Face Mask Detector for Raspberry Pi Based on Computer Vision and Edge Computing

Zhexen Y. Seitbattalov[1], Hüseyin Canbolat[2], and Sabyrzhan K. Atanov[1]

[1] *L.N. Gumilyov Eurasian National University, 2 Kanysh Satbayev St., Astana, Z01A3D7, Kazakhstan*
[2] *Ankara Yildirim Beyazit University, Gazze Cd. No:7, Ayvali, Keçiören, Ankara, 06010, Turkey*

Abstract

The pandemic of the coronavirus disease 2019 has shown weakness and threats in various fields of human activity. In turn, the World Health Organization has recommended different preventive measures to decrease the spreading of coronavirus. Nonetheless, the world community ought to be ready for worldwide pandemics in the closest future. One of the most productive approaches to prevent spreading the virus is still using a face mask. This case has required staff who would verify visitors in public areas to wear masks. The aim of this paper was to identify persons remotely who wore masks or not, and also inform the personnel about the status through the message queuing telemetry transport as soon as possible using the edge computing paradigm. To solve this problem, we proposed to use the Raspberry Pi with a camera as an edge device, as well as the TensorFlow framework for pre-processing data at the edge. The offered system is developed as a system that could be introduced into the entrance of public areas. Experimental results have shown that the proposed approach was able to optimize network traffic and detect persons without masks. This study can be applied to various closed and public areas for monitoring situations.

Keywords

Internet of Things, Raspberry Pi, Edge device, pandemic, TensorFlow, message queuing telemetry transport

1. Introduction

The current world and economy have been affected by the coronavirus disease 2019 (COVID-2019) and due to worldwide lockdowns and shutdowns of businesses. According to statistics declared by the World Health Organization (WHO), it has registered 611.42 million confirmed cases of infection by COVID-2019 and 6.512 million cases of deaths by September 2022. This respiratory virus spreads primarily through close contact and in overcrowded areas [1]. Moreover, our early research has demonstrated that values of the temperature parameter of 5-9°C, the humidity of 30-50% and no ventilation are optimal for the virus activity [2]. As previous experience has shown that mass vaccinations, quarantines, social distancing, hand sanitizing and wearing respirators, surgical masks have had positive effects on the decrease of the COVID-2019 spreading [3]. All those measures had proved highly effective in reducing cases of deaths and patient hospitalizations.

However, the world community and the WHO should be prepared for similar global pandemics in the future. In most pandemics, it is essential to wear masks and keep up a secure distance between persons to guarantee that the infection does not spread. In this setting, it may be an exceptional approach to check whether individuals wear masks at the entrance to open spaces for preventive

Proceedings of the 7th International Conference on Digital Technologies in Education, Science and Industry (DTESI 2022), October 20-21, 2022, Almaty, Kazakhstan
EMAIL: sbtl.jeks@gmail.com (Zhexen Y. Seitbattalov); huseyin.canbolat@gmail.com (Hüseyin Canbolat); atanov5@mail.ru (Sabyrzhan K. Atanov)
ORCID: 0000-0003-2607-4908 (Zhexen Y. Seitbattalov); 0000-0002-2577-0517 (Hüseyin Canbolat); 0000-0003-2115-7130 (Sabyrzhan K. Atanov)

purposes [1]. The presence of a remote system that will perform this control instead of security staff would be a way to decrease costs and provide health safety to personnel.

Recent studies have shown that most face detection systems use various approaches for image processing and computer vision such as Convolutional Neural Networks (CNN), Deep Learning (DL), Keras, OpenCV, faster regions with CNN, Sensor Fusion (SF) [1, 3, 4, 5, 6, 7, 8]. F. Ozyurt, in contrast to other studies, has introduced additional parameter to measure the body temperature of visitors through the MLX90614 sensor and used MobileNetV2 architecture. The accuracy of his proposed approach has reached about 97 percent [1]. S. Meivel has focused on monitoring streets by drones and measuring the social distance between people through faster regions with CNN and YOLOv3 algorithms [3]. P. Sertic has offered a hardware accelerated system using DL and tested it on three embedded platforms: Raspberry Pi 4B with either Google Coral USB TPU or Intel Neural Compute Stick 2 VPU, and NVIDIA Jetson Nano. The best accuracy has been achieved on the Jetson Nano platform and equals 94.2 percent [6]. Besides using an Enhanced MobileNetV2 for mask detection, R. K. Shinde has collected heart rate, body temperature and oxygen level to avoid mistakes in defining the infected people [7].

