IoT-based System for COVID-19 Indoor Safety Monitoring

Nenad Petrović and Đorđe Kocić

Abstract—**In this paper, we introduce an affordable IoT-based solution aiming to increase COVID-19 indoor safety, covering several relevant aspects: 1) contactless temperature sensing 2) mask detection 3) social distancing check. Contactless temperature sensing subsystem relies on Arduino Uno using infrared sensor or thermal camera, while mask detection and social distancing check are performed by leveraging computer vision techniques on camera-equipped Raspberry Pi.**

***Index Terms*—Arduino; computer vision; coronavirus; COVID-19; Raspberry Pi, ontology.**

I. Introduction

Since the last days of the previous year, the occurrence of novel infectious flu-alike respiratory disease *COVID-19* caused by *SARS-Cov-2* virus (also known as *coronavirus*) has affected almost every aspect of people's lives globally. First, it was discovered in China, but spread quickly to other continents in just few weeks. According to [1], until July 11th, 2020, the total number of identified cases was 12,653,451, while taking 563,517 lives worldwide.

Common symptoms of coronavirus disease include fever, tiredness, sore throat, nasal congestion [2], loss of taste and smell [3]. In most cases, it is transmitted directly (person to person) through respiratory droplets, but also indirectly via surfaces [4, 5]. Incubation period could be quite long and varies (between 14 and 27 days in extreme cases) [6, 7]. Furthermore, even asymptomatic persons (almost 45% of cases) can spread the disease [7] making the situation even worse. Therefore, the usage of face masks and sanitizers has shown positive results when it comes to disease spread reduction [8]. However, the crucial problem is the lack of approved vaccine and medication [9].

Due to these facts, many protection and safety measures were taken by governments in order to reduce the disease spread, such as obligatory indoor mask wearing, social distancing, quarantine, self-isolation, limiting citizens' movement within country boarders and abroad, often together with prohibition and cancellation of huge public events and gatherings [10]. Despite the fact that the pandemic seemed weaker at some points, most of safety regulations are still applied due to unstable situation. From workplace behavior to social relations, sport and entertainment, coronavirus disease poses many changes to our everyday routine, habits and activities.

In this paper, cost-effective IoT-based system aiming to help organizations respect the COVID-19 safety rules and guidelines in order to reduce the disease spread is presented. We focus on most common indoor measures - people with high body temperature should stay at home, wearing mask is obligatory and distance between persons should be at least 1.5-2 meters. For the first scenario, Arduino Uno microcontroller[1] board with contactless temperature sensor is used, while we rely on Raspberry Pi[2] single-board computer equipped with camera making use of computer vision techniques for other two scenarios. We decided to use these devices due to their small size and affordability.

II. Background

A. *OpenCV*

Python version of OpenCV [11], open-source computer vision library was used for implementation of mask detection and social distance check algorithms. We decide to use it, as it was approved for usage with older Raspberry Pi devices [12].

Face and body detection algorithms rely on the existing OpenCV implementation of Viola-Jones object detection framework based on Haar feature cascades [13]. It is a machine learning approach where cascade function is trained from a large set of positive and negative images. After that, this function is used to detect objects in new images. OpenCV comes with both trainer and detector. However, OpenCV offers pre-defined classifiers for detection of commonly used objects, such as human face, whole body, body and face parts (both front and back for some of them). Therefore, in this paper, we leverage the existing classifiers provided by OpenCV library, as they were enough to cover satisfy the needs of the implemented solution.

In [12], face detector provided by OpenCV library was used for control of multimedia reproduction systems based on Raspberry Pi devices within museums and cultural heritage sites, showing acceptable performance, even in real-time use cases.

B. *MQTT*

In this paper, MQTT (Message Queuing Telemetry Transport)[3] was used for machine-to-machine communication between the involved devices - Raspberry Pi, Arduino, Edge

Nenad Petrović is with the Faculty of Electronic Engineering, University of Niš, Aleksandra Medvedeva 14, 18000 Niš, Serbia (e-mail: nenad.petrovic@elfak.ni.ac.rs).

