Smart technologies for COVID-19 indoor monitoring

11

Nenad Petrović[a] *and Đorđe Kocić*[b]
[a]Department of Computer Science, Faculty of Electronic Engineering, University of Niš, Niš, Serbia, [b]Department of Power Engineering, Faculty of Electronic Engineering, University of Niš, Niš, Serbia

11.1 Introduction

In the earliest days of COVID-19 pandemic, due to lack of information, vaccine and medication, most of the countries applied strict measures in order to reduce the spread of coronavirus. Most forms of public gatherings (both indoors and outdoors) were prohibited, while many countries applied partial or total lockdown. In case of partial lockdown, citizen movement was limited to specific time frame each day. As outcome, all shops except food supply and pharmacies remained closed, all social events were canceled (such as concerts), while most of the companies and educational institutions started performing their activities online. Moreover, several new indoor safety rules have emerged and are now becoming common practice and part of our everyday life, such as each person has to sanitize hands and shoes before entry, protective masks must be worn, and the distance between persons must be at least 2 m (Cirrincione et al., 2020). Apart from that, self-isolation in case of interaction with potentially infected persons and quarantine after traveling are also widely adopted (Tin, Hart, & Ciottone, 2021). However, together with these measures, several new issues occurred as well. First, there are always people who do not behave according to these constraints (intentionally or not), exposing themselves and others to potential COVID-19 infection. On the other hand, more strict government responses, such as lockdown could have catastrophic consequences on economy (Verma et al., 2021), especially in developing countries (USGLC, 2020) and places where tourism is crucial branch. Finally, psychological consequences, which occur as side effect of long lockdown periods, such as anger, irritability, emotional exhaustion, depression, and posttraumatic stress symptom, have an overall negative impact on the society (Chaudhury, 2020).

Regarding the current situation, as of March 2021, despite the intensive ongoing vaccination process in many countries around globe (Our World in Data, 2021), the number of new cases still remains quite high (Worldometer, 2021). However, more and more advocated strategy is to focus on safety rule compliance (either indoors or outdoors) and risk reduction, rather than strict lockdown measures. This way, it would be possible to perform usual activities in almost the same way as before, even in case of international traveling (Traskevich & Fontanari, 2021) under new circumstances, at least until vaccination takes effect. Therefore, the automation of indoor space

Viruses, Bacteria, and Fungi in the Built Environment: Designing Healthy Indoor Environments.
https://doi.org/10.1016/B978-0-323-85206-7.00012-5

surveillance and compliance with rules are still of utmost importance when it comes to battle against COVID-19. In this area, the state-of-the-art information technology plays a crucial role (Asadzadeh et al., 2020).

The main topic of this chapter is the adoption of smart technology for the purpose of COVID-19 indoor safety monitoring and rule compliance. The focus is on leveraging the processing capabilities of affordable all-present IoT devices (single-board computers, microcontrollers) and smartphones in synergy with the state-of-the-art concepts, such as computer vision and robotics. Moreover, several case studies developed at the University of Niš, Faculty of Electronic Engineering, aiming COVID-19 protection within faculty building, are presented. They cover the following aspects relevant to COVID-19 indoor safety: (1) mask detection, (2) person count, (3) social distancing, (4) automated touch-free hand sanitization, and (5) contact tracing. According to the research outcome, it is achievable to make use of small-size, low-power IoT devices for development of cost-effective indoor monitoring solutions, under certain limitations, mostly related to performance.

11.2 Background and related work

11.2.1 Internet of things

The Internet of Things (IoT) refers to a network of interconnected physical objects or devices (the so-called "things"), which are equipped with sensors, and variety of software technologies with main purpose of exchanging data with other devices and systems relying on Internet. Moreover, they can include different types of actuators, which give them the possibility to respond to the changes that occur in their environment. From domestic applications to manufacturing, fitness, and healthcare, these smart devices reside around us, aiming to enable automation of particular tasks from their domain of usage. However, in most cases, what is common to them is the small size, low energy consumption, but also much lower processing power compared to traditional computers, as well. Therefore, in order to achieve their goals, they often need to interact with other devices or servers. When it comes to machine-to-machine communication, various technologies are used, from short (NFC, Bluetooth) to long range (WiFi, LTE, 5G). Moreover, in IoT systems, lightweight communication protocols over TCP/IP are often used for message exchange, such as MQTT (Message Queuing Telemetry Transport)[a] that is based on publish-subscribe model.

When it comes to COVID-19 indoor safety monitoring, IoT devices play crucial role (Nasajpour et al., 2020). Many different use cases are described in the existing literature—from temperature check and cough detection (Das et al., 2021; Rasool et al., 2020) to visitor count (Das et al., 2021) and computer vision-based applications, such as mask detection and social distancing (Rahman et al., 2020; Vedant et al., 2021).

[a] https://mqtt.org/.

In this chapter, several different types of IoT devices are leveraged. Raspberry Pi[b] single-board computer is quite powerful device, but affordable at the same time, which is used together with camera module in scenarios making use of computer vision techniques—mask detection, person count, and social distancing. On the other hand, Arduino Uno,[c] a cheaper, but less powerful microcontroller is used together with WiFi module and corresponding sensors for simpler tasks, such as RFID person identification within contact tracing use case and contactless temperature measurement. Finally, when it comes to critical scenarios under extreme weather conditions—simpler, but more durable devices are used, such as PIC16[d] family microcontrollers or Intel 8086.

11.2.2 Smartphone apps

From education to transportation and entertainment, smartphone apps cover almost any aspect of our everyday life (Visconti, 2020). Since the beginning of last decade, smartphone devices have become much more powerful, almost as traditional personal computers, when it comes to multimedia applications. Moreover, these devices are now more affordable than ever before. Therefore, it is possible to use their full potential for wide adoption of novel use cases and scenarios involving the usage of augmented reality (AR), virtual reality (VR), artificial intelligence (AI), and computer vision.

During the COVID-19 pandemic, many novel smartphone apps have emerged, covering various aspects relevant to the battle against coronavirus (Petrović, Dimovski, Peterlin, Meško, & Roblek, 2021; Petrović & Kocić, 2020; Petrović, Radenković, & Nejković, 2020; Petrović, Tošić, & Nejković, 2021). Smartphone apps usage scenarios cover quick health state assessment, shopping assistance, volunteer help, test and vaccine scheduling, resource planning (hospital places, mask), cough detection, and many others.

In this chapter, smartphone companion apps are used to make indoor spaces COVID-19 safety monitoring more convenient. The presented applications visualize the measurements and results of data analysis collected by IoT devices (camera, temperature sensor). For that purpose, AppSheet[e] technology was used. It is an online platform and development environment for rapid, automated development and deployment of multiplatform mobile applications without almost any coding. AppSheet relies on Cloud for both computation resources and data storage (such as Google Sheets). According to our previous results, it dramatically speeds up the development of smartphone apps (Petrović et al., 2020), giving the ability to provide supportive applications timely, which can reduce risk and save lives during the pandemic. Moreover, AR.js[f] library for JavaScript was used to incorporate the elements of augmented reality within AppSheet applications. This lightweight framework runs without any problems

[b] https://www.raspberrypi.org/.
[c] https://store.arduino.cc/arduino-uno-rev3.
[d] https://www.microchip.com/wwwproducts/en/PIC16F870.
[e] https://www.appsheet.com/.
[f] https://ar-js-org.github.io/AR.js-Docs/.

even on older smartphone devices, while providing all the necessary features, such as marker recognition, 3D model loading, and animation. In this chapter, it is used for user interface within the robot monitoring patrol case study, following the successful adoption in AR-enabled robot companion apps (Petrović, Tošić, & Nejković, 2021).