Nonetheless, the considered studies above have ignored the network bandwidth and traffic issues since those proposed systems accumulate an enormous amount of data because of the video stream and sensor values. Those generated data often sends to the cloud for storage and processing. For instance, a drone with camera and aircraft engines in Boeing 787 generate more than 20 gigabytes of sensor data per hour [9].

The edge computing paradigm handles the bottleneck of cloud computing for various latency-sensitive Internet of Things (IoT) applications by proposing computing resources closer to the sources of data [10, 11, 12]. Those studies have demonstrated scenarios where edge computing dominates over traditional cloud computing, for example, in cryptographic performance [11, 12, 13, 14].

The aim of the study was to develop the algorithm of face mask detection through computer vision, and after that provided prompt transmission data to an end user using edge computing and message queuing telemetry transport (MQTT).

2. Materials and Methods

To reduce the cost of the developed system, we proposed to use a Raspberry Pi 4 Model B (4 Gigabyte) as an edge device with an OmniVision OV5647 camera, which is shown in Figure 1.

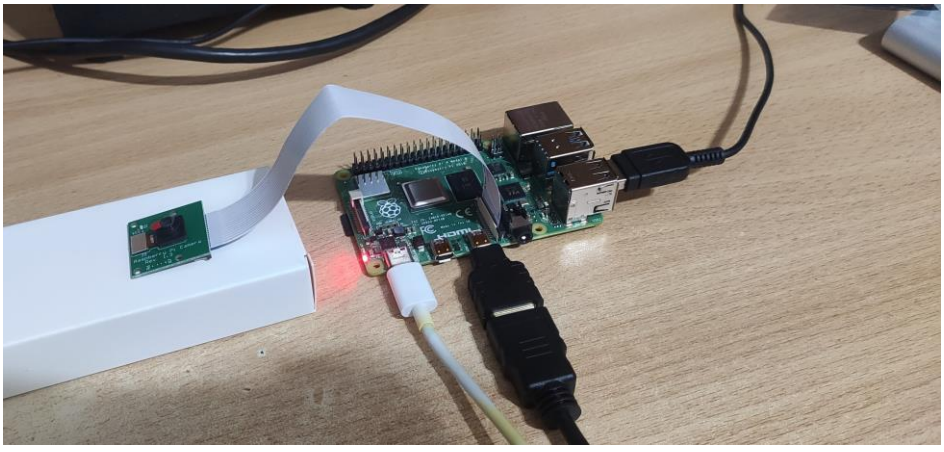

Figure 1: Edge Device (Raspberry Pi and OmniVision OV5647 Camera)

Raspberry Pi is a single-board computer that can find applications in both cloud and edge computing paradigms. Other pros of the Raspberry Pi are the intermediate speed processor, the presence of a peripheral interface and network support, which provides the opportunity to gather and analyze received data from IoT devices [15].

Raspberry Pi OS (Raspbian) has been selected as an operating system for Raspberry Pi.

As we noted earlier, the MQTT has been used as a messaging protocol between the edge device (publisher, Aedes Broker) and end users (subscribers, MQTTBox) [16]. The broker layer scheme is demonstrated in Figure 2, where the MQTT broker has been applied to transmit text data to the end user devices (smartphones, laptops and personal computers).

