more popular than MicroPython, but it also has some drawbacks. Kodali et al. compared these two platforms by a series of features. They state that the difference is in the language type since MicroPython is a scripting language, while Arduino C code needs to be compiled. According to them, MicroPython is simpler, and the development is 5-10 times faster because there is no compiling. The syntax is cleaner, and the code is more readable in MicroPython [14]. Tanganelli et al. have reviewed the available platforms for rapid prototyping of IoT solutions from a developer perspective. In addition to the Arduino IDE and MicroPython, they listed several other platforms and development tools: FreeRTOS, mbedOS, Zephyr, Contiki OS, RIOT OS and

Zerynth. They also cite perhaps the biggest drawback of the MicroPython, which is the significantly smaller number of development boards that support it. The Arduino can be used on over 1000 development boards, while the MicroPython is supported by twenty. Their paper is from 2019, and the situation may have changed somewhat [23].

2.2. MicroPython hardware on edge

In the previously presented paper review, MicroPython-enabled platforms are grouped into three individual sections. The first section contains papers that use ESP32 and ESP8266 SoC. These are Espressif Inc. products based on Tensilica LX106 and LX6 processors. The second section lists papers that use different SoCs from STMicroelectronics, while the third section includes paper that uses SoCs based on the RISC-V architecture.

The official GitHub page of the MicroPython project lists development boards that come with MicroPython installed. There are currently 35 development boards on the list, primarily based on the STM32F family of microcontrollers [24].

3. Method

The first part of this section gives the characteristics of the used development boards and microcontrollers with MicroPython installed. The second part of the section describes creating a model of a convolutional neural network and converting the model into formats that can be used on SoC with MicroPython.

3.1. Used hardware supported by MicroPython

The paper analyzes two development boards based on different architectures. The development board in Figure 1 is an OpenMV Cam H7 Plus and is based on the STM32H743II ARM Cortex M7 processor. The development board's price was € 99.00, but only one STM32H743II processor costs € 11.61. Their characteristics are listed in Table 1.

Table 1
Microprocessor's characteristics

	STM32H743II	Kendryte K210
Producer	STM	Canaan Inc.
Bit Width	32 bit	64 bit
CPU clock	480 MHz	400 MHz
RAM	1+32 MB	6 MB
FLASH	2+32 MB	16 MB

Figure 1: OpenMV Cam H7 Plus

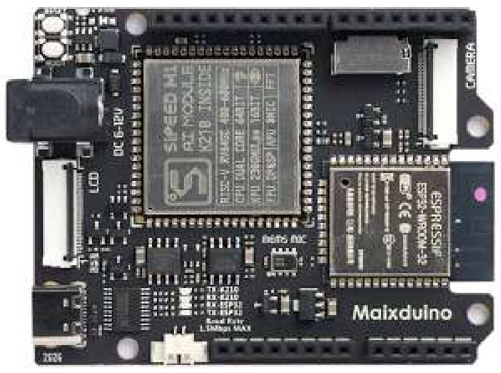


Figure 2: Sipeed Maixduino

The Development board in Figure 2 is a Sipeed Maixduino Kit for RISC-V AI + IoT, and it is based on the Kendryte K210 based on RISC-V instruction set architecture. The development board's price was $ 23.90, but only one K210 costs $ 8.64.

3.2. CNN model development and deployment

The MNIST database of handwritten digits has been used for training in this paper. This database consists of sixty thousand grayscale images of handwritten digits between 0 and 9.

The image's size is 28x28 pixels that are reduced to 14x14-pixel size. One simple convolution neural network has been developed to resolve this classification problem, and it can be seen in Figure 3.

This neural network is quite simple and consists of a single convolutional layer with a 3x3 pixel filter. The number of filters is a variable changed during testing to notice the impact of that variable on the inference speed. Adam is used as an optimizer and ReLU as an activation function. The model code can be found on the GitHub address: https://github.com/kristian1971/MicroPythonRACE.

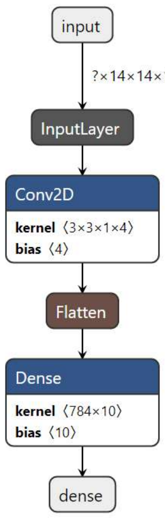


Figure 3: Used CNN model

The model was trained for only 12 epochs, and after that, it was converted to TensorFlow Lite (tflite) format. A total of 10 training processes were conducted, and the variable that was changed was the number of filters in the convolutional layer. After each training, the model in tflite format is saved. The same model was used on both development boards.