Đorđe Kocić is with the Faculty of Electronic Engineering, University of Niš, Aleksandra Medvedeva 14, 18000 Niš, Serbia (e-mail: seriousdjoka@gmail.com).

[1] https://store.arduino.cc/arduino-uno-rev3
[2] https://www.raspberrypi.org/
[3] http://mqtt.org/

servers and smartphones. It is a lightweight, publish-subscribe messaging protocol on top of TCP/IP. MQTT is designed for use cases where small code footprint is desired or network bandwidth is limited, which is suitable for IoT solution leveraging low-power computing devices presented in this paper. Moreover, publish-subscribe messaging mechanism requires a message broker. For that purpose, we use a Node.js MQTT broker implementation within Node-RED[4] deployed on a server residing within the Edge. For IoT devices, corresponding MQTT client libraries were used – PubSubClient[5] for Arduino, Paho-MQTT[6] for Raspberry Pi and Paho Android Service[7] for smartphones. The devices measuring body temperature, detecting masks and social distancing send MQTT messages to Edge servers in cases when a person does not satisfy conditions to pass some of the safety check steps. Furthermore, the Edge server processes the received messages and forwards them to corresponding security workers to notify them about breaking of COVID-19 safety rules. Each message is sent in a form of JSON-encoded string.

C. *Semantic knowledge representation*

The role of semantic technology is to enable encoding the meaning of data separately from the content itself and related applications, which provides the ability to understand data, exchange its understanding and perform reasoning on top of it [14]. In this case, the formalization of knowledge is done in a form that is understandable for both humans and machines. Within the semantic knowledge bases, the data is represented with respect to ontologies.

In IoT systems, ontologies are often used to achieve interoperability and integrate data coming from heterogeneous devices and their sensors in order to enable their control in unified way [15]. For example, in [16], we introduced Smart Grid Response Ontology to enable reasoning about the events that occurred within the IoT-based system for smart grid monitoring and generate adequate response in a particular context. In this paper, we adopt similar approach to semantic representation in context of COVID-19 safety monitoring, but extend it with the elements of spatial reasoning.

III. Related Work

There are several existing works that contain some of the elements relevant to the work presented in this paper. However, to the best of our knowledge, there is no such solution covering all these aspects together to achieve this goal while allowing execution on low-cost IoT devices at the same time.

In [17], a dataset for masked face recognition is introduced and its application by different algorithms in context of campus and enterprise coronavirus prevention discussed. Moreover, in [18], a high-accuracy method for facial mask detection using semantic segmentation based on fully convolutional networks, gradient descent and binomial cross entropy was presented. However, performance-wise, it is too heavy for low-power IoT devices, such as Raspberry Pi. On the other side, in [19], a state model-based solution for face mask detection relying on Viola-Jones algorithm in context of ATM center security was described.

When it comes to temperature sensing, there are several variants of Arduino-based solutions. In [20], Arduino was used for real-time temperature visualization using MATLAB. However, the used sensor does not allow contactless temperature sensing. Moreover, in [21], a similar system incorporating the usage of smartphones for remote temperature monitoring using Arduino Uno was presented.

The system architecture presented in this paper is inspired by our previous work on remote smart grid monitoring anomaly, power consumption monitoring and relay protection using IoT devices (smartphones and Arduino Uno) [16] and video surveillance system relying on Raspberry Pi single-board computers and Edge servers [22]. Our main goal is to provide a comprehensive solution for COVID-19 safety monitoring which relies on IoT devices as much as possible in order to be affordable at the same time.

IV. Implementation

A. *System overview*

Our solution consists of the following subsystems: 1) temperature measurement subsystem based on Arduino Uno 2) computer vision subsystem for mask detection and social distancing check based on Raspberry Pi 3) server side 4) smartphone application for security guards.