11.2.3 Computer vision

Computer vision is an interdisciplinary research area within computer science, which aims the design and implementation of algorithms and methods enabling the computers to gain high-level understanding of digital images or video sequences in a similar way as humans do this task. In context of coronavirus pandemic, computer vision techniques are leveraged for both the prevention and diagnosis.

When it comes to prevention, the most notable use cases include mask detection, person count, and social distancing (Rahman et al., 2020; Vedant et al., 2021). On the other hand, it is also used for image-based patient assessment, relying on different image sources, such as X-ray, CT, and ultrasound (Ulhaq et al., 2020).

In this chapter, the focus is on the first group of use cases related to prevention in context of COVID-19 indoor safety monitoring. The algorithms are implemented using OpenCV[g] library for Python programming language and run on Raspberry Pi devices. For face, body, mouth, and nose detection, which are important subtasks of the presented computer-vision based use cases, we rely on the implementation of Viola-Jones object detection framework based on Haar feature cascades (Viola & Jones, 2001) offered by OpenCV. As an alternative approach covering transparent mask detection, Tiny-YOLO[h] for Python was also considered, giving higher mask detection accuracy, but lower framerate on Raspberry Pi devices (Rosebrock, 2020). This framework uses a single convolutional neural network (CNN) to detect multiple objects.

11.2.4 Blockchain and smart contracts

Blockchain refers to technology for decentralized transaction execution and storage of records holding the necessary information about them (such as sender, receiver, amount, timestamp). In these transactions, virtual digital tokens can be exchanged between the participant in order to retrieve some service or physical goods. For that purpose, a distributed ledger is used, which represents append-only sequence of data blocks holding the information about the transactions, stored in immutable, reliable, public, trackable and anonymous manner, relying on peer-to-peer network of computers (often called nodes). In order to take an effect, nodes within the network have to confirm the executed transaction using the consensus protocol. Apart from typical financial use cases, blockchain has been adopted in many novel scenarios, especially since the emergence of Ethereum[i] platform and Solidity[j] smart contracts. In the context

[g] https://opencv.org/.
[h] https://pjreddie.com/darknet/yolo/.
[i] https://ethereum.org/en/whitepaper/.
[j] https://docs.soliditylang.org/en/v0.8.3/.

of blockchain technology, smart contract refers to a software code that provides the definition of protocol necessary for execution of the desired transaction within the targeted platform (Zheng, Gao, Huang, & Guan, 2021).

In case of COVID-19 pandemic, blockchain and smart contracts have been mainly adopted for vaccination and health record storage (Deka, Goswami, & Anand, 2020). In this chapter, the focus is on usage of Ethereum blockchain and Solidity smart contracts for purpose of contact tracing when potentially infected persons are involved (Fusco, Dicuonzo, Dell'Atti, & Tatullo, 2020).

11.2.5 Semantic knowledge representation and ontologies

In computer systems, the purpose of semantic technology is to encode the meaning of data separately from executable code and content itself, in a form which is understandable for both humans and machines. This way, it becomes possible to understand data, exchange its understanding and perform reasoning on top of it conveniently. Within the semantic knowledge bases, the data are represented with respect to ontologies. Ontology is explicit formal description, which is used for the conceptualization of a particular domain. It consists of classes, individuals, attributes, and relations. Class refers to abstract group of objects that belong to same kind. Individuals represent instances of these classes. Attributes are properties of classes. Furthermore, relations define ways in which either classes or individuals can be related. For both the ontologies and facts, the RDF standard language is used. It defines (*subject, predicate, object*) triplets that are persisted within the semantic triple stores. On the other hand, SPARQL[k] is used as query execution against the semantic triple stores. The results are retrieved in order to support different reasoning mechanisms that enable the inference of new facts starting from the existing knowledge base. In computer systems, ontologies are often used to solve the issues related to interoperability (Petrovic & Tosic, 2019).

In this chapter, semantic knowledge representation with respect to ontologies is used for annotation of measurements and results obtained by heterogeneous IoT devices, together with their capabilities in order to achieve their interoperability and coordination regarding the indoor safety monitoring scenarios, relying on Nejkovic, Petrovic, Tosic, and Milosevic (2020).

11.3 Case studies

11.3.1 Entrance check: Contactless temperature and mask detection

When new visitor arrives at the building's entrance, the protocol which will be described has to be followed, once human face is detected by the entrance check system (illustrated in Fig. 11.1).

[k] https://www.w3.org/TR/rdf-sparql-query/.

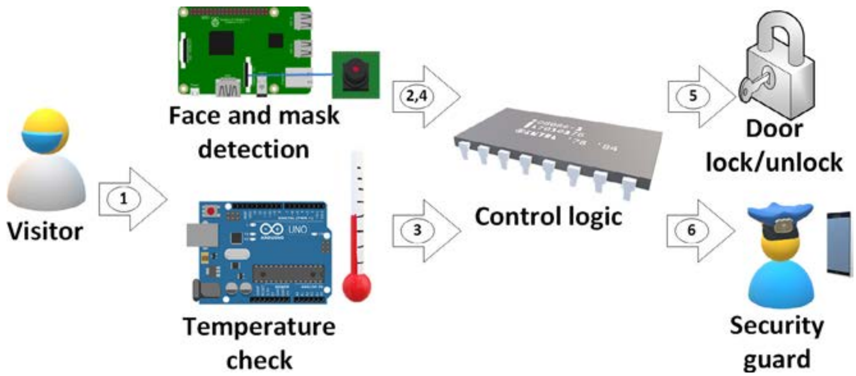

Fig. 11.1 Entrance check system performing mask detection and temperature check: (1) Arrival of new visitor at the entrance, (2) face detection outcome. (3) temperature check outcome, (4) mask check outcome, (5) door lock/unlock signal, and (6) notification for security guard.

After that, it is checked whether the temperature of the new visitor is normal (assumed to be below 37°C). In this case, we rely on Arduino Uno equipped with contactless temperature sensor, such as infrared-based MLX90614.[1] If the visitor passes this check successfully, the outcome signal is "1," while it is "0" otherwise. In case that the temperature is above normal, the door will be locked, and MQTT message will be sent to the edge server, containing the safety rule violation type (high temperature) and room/entrance number, together with timestamp. Moreover, the edge server inserts the information extracted from message into Google Sheets document behind the AppSheet-based mobile for security guards. This way, the security guards are notified about possible safety rule violations within the building.

