MicroPython or Arduino C for ESP32 - Efficiency for Neural Network Edge Devices

Kristian Dokic[1](✉), Bojan Radisic[1], and Mirko Cobovic[2]

[1] Polytechnic in Pozega, Pozega, Croatia
kdjokic@vup.hr
[2] College of Slavonski Brod, Pozega, Croatia

Abstract. In the last few years, microcontrollers used for IoT devices became more and more powerful, and many authors have started to use them in machine learning systems. Most of the authors used them just for data collecting for ML algorithms in the clouds, but some of them implemented ML algorithms on the microcontrollers. The goal of this paper is to analyses the neural networks data propagation speed of one popular SoC (Espressif System company ESP32) with simple neural networks implementation with two different development environment, Arduino IDE and MycroPython. Neural networks with one hidden layer are used with a different number of neurons. This SoC is analysed because some companies started to produce them with UXGA (Ultra Extended Graphics Array) camera implemented and it can be used to distribute computing load from central ML servers.

Keywords: Neural network · ESP32 · Arduino · MicroPython

1 Introduction

The traditional approach of artificial intelligence algorithm usage forces centralised schema. This approach is simpler, but in some cases, it makes the high load on a central server. One solution is to transfer tasks from the central server to smaller processors or microcontrollers on edge, and this is called *edge computing*. This approach reduces network traffic since edge nodes upload reduced data instead of complete input data [1].

In the last few years, many microcontroller producers have worked on artificial intelligence implementation on microcontrollers. Some of them have developed special libraries with AI functions, but the others have implemented special hardware with enhanced AI capabilities.

In this paper, a low-cost Chinese SoC ESP32 is tested with feed-forward neural networks on two different development environment, Arduino IDE and MycroPython. Espressif System, a company from China introduced ESP32 before three years. It does not have machine learning capabilities. It is manufactured using a 40 nm process by TSMC (Taiwan Semiconductor Manufacturing Company) [2]. Many projects have been developed with ESP32 SoC like water pumping solar system [3], or monitors for Liquefied petroleum gas [4]. Some authors suggested that ESP32 will play one of the major role in future Internet of Things devices development [5].

In the second section, some papers that deal with machine learning usage and ESP32 are presented, but in the third section, some development environments for ESP32 are presented. In the fourth section, neural networks forward propagation times with a different number of neurons in the hidden layer are measured. Finally, in section five, discussion and conclusion can be found.

2 ESP32 and Machine Learning

ESP32 is used for various machine learning tasks, but usually only for measuring and collecting data for dislocated machine learning servers. On the other hand, some papers deal with ESP32 usage in projects that have machine learning algorithms implemented. In the next two paragraphs, papers that deal with these two different approaches are analysed.

2.1 Machine Learning on the Cloud

Zidek et al. used ESP32 SoC with different sensors for wireless devices input data optimization. Data was accumulated into the packet and then the hole packet was sent to the cloud service [6].

Rosato and Masciadri have chosen Logistic Regression to detect sitting person with the device based on ESP32 SoC. They made monitoring system and the key component with ESP32 was installed under the person chair. Data has been analysed on the server, and ESP32 has been wirelessly connected to it [7].

Islam et al. designed small device that can record electrical activity of the human heart using electrodes placed on the skin. This device has been based on the Analog Devices AD8232. They used ESP32 SoC to send data to the server. Linux server has been used to analyse data with machine learning algorithms [8].

Fernoaga et al. used ESP32 to take pictures of gas, electricity or water counters. These devices have had implemented cameras and they have sent pictures to the cloud. Numbers from the counters are recognised by neural networks from supplied pictures [9].

Komarek et al. presented a project that provide a layer that uses MQTT (MQ Telemetry Transport) for interfacing sensors and nodes. They used ESP32 SoC to send data to other nodes and the cloud for the ambient intelligence system [10].

Chand et al. used ESP32 SoC for a person behavior and status prediction in one room. They used ultrasonic sensors and data from sensors are sent to a server. The main goal of the server was to decide whether it is normal or abnormal situation. This decision system is based on machine learning algorithms [11].