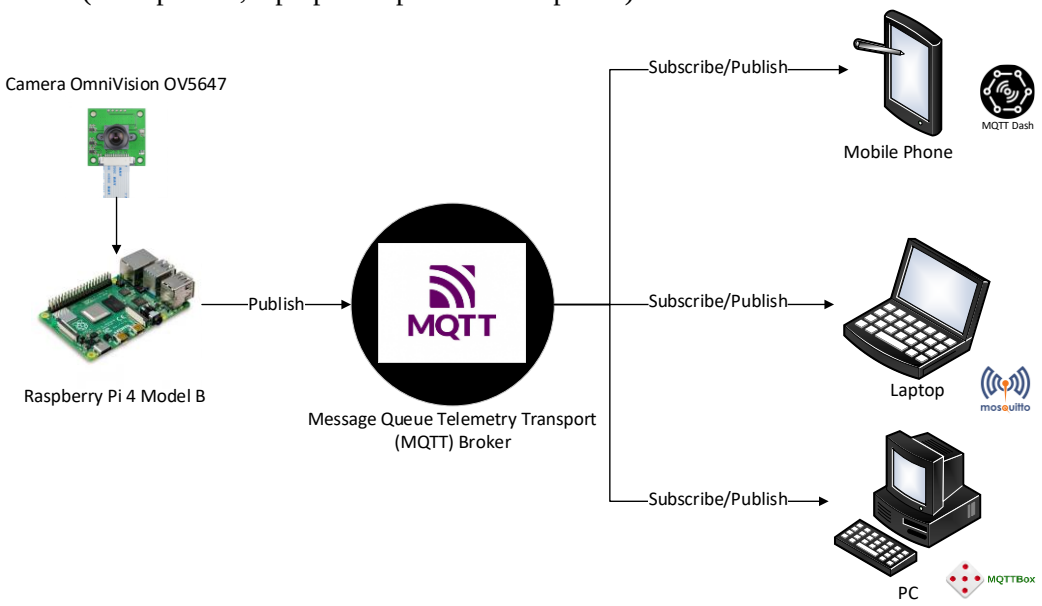

Figure 2: The Scheme of Broker Layer

2.1. Service layer

This layer was responsible for receiving data from the broker layer in which Raspberry Pi was applied as a service or a platform. We have configured Node Red to organize the data flow and visual programming for IoT devices, and also created a possibility for subscribers to obtain messages from publisher. We offered to apply a personal computer based on the Windows operation system for a subscriber. The scheme of the service layer is illustrated in Figure 3.

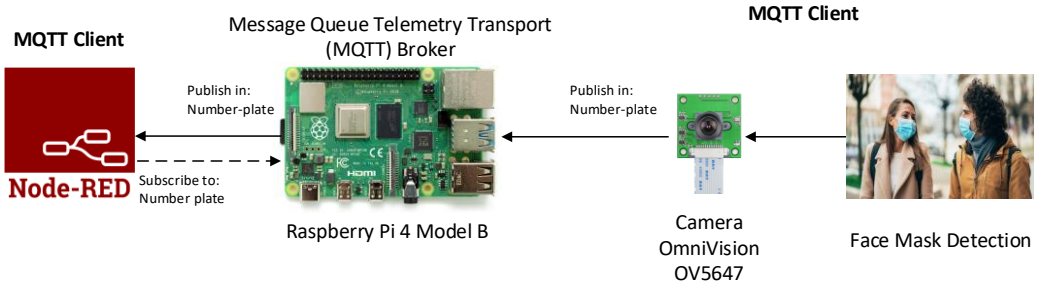

Figure 3: The Scheme of Service Layer

The OmniVision OV 5647 camera transferred video stream data to the Raspberry Pi. A single-board computer has used computer vision algorithms to detect the face mask and transmitted it to the server side in text format via the MQTT message broker based on Red Note.

2.2. Application layer

This layer has been used to develop the back-end of the application and provide an output for an operator to control the entrance. OpenCV, TensorFlow, Imutils, picamera and numpy libraries of Python language were used to code the algorithm for face mask detection [16, 17]. For training a

model of face mask detection we have used a dataset of 5774 images, which has been divided into two classes: mask-wearing images and do not wear masks [18]. Examples of those images are demonstrated in Figure 4.

Figure 4: Dataset Examples

Figure 5 presents a flowchart of the face mask detection process.

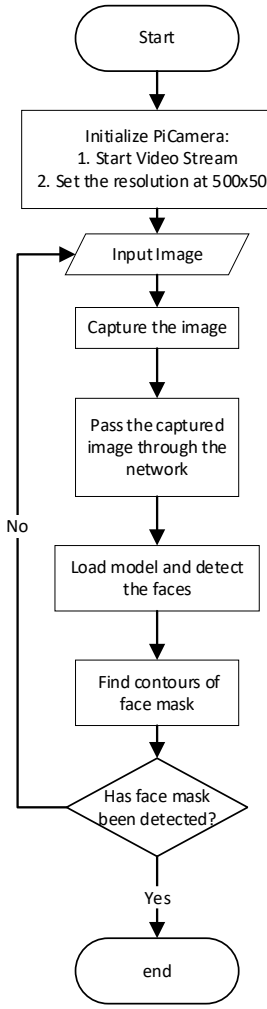

Figure 5: The Flowchart of Face Mask Detection

At the first stage, the Raspberry pi camera was initialized. Its resolution was set at 500x500 and started the video streaming process. Then started capturing frames from the Raspberry Pi camera and