Furthermore, if the visitor passes the temperature check, it is necessary to determine whether the person wears a mask or not. For that purpose, a computer vision algorithm is executed on Raspberry Pi 3 and implemented in Python using OpenCV, as described in Petrović and Kocić (2020). For face, mouth, and nose detection subtasks, it relies on predefined classifiers in OpenCV. This simple algorithm assumes that mask is worn properly when a human face is detected with both mouth and nose covered. In that case, the outcome of mask detection is "1," while it is "0" in case when the mask is not detected. If this check fails, a notification is sent to the security guard via AppSheet-based mobile app. In order to make this possible, Raspberry Pi inserts new record in Google Sheets document behind the app (using Google Sheets API for Python), containing the room number, timestamp, violation type (no mask), and link to the image taken by camera during the mask check. However, it was already stated (Petrović and Kocić, 2020) that the main weak point of this approach is transparent mask detection. In order to overcome the issue, Tiny-YOLO was used and trained on our dataset containing 1000 images for each of the classes: "no-mask," "mask," and "transparent-mask." Despite the fact that the previously

[1] https://maker.pro/arduino/projects/build-an-infrared-thermometer-arduino-and-mlx90614.

mentioned issue can be solved this way, the first approach using OpenCV is still more suitable for execution on Raspberry Pi.

When it comes to control logic, it is executed on PIC16 family microcontroller or simple microprocessor (such as Intel 8086), as described in Petrović (2020, 2021). The decision to use simpler hardware for this task is due to their low power consumption, small size, reliability, and durability even under extreme weather conditions. The main role of the control unit is to receive the temperature and mask check outcome signals and open the door only in the case when both conditions are satisfied—the new visitor wears a mask and has a normal body temperature. Moreover, it considers the number of persons currently inside the building. Even if the new visitor passes the check, the door will remain locked in cases when the number of persons currently inside the building is above predefined limit. In that case, the new visitor has to wait until at least one person exits, which is implemented as interrupt decreasing the current number of persons, as shown in Petrović (2021). In Listing 11.1, the pseudocode of the algorithm executed on the main control unit is described. It is implemented as interrupt, which is triggered when the face of a new visitor is detected by Raspberry Pi.

The mobile application for security guard notification is described in Petrović and Kocić (2020) and Petrović et al. (2020), while its screenshots are given in Fig. 11.2.

11.3.2 Touch-free automatic hand sanitation using PIC16 microcontroller

Alcohol-based hand sanitizers also play an important role in COVID-19 spread reduction (Abuga & Nyamweya, 2021). Since the beginning of the pandemic, most organizations have started to provide free hand sanitization within their buildings in different forms—bottles, sprays, or button-based dispensers. However, the risk of infection still

Listing 11.1 Pseudocode executed as interrupt on main control unit responsible for entrance check.

```
Input: face_detected, mask_outcome, temperature_outcome, person count, person_limit
Output: door lock/unlock
Steps:
when(face_detected)
    if(temperature_outcome==1 and mask_outcome==1)
        if(person_count==person_limit)
            unlock_door=false;
        else
            unlock_door:=true;
            person_count++;
        end if;
    else
        unlock_door:=fale;
    end if;
end when;
end.
```

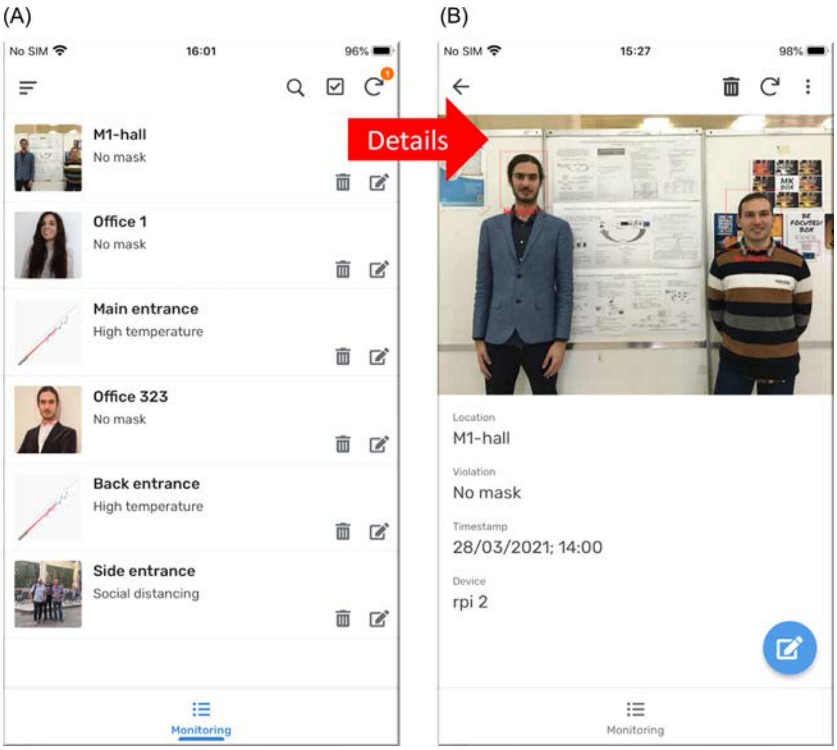

Fig. 11.2 AppSheet-based security guard notification app: (A) list of all violations and (B) detailed view for "no mask" violation.

remains high, as these packages still need to be touched by hands in order to be used. A solution to this issue are so-called touch-free dispensers, which do not require any physical contact with their users.

In this chapter, a touch-free automatic hand sanitation solution using PIC16F870 microcontroller equipped with an ultrasonic sensor SRF04 for distance measurement is presented. The choice was made due to its small size, affordability, and high reliability at the same time. Hand sanitation is the next step after temperature and mask checks. Once these two checks are passed, an interrupt is triggered via pin RB0, so the distance measurement mode is activated for next 30 s. This way, the risk of ultrasonic exposure is reduced, as sensor only works within short timeframe after the new visitor's arrival. The time since the person's arrival is measured using the internal TMR0 interrupt, which is configured to be triggered by register overflow each second. In this mode, the microcontroller will be in a distance measurement loop. Once the nearest object is closer than 5 cm from the ultrasonic sensor, it is assumed that the person has placed hands nearby and the dispenser device connected to PIC via pin RB1 will be activated for 3 s. Otherwise, the dispenser will remain closed. In Fig. 11.3, a block diagram of the implemented system within the Proteus simulation environment is shown. In this diagram, the first green LED (D1) corresponds to the activated state

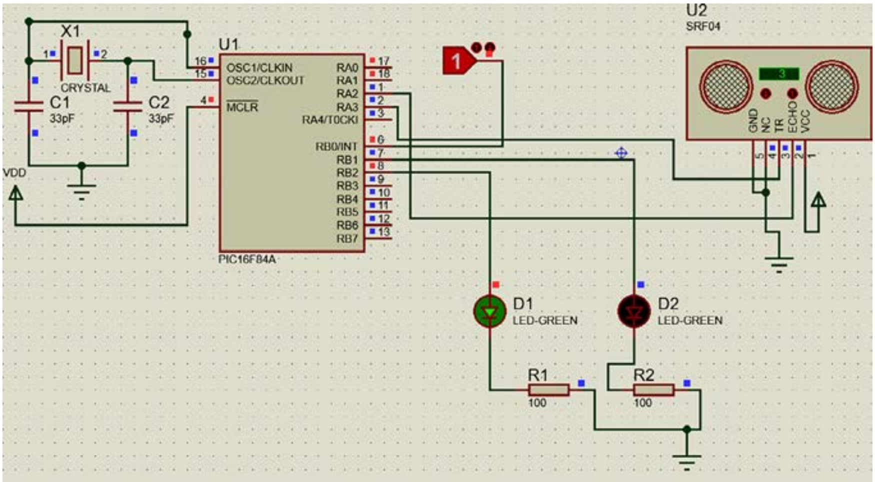

Fig. 11.3 PIC16-based touch-free hand sanitization system in Proteus simulation environment.

of dispenser, while if it is turned off, it means that the distance is above the threshold, and so the dispenser will remain closed. Moreover, in order to reduce the risk of COVID-19 infection, the visitors are not allowed to proceed without sanitizing their hands first, and so the door will be unlocked for 30 s once their hands are detected using an ultrasonic sensor (the second green LED, denoted as D2).