2.2 Machine Learning on an ESP32

Espressif Systems company developed ESP-WHO framework for face detection and recognition. This framework is based on Multi-task Cascaded Convolutional Networks model and new mobile architecture - MobileNetV2 and it is available on the GitHub.

The main difference between this framework and previously mentioned projects is in the fact that ESP-WHO has face detection and recognition algorithms on the ESP32 SoC [12, 13].

Kokoulin et al. used ESP32 SoC for the system that process video stream and detect presence of faces or silhouettes. Only images with faces or silhouette are sent to the central face recognition server. The main goal of that system were network traffic reduction as well as central face recognition server computing load reduction. Estimated network traffic decrease up to 80–90% [14].

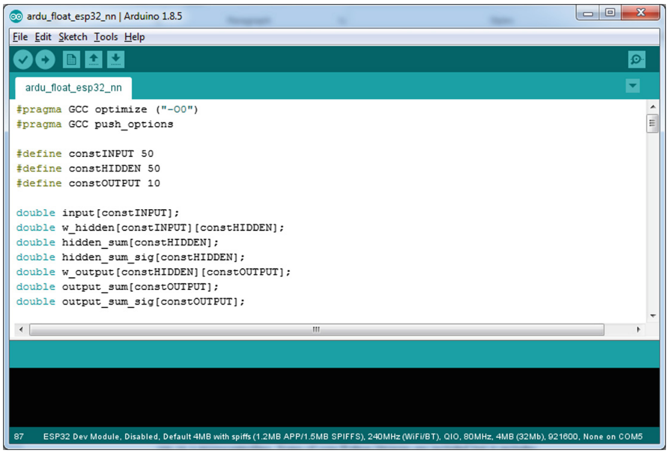
3 ESP32 Development Environments

ESP-IDF (Espressif IoT Development Framework) is the official development framework for the ESP32 SoC. ESP-IDF is the most powerful framework, and it can be used with Linux and Windows OS. Because of ESP32 popularity some other tools are adapted to ESP32. Most popular are Arduino IDE and MicroPython, and they will be analysed.

On the other hand, there are lots of development environments that can be used for ESP32 projects development. KB-IDE is open IDE available on GitHub for ESP32 SoC boards. It supports visual programming similar to Scratch, Arduino style programming, and the official ESP-IDF framework for more experienced users (https://kbide.org/). Lua programming language can be used with ESP32, too. With ChiliPeppr ESP32 Web IDE, a browser can be used for editing/uploading Lua code to ESP32 device. The ChiliPeppr IDE has a serial port JSON (JavaScript Object Notation) Server that can be used locally or remotely to let browser communicate directly to serial port and ESP32 (http://chilipeppr.com/esp32). ESP32 is supported by PlatformIO. It is a cross-platform code builder and library manager. It supports more than 200 development boards, 15 development platforms and ten frameworks.

3.1 Arduino IDE

Arduino IDE is java application used to write programs for Arduino compatible boards. It can be used with ESP32, but first some additional software components have to be downloaded before use. The Arduino IDE supports simplified versions of C and C++ languages. Arduino is well known Italian company with lots of microcontroller boards that are licensed under the LGPL (Lesser General Public License) or the GPL (General Public License). In Fig. 1, the Arduino IDE can be seen.


ardu_float_esp32_nn | Arduino 1.8.5
File Edit Sketch Tools Help
ardu_float_esp32_nn

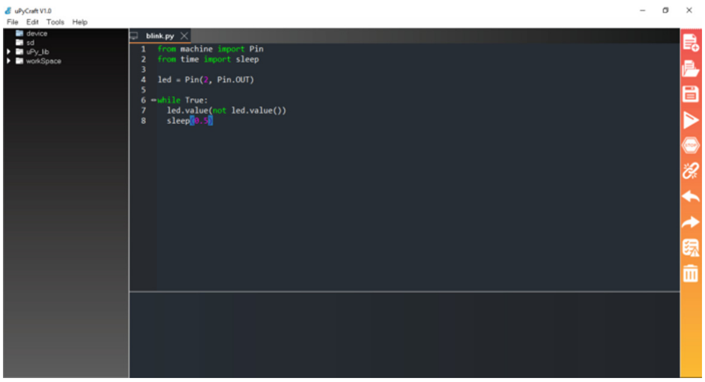
```
#pragma GCC optimize ("-O0")
#pragma GCC push_options

#define constINPUT 50
#define constHIDDEN 50
#define constOUTPUT 10

double input[constINPUT];
double w_hidden[constINPUT][constHIDDEN];
double hidden_sum[constHIDDEN];
double hidden_sum_sig[constHIDDEN];
double w_output[constHIDDEN][constOUTPUT];
double output_sum[constOUTPUT];
double output_sum_sig[constOUTPUT];
```

```
87      ESP32 Dev Module, Disabled, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), 240MHz (WiFi/BT), QIO, 80MHz, 4MB (32Mb), 921600, None on COM5
```

