# Mini Project - Team 3
# Automation of Covid SOPs in Campus

## Team Leader:

**Pranesh G**

## Team Members:

**Shravan Raviraj**
**Soundar Murugan**
**Keshav Agarwal**
**Soham Sandeep Salhotra**
**Sarun Sathis Menon**

## Additional Documents and Files:

https://drive.google.com/folderview?id=1s7SgIhy2txUg4hNXrCroNkCH7jnv2PWy

### Abstract

The Team has aimed to solve problems in the human-crewed operation of ensuring Covid SOPs on the Campus. Manipal's back to campus initiative amidst the pandemic was one of MAHE's bold decisions for the benefit of all. MAHE had commendable in organizing the back-to-campus initiative with a well laid out plan for the students to return. However, there is scope for improvement and automation of the protocols. There are fewer actions taken to ensure masks' usage, contactless attendance in hostels, and hassle-free Ion (internet provider for hostels) reactivation. Our Project aims to solve each of these problems with automation.
We have solved 3 of the pressing problems with Machine learning, Computer vision, ROS, and web automation. A short description of each solution is given below.

1. **Mask detection:** We have built a Machine learning-based model for mask detection and integrated it with ROS, and monitored the student's mask usage. CNN model is created with the various dataset to increase the robustness of the model. The model has the accuracy of close to 99% in the test set. Thus ensuring greater safety among students and also using mask detection as a node for other purposes like mask and sanitizer dispenser.

2. **Sanitizer and Mask Dispensing system interfaced with Mask Detection**: We have built a mask dispenser and sanitizer dispenser for students based on the mask detection model's result. We have used the CNN model to detect whether a person is wearing a mask or not and an Rpi based mask and sanitizer dispenser integrated with the model through ROS. Thus enabling a robust and safe solution for the students to ensure maximum precaution.

3. **Contactless Attendance system for hostels:** We have built a QR code-based attendance system as a better alternative to the conventional biometric system considering the pandemic. The student just needs to generate an encoded QR code from his SLCM and show it to the camera to give his hostel's attendance. The solution is made possible by integrating openCV for QR decoding and ROS as middleware. The data of the attendance is stored in CSV format for record-keeping. Thus ensuring the contactless attendance method and also ensuring safety.

4. **Hassle-free and full automation of ION reactivation:** We have built software web automation to automate the messy queuing process in ION counters for reactivation amidst the pandemic. We have designed a web server that can get input from students regarding their details relevant to ION and update the server's data without any human intervention—thus making the process available 24/7 , contactless, and smooth.

The Project is divided into four main sections:

1. **Mask detection-based entry and sanitization**

2. **Sanitizer and Mask Dispensing system interfaced with Mask Detection**

3. **Contactless Attendance system for hostels**

4. **Hassle-free and full automation of ION reactivation**

# MASK DETECTION


## Abstract:

We are currently living in the times of a global pandemic. While countries around the globe try to re-initiate business and education in order to recover from the economic impact of the pandemic, it is important to ensure that necessary protocol is being followed.

Masks being mandatory in such circumstances, serve as an effective means to curb the spread of the COVID-19 virus. Although, it is a matter of concern that several citizens fail to follow the norms. In our project we try to develop a system to identify people who do not wear face masks with the help of relevant image processing techniques. This could ensure safety in public places such as restaurants and malls or even Laboratories and Classrooms wherein people wouldn't be allowed to enter without a face mask


## Introduction:

Masks are a key measure to suppress transmission of the virus and save lives. Yet we find people not wearing masks and violating the rules. The Mask Detector detects if the person is wearing the mask or not. In order to achieve our goal we use relevant Deep learning techniques that ensure accurate prediction. We use the Inception-v3 Deep Learning Architecture in order to carry out identification of masks. The model is trained over data containing images of two classes : "With Mask" and "Without Mask". We also use the openCV library to capture images and feed it to the model in the required format. The Mask Detector can be integrated with a mask dispenser to ensure everyone has a mask before entering a closed space.

In this way the whole procedure is being automated. As a result of which the contact can be almost reduced to zero and we could curb the spread of the virus to a great extent possible.

## Literature

Inception-v3 is a convolutional neural network architecture from the Inception family that makes several improvements including using Label Smoothing, Factorized 7 x 7 convolutions, and the use of an auxiliary classifier to propagate label information lower down the network (along with the use of batch normalization for layers in the side head). By rethinking the inception architecture, computational efficiency and fewer parameters are realized. With fewer parameters, a 42-layer deep learning network, with similar complexity as VGGNet, can be achieved. With 42 layers, a lower error rate is obtained and makes it the 1st Runner Up for image classification in ILSVRC ( ImageNet Large Scale Visual Recognition Challenge).

The purpose of Inception architecture was to reduce computational resource usage in highly accurate image classification using deep learning. They had focused on finding an optimized position between the traditional way of increasing performance, which is to increase size and depth, and using sparsity in the layers based on the theoretical grounds. Both the approach in their own position can cost a huge amount of computational resources. For such a deep learning system like Inception which uses fully learned filters in their 22 layer architecture, this was the main goal to achieve. They focused on the approach to generate a correlation statistic analysis to generate groups of higher correlation to feed forward to the next layer.

## Problem statement & Objectives

Our project aims to accurately distinguish between people who wear masks and those who don't. This includes detecting the boundaries of the face and feeding the image of the face to the model. The Model further performs a binary classification.

## Methodology

A well known Deep Learning Architecture - "Inception - v3" was used for the project. The Inception version 3 is a Convolutional Neural Network Architecture which is 48 layers deep, it also uses Normalisations and Max Pooling Layers. The model is capable of classifying images upto 1000 categories although we aim to effectively achieve a binary classification. When fed with an input of shape (None,256,256,3), the model gives an output of shape - (None, 14,14, 768). This output is flattened and fed to a Dense layer (relu Activation) -> Dropout -> Final Neuron (Sigmoid Activation). The Model gives as an output the probability that a person has or has not worn a mask.

The model is trained on an RGB Image dataset. The Dataset Consists of 1889 Images belonging to 2 Classes - "With" and "Without Mask".
The Dataset is Split into :

1. Training : 1035 Images
    a. With Mask - 594 Images
    b. Without Mask - 441 Images
2. Validation : 498 Images
    a. With Mask - 298 Images
    b. Without Mask - 200 Images
3. Test : 356 Images
    a. With mask - 178 Images
    b. Without Mask - 178 Images

## Code Snippets :

```
from tensorflow.keras.applications.inception_v3 import InceptionV3

local_weights_file = 'inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5'

pre_trained_model = InceptionV3(input_shape = (256, 256, 3),
                                include_top = False,
                                weights = None)

pre_trained_model.load_weights(local_weights_file)

for layer in pre_trained_model.layers:
  layer.trainable = False

pre_trained_model.summary()

last_layer = pre_trained_model.get_layer('mixed7')
print('last layer output shape: ', last_layer.output_shape)
last_output = last_layer.output
```

```
from tensorflow.keras.optimizers import RMSprop

x = layers.Flatten()(last_output)
x = layers.Dense(1024, activation='relu')(x)
x = layers.Dropout(0.2)(x)
x = layers.Dense  (1, activation='sigmoid')(x)

model = Model(pre_trained_model.input, x)

model.compile(optimizer = RMSprop(lr=0.0001),
              loss = 'binary_crossentropy',
              metrics = ['accuracy'])
```

```
base_dir = '/home/hp/Desktop/roblab/face-mask-detector-master'

train_dir = os.path.join(base_dir, 'train')
validation_dir = os.path.join(base_dir, 'validation')
test_dir = os.path.join(base_dir, 'test')

train_mask_dir = os.path.join(train_dir, 'with_mask')
train_no_mask_dir = os.path.join(train_dir, 'without_mask')
validation_mask_dir = os.path.join(validation_dir, 'with_mask')
validation_no_mask_dir = os.path.join(validation_dir, 'without_mask')

train_mask_fnames = os.listdir(train_mask_dir)
train_no_mask_fnames = os.listdir(train_no_mask_dir)

train_datagen = ImageDataGenerator(rescale = 1./255.,
                                   rotation_range = 40,
                                   width_shift_range = 0.2,
                                   height_shift_range = 0.2,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)

test_datagen = ImageDataGenerator(rescale = 1.0/255.)

train_generator = train_datagen.flow_from_directory(train_dir,
                                                    batch_size = 20,
                                                    class_mode = 'binary',
                                                    target_size = (256, 256))

validation_generator =  test_datagen.flow_from_directory(validation_dir,
                                                         batch_size  = 20,
                                                         class_mode  = 'binary',
                                                         target_size = (256, 256))

test_generator = test_datagen.flow_from_directory(test_dir,
                                                  batch_size = 1,
                                                  shuffle=False,
                                                  class_mode='binary',
                                                  target_size = (256, 256))
print(train_generator.samples)
```

```
Found 1035 images belonging to 2 classes.
Found 498 images belonging to 2 classes.
Found 356 images belonging to 2 classes.
1035
```

The Model was trained for 6 Epochs over the given dataset, with a batch size of 20.