In Listing 11.2, the pseudocode of algorithm written in C programming language (XC8 compiler) and executed on PIC16F870 is presented. The distance measurement method relying on a SRF04 ultrasonic sensor is described in Petrović (2020).

11.3.3 RFID-based contact tracing relying on Arduino Uno

One more important aspect related to COVID-19 spread prevention indoors is tracing of contacts involving the infected persons (Flood, Chan, Chen, & Aspinall, 2021). In this chapter, an approach leveraging RFID-based identification of persons within the building is proposed. We decide to rely on RFID, as it is already adopted in huge number of institutions (among them, faculties and book libraries) for the purpose of person identification and presence measurement. When it comes to hardware implementation, Arduino Uno equipped with affordable RC522 RFID reader and ESP8266 WiFi module was used. In Fig. 11.4, an overview of the module responsible for person identification is given.

For that purpose, each room within the building should have such component controlling the entrance door installed. When a new visitor arrives, RFID card with identifier owned by organization should be provided in order to enter the room. The identifiers are stored in array of bytes, while each 4 bytes belong to distinct person. Once person provides a valid RFID card, MQTT message containing the identifier, room number, and time stamp will be sent to edge server via WiFi and stored into

Listing 11.2 Algorithm for smart hand sanitizer control executed on microcontroller.

```
Input: new_person_arrival
Output: dispenser_on, door_unlock
Steps:
dispenser_on:=false;
door_unlock:=false;
when(new_person_arrival is detected via RB0)
    mode:=1;
    time_since_arrival:=0;
end when;
when(tmr0 overflow)
    increment time_since_arrival;
end when;
…
while(1)
    while (mode==1 and time_since_arrival<30)
        if(distance<5cm)
            dispenser_on:=true;
            wait(3 sec);
            dispenser_off:=false;
            door_unlock:=true;
            wait(30 sec);
            door_unlock:=false;
        end if;
    end while;
end while;
end.
```

semantic database, while the door will be unlocked. Otherwise, if a person does not provide valid RFID card, the door will remain locked. In Listing 11.3, pseudocode of algorithm executed on Arduino Uno device within contact tracing system is shown.

Furthermore, in case that person who visited some of the rooms later turns out to be coronavirus-positive, the organization's administration should be notified via mobile app by the infected person herself/himself. After that, all of the visitors who were in the same room with the infected one in the last 7 days before her/his last visit will be notified via SMS or mobile app, in case that person's mobile number is not available (assuming that organizations had provided the companion mobile app previously). On the other hand, if the infected person has not visited the building at least 7 days before the infection, notifications are not sent. In this case, an algorithm shown in Listing 11.4 will be executed on the server in order to retrieve all the potentially infected persons, generate notification messages, and send them to target destination.

Once the list of all the potentially infected persons is retrieved, it is stored on Ethereum blockchain by executing the Solidity smart contract shown in Listing 11.5. For each of the contacts, the date when the interaction occurred together with the room ID is also stored on blockchain as a part of transaction. This way, it is possible to keep track of persons under high risk of COVID-19 in a transparent, public and immutable manner, but without giving out their identity. However, in order to make use of this

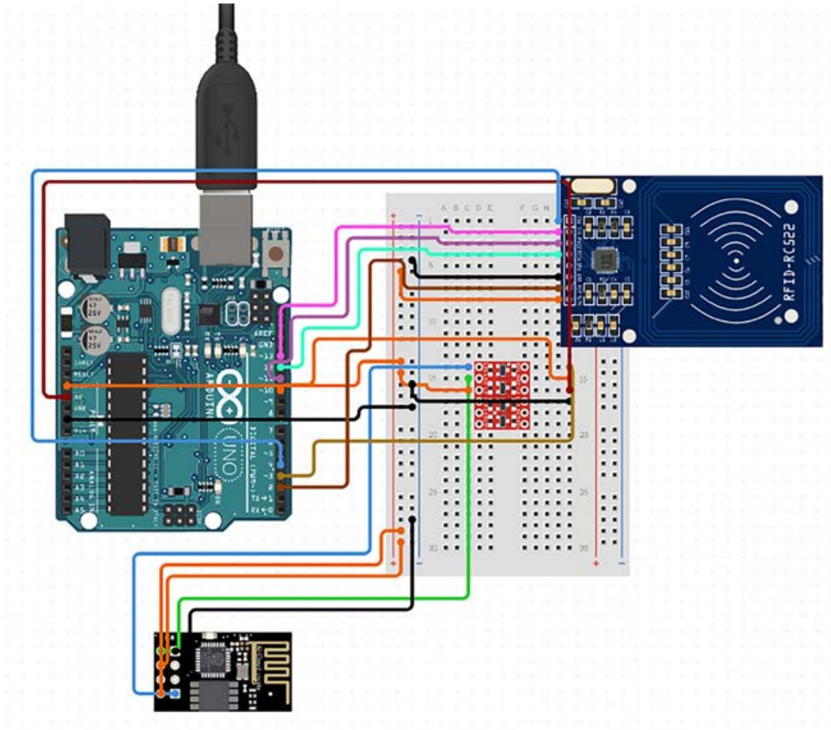

Fig. 11.4 Components for RFID-based person identification within contract tracing system based on Arduino Uno.