Fig. 1. Arduino IDE

3.2 MicroPython

MicroPython is an implementation of a Python programming language optimised to run on some microcontrollers. Some of the core Python libraries are included, but it includes modules that allow low-level hardware access to the programmer. MicroPython was created by Australian programmer Damien George. It does not have rich IDE, but on their web page spycraft IDE is suggested. In Fig. 2, uPyCraft IDE can be seen.



```
from machine import Pin
from time import sleep

led = Pin(2, Pin.OUT)

while True:
  led.value(not led.value())
  sleep(0.5)
```

Fig. 2. uPyCraft IDE

4 Neural Networks Forward Propagation Speed

Teerapittayanon suggests that their Distributed Deep Neural Networks model can reduce the communication cost by a factor of over 20x but to achieve this, so-called “edge” or “end” devices have to use simple Machine Learning models and process data from sensors [15]. Zhang describes keyword spotting device based on Cortex-M7 based STM32F746G-DISCO development board. Keyword spotting devices are typical examples of edge devices that are based on tiny microcontrollers with limited memory and compute capability. They require real-time response and high accuracy for good user experience [16]. Lai quotes that “Deep Neural Networks are becoming increasingly popular in always-on IoT edge devices performing data analytics right at the source, reducing latency as well as energy consumption for data communication”. Lai used Arm Cortex-M processor board and achieved 4.6X improvement in runtime/throughput and 4.9X improvement in energy efficiency with CMSIS-NN kernels [17].

There are lots of examples where a neural network is trained on a powerful computer and then deployed to tiny microcontroller to analyse some input data. In this paper ESP32 is tested for that task. The simple neural network is programmed with random weights and different number of nodes and data propagation time is measured.

4.1 Data and Methods

Neural networks with 50 input nodes and ten output nodes are programmed with Arduino IDE and MicroPython. The number of hidden nodes in one hidden layer is between 50 and 200. A logistic sigmoid activation function is used (Fig. 3).

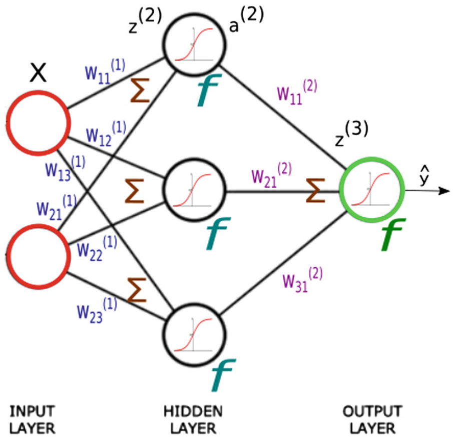


Fig. 3. Neural network with one hidden layer

Setup for testing is in Table 1.

Table 1. Testing setup

	Input layer	Hidden layer	Output layer	uC	Weights format
1	50	50	10	Arduino	float
2	50	100	10	Arduino	float
3	50	150	10	Arduino	float
4	50	200	10	Arduino	float
5	50	50	10	Arduino	double
6	50	100	10	Arduino	double
7	50	150	10	Arduino	double
8	50	200	10	Arduino	double
9	50	50	10	MicroPython	float
10	50	100	10	MicroPython	float
11	50	150	10	MicroPython	float
12	50	200	10	MicroPython	float

4.2 Microcontrollers

Two ESP32 based microcontrollers are used for testing purposes. ESP32 WROOM is in Fig. 4, and PYCOM WIPY 3.0 is in Fig. 5. The main difference between them is that PYCOM WIPY 3.0 has MicroPython installed and ready to be used. ESP32 WROOM can be used with Arduino IDE without any setup.