## Result Analysis and Discussion

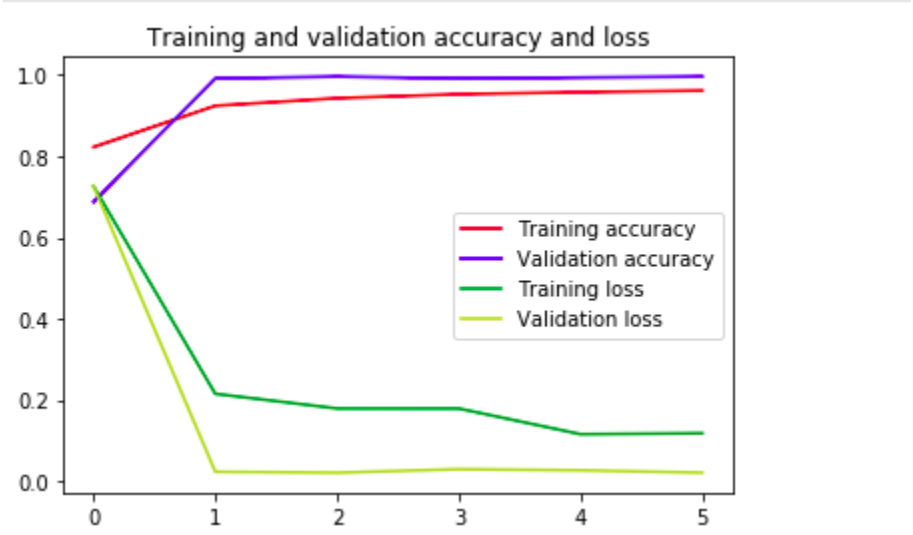
Training Results and Visualisation :

**No. of Epochs : 6**
**Steps per Epoch : 51**
**Batch Size : 20**

**Accuracy : 99.58%**

```
history = model.fit(
            train_generator,
            validation_data = validation_generator,
            steps_per_epoch = 51,
            epochs = 6,
            validation_steps = 24,
            verbose = 1)
```

```
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
Train for 51 steps, validate for 24 steps
Epoch 1/6
51/51 [==============================] - 107s 2s/step - loss: 0.7256 - accuracy: 0.8227 - val_loss: 0.7264 - val_accuracy: 0.6875
Epoch 2/6
51/51 [==============================] - 103s 2s/step - loss: 0.2156 - accuracy: 0.9241 - val_loss: 0.0233 - val_accuracy: 0.9917
Epoch 3/6
51/51 [==============================] - 98s 2s/step - loss: 0.1802 - accuracy: 0.9429 - val_loss: 0.0209 - val_accuracy: 0.9958
Epoch 4/6
51/51 [==============================] - 98s 2s/step - loss: 0.1781 - accuracy: 0.9527 - val_loss: 0.0296 - val_accuracy: 0.9917
Epoch 5/6
51/51 [==============================] - 98s 2s/step - loss: 0.1152 - accuracy: 0.9576 - val_loss: 0.0268 - val_accuracy: 0.9937
Epoch 6/6
51/51 [==============================] - 106s 2s/step - loss: 0.1188 - accuracy: 0.9616 - val_loss: 0.0209 - val_accuracy: 0.9958
```



## Output/Inference

The Model was tested on the Webcam.

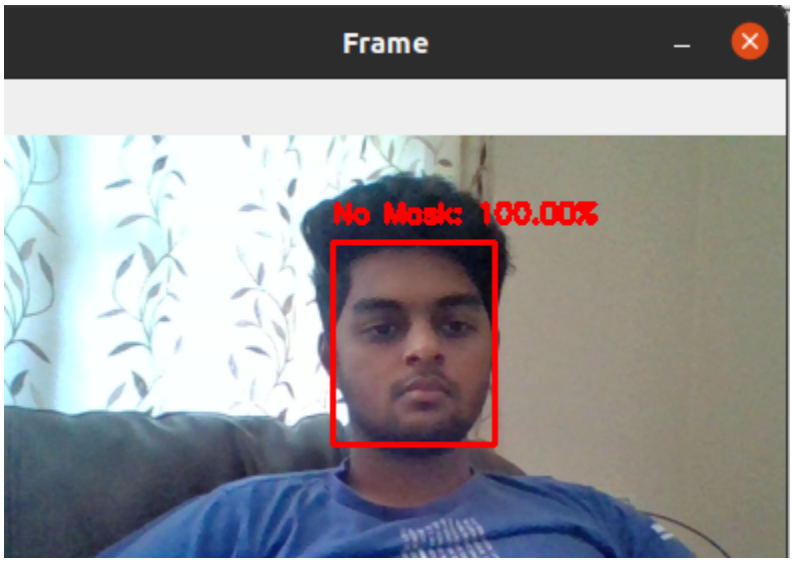
Code Snippet :

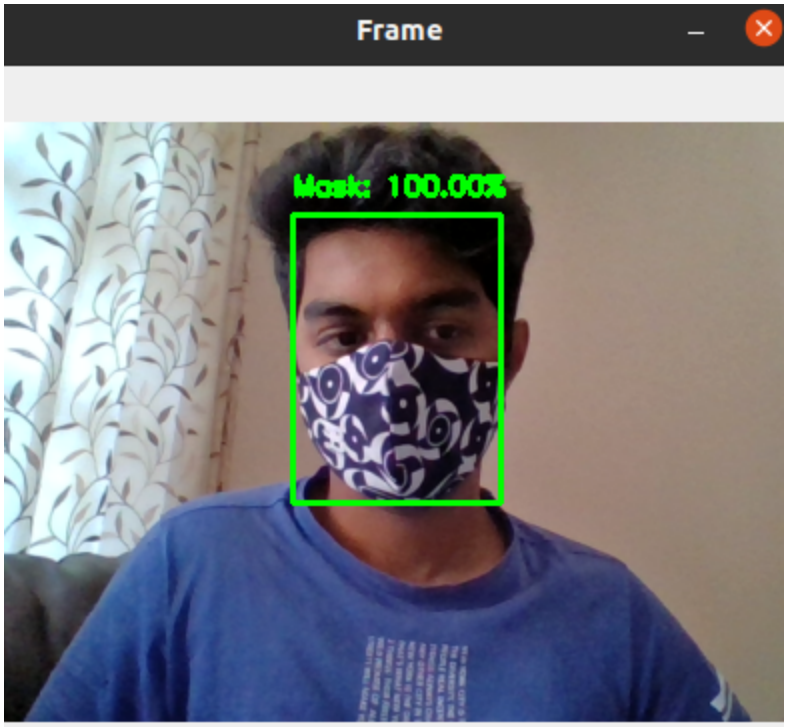
[3]:
```
def detect_and_predict_mask(frame, faceNet, maskNet):

    (h, w) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(frame, 1.0, (256, 256), (104.0, 177.0, 123.0))

    faceNet.setInput(blob)
    detections = faceNet.forward()

    faces = []
    locs = []
    preds = []

    for i in range(0, detections.shape[2]):

        confidence = detections[0, 0, i, 2]

        if confidence > 0.5:

            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (startX, startY, endX, endY) = box.astype("int")

            (startX, startY) = (max(0, startX), max(0, startY))
            (endX, endY) = (min(w - 1, endX), min(h - 1, endY))

            face = frame[startY:endY, startX:endX]
            face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
            face = cv2.resize(face, (256, 256))
            face = img_to_array(face)
            face = preprocess_input(face)

            faces.append(face)
            locs.append((startX, startY, endX, endY))

    if len(faces) > 0:

        faces = np.array(faces, dtype="float32")
        preds = maskNet.predict(faces, batch_size=32)[0]

    return (locs, preds)
```

[4]:
```
vs = VideoStream(src=0).start()
time.sleep(2.0)
```

[5]:
```
while True:

    frame = vs.read()
    frame = imutils.resize(frame, width=400)

    (locs, preds) = detect_and_predict_mask(frame, net, model)

    for (box, pred) in zip(locs, preds):

        (startX, startY, endX, endY) = box
        mask  = pred
        no_mask = 1 - mask

        label = "Mask" if mask < no_mask else "No Mask"
        color = (0, 255, 0) if label == "Mask" else (0, 0, 255)

        label = "{}: {:.2f}%".format(label, max(mask, no_mask) * 100)

        cv2.putText(frame, label, (startX, startY - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
        cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)

    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF

    if key == ord("q"):
        break

cv2.destroyAllWindows()
vs.stop()
```

Output Visualisation :






The Model accurately predicts the results as shown in the figure.

## Conclusion and Future Work

COVID-19 has proved to be a very fatal disease and masks can only shield us from it in work and public places. But if someone is not wearing a mask he/she is not only putting himself/herself in danger but also to other people in the surroundings. Therefore mask detection becomes very important and this project has been successful in implementing it.

It has many advantages and applications in the current scenario as almost all the activities have resumed throughout the country.