The OpenMV Cam H7 Plus development board's integrated development environment can be downloaded for free from the manufacturer's website and is called the OpenMV IDE. A free Integrated development environment based on the OpenMV IDE, called the MaixPy IDE, is available for the Sipeed Maixduino development board. Both are based on MicroPython. However, the use of the convolutional neural network model is somewhat different on boards. The OpenMV IDE and the associated development board allow easy transfer of the model in tflite format with a simple drag-and-drop method because the OpenMV Cam H7 Plus development board in the Windows environment is presented as a USB memory stick.

In contrast, for the Sipeed Maixduino development board, model preparation is more complicated. NNCase is used for the conversion, which is also free, and after the conversion, a model in kfpkg format is obtained. The file in the specified kfpkg format is transferred to the development board using the KFlash tool. Figure 4 shows the process of training, conversion and transfer of the model to both development boards.

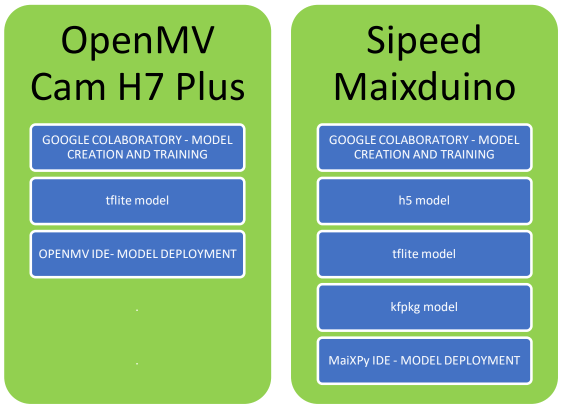
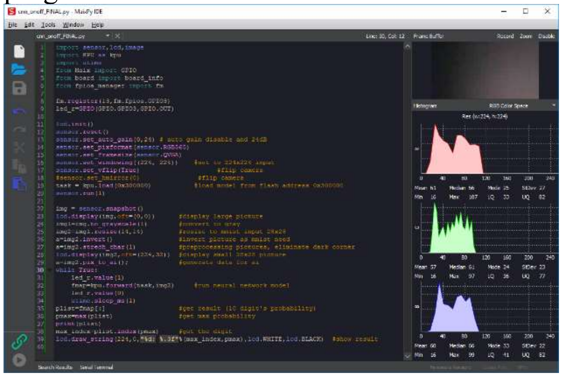

OpenMV Cam H7 Plus
GOOGLE COLABORATORY - MODEL CREATION AND TRAINING
tflite model
OPENMV IDE- MODEL DEPLOYMENT
Sipeed Maixduino
GOOGLE COLABORATORY - MODEL CREATION AND TRAINING
h5 model
tflite model
kfpkg model
MaiXPy IDE - MODEL DEPLOYMENT

Figure 4 - Process of training, conversion and transfer

After the models have been transferred to development boards, it is necessary to write a program in each integrated development environment that will apply that model. In both cases, it is written in MicroPython, and the differences between them are insignificant. The programs are available on the GitHub server.



```
import sensor,lcd,image
import KPU as kpu
import utime
from Maix import GPIO
from board import board_info
from fpioa_manager import fm

fm.register(13,fm.fpioa.GPIO3)
led_r=GPIO(GPIO.GPIO3,GPIO.OUT)

lcd.init()
sensor.reset()
sensor.set_auto_gain(0,24) # auto gain disable and 24dB
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QVGA)
sensor.set_windowing((224, 224))    #set to 224x224 input
sensor.set_vflip(True)                      #flip camera
#sensor.set_hmirror(0)                 #flip camera
task = kpu.load(0x300000)             #load model from flash address 0x300000
sensor.run(1)

img = sensor.snapshot()
lcd.display(img,oft=(0,0))        #display large picture
img1=img.to_grayscale(1)          #convert to gray
img2=img1.resize(14,14)           #resize to mnist input 28x28
a=img2.invert()                   #invert picture as mnist need
a=img2.strech_char(1)             #preprocessing pictures, eliminate dark corner
lcd.display(img2,oft=(224,32))    #display small 28x28 picture
a=img2.pix_to_ai();               #generate data for ai
while True:
    led_r.value(1)
    fmap=kpu.forward(task,img2)     #run neural network model
    led_r.value(0)
    utime.sleep_ms(1)
plist=fmap[:]                     #get result (10 digit's probability)
pmax=max(plist)                   #get max probability
print(plist)
max_index=plist.index(pmax)       #get the digit
lcd.draw_string(224,0,"%d: %.3f"%(max_index,pmax),lcd.WHITE,lcd.BLACK)  #show result
```

Figure 5: MaixPy IDE

Figure 5 shows the integrated development environment MaixPy IDE, while the OpenMV IDE is not shown because the visual differences are negligible.