constructing a blob. We have passed that blob through the network: load serialized face detector model from disk and detect the faces. Find contours of face mask and output label with a rectangle. At the final step, after receiving data about face mask detection, Raspberry pi transmits this data to subscribers.

The aedes broker, inject, exec, function, mqtt out, mqtt in, and debug nodes were used from the Node Red's menu to create the face mask detector flow. Figure 6 illustrates the flow diagram.

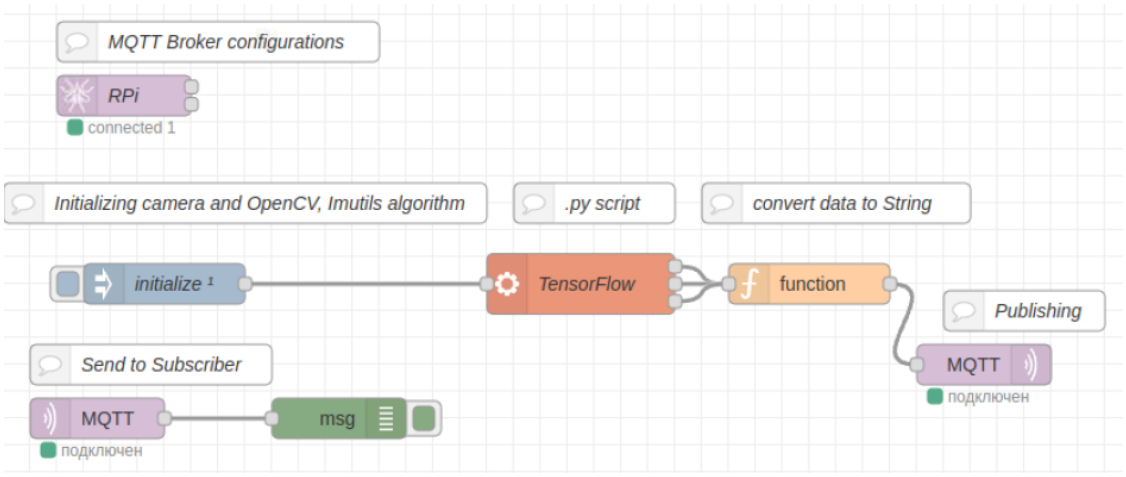

Figure 6: Flow in Node Red

The aedes broker (RPi) node allows user to configure the message broker for the Raspberry Pi and specify its port. The inject node initialized the work of the flow, the camera, and launches the TensorFlow node (computer vision algorithm), which is implemented in the Python programming language for face mask detection [19, 20]. The results are transferred to the function node in order to translate the text into a readable form. The mqtt out and mqtt in nodes (MQTT) publish data to the address of subscribers. The msg node allows user to display the result.

Figure 7 shows an example of displaying the face mask detection in the terminal.