First, all people that try to the enter building have to pass contactless temperature check. For that purpose, we rely on Arduino Uno equipped with infrared thermometer (such as MLX90614[8]) or thermal camera sensor (AMG8833[9] for example). Moreover, it uses ESP8266 WiFi module for communication with Edge servers using MQTT protocol. In case that person has body temperature higher than normal, the door is locked and MQTT message sent to server, containing both the temperature value and location where it was recorded. Server receives this message, parses it and forwards to smartphone application used by security guards, so they can arrive to make sure that person does not try to enter the building further. Otherwise, if passenger's temperature is normal, Arduino will send signal to open the door.

After that, passengers proceed to next step of checking – mask detection. For this task, computer vision subsystem based on Raspberry Pi single-board computer equipped with camera module version 1[10] revision 3 was used. In case that passenger does not wear mask or it does not cover nose, security guards will be informed via MQTT message, so they can provide a mask or warn that person to leave. Otherwise, if

[4] https://nodered.org/
[5] https://github.com/knolleary/pubsubclient
[6] https://pypi.org/project/paho-mqtt/
[7] https://github.com/eclipse/paho.mqtt.android

[8]https://maker.pro/arduino/projects/build-an-infrared-thermometer-arduino-and-mlx90614
[9]https://learn.adafruit.com/adafruit-amg8833-8x8-thermal-camera-sensor/arduino-wiring-test
[10] https://www.raspberrypi.org/documentation/hardware/camera/

the person that is being checked wears mask, the door will be opened. Furthermore, once they enter the building, Raspberry Pi devices check whether social distancing is applied properly or not at given locations. In a similar way, MQTT message will be sent to inform the security guards when social distancing is not applied properly in some of the rooms.

On the server side, the MQTT broker and semantic triple store are deployed, while message processing, event logging, reasoning and message forwarding are done. Edge servers receive messages, perform their semantic annotation and reasoning to find the right security guard that will be notified. A simple Android mobile application used by security guards receives MQTT messages from server side and visualizes the data about rule violation and location where it occurred within the building. In Fig. 1, an overview of the proposed IoT-based solution that aims to ensure that COVID-19 safety guidelines are applied properly indoors is given.

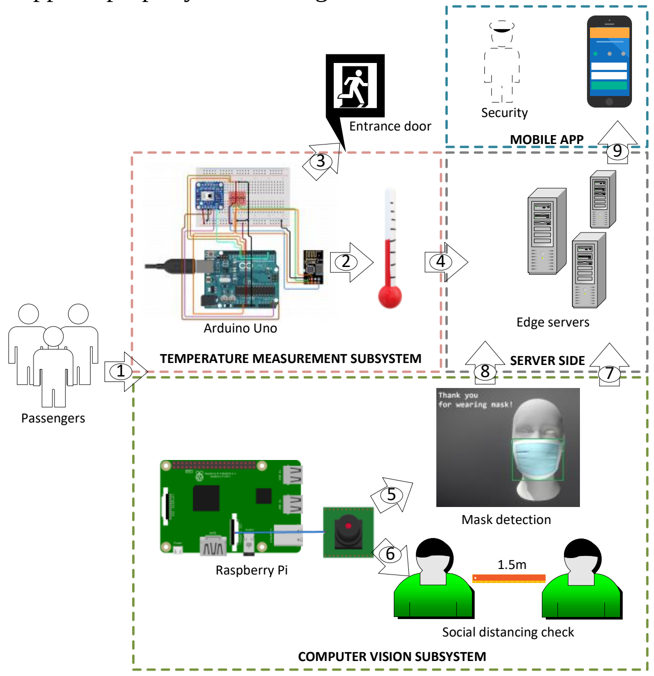

Fig. 1. COVID-19 indoor safety IoT system overview: 1-Passenger arrival 2-Temperature value 3-Door open/close signal 4-MQTT message warning that someone has higher body temperature than average 5-Wears mask/doesn't wear mask 6-Social distancing satisfied/not satisfied 7-MQTT warning message telling that person without mask tries to enter 8-MQTT warning message that passengers do not respect social distance measures 9-MQTT notification messages sent to security worker's smartphone about people breaking COVID-19 safety measures at various building rooms.