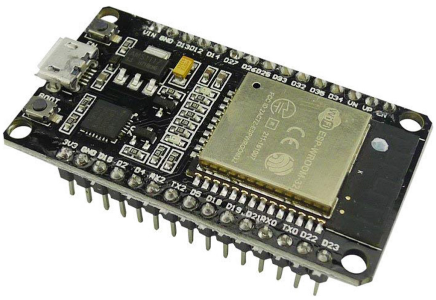

VIN GND D13 D12 D14 D27 D26 D25 D33 D32 D35 D34 VN VP EN
BOOT
3V3 GND D15 D2 D4 RX2 TX2 D5 D18 D19 D21 RX0 TX0 D22 D23
ESP-WROOM-32

Fig. 4. ESP32 WROOM (Arduino)

Both boards have Xtensa dual-core 32-bit LX6 microprocessor operating on 240 MHz. ESP32 WROOM has only 520 KB RAM, but PYCOM WIPY 3.0 has 520 KB and 4 MB additional memory. WiFi and Bluetooth modules characteristics are same, but ESP32 WROOM has serial to USB controller implemented on board.

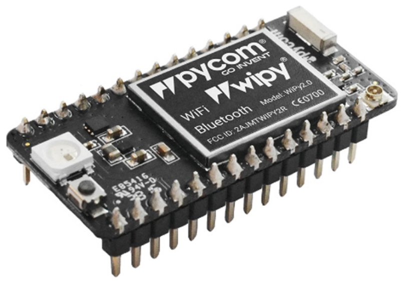


Fig. 5. PYCOM WIPY 3.0 (MicroPython)

4.3 Testing

There are different ways to test the speed of neural network data propagation, but in this research digital oscilloscope is used. Some authors used implemented functions milis() and micros() on both platforms but measuring with oscilloscope is better solution because some processes in microcontroller cannot affect measuring.

To measure the time, GPIO (general-purpose input/output) output of the microcontroller is connected to an oscilloscope. Simple code in Arduino C and MicroPython with tasks listed in Table 2 is prepared. This code has been used to measure frequency of pin output rise and fall but these periods are too small to have any effect on measuring accuracy. Periods can be seen in Table 3. All code listings can be found on GitHub.

Table 2. Code in Arduino C and MicroPython for pin output rise and fall

State	Arduino C	MicroPython
GPIO ON	digitalWrite(LED_BUILTIN, HIGH);	led.value(1)
GPIO OFF	digitalWrite(LED_BUILTIN, LOW);	led.value(0)

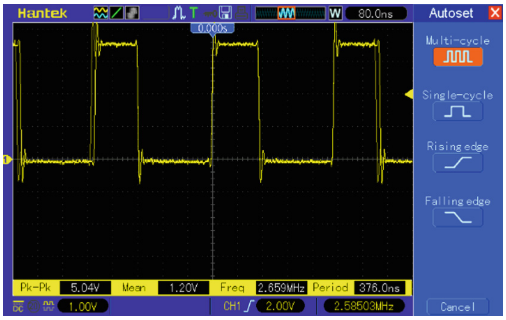
The results are in Table 3.

Table 3. Rise/fall period (Arduino C and MicroPython)

Platform	Period
Arduino C	376 ns
MicroPython	11,75 µs

Oscilloscope screenshot can be seen in Fig. 6.

In the next step, neural network code has been implemented two times after GPIO ON and GPIO OFF code on both platforms. The flowchart can be seen in Fig. 7.