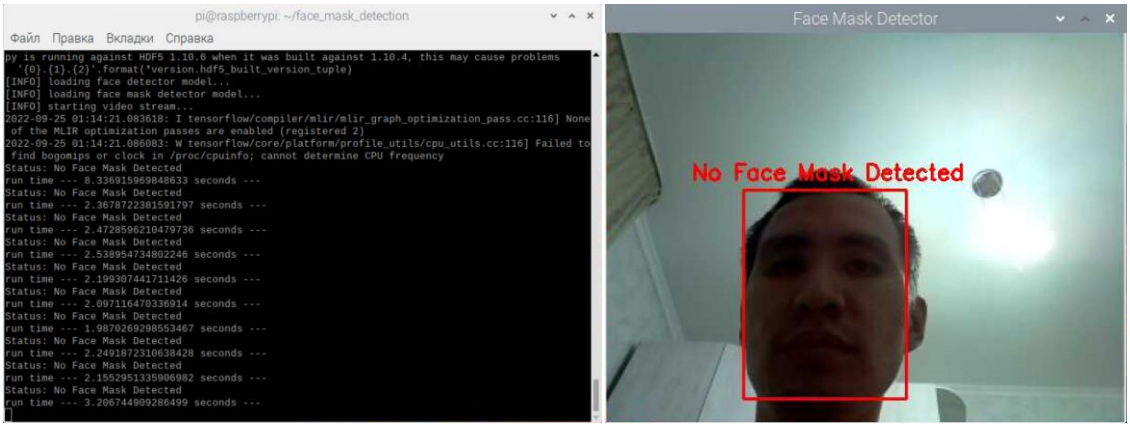

Figure 7: The Result of Face Mask Detection in the Terminal

3. Experimental Results and Discussion

Figure 8 shows the results of processing images in the terminal windows as the text. The image captured by the OmniVision OV 5647 is displayed in the Frame window.

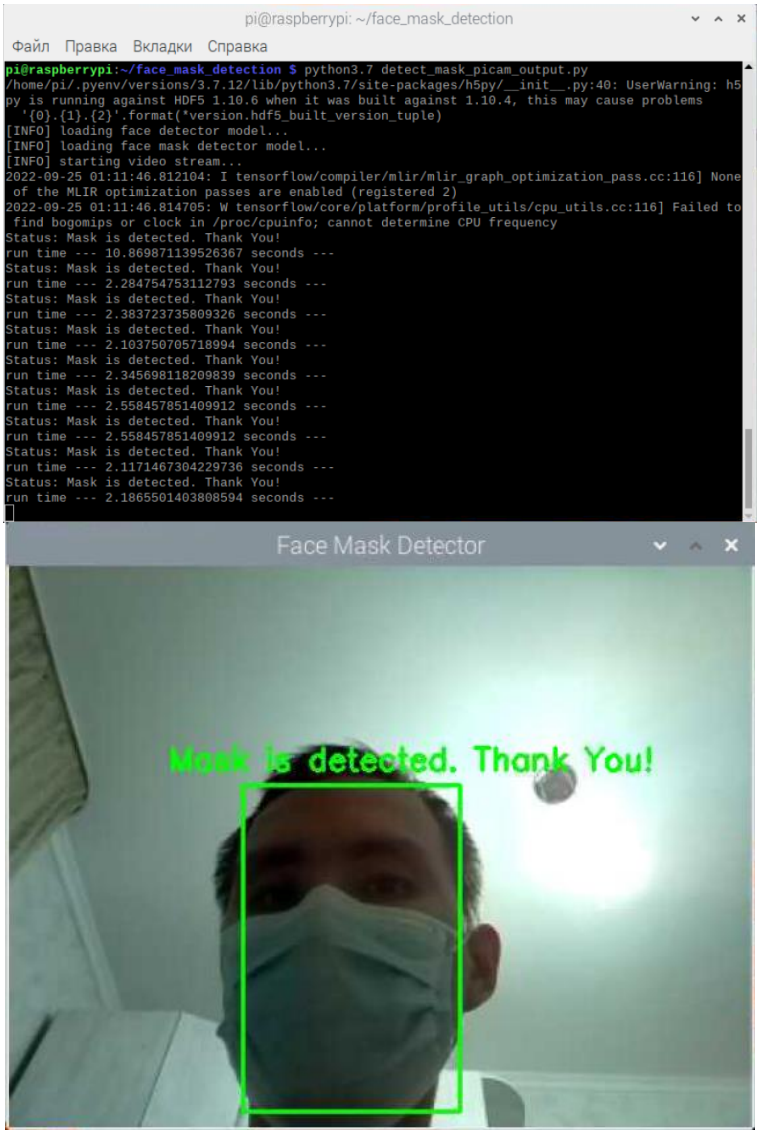

Figure 8: The Result of Face Mask Detection in the Terminal

The debug tab of the Node Red window shows the result of received data from Picamera in Figure 9. And also the results of data transmission to publishers are shown in the MQTTBox's window.