B. *Mask detection algorithm*

For implementation of mask detection algorithm, we rely on three OpenCV library classifiers[11]: 1) *haarcascade_frontalface_default* – which is used for detection of human face from frontal side 2) *haarcascade_mcs_mouth* – recognizes human mouth within the provided image 3) *haarcascade_mcs_nose* – to detect nose. For each frame coming from camera stream, the procedure given in Listing 1 is executed.

```
Input: image
Output: label
Steps:
1. gray_image = ConvertToGray(image);
2. bw_image = ConvertToBw(gray_image);
3. faces = DetectFaces(gray_image);
4. faces_bw = DetectFaces(bw_image);
5. if(faces.length==0 and faces_bw.length==0)
6.     label = "No face found!"
7. else
8.     if(faces_length==0 and faces_bw.length>0)
9.         mouths = DetectMouth(bw_image);
10.        noses = DetectNose(bw_image);
11.    else
12.        mouths = DetectMouth(gray_image);
13.        noses = DetectNose(gray_image);
14.    end if;
15.    if(mouths.length==0 && noses.length==0)
16.        label = "Thank you!"
17.        OpenDoor();
18.    else
19.        for each m in mouths
20.            for each f in faces
21.                if(m.y> y and m.y<f.y+f.h and m.x>x   and m.x<f.x+f.w)
22.                    label = "Please put mask on!";
23.                    sendMQTT("no mask", "location name");
24.                end if;
25.            end for;
26.        end for;
27.        for each n in noses
28.            for each f in faces
29.                if(n.y> y and n.y<f.y+f.h and n.x>x   and n.x<f.x+f.w)
30.                    label = "Please put mask over nose!";
31.                    sendMQTT("improper mask", "location name");
32.                end if;
33.            end for;
34.        end for;
35.    end if;
36. end if;
37. return label;
38. end.
```

Listing 1. Pseudo-code of mask detection algorithm.

First, the camera frame is converted to a gray scale image which is used for face detection, as it required by the OpenCV's Haar cascade classifier. Moreover, additional black and white version of camera frame is also created as a new copy. It was empirically noticed that in most cases of person wearing a white mask, OpenCV classifier cannot identify the face correctly if gray scale image is used, but performs better with black and white images instead.

After that, face detection is performed against both images. In case that array length of detected faces is 0 in both cases, then it is assumed that there isn't any human present within the camera's view. Otherwise, if a face was detected, mouth and nose detection are further applied to the corresponding camera frame version. In case that image does not contain mouth and nose, it means that person wears mask properly and corresponding door will be opened. However, if mouth is detected and its coordinates are within the area of the detected face, then the person is warned that mask is needed in order to proceed. If nose is detected within face area, then the person is warned to put the mask properly (covering nose). The algorithm is working in both single- or multi- person mode. In single-person mode, it is assumed that people pass one by one

[11] https://github.com/opencv/opencv/tree/master/data/haarcascades

near the mask detection system's camera. In case of multi-person mode, the check whether mouth and nose coordinates lie inside a face is done for each pair of face-mouth and face-nose pair, as more than one person can be allowed to pass near camera at once. In Fig. 2, a screenshot of successful mask detection on Raspberry Pi model 2B is given.

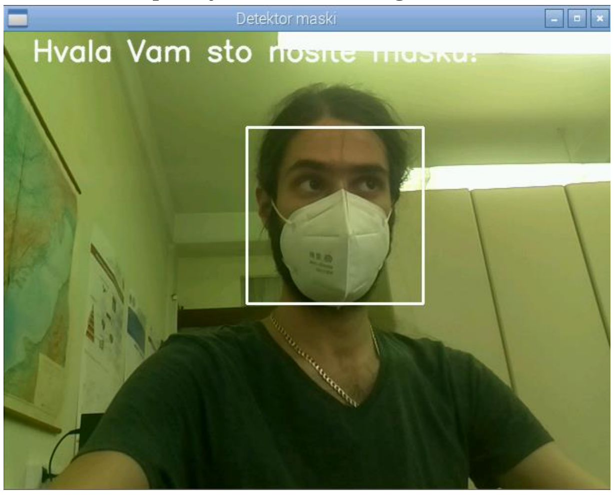

Fig. 2. Mask detection on Raspberry Pi 2B.