- The biggest advantage is that it will point out the person not wearing a mask thus protecting the people inside the room from possible infection.
- It eliminates the need of a supervisor to check if a person entering the room is wearing a mask. Thus also reducing a direct contact point.
- Even if a person is not wearing a mask, it will provide a mask to him/her when integrated with a dispenser.
- Lastly, it helps in “breaking the chain” of COVID-19 spread and eradicate the pandemic.

But like all prototypes this also has some limitations. They are listed below-

- This version is not able to detect if a person's mask is damaged.
- Also it cannot detect if a person is wearing a proper N95 mask that can prevent viral contact. A person wearing a cheap and low quality mask may be allowed to enter.
- The Model also works poorly in bad lighting as it is unable to capture the contrast in colours.

In order to overcome the shortcomings, higher quality cameras can be used or various image processing techniques can be used over the dataset to be able to feed the model with an image which has appropriate white balance. The model could also be able to distinguish people wearing cheaper or damaged masks if it was trained with datasets containing such masks as a separate class/category.

# Mask and Sanitizer dispenser:


## Abstract:

Amidst these times of the COVID-19 pandemic wherein, safety and precautions are a primary need, minimizing contact and sanitizing is of utmost importance.

We have come up with one of the best possible ways of implementing automation in college that relate to the current scenario. Having observed the current system, we came up with some ideas regarding the same. One of the feasible solutions to this problem could be the use of a Mask and sanitizer dispenser.

For wireless control and IOT control over the mask and sanitizer dispenser, the Raspberry Pi must be connected to the ROS node that detects masks and raises a flag. Once the flag has been raised, the system knows that the user is wearing a mask and sends the command for the sanitizer to be dispensed.


## Introduction:

The aim behind this Mask and Sanitizer Dispenser is to first detect if the person is wearing a mask or not.

If yes, then the Mask dispenser does not dispense the mask, but if not, the Mask dispenser would dispense a mask for safety measures. After the Mask is dispensed and worn by the individual, the person is provided with sanitizer from the Sanitizer dispenser. In this way, certain precautionary measures for COVID-19 would be dealt with.

The idea of doing so emerged because of safety measures not being followed by individuals in some areas. To reduce human labor and ease out the process of sanitization, we tried to automate the entire procedure so that the contact can be reduced to zero and the spread of the virus could be curbed to a great possible extent. To further reduce contact, we have implemented the automation over IOT. This makes the system more robust and easily implementable with lesser chance of failure and more chance of quick repair and changes

## Literature:

COVID-19 is caused by the virus, mainly when an infected person is in close contact with another person. The virus can spread from an infected person's mouth or nose in small liquid particles when they cough, sneeze, speak, sing, or breathe heavily. These liquid particles are of different sizes, ranging from larger 'respiratory droplets' to smaller 'aerosols.

Other people can catch the virus when it gets into their mouth, nose, or eyes, which is more likely to happen when people are in direct or close contact (less than 1 meter apart) with an infected person.

Medical masks are surgical or procedure masks that are flat or pleated (some are like cups); they are affixed to the head with straps. Wearing a medical mask is one of the preventive measures to limit the spread of COVID-19 as the chances of inhaling the virus is minimized to a great possible extent.

Washing hands with soap and water or using alcohol-based hand sanitizer kills viruses that may be handed. If infected individuals cough/sneeze droplets land on a surface, the transmission is possible by touching such objects or surfaces. Cleaning hands can help get rid of viruses present on hands.

A mask dispenser not only provides a mask but also reduces contact as it is completely automated. It is a 2-step way of ensuring minimal COVID-19 spread.

The hand sanitizer is a mandatory precautionary measure as keeping hands clean especially before eating is of prime importance.

## Problem statement:

The need for wearing a mask and timely sanitizing of hands for safeguard measures against COVID-19.

The main objective is to

1. Reduce human labor as it would reduce an individual's chances of contracting the virus

2. Reduce expenditure of the institute as a person need not be physically present to supervise this operation (as it is completely automated).

3. Social distancing would also be maintained as people stand in line for the above process and crowding in any form, for anything can be completely avoided.

## Methodology:

In the present scenario due to Covid-19, there is a high demand and need for a contactless face mask dispenser in densely populated areas like schools, colleges, residential districts, large-scale manufacturers, and other enterprises to ensure safety. The use of software like ROS can yield an effective solution to this problem. Using machine learning algorithms, we've created a mask detector that branches out into two parts.

1. Yes (the subject is wearing a mask)

When the subject is wearing a mask, the mask detector approves his/her entry into the building.

2. No (the subject is not wearing a mask)

When the subject is not wearing a mask, the dispenser comes into the picture and dispenses a mask. Following the approval of the software, the door opens.

3. No person detected: Nothing is dispensed, and door remains closed

### ROS:

The Robot Operating System (ROS) is not an actual operating system, but a framework and set of tools that provide the functionality of an operating system on a heterogeneous computer cluster. Its usefulness is not limited to robots, but most tools provided are focused on working with peripheral hardware.

ROS provides functionality for hardware abstraction, device drivers, communication between processes over multiple machines, tools for testing and visualization, and much more.

The key feature of ROS is the way the software is run and the way it communicates, allowing you to design complex software without knowing how certain hardware works. ROS provides a way to connect a network of processes (nodes) with a central hub. Nodes can be run on multiple devices, and they connect to that hub in various ways. In this project when the mask detection node publishes the mask flag, the Raspberry Pi sends the command to the servo motor to start dispensing the sanitizer.

The MQTT protocol stands for MQ Telemetry Transport. It is an open OASIS and ISO standard lightweight, publish-subscribe network protocol that transports messages between devices. The protocol usually runs over TCP/IP; however, any network protocol that provides ordered, lossless, bi-directional connections can support MQTT. MQTT works by publishing messages to a broker on a topic, the broker filters messages and then distributed to subscribers. A client receives messages by subscribing to the topic on the same broker. All clients can publish and subscribe to messages. MQTT is used as it has a minimum code footprint and requires a small bandwidth network connection.

MQTT brings many powerful benefits to your process:

- Distribute information more efficiently
- Increase scalability
- Reduce network bandwidth consumption dramatically
- Reduce update rates to seconds
- Very well-suited for remote sensing and control
- Maximize available bandwidth
- Extremely lightweight overhead
- Very secure with permission-based security

For our application, we have used the ClientID ‘maskServo’, which publishes to the topic ‘sensors/servo’. When the mask detector algorithm detects a mask, it publishes the flag to the topic, which is picked up by the client and the servo is run

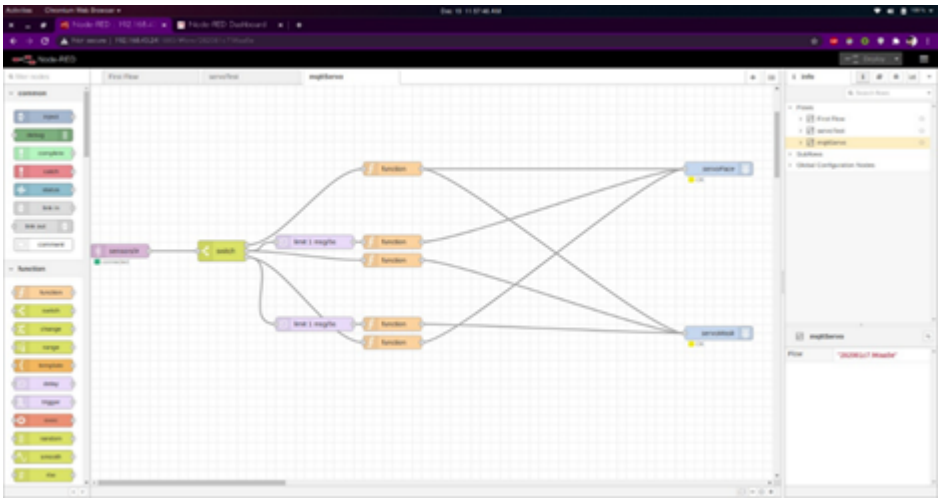


Fig1: Basic layout of RPi connections

**Mask detection- features :**

Face Mask Detection Platform uses Artificial Network to recognize if a user is not wearing a mask. A webcam is integrated with the software to make this happen.