The HANTEK DSO5102P oscilloscope was used to measure the inference speed. Before the command that triggers the data inference through the neural network, one pin on the development board is lifted from LOW to HIGH. After the inference process, the state of that same pin is lowered from HIGH to LOW. The commands for the OPENMV IDE are as follows:

pin1.value(1)
out = net.classify(graytmp)
pin1.value(0)

The commands for the MaixPy IDE are as follows:

led_r.value(1)
fmap=kpu.forward(task,img2)
led_r.value(0)

The described method is used to measure the propagation time of data through a convolutional neural network. Figure 6 shows the measurement procedure on the oscilloscope.

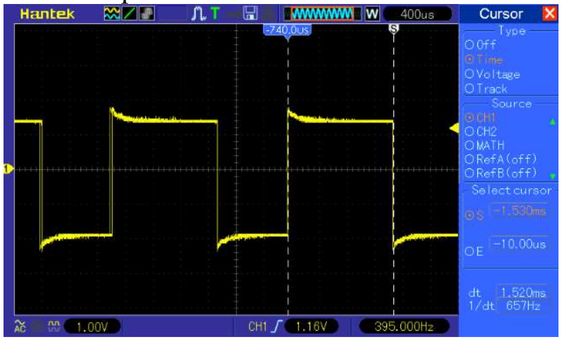


Figure 6: Measurement with an oscilloscope

4. Testing and measurement

In the measurement procedure, ten models were used on both development boards that differed in the number of filters in the convolutional layer. The number of filters was changed from one to ten. Table 2 shows the data obtained for the Sipeex Maixduino development board, while Table 3 shows the data for the OpenMV Cam H7 Plus.

Table 2
Sipeex Maixduino data

	ACCURACY	Tflite	t
1	0.8969	8 kB	1,2 ms
2	0,9220	16 kB	1,3 ms
3	0.9479	24 kB	1,4 ms
4	0.9330	31 kB	1,5 ms
5	0,9520	39 kB	1,6 ms
6	0.9637	47 kB	2,0 ms
7	0,9623	54 kB	2,3 ms
8	0.9624	62 kB	2,3 ms
9	0,9632	70 kB	2,6 ms
10	0,9620	78 kB	2,6 ms

The first column of both tables contains the number of filters in the convolution layer. In the second column, the accuracy is obtained in the model creation phase, and in the third column are the *tflite* file sizes. The first three columns are the same for both development boards.

Table 3
OpenMV Cam H7 Plus data

	ACCURACY	Tflite	t
1	0.8969	8 kB	0,22 ms
2	0,9220	16 kB	0,33 ms
3	0.9479	24 kB	0,42 ms
4	0.9330	31 kB	0,52 ms
5	0,9520	39 kB	0,63 ms
6	0.9637	47 kB	0,74 ms
7	0,9623	54 kB	0,85 ms
8	0.9624	62 kB	0,97 ms
9	0,9632	70 kB	1,08 ms
10	0,9620	78 kB	1,20 ms

The fourth columns show the neural network inference speed, and these values are significantly less for the OpenMV Cam H7 Plus development board. Figure 7 shows a graph of speed for both development boards.