C. *Social distancing check algorithm*

When it comes to social distancing check algorithm, it leverages OpenCV's *haarcascade_fullbody* classifier for human body detection within the captured image. In Listing 2, the previously described algorithm for social distancing check based on computer vision is given.

```
Input:  image, threshold_d
Output:  label
Steps:
 1. gray_image=ConvertToGray(image);
 2. bodies=DetectFaces(gray_image);
 3. if(bodies.length≤1)
 4.     text="Not enough people for check!"
 5. else
 6.     for each b1 in bodies
 7.         for each b2 in bodies
 8.             d=sqrt((b1.x-b2.x)^2 +(b1.y-b2.y)^2);
 9.             d_m=ConvertPixelsToMeters(d);
10.                 if(b1 ≠ b2 and d_m< threshold_d)
11.                     label="Social distancing not applied!";
12.                     sendMQTT("social distancing alert", "location name");
13.               end if;
14. end if;
15. return label;
16. end.
```

Listing 2. Pseudo-code of social distancing check algorithm.

In a similar way as mask detection algorithm, each camera frame is converted to a gray scale image. Furthermore, body detection is applied. If more than one human body is detected, the distance between each two persons is calculated and compared against a given threshold distance *threshold$_d$*, given in meters. However, all distances should be normalized depending on the camera characteristics and object position before comparison. The mapping of pixels to real world distances in meters is performed with respect to formula [23]:

$$\frac{image\ dimension}{focal\ length} = \frac{object\ dimension}{distance\ to\ object} \quad (1)$$

If the distance between each two bodies is greater or equal to *threshold$_d$*, then social distancing is applied correctly in a given scenario. Otherwise, if this condition does not hold for at least one pair of bodies, then the message will be sent to the server and security operator notified. Fig. 3 shows the screenshot of social distance check application.

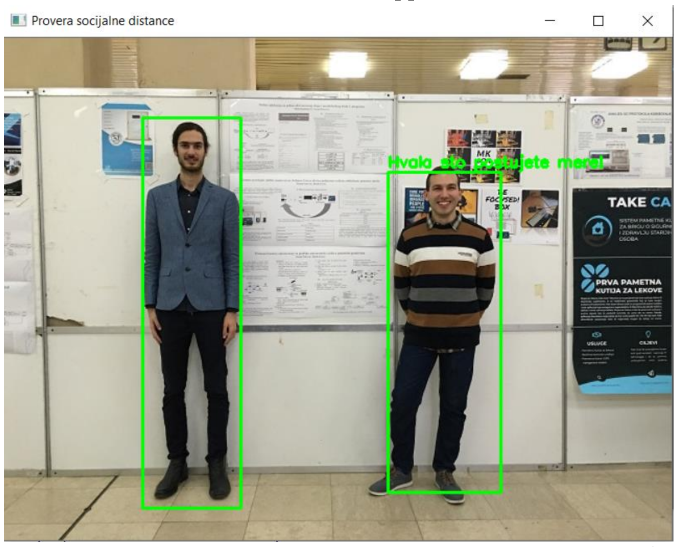

Fig. 3. The authors testing the social distancing check algorithm for threshold$_d$=1m.

D. *Temperature measurement*

In Listing 3, an excerpt of code running on Arduino is given.

```
void loop() {
    temp = sensor.readCelsius();
    if (temp > 37) {
            digitalWrite(13, HIGH);
            snprintf (msg, 100, "Temperature:#%ld;
location:faculty entrance", temp);
            client.publish("tempTopic", msg);
    }
    else {
            digitalWrite(13, LOW);
    }

}
```

Listing 3. Excerpt from temperature measurement code run on Arduino Uno positioned at faculty building entrance.