80.0ns 0.000s

Pk-Pk 5.04V Mean 1.20V Freq 2.659MHz Period 376.0ns

DC 1.00V CH1 2.00V 2.58503MHz

Fig. 6. Oscilloscope screenshot

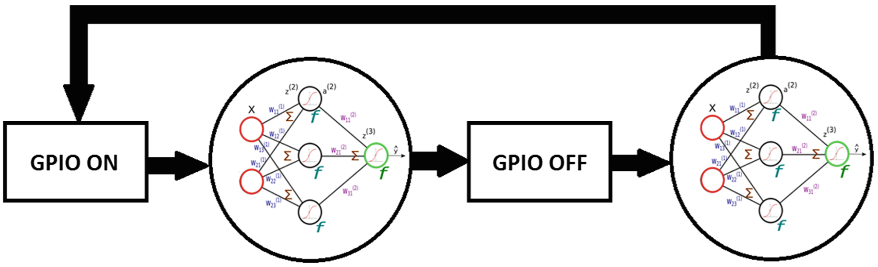


Fig. 7. Flowchart

Table 4. Final results

	Input layer	Hidden layer	Output layer	uC	Weights format	Period
1	50	50	10	Arduino	float	2,58 ms
2	50	100	10	Arduino	float	5,05 ms
3	50	150	10	Arduino	float	7,56 ms
4	50	200	10	Arduino	float	10,02 ms
5	50	50	10	Arduino	double	6,50 ms
6	50	100	10	Arduino	double	13,04 ms
7	50	150	10	Arduino	double	19,63 ms
8	50	200	10	Arduino	double	26,01 ms
9	50	50	10	MicroPython	float	153,84 ms
10	50	100	10	MicroPython	float	304,41 ms
11	50	150	10	MicroPython	float	423,91 ms
12	50	200	10	MicroPython	float	586,51 ms

Results are in Table 4. Arduino C can use two floating-point number precisions, and both of them are used in measuring. New MicroPython firmware support double-precision floating-point numbers but it was unavailable to us, and in Table 4 there are only four rows concerning MicroPython.

Results can also be seen in Fig. 8.

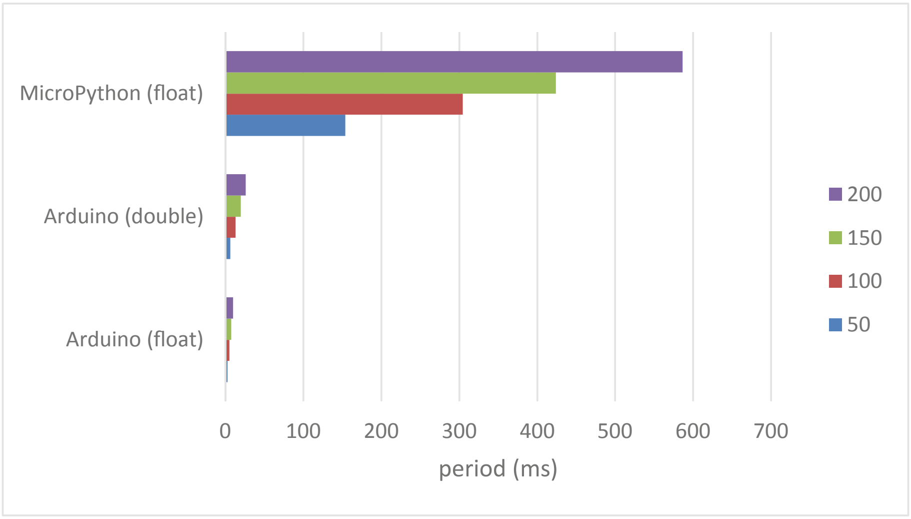


Fig. 8. Periods for different platforms and hidden nodes

5 Discussion and Conclusion

It can be seen that Arduino C based neural network has significant lower data propagation latency than MicroPython based neural network. Some authors indicated this, but they have tested propagation speed with integers. On the GitHub platform number of counts running for 10 s for three different setups can be found. MicroPython on Teensy 3.1 (96 MHz ARM) counts to 1,098,681 for 10 s, but Arduino C on Teensy 3.1: (96 MHz ARM) counts to 95,835,923 for the same period. Author used milis() function for time measuring [18].

Arduino IDE is faster platform than MicroPython for neural networks development on edge devices. This research can be expanded with other development environments, especially with ESP-IDF. Some authors suggested that floating-point neural networks can be replaced with integer neural networks when cost is primary design concern, so it would be interesting to test performance and speed with integer neural networks [19, 20].