Listing 11.3 RFID-based person identification algorithm pseudocode executed on Arduino Uno.

```
Input: card_detected, room_id
Output: door_unlock
Steps:
    If(card_detected)
       person_id:=ReadCard();
       in_list:=checkInList(person_id);
       if(person_id is in_list)
          message:=newMQTT(person_id, timestamp, room_id);
          sendMQTT(message);
          door_unlock:=true;
       end if;
    else
     door_unlock:=false;
    end if;
    return door_unlock;
    end.
```

Listing 11.4 Pseudocode of algorithm executed on edge server responsible for notifying the persons who were in the same room with the infected one.

```
Input: infected_id, date_infected
Output: persons_at_risk
Steps:
  1.  room_id, last_visit:=QueryLog(infected_id);
  2.  days_passed:=date_infected-last_visit;
  3.  if(days_passed > 7)
  4.      persons_at_risk:=null;
  5.  else
  6.      day_range_begin:=last_visit-7 days;
  7.      day_range_end:=last_visit;
  8.      rooms_at_risk:=QueryLog(infected_id, day_range_begin, day_range_end);
  9.      persons_at_risk = QueryLog(get all persons who were in rooms_at_risk, day_
          range_begin, day_range_end);
  10.     For each person_id in persons_at_risk
  11.         date_interacted:=QueryLog(person_id, infected_id);
  12.         SendNotification(person_id, "Warning! You were in the same room with
              infected person on "+date_interacted);
  13.     End for;
  14. end if;
  15. return persons_at_risk;
  16. end.
```

Listing 11.5 Solidity smart contract for infected person's contact tracing using Ethereum blockchain platform.

```
pragma solidity ^0.8.1;

contract ContactTracing {
    uint32 personId;
    uint interactionDate;
    uint16 roomId;

    function setPerson(uint32 p, uint d, uint16 r) public {
        personId = p;
        interactionDate=d;
        r=roomId;
    }

    function getPerson() public view returns (uint32 retVal)  {
        return personId;
    }

    function getDate() public view returns (uint retVal)  {
        return interactionDate;
    }

    function getRoom() public view returns (uint16 retVal)  {
        return roomId;
    }
}
```

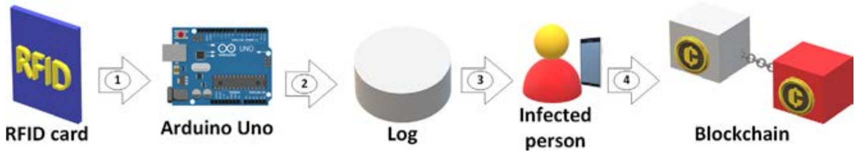

Fig. 11.5 RFID-based contact tracing workflow: (1) Person identifier, (2) MQTT message, (3) notification of potentially infected persons, and (4) adding new block about potentially infected persons to Ethereum blockchain as a result of smart contract execution.

approach, the same RFID tag should be used across many organizations (such as NFC module on smartphone in card emulation mode).

In Fig. 11.5, workflow of RFID-based contact tracing is depicted.

11.3.4 Patrol of coordinated ground robots

Finally, a set of coordinated mobile ground robots equipped with camera is used for indoor monitoring. These robots patrol inside the building, collect images, which are further processed on edge server, and perform predefined activities as a response to the detected events. In this chapter, we use TurtleBot3 kit, model Waffle Pi for this purpose, which is based on Raspberry Pi 3. TurtleBot refers to series of affordable, battery-powered, vehicle-alike, ROS-based robots. Moreover, the Robot Operating System (ROS) covers a collection of open-source software components whose goal is to provide the possibility of convenient robot application development without being aware of low-level details regarding the robot hardware implementation.

For each image streamed from robot, the following checks are performed on edge server relying on computer vision algorithms, as described in Petrović and Kocić (2020): (1) person count, (2) mask detection, and (3) social distancing check. For the first one, a predefined human body detector in OpenCV is used, while the second algorithm was previously mentioned. When it comes to social distancing, the algorithm relies on body detector, while the distance between each two persons is calculated, converted to meters, with respect to calibration formula from Petrović and Kocić (2020) and compated to predefined minimal value. Moreover, a companion AR-enabled mobile app (Petrović, Tošić, & Nejković, 2021) based on AppSheet is used to notify the security guards about the detected COVID-19 safety violations. For each violation event (such as number of persons in room above threshold, presence of a person without mask, social distancing rule not respected), the number of room is also recorded, together with timestamp and corresponding image captured by robot that is shown to the security guard. Moreover, there is a map of events within the application.

Moreover, it is possible to define coordinated robot behavior in form of simple if-then rules (Petrović, Tošić, & Nejković, 2021):

$$\begin{gathered} if\left(robot_k\ detects\ violation_k\ in\ room_k\right) then\, send\ patrol_{i=1\ldots n} \\ \left(robot_i\ with\ device_i\ in\ room_i\right) \end{gathered} \tag{11.1}$$

Therefore, if a robot detects some indoor safety rule violation event in one of the rooms, it is possible to send a patrol of up to *n* robots at different locations, equipped with target sensor/actuators (denoted as *device*). For example, if a person's mask is damaged, a patrol robot will detect that there is a person without proper mask in a room, so another robot holding a box of masks will be sent to that room, so this person can take a new mask. Another example would be the case the number of persons inside room is above the threshold for this room size, so another robot equipped with speakers and text-to-speech enabled will be sent to warn them. For coordination capabilities, the robots rely on semantic protocol implemented within SCOR framework (Nejkovic et al., 2020).

In Fig. 11.6, screenshots of the companion mobile app are given. It covers the following aspects: robot management, violation event list (together with map), and coordination scenario design.

Furthermore, the app contains AR-enabled GUI (implemented using AR.js library for JavaScript), which gives the ability to see the robot status and list of detected events by pointing smartphone camera to barcode maker attached to robotic device, as shown in Fig. 11.7A. The user will be redirected to the corresponding AppSheet page related to robotic device by touching the box that appears over its marker, as shown in Fig. 11.7B.

In Fig. 11.8, overview of this case study is given.

11.3.5 COVID-19 indoor safety monitoring ontology

The main role of this ontology is to enable interoperability and coordination of heterogeneous IoT devices within COVID-19 indoor safety monitoring system. It is used for semantic annotation of device capabilities and sensor measurements in context of coordination scenario definition. This ontology builds upon the work presented in Petrović and Kocić (2020) and relies on semantic coordination protocol from Nejkovic et al. (2020). In Fig. 11.9, the visual representation of this ontology is shown.

In this ontology, the top-level concept is *Monitoring System*. Such system consists of one or many *Devices*—either IoT low-power boards such as PIC microcontroller, Arduino Uno and Raspberry Pi, ground robots, or conventional edge servers. Furthermore, each device can be equipped with *Add-On*, including various *Sensors* (*Camera*, *Temperature, Microphone*, and *Distance* sensors) and *Actuators* (*Speakerphone*, *Mask Package*). Within the monitored space, different types of *Events* can be detected using the available sensors. Among them, we have safety rule *Violations* (*no mask, too many people, Social distancing*—camera; *high temperature*—sensor; *cough*—microphone), *Face* (camera) and *Hands Close* detection (distance sensor), *Arrival* (RFID reader). Moreover, this ontology covers the definition of coordination rules (as defined in Eq. 11.1), so it is possible to define the desired action, which is executed as a *Response* when a specific type of event occurs. One of the following actions is supported: (1) notifying the *Security Guard* via mobile app that violation has occurred (*Notify Guard*); (2) locking the room door when a new visitor does not pass the entrance check (*Lock Door*); (3) bringing a package of masks using a robot for persons without mask (*Mask Package*); (4) warning the persons for social distancing