The temperature measurement subsystem based on Arduino Uno measures passenger's temperature using contactless IR sensor. The passengers pass one by one. In case that passenger's temperature exceeds average human body (37 °C), then Arduino Uno generates signal to lock the door in order to prevent the person from entering the building and sends MQTT message which tells that person with high body temperature was detected at a certain location. Otherwise, the door is opened to let the person in.

E. *COVID-19 Indoor Safety Monitoring Ontology*

The highest-level concept in this ontology is *Monitoring System*. It consists of heterogeneous *Devices*, such as Raspberry Pi, Arduino Uno and conventional laptop. Each device is equipped with different sensors and is able to detect different types of safety rule *Violations*. Several types of safety rule violations are considered: *Social Distancing*, *No Mask* and *High Temperature*. In case that some violation occurs, then the corresponding *Action* is taken as a response, such as closing *Door* or notifying *Security Guard*. For both the physical objects (*Device*, *Door*, *Security Guard*) and *Violation* events, the *Room* where it resides or occurs is relevant. Each *Room* is located on a *Floor*. This way, it is enabled to find the available *Security Guard* from the same *Floor* where *Violation* occurred and send him/her notification. Otherwise, the first guard that is available is selected. In Fig. 4, an excerpt from the described ontology is given.

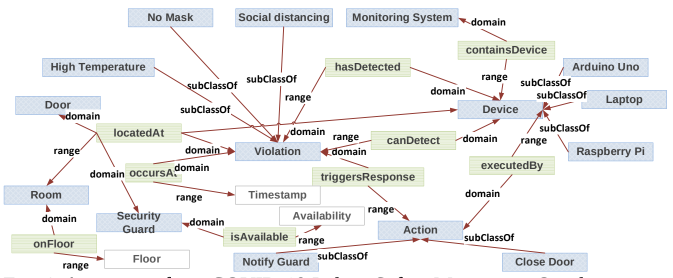

Fig. 4. An excerpt from COVID-19 Indoor Safety Monitoring Ontology.

In Listing 4., an example of SPARQL query for simple location matching in case of Violation of a type *Social Distancing* violation leveraging *Floor* is given.

```
PREFIX cismo: http://www.example.com/CISMO/
SELECT ?sg
WHERE {
  GRAPH <http://www.example.com/example1> {
    ?sg cismo:locatedAt ?r1.
    ?v  cismo:locatedAt ?r2.
    ?v  rdf:Type cismo:SocialDistancing.
    ?r1 cismo:onFloor ?f1.
    ?r2 cismo:onFloor ?f2.
    ?sq cismo:isAvailable 1.
        FILTER(?f1=?f2)
        }
}
```
Listing 4. SPARQL query for finding the security guard from the same floor where social distancing violation occurred.

V. Experiments and Evaluation Results

For evaluation, the following, the following devices were used: a laptop equipped with Intel i7 7700-HQ quad-core CPU running at 2.80 GHz with 16GB of DDR4 RAM and 1TB HDD acting as Edge server, Raspberry Pi 2B (RPi 2B), Raspberry Pi 3 (RPi 3) and Arduino Uno Rev. 3 with both IR sensor and thermal camera. In Table I, the results of performance evaluation for various scenarios, devices and settings are given. The first column denotes the name of the scenario (mask detection, distance check, contactless temperature measurement). The second column shows the hardware configuration used in the experiment. Moreover, the third column is the frame size expressed as number of horizontal multiplied by vertical pixels. Furthermore, the next column shows performance results achieved for given configuration expressed as number of processed frames (fps) or measurements per second (mps). Finally, the last column represents the accuracy achieved for the observed scenario. It is expressed as average percentage of successfully detected cases for mask detection and social distancing, while in case of temperature measurement it is average measurement error for corresponding sensor.