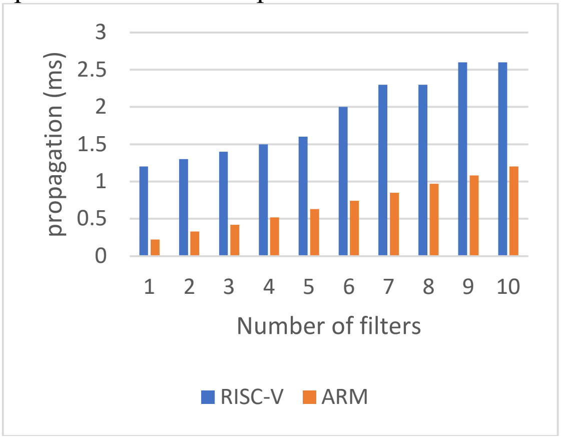


Figure 7: Inference speed

If we look at the data more closely, we can see a linear increase in the inference speed in the OpenMV Cam H7 Plus development board depending on the number of filters, while for Sipeed MaixPy, this dependence is not linear. To determine this difference, Figure 8 shows the inference speed with x filters (t_x) relative to the inference speed measured with only one filter in the convolution layer (t_1). It is evident that by increasing the number of filters, the inference speed increases faster on the OpenMV Cam H7 Plus development board, while this increase is slower on Sipeed MaixPy.

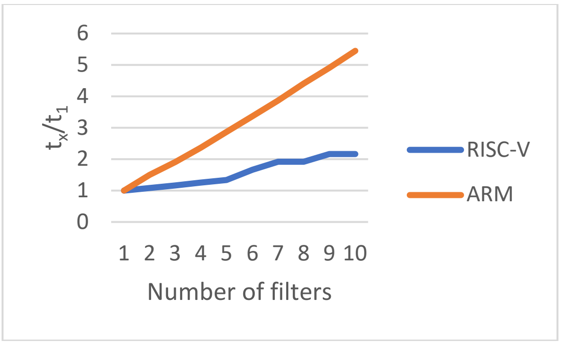


Figure 8: Relative inference speed

5. Discussion and conclusion

The paper compares the convolutional neural network inference speed of two various developmental boards. One development board is based on the STM32H743II ARM Cortex M7 processor, while the other is based on the Kendryte K210 RISC-V processor. The same neural network model was used in both cases, and MicroPython was used for the model implementation on development boards. TensorFlow 2.x and Keras were used to develop and train the model.

It can be observed that the inference speed under the given conditions is significantly lower on the STM32H743II ARM Cortex M7 processor. However, we must emphasize that the convolutional neural network used is simple and that the input image is 14x14 pixels in size.

A detail that should also be emphasized is the influence of the number of used filters on the inference speed. With only one filter in the convolutional layer, the development board based on the ARM processor was 6x faster, while when using ten filters, this difference dropped to only 2x. The increase in the number of filters reduces the difference in inference speed between boards. It is possible that at some point, the development board based on the RISC-V processor would be faster.

The cause of this slight slope curve for the RISC-V processor in Figure 8 is a Neural Network processor in K210 with built-in convolution, batch normalization, activation, and pooling operations. This is sparsely described in the documentation [25].
In future research, the number of filters and other parameters should be further increased, and the influence of filter size should be studied. In this paper, a 3x3 filter has been used, and the documentation for the K210 states that the Neural Network processor speeds up the data propagation through a convolutional neural network for 3x3 filters.

6. References

[1] A. S. Gillis, »internet of things (IoT),« TechTarget, [Mrežno]. Available: https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT. [Pokušaj pristupa 12 January 2021].

[2] T. M. Mitchell, Machine Learning, New York: McGraw-hill, 1997.

[3] Y. Zhang, N. Suda, L. Lai i V. Chandra, *Hello Edge: Keyword Spotting on Microcontrollers,* 2018.

[4] F. Sakr, F. Bellotti, R. Berta i A. De Gloria, »Machine Learning on Mainstream Microcontrollers,« *Sensors,* svez. 20, 2020.

[5] J. Beningo, »Prototype to production: MicroPython under the hood,« Aspeccore Inc., 11 July 2016. [Mrežno]. Available: https://www.edn.com/prototype-to-production-micropython-under-the-hood/. [Pokušaj pristupa 13 December 2020].

[6] George Robotics Limited, »MicroPython,« George Robotics Limited, [Mrežno]. Available: https://micropython.org/. [Pokušaj pristupa 12 December 2020].

[7] S. Sentance, J. Waite, L. Yeomans i E. MacLeod, »Teaching with physical computing devices: the BBC micro: bit initiative,« u *Proceedings of the 12th Workshop on Primary and Secondary Computing Education*, 2017.

[8] M. Cápay i N. Klimová, »Engage Your Students via Physical Computing!,« u *2019 IEEE Global Engineering Education Conference (EDUCON)*, 2019.