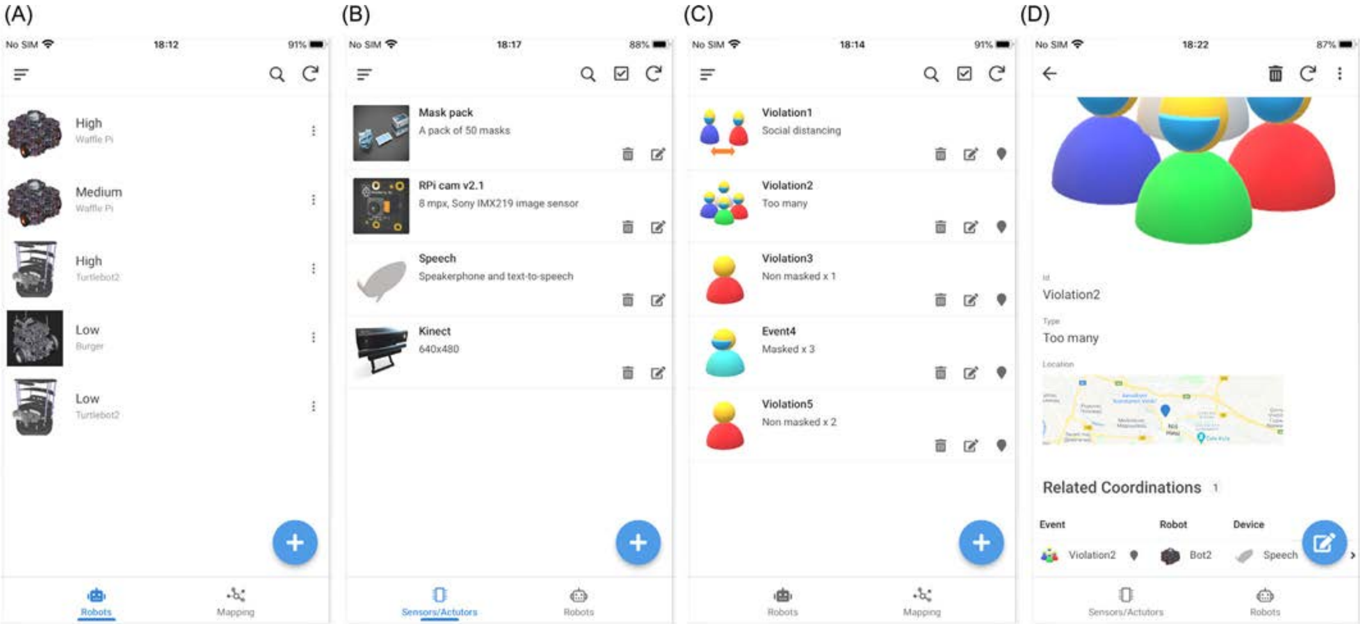

Fig. 11.6 Screenshots of robot patrol AppSheet mobile companion app: (A) robot list, (B) sensor and actuator list, (C) violation list, and (D) coordination scenario design.

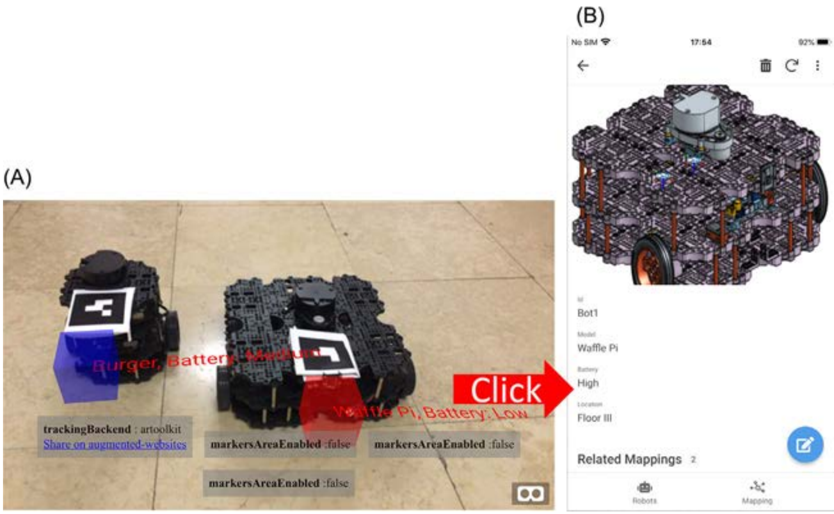

Fig. 11.7 AR robot patrol companion app: (A) AR.js interface and (B) details view in AppSheet.

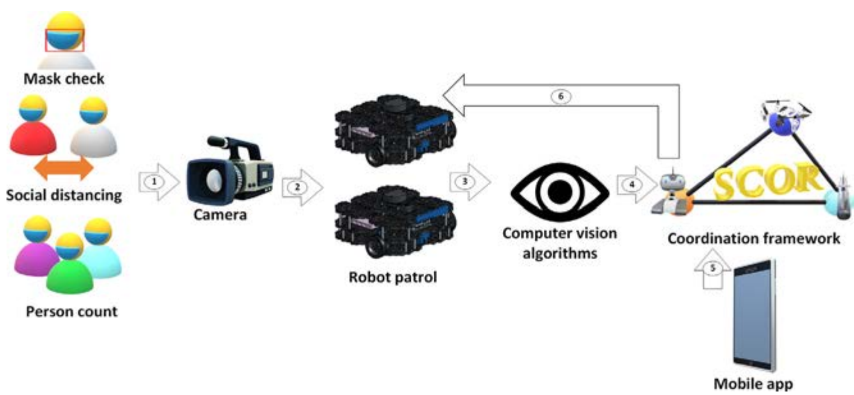

Fig. 11.8 Coordinated robot patrol for indoor monitoring case study: (1) Observation, (2) camera images, (3) image sent to server, (4) computer vision algorithm results, (5) coordination scenarios, and (6) robot patrol coordination.

or no mask violation using speech-enabled robot (*Warn*); and (5) spray sanitizer when hands are detected (*Open Dispenser*). For both the physical objects (*Device, Door, Room, Person*) and *Events*, the *Room* where it resides or occurs is relevant. This way, it is enabled to find the available *Security Guard* from the same *Floor* where *Violation* occurred and send him/her the corresponding notification. Otherwise, the first guard available is selected, using the SPARQL query from Petrović and Kocić (2020). When it comes to contact tracing scenario, it is necessary to know which are the potentially infected persons (*Infected* property). Furthermore, for each room, two numbers are relevant: (1) maximum number of persons allowed (*Person Limit*) and (2) current number of persons inside it (*Person Count*). In Listing 11.6, a SPARQL query, which returns all rooms where the current number of persons is greater than limit, is given.