TABLE I
EVALUATION RESULTS

Scenario	Device	Frame size [W x H]	Frame rate [fps/mps]	Accuracy
Mask detection	RPi 2B	640x480	0.48	84-91%
		320x240	1.71	
	RPi 3	640x480	0.76	
		320x240	2.83	
	Laptop	640x480	11.94	
		320x240	38.46	
Distancing check	RPi 2B	640x480	0.72	65-73%
		320x240	2.65	
	RPi 3	640x480	1.12	
		320x240	4.29	
	Laptop	640x480	16.77	
		320x240	61.17	
Temperature sensing	IR (MLX90614)	1	8	0.5°C
	Thermal camera (AMG8833)	8x8	2	2.5°C

Considering the results shown in Table I, it can be concluded that RPi 3 has better performance than RPi 2B. It can be explained by the fact that RPi 3 utilizes newer quad-core ARM Cortex-A53 running at 1200 MHz then RPi 2B's 900 MHz ARM Cortex-A7. However, they both have only 1GB of RAM and their performance is still much behind the laptop. For lower resolutions, all devices show better performance, as expected.

Comparing the mask detection and social distancing check performance, we see that the second is faster as it only uses one classifier (full body), while mask detection uses three of them (face, nose and mouth). In all social distancing check experiments, the performance for two persons was evaluated, while it is expected to reduce as the number of people within the camera view increases. The performance of distancing check varies together with distance of objects from camera, as it changes with respect to initially calculated ratio between pixels and meters. The accuracy of both computer vision scenarios increases with resolution, but the cost is paid with performance decrease. Despite the acceptable accuracy of mask detection algorithm, it is not designed to detect transparent masks and face shields, which is a potential drawback.

On the other side, regarding the contactless temperature measurement, we can see that thermal camera is less accurate more demanding for computation, as it includes 64 measurements (8x8 matrix). However, its main advantage over IR sensor is the ability to measure the temperature of several persons at once, but requires additional data processing.

Finally, when it comes to time needed to find the appropriate security guard and generate MQTT message, in all the experiments, it does not exceed 1 second.

VI. Conclusion and Future Work

According to the achieved results, the proposed solution is usable for its purpose under certain performance limitations (such as number of processed frames or measurements per second). Moreover, it relies on both open hardware and free software, being definite and desirable advantage for such systems.

In future, it is planned to experiment with various deep learning and computer vision frameworks for object detection on Raspberry Pi in order to achieve higher framerate. Moreover, we would like to extend this solution with environment sensing mechanisms for adaptive building air conditioning and ventilation airborne protection in order to reduce the spread of coronavirus indoors [4, 8, 24], especially during summer. Furthermore, we will consider the implementation of mechanisms for transparent face shield detection. Finally, the ultimate goal is to integrate the system presented in this paper with our framework for efficient resource planning during pandemic crisis [25] in order to enable efficient security personnel scheduling and mask allocation, together with risk assessment based on statistics about respecting the safety guidelines and air quality.

Acknowledgment

This work has been supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia.

References

[1] Coronavirus Update (Live) [online]. Available on: https://www.worldometers.info/coronavirus/, last accessed: 11/07/2020.

[2] P. Zhai et al., “The epidemiology, diagnosis and treatment of COVID-19”, International Journal of Antimicrobial Agents vol. 55 issue 5, May 2020, 105955, pp. 1-13, 2020. https://doi.org/10.1016/j.ijantimicag.2020.105955

[3] P. Dawson et al., “Loss of Taste and Smell as Distinguishing Symptoms of COVID-19”, Clinical Infectious Diseases June 2020, pp. 1-4, 2020. https://doi.org/10.1093/cid/ciaa799

[4] L. Morawska, “How can airborne transmission of COVID-19 indoors be minimised?”, Environment International vol. 142, September 2020, 105832, pp. 1-7, 2020. https://doi.org/10.1016/j.envint.2020.105832

[5] T. Galbadage, B. Peterson, R. Gunasekera, “Does COVID-19 Spread Through Droplets Alone?”, Frontiers in Public Health, vol. 8, April 2020, pp. 1-4, 2020. https://doi.org/10.3389/fpubh.2020.00163

[6] Coronavirus Incubation Period [online]. Available on: https://www.worldometers.info/coronavirus/coronavirus-incubationperiod/ , last accessed: 11/07/2020.