Automatically sends alert

Multi-Channel Recognition

**Snapshots of code used to detect mask:**

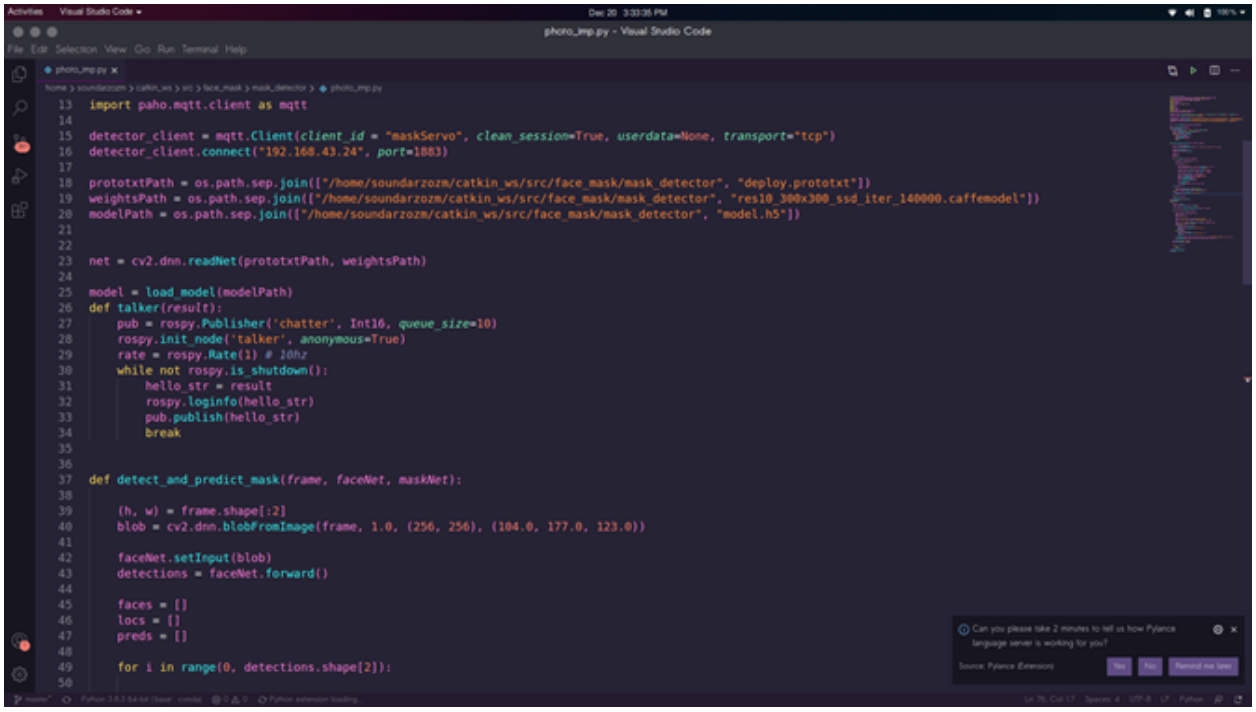



Fig:1

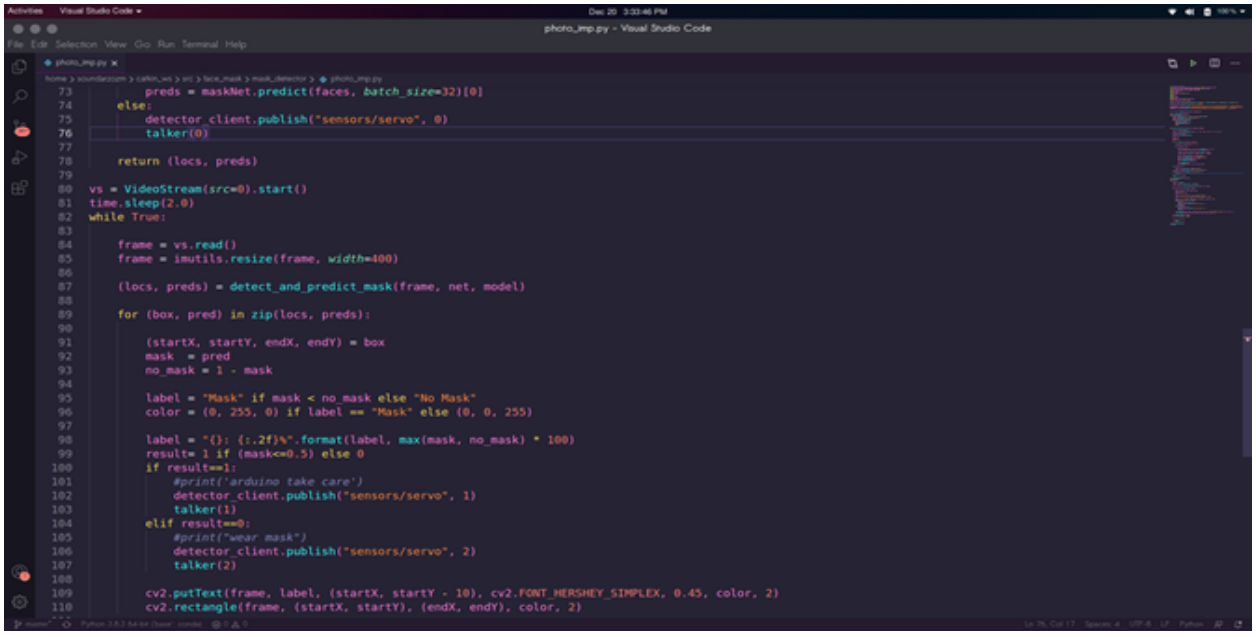
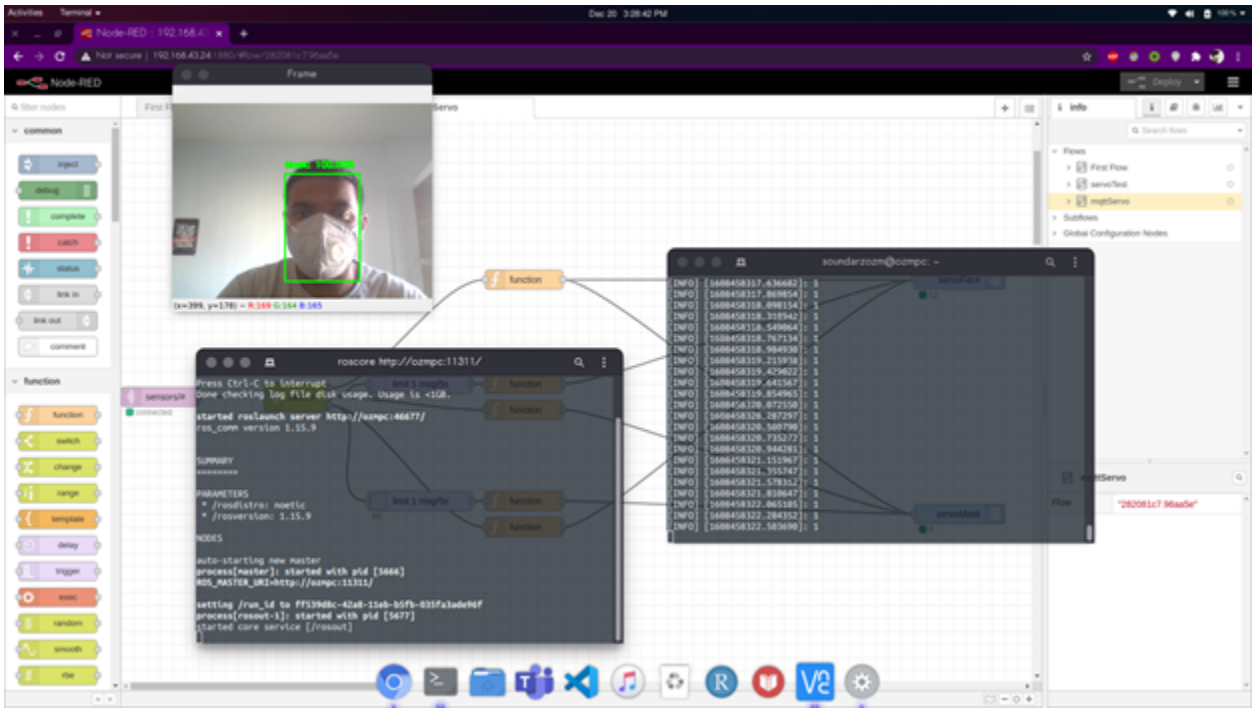



Fig: 2

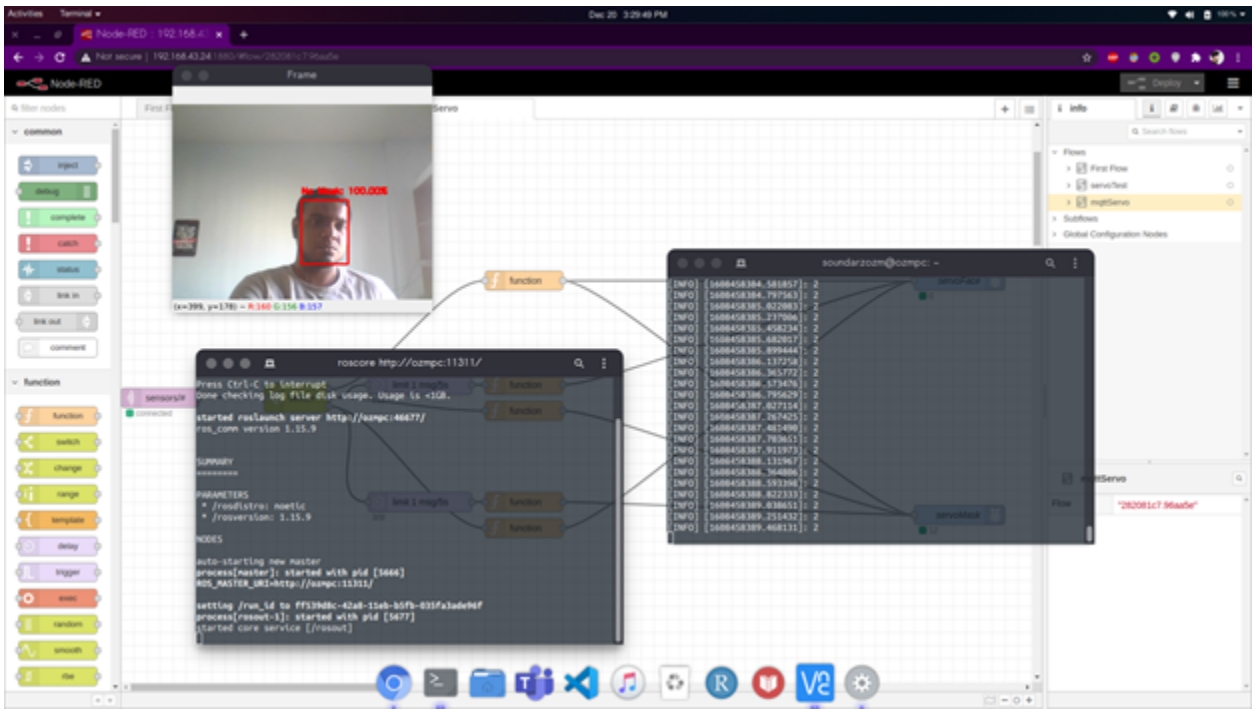
## Result:


Following the detection of a mask on the subject's face, the software allows entry into the building. Otherwise, it doesn't allow and remind the subject to wear a mask by dispensing one

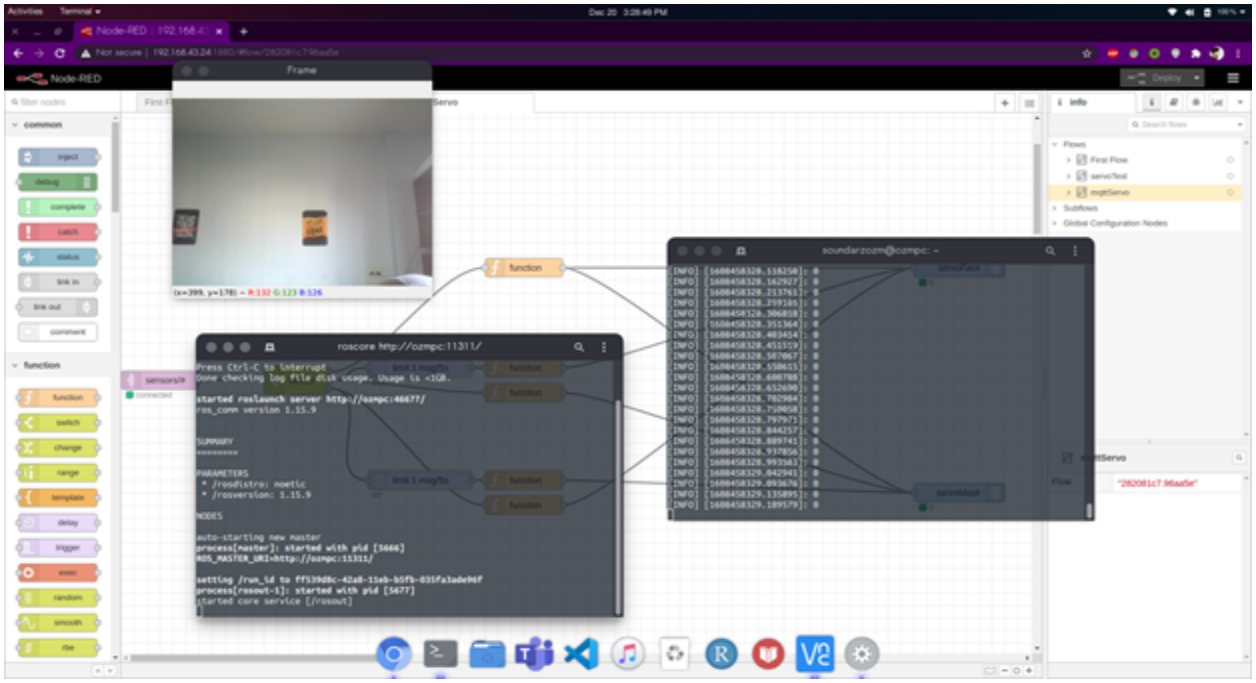


.

Screenshot 1: when the mask is detected



Screenshot 2: when a person is not wearing a mask



Screenshot 3: When in idle condition

## Output:

The solution using ROS works well in terms of making sure that people are complying with public health guidelines. But this feature raises questions about data privacy. It's particularly important to make sure the user data is safe and secured.

## Future works:

1. This dispenser model can be modified to be more efficient by reducing the number or servo motors and sensors used. This way we will be able to reduce the size of the apparatus.
2. More efficient and greener ways can be used to power the dispenser. Many components can be manufactured to bare minimum specifications to save on production costs and make it easy-to-install.
3. The mechanical consideration and the electronic casing need to be better designed to withstand any outdoor condition

## Conclusion:

A possible upgrade to this would be the addition of embedded software and safe storage and protection of user data.

# QR code-based hostel attendance system

## Abstract

Considering the current pandemic scenario, it is highly recommended not to touch common surfaces. Gate handles, railings, elevator buttons are some unavoidable surfaces but hostel biometrics, signature-based attendance system is something that can be avoided provided some contactless substitute is used. QR code-based attendance system is one such solution.

## Goals

1. Working towards making pandemic related SOPs fully contactless.
2. Avoiding chances of spread of virus through physical touch.
3. Replace obsolete systems of register signature and biometrics attendance.

## Introduction

A unique QR code is to be attached to every student's health card. At the place of biometrics in hostels QR code scanning cameras are to be used. Student shows his QR in the camera. The system marks the student as present and the screen shows the basic details of the student, so that the student can verify whether he/she is marked or not.

## Softwares Used

- OpenCV
- ROS
- Python
- ZBar Library (pyzbar package)

## Methodology

OpenCV, a library of programming functions mainly aimed at real-time computer vision, helps us to extract the QR code from the view visible on camera in front of which a health card containing QR code is shown by the student.

Using the OpenCV library, the information encoded in pixels of the QR code is extracted. This data is then stored in the form of a variable.

Once the variable is stored, using ROS the data will be published to the screen which will be visible to the student for verification, and can further be used by various systems for logging and other calculations, such as percentage of attendance, number of absences, etc.

The ROS Publisher uses a String data type topic to publish the data from the QR Codes, and any other program

The data will be converted to the dictionary and finally will be updated in .csv file where data of all other students are stored thus providing a common database for all students present in the hostel that too without any physical contact.

Although this program has the camera feed showing on the computer, this would not be shown to the end-user. The confirmation of attendance will be given to the end-users via the text output on the screen, or a separate display/feedback mechanism that it's subscribed to the ROS Publisher (Such as a buzzer and a standard 2 line LCD text display).

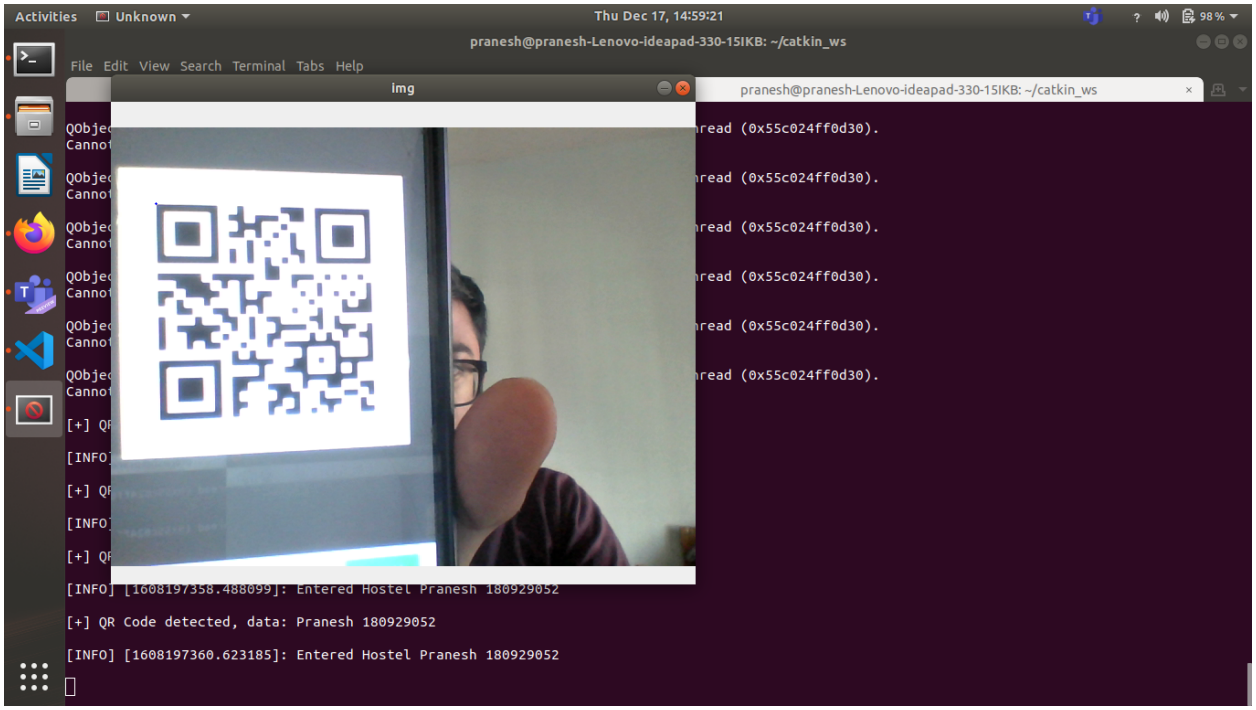



Fig. 1: The ROS Integrated QR Code Scanner, with the output being published in a ROS Topic in the Terminal behind