References

1. Li, H., Ota, K., Dong, M.: Learning IoT in edge: deep learning for the Internet of Things with edge computing. IEEE Network **32**(1), 96–101 (2018)
2. Espressif System: ESP32 Overview. https://www.espressif.com/en/products/hardware/esp32/overview. Accessed 15 June 2019
3. Biswas, S.B., Iqbal, M.T.: Solar water pumping system control using a low cost ESP32 microcontroller. In: IEEE Canadian Conference on Electrical & Computer Engineering, Quebec City (2018)
4. Abdullah, A.H., et al.: Development of ESP32-based Wi-Fi electronic nose system for monitoring LPG leakage at gas cylinder refurbish plant. In: 2018 International Conference on Computational Approach in Smart Systems Design and Applications (ICASSDA), Kuching (2018)
5. Maier, A., Sharp, A., Vagapov, Y.: Comparative analysis and practical implementation of the ESP32 microcontroller module for the Internet of Things. In: 7th IEEE International Conference on Internet Technologies and Applications, Wrexham (2017)
6. Zidek, K., Janacova, D., Hosovsky, A., Pitel, J., Lazorik, P.: Data optimization for communication between wireless IoT devices and cloud platforms in production process. In: 3rd EAI International Conference on Management of Manufacturing Systems, Dubrovnik (2018)
7. Rosato, D., Masciadri, A.: Non-invasive monitoring system to detect siting people. In: Goodtechs, Bologna (2018)
8. Islam Chowdhuryy, M.H., Sultana, M., Ghosh, R., Ahamed, J.U., Mahmood, M.: AI assisted portable ECG for fast and patient specific diagnosis. In: International Conference on Computer, Communication, Chemical, Materials and Electronic Engineering, Rajshahi (2018)
9. Fernoaga, V.P., Stelea, G.-A., Balan, A., Sandu, F.: OCR-based solution for the integration of legacy and-or non-electric counters in cloud smart grids. In: IEEE 24th International Symposium for Design and Technology in Electronic Packaging (SIITME), Iași (2018)
10. Komarek, A., Pavlik, J., Mercl, L., Sobeslav, V.: Hardware layer of ambient intelligence environment implementation. In: Nguyen, N.T., Papadopoulos, G.A., Jędrzejowicz, P., Trawiński, B., Vossen, G. (eds.) ICCCI 2017. LNCS (LNAI), vol. 10449, pp. 325–334. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67077-5_31
11. Chand, G., Ali, M., Barmada, B., Liesaputra, V., Ramirez-Prado, G.: Tracking a person's behaviour in a smart house. In: Liu, X., et al. (eds.) ICSOC 2018. LNCS, vol. 11434, pp. 241–252. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17642-6_21
12. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.-C.: MobileNetV2: inverted residuals and linear bottlenecks. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City (2018)
13. Zhang, K., Zhang, Z., Li, Z., Qiao, Y.: Joint face detection and alignment using multi-task cascaded convolutional networks. IEEE Signal Process. Lett. **23**(10), 1499–1503 (2016)
14. Kokoulin, A.N., Tur, A.I., Yuzhakov, A.A., Knyazev, A.I.: Hierarchical convolutional neural network architecture in distributed facial recognition system. In: IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), Saint Petersburg and Moscow (2019)
15. Teerapittayanon, S., McDanel, B., Kung, H.: Distributed deep neural networks over the cloud, the edge and end devices. In: IEEE 37th International Conference on Distributed Computing Systems (ICDCS) (2017)
16. Zhang, Y., Suda, N., Lai, L., Chandra, V.: Hello edge: keyword spotting on microcontrollers. arXiv preprint arXiv:1711.07128 (2017)

17. Lai, L., Suda, N., Chandra, V.: CMSIS-NN: efficient neural network kernels for arm cortex-M CPUs. The Computing Research Repository (2018)
18. George Robotics Limited: Performance, 1 June 2014. https://github.com/micropython/micropython/wiki/Performance. Accessed 1 Sept 2019
19. Behan, T., Liao, Z., Zhao, L.: Integer Neural Networks On Embedded Systems. In: Recent Advances in Technologies. IntechOpen (2009)
20. Wang, N., Choi, J., Brand, D., Chen, C.-Y., Gopalakrishnan, K.: Training deep neural networks with 8-bit floating point numbers. In: 32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montreal (2018)