[7] D. Oran, E. Topol, Prevalence of Asymptomatic SARS-CoV-2 Infection: A Narrative Review, Annals of Internal Medicine, June 2020, pp. 1-7, 2020. https://doi.org/10.7326/M20-3012

[8] T. Dbouk, D. Drikakis, “On respiratory droplets and face masks”, Physics of Fluids 32, 063303, pp. 1-11, 2020. https://doi.org/10.1063/5.0015044

[9] Y. Song et al., “COVID-19 Treatment: Close to a Cure? – A Rapid Review of Pharmacotherapies for the Novel Coronavirus” [preprint], pp. 1-25, 2020. https://doi.org/10.20944/preprints202003.0378.v1

[10] V. Balachandar et al., “COVID-19: emerging protective measures”, European Review for Medical and Pharmacological Sciences vol. 24 (6), pp. 3422-3425, 2020. https://doi.org/10.26355/eurrev_202003_20713

[11] Open Computer Vision [Online]. Available on: https://opencv.org/ , last accessed: 07/07/2020.

[12] N. Petrović, “Upravljanje multimedijalnim sistemom pomoću algoritma za detekciju lica na Raspberry PI arhitekturi u realnom vremenu”, IEEESTEC – 8th Student Projects Conference, Niš, Serbia, pp. 21-24, 2015.

[13] P. Viola, M. Jones, “Rapid Object Detection using a Boosted Cascade of Simple Features”, 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 511-518, 2001.

[14] J. Davies, R. Studer and P. Warren, *Semantic Web Technologies: Trends and Research in Ontology-based Systems*, John Wiley & Sons, 2006.

[15] A. Venceslau et al., “IoT Semantic Interoperability: A Systematic Mapping Study”, ICEIS 2019, pp. 535-544, 2019. https://doi.org/10.5220/0007732605350544

[16] N. Petrović, Đ. Kocić, “Data-driven Framework for Energy-Efficient Smart Cities”, Serbian Journal of Electrical Engineering, Vol. 17, No. 1, Feb. 2020, pp. 41-63. https://doi.org/10.2298/SJEE2001041P

[17] Z. Wang et al., “Masked Face Recognition Dataset and Application” [preprint], pp. 1-3, 2020. https://arxiv.org/pdf/2003.09093.pdf

[18] T. Meenpal, A. Balakrishnan, A. Verma, “Facial Mask Detection using Semantic Segmentation”, 2019 4th International Conference on Computing, Communications and Security (ICCCS), pp. 1-5, 2020. https://doi.org/10.1109/CCCS.2019.8888092

[19] M. Kavitha, S. M. M. Roomi, K. Priya, K. B. Devi, “State model based face mask detection”, International Journal of Engineering & Technology, 7 (2.22), pp. 35-38, 2018.

[20] R. Biswas, A. Roy, “Real Time Temperature Graph using MATLAB and Arduino”, International Journal of Engineering Research & Technology (IJERT) vol. 9 issue 5, pp. 624-625, 2020. https://doi.org/10.17577/IJERTV9IS050482

[21] M. J. Pramila, P. S. Shewta, “Wireless Temperature detector System using ARDUINO and IOT”, International Journal of Computer Trends and Technology (IJCTT) vol. 67 issue 11, pp. 82-83, 2019. https://doi.org/10.14445/22312803/IJCTT-V67I11P113

[22] N. Petrovic, “Surveillance System Based on Semantic Video and Audio Annotation Leveraging the Computing Power within the Edge”, XIV International SAUM 2018, pp. 281-284, 2018.

[23] How to calculate meters per pixel for a given camera? [online]. Available on: https://engineering.stackexchange.com/questions/32892/how-to-calculate-meters-per-pixel-for-a-given-camera, last accessed: 11/07/2020.

[24] R. V. Gomeseria, “Building Services Design to Prevent the Spread of COVID19”, pp. 1-18, 2020. https://doi.org/10.17605/OSF.IO/BQM2F

[25] N. Petrovic, Dj. Kocic, “Framework for Efficient Resource Planning in Pandemic Crisis”, CIIT 2020, pp. 1-6, 2020.