## Code:

### File: qr_code.py

```
#!/usr/bin/env python3
 # license removed for brevity
import rospy
from cv2 import cv2
import time
```

```
import datetime
from std_msgs.msg import String
import csv
from datetime import datetime

def talker(data):
    pub = rospy.Publisher('chatter', String, queue_size=10)
    rospy.init_node('talker', anonymous=True)
    rate = rospy.Rate(10) # 10hz
    while not rospy.is_shutdown():
        hello_str = "Entered Hostel %s" % data
        rospy.loginfo(hello_str)
        pub.publish(hello_str)
        rate.sleep()
        break
# initialize the cam
cap = cv2.VideoCapture(0)
# initialize the cv2 QRCode detector
detector = cv2.QRCodeDetector()
csv_file = "data.csv"
csv_columns = ['name','reg_no', 'College','Hostel Block', 'Date','Time_Punched']
while True:
    _, img = cap.read()
    # detect and decode
    data, bbox, _ = detector.detectAndDecode(img)
    # check if there is a QRCode in the image
    if bbox is not None:
        # display the image with lines
```

```
        for i in range(len(bbox)):
            # draw all lines
            cv2.line(img, tuple(bbox[i][0]), tuple(bbox[(i+1) % len(bbox)][0]), color=(255, 0, 0), thickness=2)
        if data:
            print("[+] QR Code detected, data:", data)
            today = date.today()
            data_dict= list(data)
            now = datetime.now()
            Time_Punched= now.strftime("%H:%M:%S")


            date = today.strftime("%d/%m/%Y")
            dict = {'name': data_dict[0], 'reg_number': data_dict[1], 'College': data_dict[2], 'Hostel Block': data_dict[3], 'Date': date, 'Time_Punched': Time_Punched }
            try:
                with open(csv_file, 'w') as csvfile:
                    writer = csv.DictWriter(csvfile, fieldnames=csv_columns)
                    writer.writeheader()
                    for data in dict:
                        writer.writerow(data)
            except IOError:
                print("I/O error")


            talker(data)
            time.sleep(2)
    # display the result
    cv2.imshow("img", img)
    if cv2.waitKey(1) == ord("q"):
        break
```

```
cap.release()

cv2.destroyAllWindows()
```

Another program has also been created for the end-users to verify their QR Codes. It consists of a window with live webcam feed, that shows the QR Code and it's data live on the camera feed itself. The QR Code detected is shown in a blue box, and the contents of it are displayed in red text right above it, in an augmented reality-style display format. This program can be run on one system with a screen, while the main attendance logging program can be deployed on multiple systems without a screen. This file can also save a screenshot of the last recorded QR Code when the ‘S’ Key is pressed.

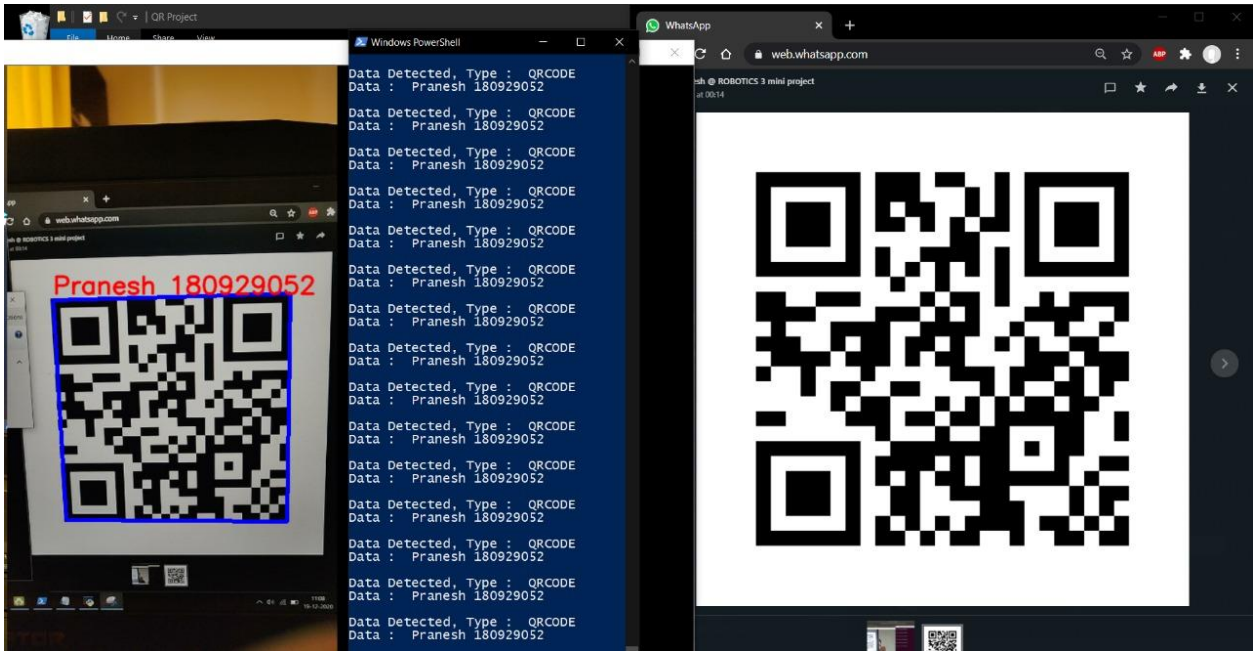



Fig. 2: The Laptop Screen, QR Code and Terminal Output when the QR Code is shown.

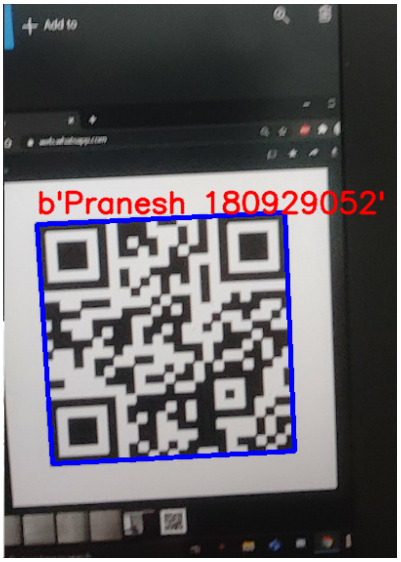



Fig. 3:The Image saved by pressing S in the program.

## Code:

### File: QR_Tester.py

```
# Required to install: OpenCV Python, Numpy, Pyzbar
from __future__ import print_function
import pyzbar.pyzbar as pyzbar
import numpy as np
import cv2
import time
cap = cv2.VideoCapture(0)
cap.set(3,640)
cap.set(4,480)
time.sleep(2)
def decode(im) :
```

```
    # Find barcodes and QR codes
    decodedObjects = pyzbar.decode(im)
    for obj in decodedObjects:
        return decodedObjects
font = cv2.FONT_HERSHEY_SIMPLEX
while(cap.isOpened()):
    ret, frame = cap.read()
    im = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    decodedObjects = decode(im)
    for decodedObject in decodedObjects:
        points = decodedObject.polygon
        if len(points) > 4 :
          hull = cv2.convexHull(np.array([point for point in points], dtype=np.float32))
          hull = list(map(tuple, np.squeeze(hull)))
        else :
          hull = points;
        n = len(hull)
        for j in range(0,n):
          cv2.line(frame, hull[j], hull[ (j+1) % n], (255,0,0), 3)
        x = decodedObject.rect.left
        y = decodedObject.rect.top
        barCode = str(decodedObject.data)
        overlay1 = barCode.replace("b\'","")
        overlay2 = overlay1.replace("\'","")
        print('Data Detected, Type : ', decodedObject.type)
        print('Data : ', overlay2,'\n')
        cv2.putText(frame, overlay2, (x, y), font, 1, (0,0,255), 2, cv2.LINE_AA)
```

```
    cv2.imshow('frame',frame)
    key = cv2.waitKey(1)
    if key & 0xFF == ord('q'):
        break
    elif key & 0xFF == ord('s'): # wait for 's' key to save
        cv2.imwrite('Last_QR_Saved.png', frame)
cap.release()
cv2.destroyAllWindows()
```

## Conclusion and Future Work

1. This model is a prototype of a bigger picture. To make the process more efficient in future, we can do two-step verification i.e. along with QR code the face of the student should also match then only he/she will be marked present.
2. Implementation of this QR code system to lab/classroom attendance as well. This will help to save 15% of the teaching time available per class. Further, it will solve the issue of false attendance.
3. The system can be modified to have one-time-use QR Codes (ex: they would be generated only once a day/on one device/ if you're in range of that hostel blocks' WiFi.) This would improve its security. Similar one-time use QR codes are already used in the campus for some tasks such as for participation in events and distribution of food tokens during the college fests.