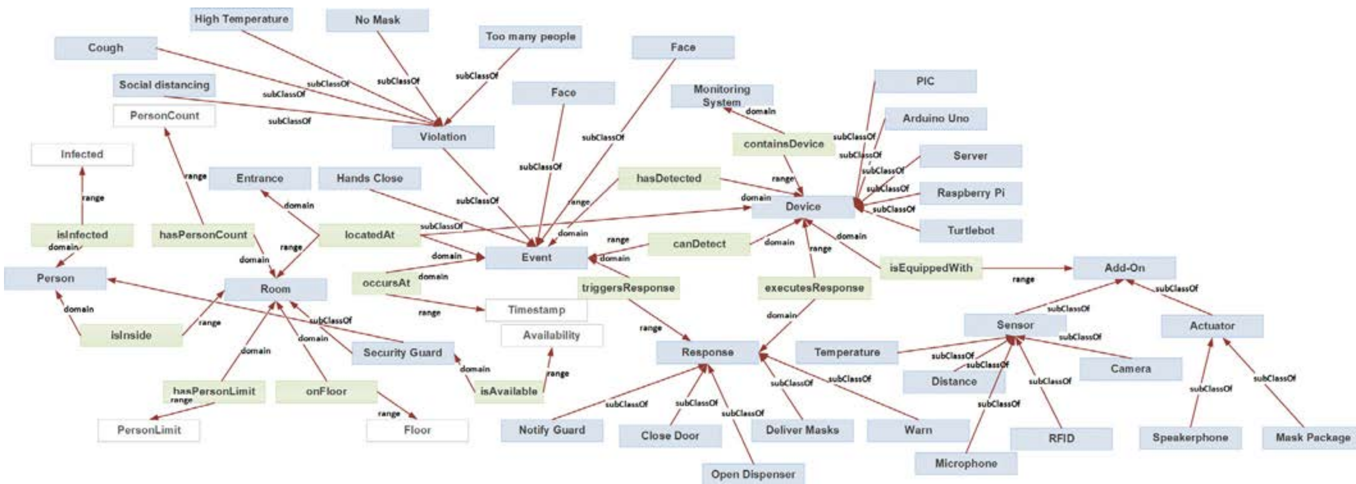

Fig. 11.9 Visual representation of COVID-19 Indoor Safety Monitoring Ontology (CISMO).

Listing 11.6 SPARQL query used for checking if there is room with greater number of persons inside than allowed.

```
PREFIX cismo: http://www.penenadpi.com/CISMO/
SELECT ?r
WHERE {
      GRAPH <http://www.penenadpi.com/example2> {
            ?r cismo:hasPersonCount ?pc.
            ?r cismo:hasPersonLimit ?pl.
            FILTER(?pc>?pl)
    }
}
```

11.4 Results and evaluation

In this section, evaluation of relevant aspects related to performance of the presented case studies is given. For the purpose of evaluation, a laptop running on Intel i7 7700-HQ quad-core 2.80GHz CPU, equipped with 16GB of DDR4 RAM and 1TB HDD was used as edge server in experiment scenarios. For computer vision-based monitoring tasks and robot patrol, Raspberry Pi 3 devices were used. Moreover, for temperature measurement, we leveraged Arduino Uno Rev. 3, while PIC16F870 served as control unit for entrance check system and hand sanitizer. Finally, semantic knowledge base was deployed on faculty's cloud server. When it comes to smartphone devices, due to multiplatform support, both iOS and Android models were involved—iPhone 6s Plus and Redmi Note 8, respectively.

In Table 11.1, the results of performance evaluation under given conditions for various aspects relevant to the presented case studies are shown. The first column refers to the name of the considered case study. Moreover, the second column denotes the aspect relevant to the case study, which is being evaluated. Furthermore, the third column holds the information about the settings and conditions related to experiment environment, such as image resolution or framework used—OpenCV or YOLO (computer vision tasks), sensor type (distance and temperature measurement), and smartphone model (mobile app response time). Finally, the last column shows the performance evaluation results for the target measurement, related to some aspect within the considered case study. For example, in case of computer vision tasks, it could be either framerate or detection accuracy; regarding the end-user applications, the average response time is considered; in case of massive notification, it is the number of persons involved; for semantic knowledge base, it is the average query execution speed, while for other sensor measurements, it refers to relative/absolute error or failure rate.

According to the achieved results, it can be concluded that IoT devices and smartphones are usable in the presented case studies with performance, which is at acceptable level. However, for more complex scenarios, their processing power is a limiting factor, so more powerful devices, such as traditional servers would be needed. On the other hand, the main advantage of IoT devices is their low price, making such systems affordable for institutions in developing countries.

Table 11.1 Case study evaluation results.

Case study	Aspect	Conditions/settings		Performance	
Entrance check	Face detection	Resolution ($w \times h$) (Framework)		Framerate (fps)	Accuracy (%)
		320×240 (OpenCV)		35.6	92
		640×480 (OpenCV)		23.4	94
	Mask detection	Resolution ($w \times h$) (Framework)		Framerate (fps)	Accuracy (%)
		320×240 (OpenCV)		25.42	87
		640×480 (OpenCV)		11.24	91
		320×240 (YOLO)		4.41	92
		640×480 (YOLO)		1.72	96
	Temperature check	Sensor		Measurements per second	Relative error (%)
		MLX90614		8	2
	Security guard notification	Mobile device		Response time (s)	
		Redmi Note 8		0.94	
Automatic hand sanitizer	Distance measurement	Sensor		Measurements per second	Average error (cm)
		SRF04		10	0.5
RFID contact tracing	RFID read	Sensor		Average entry time (s)	Fail rate (%)
		RC522		0.74	12.5
	Notification delivery	Number of persons		Delivery time (s)	
		19		3.8	
Robot patrol	Person counting	Res. (OpenCV)	Num. persons	Framerate (fps)	Accuracy (%)
		320×240	4	24.51	78
		640×480		10.92	81
	Social distancing	320×240	4	19.74	72
		640×480		9.15	76
Semantic knowledge base		Server location		Average query execution time (s)	
	Infected person retrieval	Faculty's cloud		1.16	
	Security guard notification			0.78	
	AR companion app	Mobile device		Framerate (fps)	
		iPhone 6s Plus		28	

11.5 Conclusion and future works

In this chapter, an overview of smart technologies for COVID-19 indoor safety monitoring is given, and their effectiveness in this context was evaluated on several case studies from the University of Niš, Faculty of Electronic Engineering, Serbia. According to the achieved results, it can be confirmed that these smart technologies are key-enablers in case of COVID-19 spread prevention, especially computer vision and IoT devices, which are crucial for the implementation of cost-effective solutions. Furthermore, the presented case studies present cost-effective solutions, which are affordable for most institutions.

However, our future plan is to focus on adoption of embedded artificial intelligence for indoor monitoring, especially in light of new generation AI-enabled embedded hardware, such as Arduino Nano 33 BLE Sense.[m] These new models of small-size microcontrollers are compatible with lightweight machine learning frameworks, such as TinyML and TensorFlow Lite[n] (Warden & Situnayake, 2019), opening new horizons of innovative usage scenarios relevant to coronavirus pandemic leveraging other environment data sources than images (which is one of the main topics covered in this chapter), such as cough classification (Bansal, Pahwa, & Kannan, 2020; Melek, 2021; Pahar, Klopper, Warren, & Niesler, 2021).

Acknowledgments

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

References

Abuga, K., & Nyamweya, N. (2021). Alcohol-based hand sanitizers in COVID-19 prevention: A multidimensional perspective. *Pharmacy*, *9*(1), 64. 1-14 https://doi.org/10.3390/pharmacy9010064.

Asadzadeh, A., et al. (2020). Information technology in emergency management of COVID-19 outbreak. *Informatics in Medicine Unlocked*, *21*, 1–11. https://doi.org/10.1016/j.imu.2020.100475.

Bansal, V., Pahwa, G., & Kannan, N. (2020). Cough classification for COVID-19 based on audio mfcc features using convolutional neural networks. In *2020 IEEE international conference on computing, power and communication technologies (GUCON)* (pp. 604–608).