[9] J. Adams, »Meet Raspberry Silicon: Raspberry Pi Pico now on sale at $4,« RASPBERRY PI FOUNDATION, 21 January 2021. [Mrežno]. Available: https://www.raspberrypi.org/blog/raspberry-pi-silicon-pico-now-on-sale/. [Pokušaj pristupa 22 Februray 2021].

[10] E. Regnath i S. Steinhorst, »LeapChain: efficient blockchain verification for embedded IoT,« u *Proceedings of the International Conference on Computer-Aided Design*, 2018.

[11] A. M. Cardenas, M. K. N. Pinto, E. Pietrosemoli, M. Zennaro, M. Rainone i P. Manzoni, »A low-cost and low-power messaging system based on the LoRa wireless technology,« *Mobile Networks and Applications,* p. 1–8, 2019.

[12] A. Tzounis, N. Katsoulas, T. Bartzanas i C. Kittas, »Internet of Things in agriculture, recent advances and future challenges,« *Biosystems Engineering,* svez. 164, p. 31–48, 2017.

[13] M. E. Sayed, M. P. Nemitz, S. Aracri, A. C. McConnell, R. M. McKenzie i A. A. Stokes, »The limpet: A ROS-enabled multi-sensing platform for the ORCA hub,« *Sensors,* svez. 18, p. 3487, 2018.

[14] R. K. Kodali i K. S. Mahesh, »Low cost ambient monitoring using ESP8266,« u *2016 2nd International Conference on Contemporary Computing and Informatics (IC3I)*, 2016.

[15] M. Khamphroo, N. Kwankeo, K. Kaemarungsi i K. Fukawa, »Integrating MicroPython-based educational mobile robot with wireless network,« u *2017 9th International Conference on Information Technology and Electrical Engineering (ICITEE)*, 2017.

[16] M. Khamphroo, N. Kwankeo, K. Kaemarungsi i K. Fukawa, »MicroPython-based educational mobile robot for computer coding learning,« u *2017 8th International Conference of Information and Communication Technology for Embedded Systems (IC-ICTES)*, 2017.

[17] H. Tariq, F. Touati, M. A. E Al-Hitmi, D. Crescini i A. Ben Mnaouer, »A real-time early warning seismic event detection algorithm using smart geo-spatial bi-axial inclinometer nodes for Industry 4.0 applications,« *Applied Sciences,* svez. 9, p. 3650, 2019.

[18] C. Zhang, X. Chai i J. S. Dai, »Preventing Tumbling With a Twisting Trunk for the Quadruped Robot: Origaker I,« u *ASME 2018 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 2018.

[19] Y. Wei, Q. Cao, L. Hargrove i J. Gu, »A Wearable Bio-signal Processing System with Ultra-low-power SoC and Collaborative Neural Network Classifier for Low Dimensional Data Communication,« u *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, 2020.

[20] P. Ibba, M. Crepaldi, G. Cantarella, G. Zini, A. Barcellona, M. Petrelli, B. D. Abera, B. Shkodra, L. Petti i P. Lugli, »Fruitmeter: An ad5933-based portable impedance analyzer for fruit quality characterization,« u *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2020.

[21] M. Crepaldi, A. Barcellona, G. Zini, A. Ansaldo, P. M. Ros, A. Sanginario, C. Cuccu, D. De Marchi i L. Brayda, »Live wire-a low-complexity body channel communication system for landmark identification,« *IEEE Transactions on Emerging Topics in Computing,* 2020.

[22] M. Bahmanian, S. Fard, B. Koppelmann i J. C. Scheytt, »Wide-Band Frequency Synthesizer with Ultra-Low Phase Noise Using an Optical Clock Source,« u *2020 IEEE/MTT-S International Microwave Symposium (IMS)*, 2020.

[23] G. Tanganelli, C. Vallati i E. Mingozzi, »Rapid Prototyping of IoT Solutions: A Developer's Perspective,« *IEEE Internet Computing,* svez. 23, p. 43–52, 2019.

[24] M. Sielenkemper , »Boards Summary,« GitHub, 20 Octobar 2019. [Mrežno]. Available: https://github.com/micropython/micropython/wiki/Boards-Summary. [Pokušaj pristupa 27 Februray 2021].

[25] Sipeed, "K210," Sipeed, [Online]. Available: https://maixduino.sipeed.com/en/hardware/k210.html. [Accessed 13 January 2021].