# Online ION Registration

## Abstract:

The Covid-19 pandemic has accelerated digitization, which in turn has improved ease of living and access through the internet . The takeaway from the pandemic would be social distancing and how digitization improves overall quality of living .

We have come up with a solution for improving student's access to the internet and making their lives easier and to prevent violation of COVID guidelines . Having witnessed long lines of students waiting for access to something as basic as the internet , the solution that came to our minds was to create an online registration portal as part of a web automation solution .

## Introduction:

The aim behind an online registration is to provide the students with a portal , where they can fill in their details such as email id , registration no and block/room number .

The request will directly go to Ion , which will then process the request , confirm the details and activate your Ion internet directly in the hostel block without any physical appearance at the incubation center..

The idea came as a consequence of safety measures not being followed at the physical registration desk and to reduce contact as well as to improve the students ease of living by simplifying and shifting the registration process completely online .

## Literature:

In the current offline registration system, the student is given a registration form, in which the student fills in the details such as registration no, name, hostel/room no, etc. After filling the form, it is given to the Ion worker present there. which is then taken back to the main Ion office and the form is processed manually.

Online registration will reduce human labor and the chances of error that come associated with human labor

It will also prevent long queues at the registration desk, improving social distancing and thus ensuring better service and safety of students.

## Problem statement:

To improve the following of social distancing norms and reduce human capital

The main objective is to

i) Reduce human labor as it will reduce the cost and chances of contracting the virus.

ii) Social distancing would be maintained as people stand in line for the above process and crowding in any form, for anything can be completely avoided.

iii) Reducing the chances of the error caused due to human labor.

## Methodology:

The project is done using ROS and it is having the best features. In this project by using this we are done with an automatic internet excess portal, by using some details you can excess your WIFI portal. In this we created a web page,that will ask your details like your reg no, hostel block and room no, email. After submitting your details the user is prompted with a thanks message and his data is stored in our custom build mailchimp servers .Then the creator will download the details in any format (mainly csv) and after that access to your WIFI portal is given.

## Code snippets:

### Local servers status

```
Keshav agarwal@DESKTOP-3T7KPJ9 MINGW64 ~/Desktop
$ cd Ion_registration-Signup/

Keshav agarwal@DESKTOP-3T7KPJ9 MINGW64 ~/Desktop/Ion_registration-Signup
$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (ion_registration-signup)
version: (1.0.0)
description:
git repository:
keywords:
author:
license: (ISC)
About to write to C:\Users\Keshav agarwal\Desktop\Ion_registration-Signup\package.json:

{
  "name": "ion_registration-signup",
  "version": "1.0.0",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.19.0",
    "express": "^4.17.1",
    "request": "^2.88.2"
  },
  "devDependencies": {},
  "description": ""
}


Is this OK? (yes)

Keshav agarwal@DESKTOP-3T7KPJ9 MINGW64 ~/Desktop/Ion_registration-Signup
$
```

```
Keshav agarwal@DESKTOP-3T7KPJ9 MINGW64 ~/Desktop/Ion_registration-Signup
$ atom .


Keshav agarwal@DESKTOP-3T7KPJ9 MINGW64 ~/Desktop/Ion_registration-Signup
$ nodemon app.js
[nodemon] 2.0.6
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
Server is running on port 3000
```

## Code Samples:

```
const express=require("express");
const bodyParser=require("body-parser");
const request=require("request");
const https=require("https");

const app=express();

app.use(express.static("public"));
app.use(bodyParser.urlencoded({extended: true}));

app.get("/",function(req,res){
  res.sendFile(__dirname +"/signup.html");
});

app.post("/",function(req,res){
  var firstName= req.body.fname;
  var lastName= req.body.lname;
  var email=req.body.email;

  var data={
    members:[
      {
        email_address:email,
        status: "subscribed",
        merge_fields:{
          FNAME:firstName,
          LNAME:lastName
        }
      }
    ]
```

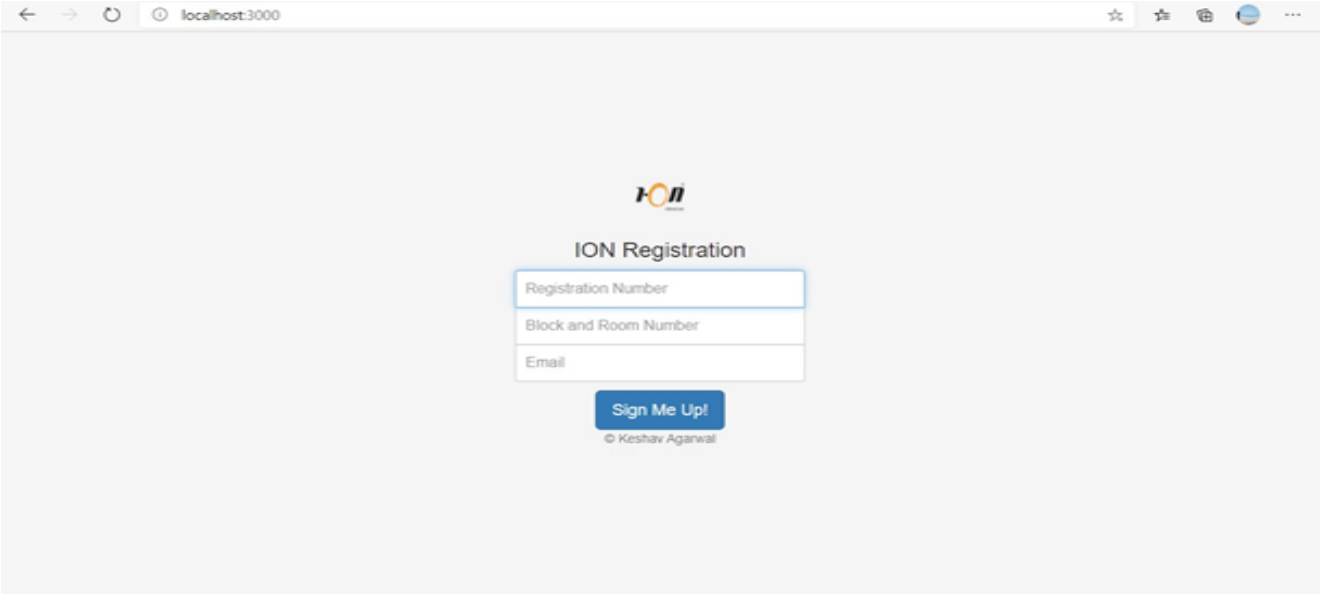
## Signup code snippets:

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="description" content="">
    <meta name="author" content="Mark Otto, Jacob Thornton, and Bootstrap contributors">
    <meta name="generator" content="Hugo 0.79.0">
    <title>ION REGISTRATION</title>

    <link rel="canonical" href="https://getbootstrap.com/docs/5.0/examples/sign-in/">



    <!-- Bootstrap core CSS -->
    <!-- Latest compiled and minified CSS -->
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4

    <!-- Favicons -->
<link rel="apple-touch-icon" href="/docs/5.0/assets/img/favicons/apple-touch-icon.png" sizes="180x180">
<link rel="icon" href="/docs/5.0/assets/img/favicons/favicon-32x32.png" sizes="32x32" type="image/png">
<link rel="icon" href="/docs/5.0/assets/img/favicons/favicon-16x16.png" sizes="16x16" type="image/png">
<link rel="manifest" href="/docs/5.0/assets/img/favicons/manifest.json">
<link rel="mask-icon" href="/docs/5.0/assets/img/favicons/safari-pinned-tab.svg" color="#7952b3">
<link rel="icon" href="/docs/5.0/assets/img/favicons/favicon.ico">
<meta name="theme-color" content="#7952b3">


    <style>
      .bd-placeholder-img {
        font-size: 1.125rem;
        text-anchor: middle;
        -webkit-user-select: none;
        -moz-user-select: none;
        user-select: none;
      }

      @media (min-width: 768px) {
        .bd-placeholder-img-lg {
          font-size: 3.5rem;
```

```
      <style>
        .bd-placeholder-img {
          font-size: 1.125rem;
          text-anchor: middle;
          -webkit-user-select: none;
          -moz-user-select: none;
          user-select: none;
        }

        @media (min-width: 768px) {
          .bd-placeholder-img-lg {
            font-size: 3.5rem;
          }
        }
      </style>


      <!-- Custom styles for this template -->
      <link href="css/styles.css" rel="stylesheet">
    </head>
    <body class="text-center">