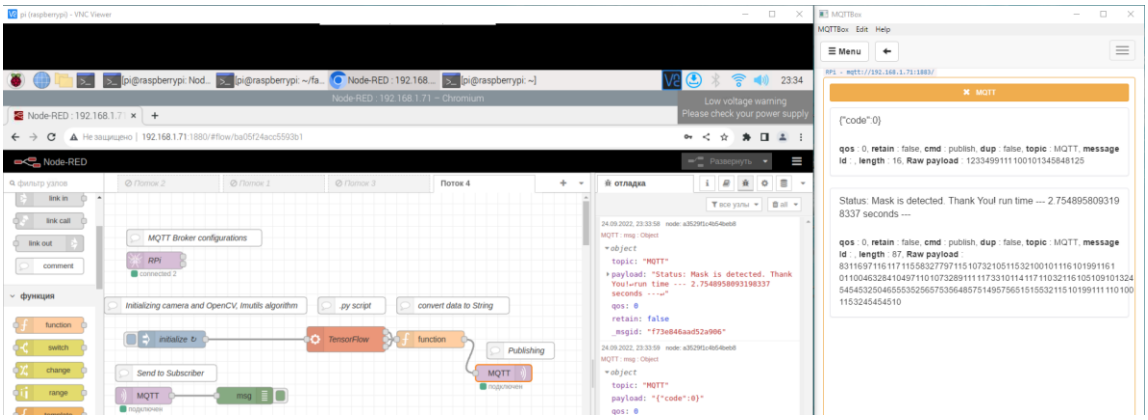

Figure 9: The Result of Processing Images in Node Red and Transferring Data to Subscriber

The average time to execute the algorithm was 2.5 seconds.

Table 1 presents various image processing methods, platforms, resolution and execution time in milliseconds [16].

Table 1

Analysis of the comparison of execution time of different image processing algorithms in central processing unit

Image processing method	Platform	Resolution	Execution time, milliseconds
High Definition Number Plate Localization	Zynq-7000 SoC	960×720 (HD)	16.17
Standard Definition Number Plate Localization	FPGA Virtex-4	640×480 (SD)	6.7
Oily style NPR	CPU: Q9450 with 42.56 GFLOPS GPU: NVIDIA G92 (GeForce 9800 GTX) with 128 Cores and 512 MB Video Memory, GTX 280 for Speed UP Test	512×512	7172
Corners and edge detection	OS: Ubuntu 11.04 CPU: Dual Core 6600, 2.40 GHz, Mem: 2 GB GPU: GeForce GTX 280, 240 CUDA cores, Memory: 1GB GPU: Tesla C1060, 240 CUDA cores, Memory: 4GB	2048×2048	4006
Content authentication	CPU Intel Xeon 5520 (2.26GHz) RAM 12GB DDR3 (1333MHz) GPU Architecture Tesla C1060 OS Centos 5.3 (64 bit) V2.3 CUDA	1024×1024	28877.66
Linear feature extraction	CPU: Q9450 with 42.56 GFLOPS GPU: NVIDIA G92 (GeForce 9800 GTX) with 128 Cores and 512 MB Video Memory, GTX 280 for Speed UP Test	1800×1400	500.958685
Inverse sinusoidal contrast transformation	AMD Phenom II Quad-core to 3.2 GHz, 12 GB of RAM, Operating System: 64-bit Linux Fedora 14 GPU: GeForce 430 GT video card with 96 cores and 1 GB of RAM DDR3 is used	1024×1024	151.256079

Thus, the execution time of various image processing methods on field-programmable gate arrays (FPGAs) is actually faster than on a Raspberry Pi. However, it should be noted that the execution time of algorithm on a Raspberry Pi is slightly inferior to the computing power of a personal computer when its processing an image by a central processing unit.

To measure the load on network traffic for transferring data to the message broker, we have used the NetWorx, which is shown in Figure 10.