Chaudhury, S. (2020). COVID-19 lockdown psychological effects. *Medical Journal of Dr D Y Patil Vidyapeeth*, *13*(6), 580–584 (1-5).

Cirrincione, L., et al. (2020). COVID-19 pandemic: Prevention and protection measures to be adopted at the workplace. *Sustainability*, *12*(9), 3603. 1-18 https://doi.org/10.3390/su12093603.

[m] https://store.arduino.cc/arduino-nano-33-ble-sense.
[n] https://www.tensorflow.org/lite/microcontrollers.

Das, S., et al. (2021). Design and development of IoT based smart security system in Covid19 situation. *Journal of Physics Conference Series*, *1797*, 012048. 1-7.

Deka, S., Goswami, S., & Anand, A. (2020). Blockchain based technique for storing vaccination records. In *2020 IEEE Bombay section signature conference (IBSSC)* (pp. 135–139). https://doi.org/10.1109/IBSSC51096.2020.9332171.

Flood, R., Chan, S., Chen, W., & Aspinall, D. (2021). Checking contact tracing app implementations. In *Proceedings of the 7th international conference on information systems security and privacy—Volume 1: ICISSP, Vienna, Austria* (pp. 133–144).

Fusco, A., Dicuonzo, G., Dell'Atti, V., & Tatullo, M. (2020). Blockchain in healthcare: Insights on COVID-19. *International Journal of Environmental Research and Public Health*, *17*(19), 7167. 1-12 https://doi.org/10.3390/ijerph17197167.

Melek, M. (2021). Diagnosis of COVID-19 and non-COVID-19 patients by classifying only a single cough sound. *Neural Computing and Applications*, 1–12. https://doi.org/10.1007/s00521-021-06346-3.

Nasajpour, M., et al. (2020). Internet of things for current COVID-19 and future pandemics: An exploratory study. *Journal of Healthcare Informatics Research*, *4*, 325–364. https://doi.org/10.1007/s41666-020-00080-6.

Nejkovic, V., Petrovic, N., Tosic, M., & Milosevic, N. (2020). Semantic approach to RIoT autonomous robots mission coordination. *Robotics and Autonomous Systems*, *126*, 103438. 1–19 https://doi.org/10.1016/j.robot.2020.103438.

Our World in Data. (2021). *COVID-19 vaccine doses administered per 100 people*. https://ourworldindata.org/covid-vaccinations. (Accessed 31 March 2021).

Pahar, M., Klopper, M., Warren, R., & Niesler, T. (2021). COVID-19 cough classification using machine learning and global smartphone recordings. *Computers in Biology and Medicine*, *135*, 1–10. https://doi.org/10.1016/j.compbiomed.2021.104572. 104572.

Petrović, N. (2020). Prototyping PIC16-based COVID-19 indoor safety solutions within microcomputer systems course. In *Proceedings of IEEESTEC—13th student projects conference, Niš, Serbia* (pp. 185–189).

Petrović, N. (2021). COVID-19 safety systems design exercises in computer science university courses. In *Proceedings of YuInfo 2021, Kopaonik, Serbia* (pp. 1–6).

Petrović, N., Dimovski, V., Peterlin, J., Meško, M., & Roblek, V. (2021). Data-driven solutions in smart cities: The case of Covid-19 apps. In *Proceedings of WWW '21 companion—7th WebAndTheCity—Web intelligence and resilience in smart cities, Ljubljana, Slovenia*. https://doi.org/10.1145/3442442.3453469.

Petrović, N., & Kocić, Đ. (2020). IoT-based system for COVID-19 indoor safety monitoring. In *Proceedings of IcETRAN 2020, Belgrade, Serbia* (pp. 603–608).

Petrović, N., Radenković, M., & Nejković, V. (2020). Data-driven mobile applications based on AppSheet as support in COVID-19 crisis. In *Proceedings of IcETRAN 2020, Belgrade, Serbia* (pp. 609–614).

Petrovic, N., & Tosic, M. (2019). SMADA-fog: Semantic model driven approach to deployment and adaptivity in fog computing. *Simulation Modelling Practice and Theory*, *101*, 102033. 1-25 https://doi.org/10.1016/j.simpat.2019.102033.

Petrović, N., Tošić, M., & Nejković, V. (2021). AR-enabled mobile apps for robot coordination. In *Proceedings of the 7th conference with international participation knowledge management and informatics, Kopaonik, Serbia* (pp. 1–10).

Rahman, M., et al. (2020). An automated system to limit COVID-19 using facial mask detection in smart city network. In *2020 IEEE international IOT, electronics and mechatronics conference (IEMTRONICS)* (pp. 1–5).

Rasool, R., et al. (2020). Detection and diagnosis system of novel COVID-19 using internet of things and send alert. *Journal of Information and Computational Science*, *10*(7), 129–135.

Rosebrock, A. (2020). *YOLO and Tiny-YOLO object detection on the Raspberry Pi and Movidius NCS*. https://www.pyimagesearch.com/2020/01/27/yolo-and-tiny-yolo-object-detection-on-the-raspberry-pi-and-movidius-ncs/. (Accessed 31 March 2021).

Tin, D., Hart, A., & Ciottone, G. (2021). The Covid19 pandemic and the emergent need for quarantine medicine. *Journal of High Threat & Austere Medicine*, *3*(1), 1–3. https://doi.org/10.33553/jhtam.v3i1.36.

Traskevich, A., & Fontanari, M. (2021). Tourism potentials in post-COVID19: The concept of destination resilience for advanced sustainable management in tourism. *Tourism Planning & Development*. https://doi.org/10.1080/21568316.2021.1894599.

U. S. Global Leadership Coalition. (2020). *COVID-19 Brief: Impact on the economies of developing countries*. online https://www.usglc.org/coronavirus/economies-of-developing-countries/. (Accessed 31 March 2021).

Ulhaq, A., et al. (2020). COVID-19 control by computer vision approaches: A survey. *IEEE Access*, *8*, 179437–179456. https://doi.org/10.1109/ACCESS.2020.3027685.

Vedant, S., et al. (2021). Leveraging deep learning and IoT for monitoring COVID19 safety guidelines within college campus. In *Advanced computing (IACC 2020) communications in computer and information science, vol 1367. Springer, Singapore* (pp. 31–53). https://doi.org/10.1007/978-981-16-0401-0_3.

Verma, P., et al. (2021). A statistical analysis of impact of COVID19 on the global economy and stock index returns. *SN Computer Science*, *27*, 1–13. https://doi.org/10.1007/s42979-020-00410-w.

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted Cascade of simple features (pp. 511–518).

Visconti, R. M. (2020). The valuation of mobile apps. In *The Valuation of Digital Intangibles*. Cham: Palgrave Macmillan. https://doi.org/10.1007/978-3-030-36918-7_12.

Warden, P., & Situnayake, D. (2019). *TinyML*. O'Reilly Media.

Worldometer. (2021). *Coronavirus update (Live)*. online https://www.worldometers.info/coronavirus/. (Accessed 31 March 2021).

Zheng, G., Gao, L., Huang, L., & Guan, J. (2021). *Ethereum smart contract development in solidity*. Springer.