  <form action="/" class="form-signin" method="POST">
      <img class="mb-4" src="image/logoion.PNG" alt="" width="72" height="57">
      <h1 class="h3 mb-3 fw-normal">ION Registration </h1>

      <input type="text" name="fname" class="form-control top" placeholder="Registration Number" required autofocus>
      <input type="text" name="lname" class="form-control middle" placeholder="Block and Room Number" required>
      <input type="email" name="email" class="form-control bottom" placeholder="Email" required>

      <button class="w-100 btn btn-lg btn-primary" type="submit">Sign Me Up!</button>
      <p class="mt-5 mb-3 text-muted">&copy; Keshav Agarwal</p>
    </form>

    </body>
  </html>
```

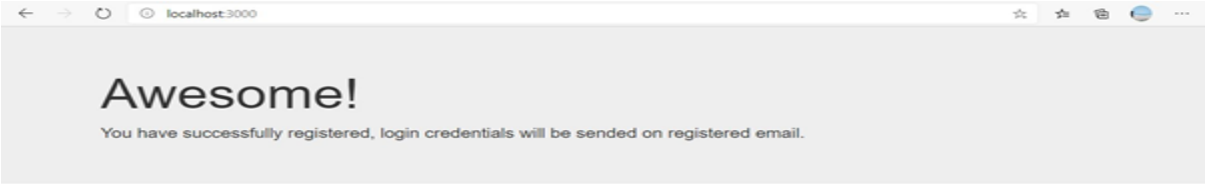
## Result Analysis and Discussion:

We will get a web page for entering our details.

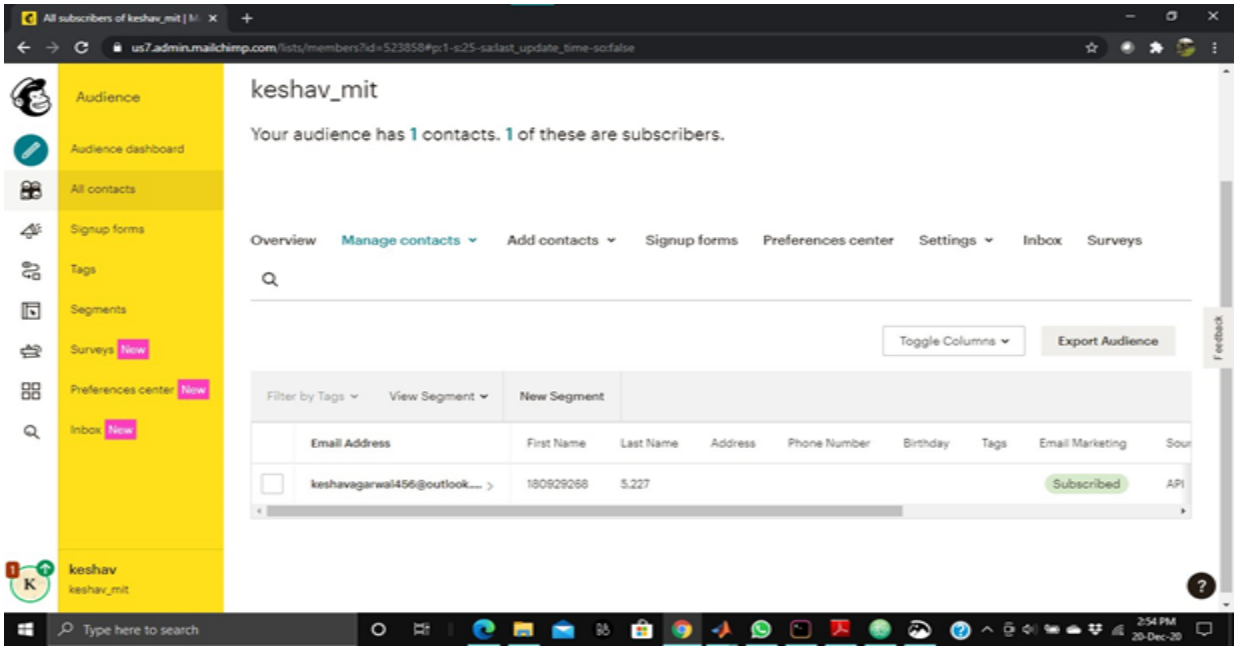



## Output:

The creator will access the data that the user entered and the user will get a message stating that his/her login details is registered successfully.




After the user enters his details and he/she receives a success message ,their data through our local setup servers on custom pc as stated above is sended to custom build mailchimp servers. Mailchimp servers are a tool that stores the user data in the form of tables and later can be downloaded easily. Screenshot the data imported to servers through web portal is shown below:




After giving login details the creator gets the message of your details and after the verification of the same, your Ion WIFI starts.

## Conclusion:

The covid-19 pandemic has accelerated digitalization, which in turn has improved ease of living and access through the internet. Here we learned how we all can access the WIFI directly. By this we are consuming less time for access to the internet connection. By simple steps we are accessing the internet. In this pandemic situation we should follow social distancing norms so by this we can't be affected by this situation. This is best for digitalization.

## Improvement:

1.Data verification too should be automated to reduce human error and reduce reaction time.

2. Our current servers are able to handle only 2000 contacts in a single go, after that it needs to be downloaded and data in servers has to be cleaned up. So that thing can be improved.

3. This prototype can be extended to other departments as well we're still paperwork is going on and students have to visit physically.

# Environmental And Safety Issues

All parts used in the project are individual components. There is no specific manufacturing or changes made in the sensors and actuators used. As a result, there is no wastage of material. Once the project loses importance or active widespread use in the near post-vaccine future, it can be disassembled and the parts can serve other purposes.

In the covid scenario, the solution gives the option of a contactless safety procedure that is commonplace now. That increases safety and improves the quality of already existing procedures.

In spite of all possible precautions, although contact can't be entirely eliminated, it can be reduced as much as possible. All of these implementations require minimal contact so there is almost no scope of any safety hazards. Some things such as refilling of Masks and Sanitizers, Cross checking of Attendance, etc. should be done periodically for effective operation.

To respect everyone's privacy, signage should be put up stating that footage is being recorded, wherever applicable, and also whether it is saved for any purposes. Also, Local and State Laws must also be applied, these may differ from the national SOP's being implemented.

In the case of the dispensers, Since the materials need not be individually packaged, it does lead to reduction in generation of single-use plastic. However, use of reusable masks etc. should be promoted to further reduce waste.

# Impacts on health:

Positive:

- No need for human supervision so the chances of transmission through humans are reduced as the process is completely automated and no human intervention is needed.
- Use of sanitizer and mask is one of the best ways to prevent the spread of the virus which is well taken care of in the project.

Negative:

- Extensive use of sanitizer is not good for the skin and the quantity of it cannot be regulated person to person.

# Ethics:

- The idea or concept should not be taken or used without permissions and should be acknowledged.
- The project should be made with an intention of improvement and benefit for human kind without any hidden motives or objectives.
- Use of pirated soft wares and unauthenticated sources should be refrained completely.

# Standards:

· Operating system must be patched according to patching guidelines

- Servers must be installed in an established server (not client) network
- Servers must be configured to utilize a host or network firewall
- Servers must be configured to emit logs to a dedicated log collection server
- Vulnerability scans must be scheduled to run and be reviewed at least monthly
- All unnecessary services must be disabled

ISO/IEC JTC 1/SC 42 is the international standards committee responsible for standardization in the area of Artificial Intelligence (AI). It is setup as a joint committee between ISO and IEC, the international standards development organizations (SDOs).

As the focal point of standardization on AI within ISO and IEC, SC 42's program of work looks at the entire AI ecosystem. Additionally, SC 42 is scoped to provide guidance to ISO and IEC committees developing Artificial Intelligence applications.

Its current program of work includes standardization in the areas of foundational AI standards, Big Data, AI trustworthiness, use cases, applications, and governance implications of AI, computational approaches of AI, ethical and societal concerns.

# References:


1. https://getbootstrap.com/docs/5.0/examples/

2. https://nodejs.org/dist/latest-v15.x/docs/api/

3. https://mailchimp.com/developer/api/

4. https://flight-manual.atom.io/api/v1.53.0/AtomEnvironment/

5. Prajna Bhandary - Github Repository, served as a contributor to the dataset

6. https://www.kaggle.com/dhruvmak/face-mask-detection - Contributor to Dataset

7. https://www.kaggle.com/andrewmvd/face-mask-detection

8. Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z (2016). Rethinking the inception architecture for computer vision. In:Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2818-2826

9. Jain, R., Gupta, M., Taneja, S. et al. Deep learning based detection and analysis of COVID-19 on chest X-ray images. Appl Intell (2020).

10. https://doi.org/10.1007/s10489-020-01902-1

11.https://www.linkedin.com/pulse/artificial-intelligence-ai-standards-ray-walshe-

12.https://security.uconn.edu/server-management-standards/

13. Wiki documentation on ROS

14. GitHub for collaboration.

15. MQTT - The Standard for IoT Messaging

16. DIY ESP8266 Home Security with Lua and MQTT