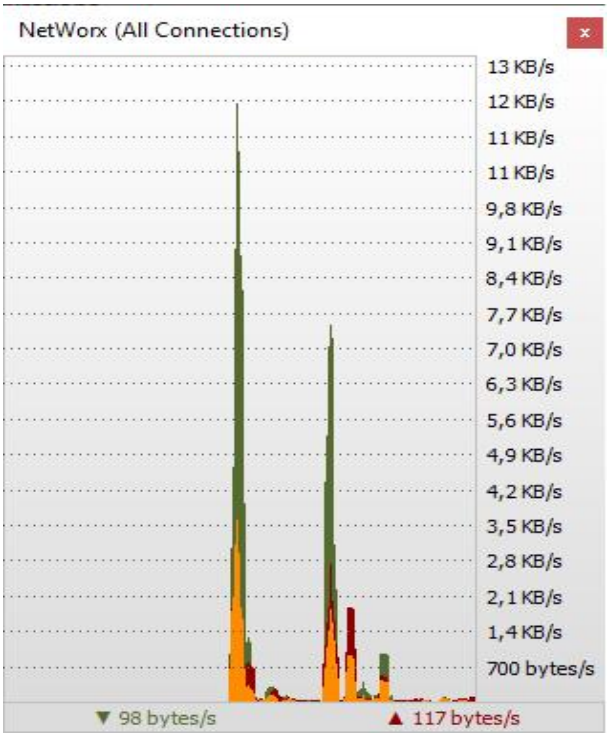

Figure 10: A graphic diagram of network traffic for transferring data to a message broker

The peak value of the network traffic was approximately 11.9 kilobits per second. Thus, it can be noted that the process of data transmission using the edge computing paradigm contributes to the optimization of network traffic, since under the cloud paradigm, the amount of transmitted sensor data is measured in megabits per second and gigabits per second.

4. Conclusion

In this study, we presented an edge device based on Raspberry Pi 4 Model B with camera OmniVision OV 5647 for face mask detection that takes the benefit of edge computing devices for data pre-processing. The recommended solution gives low-cost computation to the IoT network edge and also can be extended by integrating new modules and sensors. This not only decreases the computational price but also optimizes the network traffic compared to the cloud solution. Moreover, we have added an option for remote control through MQTT Broker.

For the future research, we are going to include in the developed system a new sensor for measuring body temperature and an algorithm that works similarly to the thermographic camera for comparison.

5. References

[1] F. Ozyurt, A. Mira, and A. Coban, Face Mask Detection Using Lightweight Deep Learning Architecture and Raspberry Pi Hardware: An Approach to Reduce Risk of Coronavirus Spread While Entrance to Indoor Spaces, Traitement Du Signal 39.2 (2022) 645–650. doi: 10.18280/ts.390227.

[2] Z. S. Moldabayeva, P. Schmidt, Z. Y. Seitbattalov, T. A. Amankeldinov, and A. E. Kyzyrkanov, Simulation of An Intelligent Detection System Virus SARS-CoV-2 in Enclosed Spaces, in: Proceedings of the 2021 16th International Conference on Electronics Computer and Computation, ICECCO, 2021. doi: 10.1109/icecco53203.2021.9663840.

[3] S. Meivel, N. Sindhwani, R. Anand, D. Pandey, A. A. Alnuaim, A. S. Altheneyan, M. Y. Jabarulla, and M. E. Lelisho, Mask Detection and Social Distance Identification Using Internet of

Things and Faster R-CNN Algorithm, Computational Intelligence and Neuroscience (2022) 2103975. doi: 10.1155/2022/2103975.

[4] M. I. Amin, M. A. Hafeez, R. Touseef, and Q. Awais, Person Identification with Masked Face and Thumb Images under Pandemic of COVID-19, in: Proceedings of the 7th International Conference on Control, Instrumentation and Automation, ICCIA, Univ Tabriz, Fac Elect & Comp Engn, Tabriz, IRAN, Feb 2021, pp. 273-276. doi: 10.1109/iccia52082.2021.9403577.

[5] P. Nagrath, R. Jain, A. Madan, R. Arora, P. Kataria, and J. Hemanth, SSDMNV2: A real time DNN-based face mask detection system using single shot multibox detector and MobileNetV2, Sustainable Cities and Society, 66 (2021) 102692. doi: 10.1016/j.scs.2020.102692.

[6] P. Sertic, A. Alahmar, T. Akilan, M. Javorac, and Y. Gupta, Intelligent Real-Time Face-Mask Detection System with Hardware Acceleration for COVID-19 Mitigation, Healthcare, 10.5 (2022). doi: 10.3390/healthcare10050873.

[7] R. K. Shinde, M. S. Alam, S. G. Park, S. M. Park, and N. Kim, Intelligent IoT (IIoT) Device to Identifying Suspected COVID-19 Infections Using Sensor Fusion Algorithm and Real-Time Mask Detection Based on the Enhanced MobileNetV2 Model, Healthcare 10.3 (2022). doi: 10.3390/healthcare10030454.

[8] K. Suganeswaran, N. Nithyavathy, S. Arunkumar, K. Dhileephan, P. Ganeshan, and A. J. Antony, Machine Learning based Mechanism for Crowd Mobilization and Control, in: Proceedings of the 6th International Conference on Inventive Computation Technologies, ICICT 2021, pp. 1334-1339. doi: 10.1109/icict50816.2021.9358631.

[9] S. K. Atanov, Z. Y. Seitbattalov and Z. S. Moldabayeva, Development an Intelligent Task Offloading System for Edge-Cloud Computing Paradigm, in: Proceedings of the 2021 16th International Conference on Electronics Computer and Computation, ICECCO, November 2021. doi: 10.1109/icecco53203.2021.9663797.

[10] R. Mahmud and A. N. Toosi, Con-Pi: A Distributed Container-Based Edge and Fog Computing Framework, IEEE Internet of Things Journal 9.6 (2022) 4125-4138. doi: 10.1109/jiot.2021.3103053.

[11] T. Gizinski, and X. Cao, Design, Implementation and Performance of an Edge Computing Prototype Using Raspberry Pi, in: Proceedings of the 2022 IEEE 12th Annual Computing and Communication Workshop and Conference, CCWC, 2022, pp. 592-601. doi: 10.1109/ccwc54503.2022.9720848.

[12] D. Hawthorne, M. Kapralos, R. W. Blaine, and S. J. Matthews, Evaluating Cryptographic Performance of Raspberry Pi Clusters, in: Proceedings of the IEEE High Performance Extreme Computing Conference, HPEC, Sep 2020. doi: 10.1109/HPEC43674.2020.9286247.

[13] N. Rathour, Z. Khanam, A. Gehlot, R. Singh, M. Rashid, A. S. AlGhamdi, S. S. Alshamrani, Real-Time Facial Emotion Recognition Framework for Employees of Organizations Using Raspberry-Pi, Applied Sciences-Basel 11.22 (2021). doi: 10.3390/app112210540.

[14] H. Yar, A. S. Imran, Z. A. Khan, M. Sajjad, and Z. Kastrati, Towards Smart Home Automation Using IoT-Enabled Edge-Computing Paradigm, Sensors 21.14 (2021). doi: 10.3390/s21144932.

[15] S. S. Brimzhanova, S. K. Atanov, M. Khuralay, K. S. Kobelekov, and L. G. Gagarina, Cross-platform compilation of programming language Golang for Raspberry Pi, in: Proceedings of the 5th International Conference on Engineering and MIS, ICEMIS 2019, vol. 10, June 2019. doi: 10.1145/3330431.3330441.

[16] Z. Y. Seitbattalov, H. Canbolat, Z. S. Moldabayeva, and A. E. Kyzyrkanov, An Intelligent Automatic Number Plate Recognition System Based on Computer Vision and Edge Computing, in: Proceedings of the 2022 IEEE International Conference on Smart Information Systems and Technologies, 2022 IEEE SIST, April 2022. To appear.

[17] Y. Arslan and H. Canbolat, Performance of Deep Neural Networks in Audio Surveillance, in: 2018 6th International Conference on Control Engineering & Information Technology, CEIT, 2018. doi: 10.1109/ceit.2018.8751822.

[18] D. O. Toybazarov, S. K. Atanov, Z. R. Burnayev, G. N. Baiseitov, and D. D. Kassenov, Modeling and Cluster Analysis of Antagonistic Systems, in: Proceedings of the 2021 16th International Conference on Electronics Computer and Computation, ICECCO, November 2021. doi: 10.1109/icecco53203.2021.9663751.

[19] A. E. Kyzyrkanov, S. K. Atanov, and S. Aljawarneh, Coordination of movement of multiagent robotic systems, in: Proceedings of the 2021 16th International Conference on Electronics Computer and Computation, ICECCO, November 2021. doi: 10.1109/icecco53203.2021.9663796.
[20] Z. Y. Seitbattalov, S. K. Atanov, and Z. S. Moldabayeva, An Intelligent Decision Support System for Aircraft Landing Based on the Runway Surface, in: 2021 IEEE International Conference on Smart Information Systems and Technologies, 2021 IEEE SIST, 2021. doi: 10.1109/sist50301.2021.9466000.